Introduction

Compact wireless keyboards are popularly used with a variety of smart devices such as PC, Mobile, Tablet, and High-end Television. Multifunction Compact Keyboard is a turnkey solution for a capacitive touch Bluetooth keyboard based on Atmel® ATSAMD21 and ATBTLC1000. It features Atmel Peripheral Touch Controller (PTC) based capacitive touch and Atmel SmartConnect Bluetooth® technology. This solution embodies a low-cost, low power compact design, seamless Bluetooth communication and a variety of functions including air mouse and predictive LED backlighting for ease of use.

This document describes the hardware and firmware design of Multifunction Compact Keyboard. The associated package contains the hardware design files and application firmware.

Features

- 40 mutual capacitance touch keys
- Coplanar sensor design
- Air-Mouse function
- Predictive LED backlighting
- Connectivity through Bluetooth® SMART (BLE 4.1)
- Standard USB 2.0 interface
- Battery powered with ~150 hours operation
- Firmware upgrade through on-board USB Boot Loader
# Table of Contents

Introduction ................................................................................................................. 1

Features ...................................................................................................................... 1

1. Abbreviations and Definitions .............................................................................. 3

2. Hardware Design .................................................................................................. 4
   2.1. Power Supply .................................................................................................... 4
       2.1.1. Battery Charging ..................................................................................... 4
       2.1.2. Battery Voltage Monitoring ................................................................. 5
       2.1.3. Voltage Regulator .................................................................................. 5
       2.2. Switch and Buttons ..................................................................................... 6
       2.3. Capacitive Touch Sensors .......................................................................... 6
       2.4. Bluetooth .................................................................................................... 7
       2.5. Angular Motion Sensor .............................................................................. 7
       2.6. LED Driving ............................................................................................... 8
       2.7. Front Panel .................................................................................................. 9

3. Firmware Design .................................................................................................. 10
   3.1. Source Files ................................................................................................... 10
   3.2. ASF Driver and Clock Configuration ............................................................. 11
   3.3. Initialization ................................................................................................... 12
   3.4. Scheduler ....................................................................................................... 12
   3.5. Major Tasks .................................................................................................. 13
       3.5.1. Capacitive Touch Sensor .............................................................. 13
       3.5.2. Detect Single and Double Tap ...................................................... 16
       3.5.3. Angular Motion Sensor ................................................................. 17
       3.5.4. LEDs ................................................................................................. 19
       3.5.5. Bluetooth ........................................................................................... 21
       3.5.6. USB .................................................................................................... 23
   3.6. System Support Tasks .................................................................................... 25
       3.6.1. Time Reference ................................................................................... 25
       3.6.2. Mode Switch ....................................................................................... 25
       3.6.3. Battery Monitoring ............................................................................. 25
       3.6.4. Boot Loader ......................................................................................... 26
       3.6.5. Low Power ......................................................................................... 26

4. Power Consumption ............................................................................................... 27

5. Design Files ......................................................................................................... 28

6. References ............................................................................................................ 29

7. Revision History .................................................................................................. 30
1. **Abbreviations and Definitions**

**Air-Mouse:** A mouse that does not need to be in contact with any surface. To move the cursor on a computer or television screen the user can hold the mouse and move it in air.

**ASF:** Atmel Software Framework is a MCU software library providing a large collection of embedded software for Atmel MCUs.

**Bluetooth Low Energy (BLE):** It is a wireless network technology designed and marketed by the Bluetooth Special Interest Group aimed at novel applications in the healthcare, fitness, beacons, security, and home automation. Compared to Classic Bluetooth, Bluetooth Low Energy or Bluetooth Smart provides considerably reduced power consumption and cost while maintaining a similar communication range.

**Coplanar Sensor Design:** Both X and Y electrodes fabricated on the same layer of the PCB in mutual capacitance sensor design.

**Electrode:** The patch of conductive material on the substrate that forms the sensor. An electrode is usually made from copper, carbon, silver ink, and Indium Tin Oxide (ITO).

**Gyroscope:** A device that sense angular velocity. Rate of rotation of the device in space can be found using 3-axis gyroscope.

**HID:** Human Input Device is a type of compute device that takes input from a human and delivers it to a machine.

**HID Over GATT Profile (HOGP):** This profile defined by Bluetooth SIG enables support of HID services over a BLE protocol stack using Generic Attribute (GATT) profile.

**Mutual Capacitance Sensor:** A sensor with connections to two parts of the sensor, an X (transmit) electrode and a Y (receive) electrode. The controller measures the mutual capacitance from X to Y.

**Peripheral Touch Controller (PTC):** This is a microcontroller peripheral, which acquires signals to detect touch on capacitive sensors.

**Predictive Backlighting:** This feature monitors hovering finger position and glows the LED of keys near to finger position. This feature is useful when the keyboard is operated in low light conditions.

**Sensor:** The component that detects a touch. Sensors consists of one or more electrodes. It can be a button, slider, or wheel.
2. **Hardware Design**

The keyboard hardware consist of multiple major modules as follows.

- Power Supply
- Switch and Button
- Capacitive Touch Sensor
- Bluetooth Module
- LEDs

The following schematic explains various components and the associations among them.

*Figure 2-1  System Block Diagram*

2.1. **Power Supply**

Keyboard is powered using a standard 3.7V single-cell Li-ion battery or three 1.5V AAA alkaline batteries connected in series. The battery voltage is regulated using LDO. If Li-ion battery is used, 5V USB supply can be used for charging. When alkaline battery is used, the USB supply is not utilized.

2.1.1. **Battery Charging**

The keyboard includes a single cell Li-ion battery charging circuitry using the charger IC (U7, RT9524). When the keyboard connects to a USB port, the charging circuitry charges the battery. Following schematic shows the battery charging circuit.
Note: The firmware is capable of enabling/disabling the Li-ion battery charging. By default, Li-ion battery charging is disabled (assuming alkaline batteries are used).

2.1.2. Battery Voltage Monitoring
Battery voltage is monitored using on-chip ADC peripheral. A voltage divider circuitry divides 4.7V to 0.95V that is suitable for use with ADC 1V internal voltage reference. Following schematic shows the battery voltage monitoring circuit.

Figure 2-3 Battery Voltage Monitoring

2.1.3. Voltage Regulator
A linear dropout regulator (LDO) (U3, SPX381R2) uses battery supply as input and generates stable 3.3V output. The regulator can handle a continuous load current up to 500mA. Following schematic shows the voltage regulator circuit.

Figure 2-4 Voltage Regulator Circuit
2.2. **Switch and Buttons**

An SPDT switch is used to control power supply of the keyboard. To power ON the keyboard, slide the switch to the left.

The push button “Mode Switch” is used to switch the operating mode between keyboard and air-mouse.

*Figure 2-5 SPDT and Push Button*

The slide switch and push button are ergonomically positioned to enhance the user experience.

2.3. **Capacitive Touch Sensors**

The keyboard uses touch sensors, which are measured by an on-chip Peripheral Touch Controller (PTC). A mutual capacitance sensor electrode consists of an X-electrode (Transmitter) and a Y-electrode (Receiver). The sensor electrode sets up an electric field between the electrodes. One X-line (drive line) and one Y-line (sense line) are required to form a mutual capacitive key.

The mutual capacitance method allows keys to be organized as touch matrices in different X-Y configurations. In this design, 40 mutual capacitance keys are arranged in a 10x4 matrix form. The following figure shows the touch key arrangement.

*Figure 2-6 Capacitive Touch Sensor*

Coplanar sensor design has been used to achieve optimum sensitivity for predictive backlighting and to accommodate back lighting LEDs. The size of each sensor electrode is 10mm x 10mm. In coplanar design, each sensor electrode consists of multiple X and Y interlocking fingers.

The following figure represents the sensor design used in the keyboard.
The X-Y separation is selected to achieve optimum sensitivity. The thickness of the center X finger is reduced to accommodate the hole for backlighting LED.

2.4. Bluetooth

The keyboard uses the Atmel Bluetooth module, ATBTL1000-MR110CA, for BLE communications. ATBTL1000-MR110CA module is a complete Bluetooth connectivity module comprising ATBTL1000 SoC and other required circuitry. An external 32kHz crystal is connected to the module as required. The following schematic shows the Bluetooth module circuit.

2.5. Angular Motion Sensor

A gyroscope is used to detect angular motion for air-mouse functionality. The BMG160 is a digital 3-axis angular rate sensor. It supports a measurement range up to 2000 degree/second and a digital resolution of 16-bit. The following schematic shows the gyroscope circuit.
2.6. LED Driving

LEDs placed behind the keys are useful for predictive backlighting while an approaching finger is a few millimeters away from the keys. The LEDs also indicates the touch status. The keyboard uses a general-purpose LED matrix driver IC (IS31FL3728). The IS31FL3728 supports I²C communication and it is configured in 6x10 matrix mode. The host can disable the LED driver by using a GPIO. This feature is highly useful to suppress LED noise during touch measurement and to reduce power consumption. The following schematic shows the LED driver circuit.
2.7. **Front Panel**
Atmel always recommends a front panel with good *dielectric properties*. For the keyboard, a 1.8mm thick white acrylic is used as front panel. White acrylic allows uniform diffusion for LED backlighting. The diffusion area increases with thickness. With 1.8mm, the diffusion is sufficient to cover individual keys. The front panel is painted using non-conductive black color on top side. While painting, cutout for text, border, and LEDs are provided. The following figure represents the front panel.

*Figure 2-11  Front Panel Design*

The front panel is glued to the PCB using a 3M double-sided adhesive sheet as shown.

*Figure 2-12  Stack-up Details*
3. **Firmware Design**

The firmware is divided into multiple tasks:

- Capacitive touch sensing
- Single and double tap detections
- Predictive Backlighting
- Update LEDs based on touch sensor states
- Process gyroscope data
- Monitor battery voltage
- Initialize Bluetooth and USB
- Transfer data through Bluetooth or USB

The following figure represents the interface between application and various tasks.

**Figure 3-1 Firmware Tasks**

3.1. **Source Files**

The following table provides the list of source files and their contents.
### Table 3-1 Source File

<table>
<thead>
<tr>
<th>File Name</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>main.c</code></td>
<td>• main() function • Peripherals initialization • Firmware task scheduling</td>
</tr>
<tr>
<td><code>touch.c</code> and <code>touch.h</code></td>
<td>• Touch sensor configuration and measurement</td>
</tr>
<tr>
<td><code>led.c</code> and <code>led.h</code></td>
<td>• Driver for IS31FL3728 device • Predictive Backlighting • Turn ON/OFF backlighting LEDs based on touch status</td>
</tr>
<tr>
<td><code>tch_process.c</code> and <code>tch_process.h</code></td>
<td>• Process touch sensor state and resolve preferences</td>
</tr>
<tr>
<td><code>gesture.c</code> and <code>gesture.h</code></td>
<td>• Detect single and double tap</td>
</tr>
<tr>
<td><code>bmg160.c</code> and <code>bmg160.h</code></td>
<td>• Driver for BMG160 device Gyroscope data processing</td>
</tr>
<tr>
<td><code>i2c.c</code> and <code>i2c.h</code></td>
<td>• Low level I²C driver for IS31FL3728 and BMG160</td>
</tr>
<tr>
<td><code>bt_app.c</code> and <code>bt_app.h</code></td>
<td>• Bluetooth initialization • Send response to pairing request</td>
</tr>
<tr>
<td><code>usb_app.c</code> and <code>usb_app.h</code></td>
<td>• USB initialization • Update system status based on various USB events</td>
</tr>
<tr>
<td><code>sys_assit.c</code> and <code>sys_assit.h</code></td>
<td>• System status initialization and updates • Battery voltage monitoring • Mode switching • Status LED update</td>
</tr>
<tr>
<td><code>hid_app.c</code> and <code>hid_app.h</code></td>
<td>• Fill the circular buffer with HID key codes • Send HID data via USB/BLE</td>
</tr>
<tr>
<td><code>ble_hid.c</code> and <code>ble_hid.h</code></td>
<td>• HID profile initialization • Update system status based on various BLE events</td>
</tr>
</tbody>
</table>

### 3.2. ASF Driver and Clock Configuration

The application uses drivers from Atmel Software Framework (ASF) to interact with various peripherals. The following table provides the list of ASF drivers.
### Table 3-2 ASF Drivers

<table>
<thead>
<tr>
<th>ASF Driver</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADC</td>
<td>Battery voltage monitoring</td>
</tr>
<tr>
<td>Delay</td>
<td>Application</td>
</tr>
<tr>
<td>EXTINT</td>
<td>To switch between keyboard and mouse mode</td>
</tr>
<tr>
<td>GPIOs</td>
<td>ATBTLC1000, LED driver, charging circuitry, and LED indication</td>
</tr>
<tr>
<td>SERCOM - I2C Master</td>
<td>Gyroscope and LED Driver</td>
</tr>
<tr>
<td>SERCOM - USART</td>
<td>ATBTLC1000</td>
</tr>
<tr>
<td>USB composite device</td>
<td>USB keyboard and mouse</td>
</tr>
</tbody>
</table>

The following schematic represents the clock source used by various MCU peripherals.

**Figure 3-2 Clock Tree**

#### 3.3. Initialization

After system reset, clock sources, peripherals, and firmware tasks are initialized. Detailed initialization of major tasks are covered in their respective sections.

#### 3.4. Scheduler

Scheduler monitors system time and invokes various tasks based on their scheduled time. The following table lists major tasks invoked by the scheduler.
Table 3-3 Task Details

<table>
<thead>
<tr>
<th>Major Task</th>
<th>Schedule Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Touch measurement</td>
<td>Every 20msec</td>
</tr>
<tr>
<td>Touch state process</td>
<td>End of every touch measurement</td>
</tr>
<tr>
<td>LED driver</td>
<td>End of every touch measurement</td>
</tr>
<tr>
<td>Read and process gyroscope data</td>
<td>Every 10msec</td>
</tr>
<tr>
<td>BLE / USB – keyboard data transaction</td>
<td>Every 20msec</td>
</tr>
<tr>
<td>BLE / USB – mouse data transaction</td>
<td>End of gyroscope data processing</td>
</tr>
<tr>
<td>BLE event manager</td>
<td>As frequently as possible. Called once in infinite while loop. Maximum interval is 5msec (RTC interrupt)</td>
</tr>
</tbody>
</table>

The following graphic represents high-level program flow of major tasks and their respective trigger conditions.

**Figure 3-3 Scheduler**

3.5. **Major Tasks**

3.5.1. **Capacitive Touch Sensor**

Sensor Configuration
The keyboard has 40 mutual capacitance sensor channels and scanning these channels takes considerable amount of time. This would increase the response time and power consumption. To optimize response time and power consumption, lump feature of PTC QTTouch library is used.

The mutual capacitance touch sensors are configured as horizontal and vertical lump sensors as shown in the following illustrations.

**Figure 3-4 Horizontal Lump Configuration**

<table>
<thead>
<tr>
<th>X11</th>
<th>X10</th>
<th>X9</th>
<th>X8</th>
<th>X7</th>
<th>X5</th>
<th>X4</th>
<th>X14</th>
<th>X13</th>
<th>X12</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y6</td>
<td>LUMP 0</td>
<td></td>
<td>LUMP 1</td>
<td></td>
<td>LUMP 2</td>
<td></td>
<td>LUMP 3</td>
<td></td>
<td>LUMP 4</td>
</tr>
<tr>
<td>Y7</td>
<td>LUMP 1</td>
<td></td>
<td>LUMP 2</td>
<td></td>
<td>LUMP 3</td>
<td></td>
<td>LUMP 4</td>
<td></td>
<td>LUMP 5</td>
</tr>
<tr>
<td>Y8</td>
<td>LUMP 2</td>
<td></td>
<td>LUMP 3</td>
<td></td>
<td>LUMP 4</td>
<td></td>
<td>LUMP 5</td>
<td></td>
<td>LUMP 6</td>
</tr>
<tr>
<td>Y9</td>
<td>LUMP 3</td>
<td></td>
<td>LUMP 4</td>
<td></td>
<td>LUMP 5</td>
<td></td>
<td>LUMP 6</td>
<td></td>
<td>LUMP 7</td>
</tr>
</tbody>
</table>

**Figure 3-5 Vertical Lump Configuration**

<table>
<thead>
<tr>
<th>X11</th>
<th>X10</th>
<th>X9</th>
<th>X8</th>
<th>X7</th>
<th>X5</th>
<th>X4</th>
<th>X14</th>
<th>X13</th>
<th>X12</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y6</td>
<td>LUMP 8</td>
<td>LUMP 9</td>
<td>LUMP 10</td>
<td>LUMP 11</td>
<td>LUMP 12</td>
<td>LUMP 13</td>
<td>LUMP 14</td>
<td>LUMP 15</td>
<td>LUMP 16</td>
</tr>
</tbody>
</table>

This lump configuration does not result in PTC saturation and increases proximity range for predictive backlighting. Using the lump feature, the total number of touch sensors are reduced to 18.

**Touch Sensor Post Processing**

Touch status of 40 keys is derived from 18 lump sensor states. The touch status of horizontal and vertical lump sensors is interpolated to get the actual key state. If vertical and horizontal lump sensors are in detect, the intersecting key is declared as touched as shown in following figure.
Resolve Lazy Finger Touch

Due to compact sensor design, multiple sensors will be touched if the user is not careful while typing. For example, when the user wants to touch the ‘Y’ key, the user can stretch the thumb and touch the ‘Y’. It is possible that adjacent keys, ‘H’, ‘U’, and ‘J’ are also touched as shown in the following figure.

Similarly on left side, when the user stretch and touch the ‘V’ key, adjacent keys ‘C’, ‘,’ and ‘Space’ could also be touched.

AKS feature is not used to solve this issue due to:
• AKS feature reports the first sensor touched by the user which could be unintended key
• Using AKS feature increases power consumption and response time as it conflicts with Quick Reburst

Application post processing is modified to avoid reporting of multiple keys during careless touches. Keyboard is split in to left and right side corresponding to left and right thumb. The application sets preference to left and right side keys. The preference is derived based on the way the user intended to type and is actually typing.
The key preference is illustrated in the above figure. Higher number means higher preference. On the left side, preference increases from left to right for column. On the right side, preference increases from right to left for column. Both left and right side preference is higher for upper rows compared with lower rows.

For example, if the user carelessly touches ‘F’ key, adjacent keys such as ‘D’, ‘X’, and ‘C’ will also be touched. Here, ‘F’ key has higher preference in both column and row as comparing to other keys. So, ‘F’ key is reported as touched. Similarly, if the user carelessly touches ‘Y’ key, adjacent keys such as ‘H’, ‘U’, and ‘J’ will also be touched. The key ‘Y’ has higher preference and it is reported as touched.

3.5.2. **Detect Single and Double Tap**

Key tap is detected based on touch status of 40 keys. When a key is in detect, the key press is reported and masked for further reporting. The key is unmasked when key goes out of detect. With this, when user touches a key, a report for that key is sent once. The following diagram illustrates the flow to detect a single tap.

*Figure 3-8 Detect Single Tap*

The double tap is detected on Shift and Alternative-Character key. The double tap detection is similar to single tap. A timeout of 200msec is introduced between two consecutive tap detections. If second tap did not occur within 200msec, then it is considered as single tap.

For every key tap, the corresponding HID key code is inserted in to the circular buffer. Some keys require a sequence of key code to be sent. For example, if the user intend to type ‘@’, the user needs to press ‘shift’, press ‘2’, release ‘2’, and then releases ‘shift’. Similar to this, the keyboard inserts four character codes, ‘Shift’ down, ‘2’ down, ‘2’ up and ‘Shift’ up to transmit buffer if user types ‘@’. For each code, write pointer is incremented.
3.5.3. Angular Motion Sensor Initialization

The BMG160 gyroscope supports a maximum range of ±2000 degree/second and has inbuilt low-pass filter with configurable cutoff frequency. The BMG160 is configured as follows.

- **Range**: ±2000 degree/second
- **Filter Bandwidth**: 116Hz

**Process Gyroscope Data**

The BMG160 provides 3-axis angular velocity. When the keyboard is stationary, the BMG160 reports zero on all three axis. When the keyboard is rotated, the angular velocity of each axis varies in positive or negative direction. The MCU reads angular velocity data every 10 milliseconds. Based on the angular velocity, the angular displacement and mouse pixel data is computed. An acceleration algorithm is applied on the mouse pixel data to improve the user experience. The acceleration algorithm is implemented in the firmware. The following figure represents the flow to compute mouse data from gyroscope reading.
Figure 3-10 Process Gyroscope Data

To improve the precision, gyroscope related calculations are performed in floating point notation. The X and Y-axis data of the gyroscope is used to compute X and Y-axis mouse data respectively.

The absolute angle to which keyboard is rotated can be found by using the following formula.

\[
\text{Absolute Angle} = \text{Absolute Angle} + \left( \frac{\text{Gyroscope Data}}{16.4} \right) \times \text{Sample Interval}
\]

Where,
- 16.4 is the sensitivity at 2000 degree/seconds with an unit of LSB/degree/second
- Sample interval is 0.01 (10msec)

Considering that the gyroscope has an offset error, the formula to calculate the angle is modified as follows.

\[
\text{Absolute Angle} = \text{Absolute Angle} + \left( \frac{\text{Gyroscope Data} - \text{Offset Error}}{16.4} \right) \times \text{Sample Interval}
\]

To compute offset error, 32 samples are collected during initialization at 5msec intervals. The average value of 32 samples is considered as offset error.

Angular Displacement between previous and current absolute angle value is used to compute pixel value. Based on pixel value, the mouse pointer does relative movements on the screen. The pixel value is required in the mouse HID report.

\[
\text{Angular Displacement} = \text{Absolute Angle}_{N} - \text{Absolute Angle}_{N-1}
\]

Mouse data is calculated by multiplying the difference by Pixel Factor.

\[
\text{Pixel Value} = \text{Angular Displacement} \times \text{Pixel Factor}
\]

- Pixel Factor is an empirical value, which is configured as 15 for both X and Y-axis

To improve the user experience, acceleration algorithm is applied on Pixel Value as follows.
The above formula accelerates the mouse pointer in exponential fashion. The Acceleration Coefficient1 and Acceleration Coefficient2 are empirical values derived on experiments. Coefficient1 and Coefficient2 are configured as 10 and 0.5 respectively.

3.5.4. LEDs

Initialization

LED driver, IS31FL3728, supports a maximum of 64 LEDs in matrix configuration. By default, IS31FL3728 configures in 8x8 mode. For keyboard, the configuration is changed to 6x10. The LED drive intensity is set to 35mA per row.

Touch Status

LED frame buffer is updated based on touch status of keys. Based on Shift and Alternative-character key status LED frame buffer is updated. If no sensor is in detect, sensors are checked for predictive backlighting.

Figure 3-11 LED Indication Flow

Predictive Backlighting

The predictive backlighting monitors hovering finger position and glows LEDs to assist the user. When the user holds a finger above (not touching) the keys, the LEDs are turned ON for respective and surrounding keys as shown in the following figure.
This feature is useful when the keyboard is used in low-light conditions. To improve the user experience, the left and right side keys are monitored separately.

The predictive backlight algorithm monitors touch delta and glows the LED when touch sensor’s delta crosses configured threshold. A delay is introduced between when the user holds a finger over some keys and the LED glows. This delay ensures that prediction algorithm does not glow LEDs when the user types in normal speed (assuming no backlight is required). Moving average filter is used to filter noise and introduces the required delay as mentioned above. The following figure represents the flow to perform predictive backlighting.
3.5.5. **Bluetooth**

ATBTLC1000 provides Bluetooth Smart Link Controller that includes RF, Link Layer, GAP, GATT, and SMP in a single SoC. It provides the host microcontroller with methods to perform standard Bluetooth Smart GAP, GATT server, and client operations as well as security management with peer devices.

**BluSDK** is an extension for Atmel Studio using, which supported profiles can be integrated to the application. After installing BluSDK extension, Bluetooth services can be integrated using ASF wizard. Keyboard application uses HID Over GATT Profile (HOGP) profile available as part of BluSDK.

**Configuration**

Keyboard acts as Bluetooth HID device (GATT server) and sends HID data using HOGP. The keyboard is configured as HID generic device with two reports. Report 0 represents mouse IN data and Report 1 represents keyboard IN data. The following figure represents high level Bluetooth configuration.

**Figure 3-14 Bluetooth Configuration**

![Bluetooth Configuration](image)

**Data Transactions**

BLE library uses the UART physical layer for data transaction between ATBTLC1000 and ATSAM D21 device. The data transaction can be split in to various stages. The following section captures high-level program flow between application and BLE library.

**Bluetooth Initialize**

During system initialization, BLE library is initialized with profile and HID configuration data. The library initializes ATBTLC1000 with HID configuration and starts advertisement. After initialization, the application should frequently check for a new event, as shown in the following figure. BLE library processes the new event. If the new event requires application attention, BLE library indicates this to application.
Pair Key Request

When BLE host request pairing with keyboard, the BLE event manager communicates the pairing request to the application, as shown in the following figure.

Report Notification

After report notification, the application sends new keyboard data every 20 milliseconds using “report update” API (mouse data every 10 milliseconds). This data is sent to BLE host by ATBTLC1000.
Send HID Data

After report notification, the application sends new keyboard data every 20 milliseconds using “report update” API (mouse data every 10 milliseconds). This data is sent to BLE host by ATBTL1000.

Configuration

The keyboard is configured as generic HID device with two interfaces. Each interface has one endpoint. Endpoint 1 represents keyboard IN data and Endpoint 2 represents mouse IN data. The following figure represents USB configuration.
Data Transaction
ASF Driver is used to initialize and transfer data between the application and the USB peripheral. The data transaction can be split into various stages as follows.

USB Initialization
The USB driver is configured with HID configuration data during system initialization. The ASF driver initializes the USB peripheral.

USB Host Connection
The ASF driver indicates USB host connection status to application via callback. The application uses this notification to know whether the keyboard and mouse data can be transmitted.
Send HID Data

When the keyboard is connected to USB, keyboard data is transmitted every 20 milliseconds to USB host using ASF driver (mouse data every 10 milliseconds), as shown in the following figure.

3.6. System Support Tasks

3.6.1. Time Reference
Real Time Counter (RTC) is used for time reference. RTC is configured to issue interrupt every 5msec. On RTC interrupt, different system time references are updated.

3.6.2. Mode Switch
External Interrupt (EXTINT) is configured to issue interrupt whenever “Mode switch” pin goes low. In interrupt callback, system status is switched between keyboard and mouse mode.

3.6.3. Battery Monitoring
Battery voltage is monitored using 12-bit ADC. Internal voltage reference (1V) is used. The battery voltage is read once during initialization and read every 5 seconds after initialization. If battery voltage is less than 3.3V status LED starts toggling every 500msec to alert the user about low voltage. After the battery voltage drops below 3.0V, all devices and peripherals are configured to sleep and no operation is performed. The battery must be replaced.
3.6.4. **Boot Loader**

USB CDC based **SAM-BA boot loader** is used for upgrading firmware. The SAM-BA boot loader uses PA15 pin for hardware boot loader entry. For keyboard, hardware entry is changed to “Mode switch” (PA00) pin. After system reset, the boot loader monitors "Mode switch" pin status and application flash regions.

The control remains in boot loader if,

- Mode switch button is pressed
- If flash content of application region is 0xFFFFFFF

New firmware can be programmed using **SAM-BA GUI** based tool.

3.6.5. **Low Power**

All the devices are configured to operate in low power mode. This helps in reducing the average current consumption of keyboard. Along with this, the application monitors the inactivity period. If no activity is detected beyond 2 minutes, all peripherals and devices are configured to sleep and no operation is performed. Under this condition, a power cycle is required.
4. **Power Consumption**

The following table provides the power consumption of various hardware modules at 3.3V.

**Table 4-1 Power Consumption of Different Hardware Modules**

<table>
<thead>
<tr>
<th>Module</th>
<th>Active Mode</th>
<th>Current Consumption [mA]</th>
<th>Low Power Mode [μA]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Condition</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SAM D21</td>
<td>Without USB</td>
<td>3.2</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>With USB</td>
<td>7.2</td>
<td></td>
</tr>
<tr>
<td>ATBTL1000</td>
<td>During Advertisement</td>
<td>0.28</td>
<td>32</td>
</tr>
<tr>
<td></td>
<td>Post Pairing</td>
<td>1.4</td>
<td></td>
</tr>
<tr>
<td>LED Driver</td>
<td>No LED Glowing</td>
<td>0.0006</td>
<td>0.36</td>
</tr>
<tr>
<td></td>
<td>While LED Glowing</td>
<td>3 + 3mA/LED</td>
<td></td>
</tr>
<tr>
<td>BMG160</td>
<td></td>
<td>2.853</td>
<td>85</td>
</tr>
</tbody>
</table>

The following table provides average power consumption of keyboard in different conditions. The power numbers are captured with a battery voltage of 4.5V.

**Table 4-2 Average Current Consumption of Keyboard**

<table>
<thead>
<tr>
<th>Firmware Features</th>
<th>Bluetooth [mA]</th>
<th>USB [mA]</th>
</tr>
</thead>
<tbody>
<tr>
<td>All features enabled</td>
<td>14.6</td>
<td>17.8</td>
</tr>
<tr>
<td>Without predictive backlighting</td>
<td>10.5</td>
<td>14.2</td>
</tr>
<tr>
<td>Without LEDs</td>
<td>6.7</td>
<td>10.2</td>
</tr>
<tr>
<td>Low power mode (after inactive</td>
<td>0.14</td>
<td>0.14</td>
</tr>
<tr>
<td>period of 2 minutes)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1000mAh battery when used without using LEDs, keyboard can work for ~150 hours with Bluetooth connection.
5. **Design Files**

The design files are available in the associated zip package. The hardware design files include schematic, Gerber, and BOM. The firmware design files include application source files and binary file of SAM-BA boot loader.
6. References


[2] SAM D21 USB


[4] HID Over GATT Profile
https://developer.bluetooth.org/TechnologyOverview/Pages/HOGP.aspx

[5] ATB TLC1000 BluSDK Release Package

### 7. Revision History

<table>
<thead>
<tr>
<th>Doc Rev.</th>
<th>Date</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>42579B</td>
<td>11/2015</td>
<td>Non-technical enhancements to the images.</td>
</tr>
</tbody>
</table>
Atmel Corporation
1600 Technology Drive, San Jose, CA 95110 USA
T: (+1)(408) 441.0311 F: (+1)(408) 436.4200
www.atmel.com


Atmel®, Atmel logo and combinations thereof, Enabling Unlimited Possibilities®, and others are registered trademarks or trademarks of Atmel Corporation in U.S. and other countries. ARM®, ARM Connected® logo, and others are the registered trademarks or trademarks of ARM Ltd. Windows® is a registered trademark of Microsoft Corporation in U.S. and or other countries. Other terms and product names may be trademarks of others.

DISCLAIMER: The information in this document is provided in connection with Atmel products. No license, express or implied, by estoppel or otherwise, to any intellectual property right is granted by this document or in connection with the sale of Atmel products. EXCEPT AS SET FORTH IN THE ATMEL TERMS AND CONDITIONS OF SALES LOCATED ON THE ATMEL WEBSITE, ATMEL ASSUMES NO LIABILITY WHATSOEVER AND DISCLAIMS ANY EXPRESS, IMPLIED OR STATUTORY WARRANTY RELATING TO ITS PRODUCTS INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. IN NO EVENT SHALL ATMEL BE LIABLE FOR ANY DIRECT, INDIRECT, CONSEQUENTIAL, PUNITIVE, SPECIAL OR INCIDENTAL DAMAGES (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS AND PROFITS, BUSINESS INTERRUPTION, OR LOSS OF INFORMATION) ARISING OUT OF THE USE OR INABILITY TO USE THIS DOCUMENT, EVEN IF ATMEL HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Atmel makes no representations or warranties with respect to the accuracy or completeness of the contents of this document and reserves the right to make changes to specifications and products descriptions at any time without notice. Atmel does not make any commitment to update the information contained herein. Unless specifically provided otherwise, Atmel products are not suitable for, and shall not be used in, automotive applications. Atmel products are not intended, authorized, or warranted for use as components in applications intended to support or sustain life.

SAFETY-CRITICAL, MILITARY, AND AUTOMOTIVE APPLICATIONS DISCLAIMER: Atmel products are not designed for and will not be used in connection with any applications where the failure of such products would reasonably be expected to result in significant personal injury or death ("Safety-Critical Applications") without an Atmel officer's specific written consent. Safety-Critical Applications include, without limitation, life support devices and systems, equipment or systems for the operation of nuclear facilities and weapons systems. Atmel products are not designed nor intended for use in military or aerospace applications or environments unless specifically designated by Atmel as military-grade. Atmel products are not designed nor intended for use in automotive applications unless specifically designated by Atmel as automotive-grade.