Atmel QTouch

AT12699: Multifunction Compact Keyboard

APPLICATION NOTE

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

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.

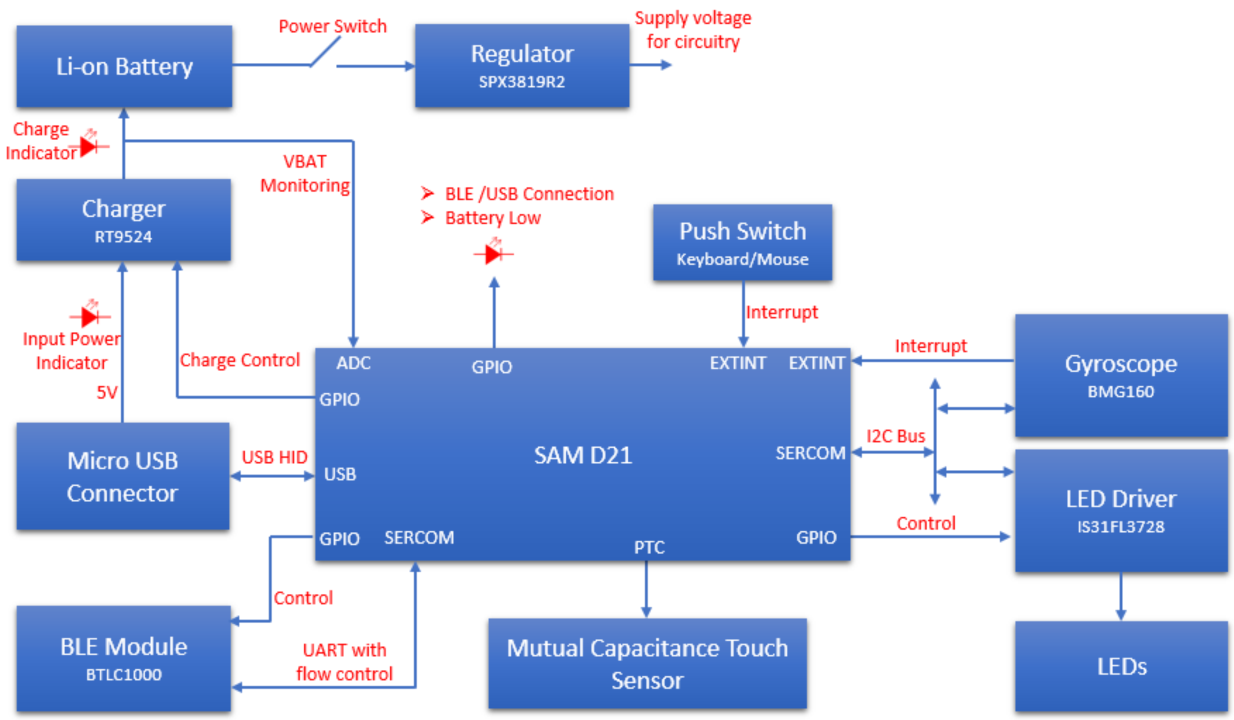
2. Hardware Design

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



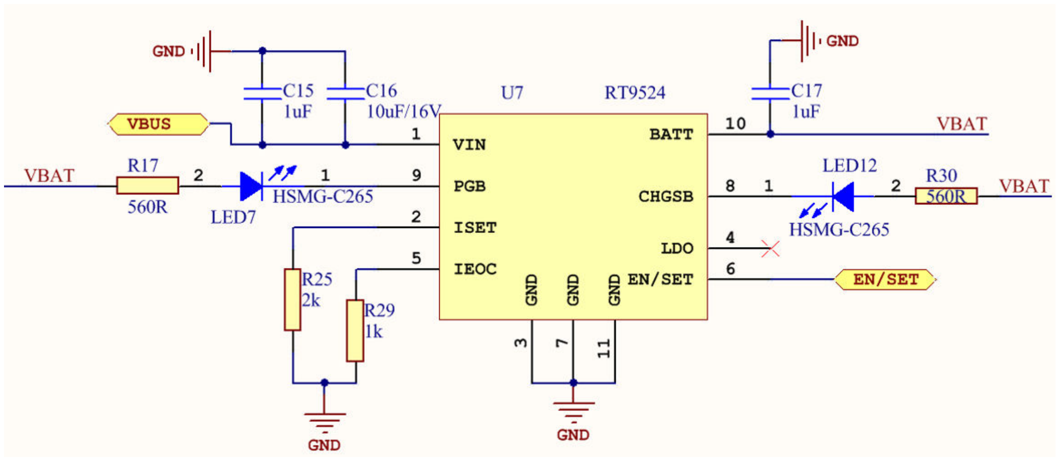
2.1. Power Supply

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.

Figure 2-2 Battery Charging Circuit

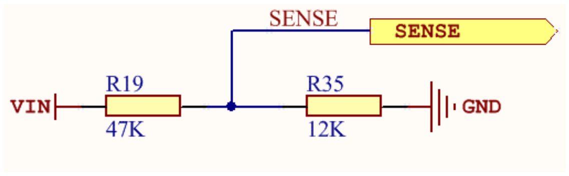


Note: The firmware is capable of enabling/disabling the Li-ion battery charging. By default, Li-ion battery charging is disabled (assuming alkaline batteries are used).

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.

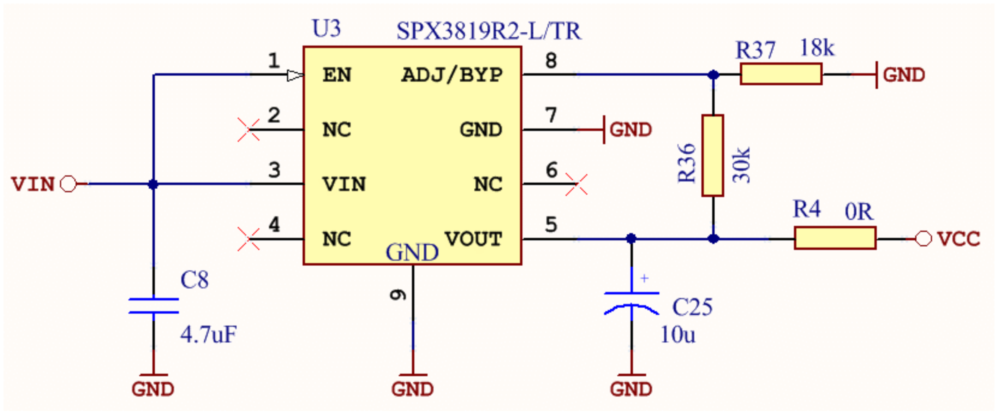
Figure 2-3 Battery Voltage Monitoring



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

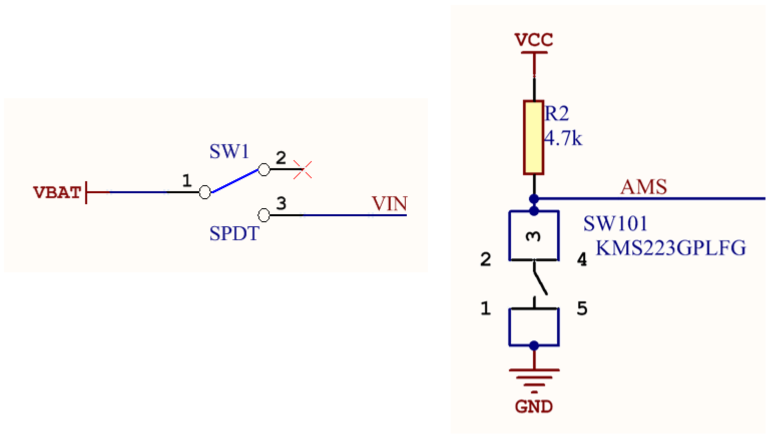


2.2. Switch and Buttons

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



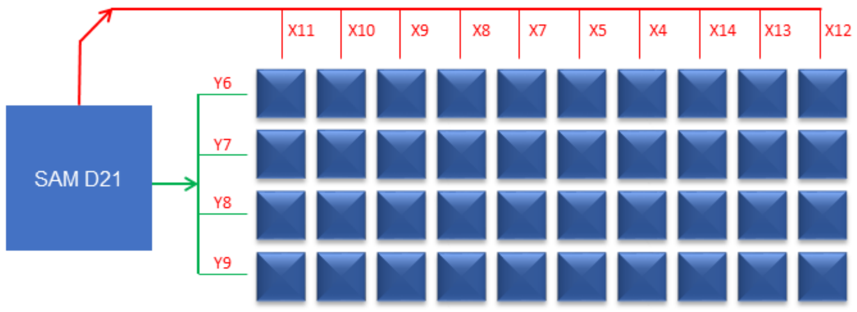
The slide switch and push button are ergonomically positioned to enhance the user experience.

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.

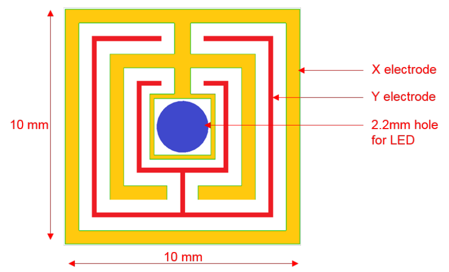
Figure 2-6 Capacitive Touch Sensor



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.

Figure 2-7 Coplanar Sensor Electrode

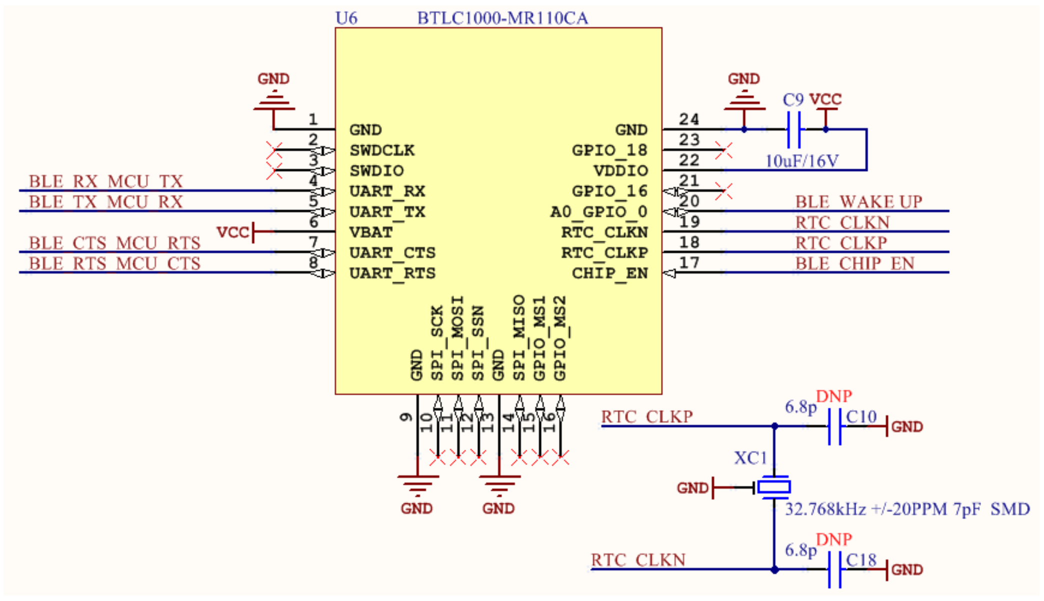


The X-Y separation is selected to achieve optimum sensitivity. The thickness of the center X finger is reduced to accommodate the hole for backlighting LED.

2.4. Bluetooth

The keyboard uses the Atmel Bluetooth module, ATBTLC1000-MR110CA, for BLE communications. ATBTLC1000-MR110CA module is a complete Bluetooth connectivity module comprising ATBTLC1000 SoC and other required circuitry. An external 32kHz crystal is connected to the module as required. The following schematic shows the Bluetooth module circuit.

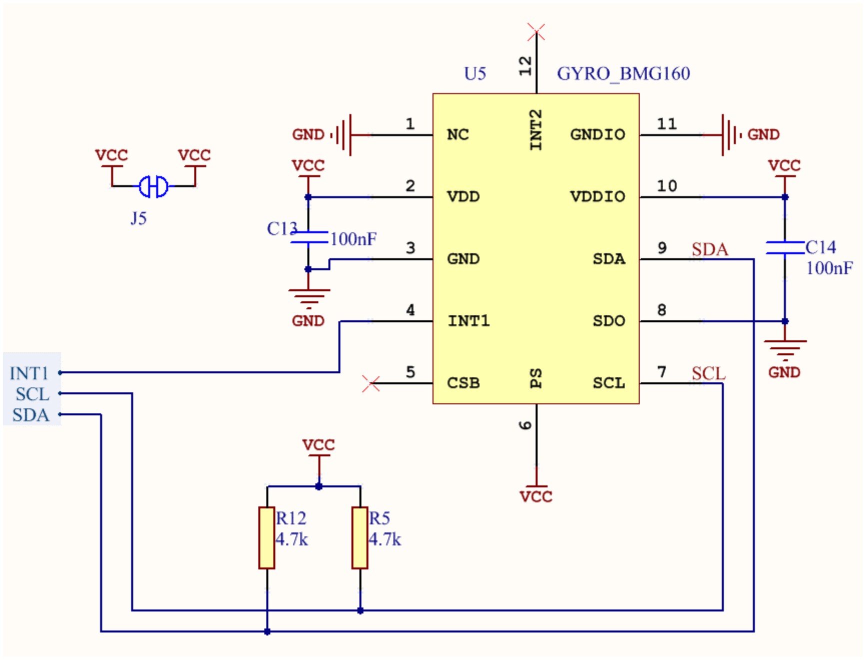
Figure 2-8 BLE Circuit



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.

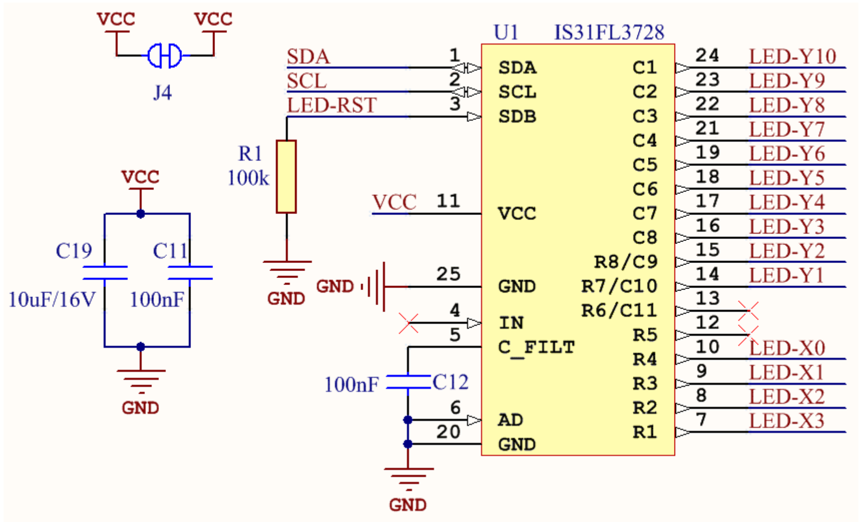
Figure 2-9 Gyroscope Circuit



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 I^{2}C 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.

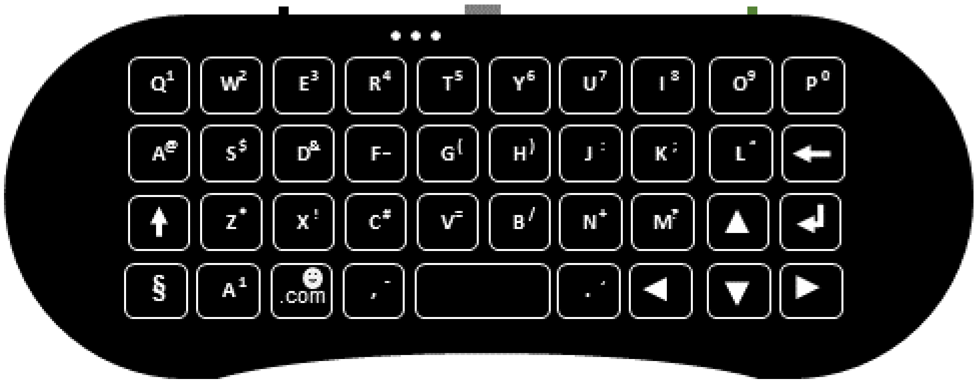
Figure 2-10 LED Driver Circuit



2.7. Front Panel

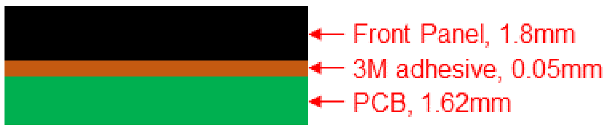
Atmel always recommends a front panel with good dielectric properties. For the keyboard, a 1.8mm thick white acrylic is used as front panel. White acrylic allows uniform diffusion for LED backlighting. The diffusion area increases with thickness. With 1.8mm, the diffusion is sufficient to cover individual 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



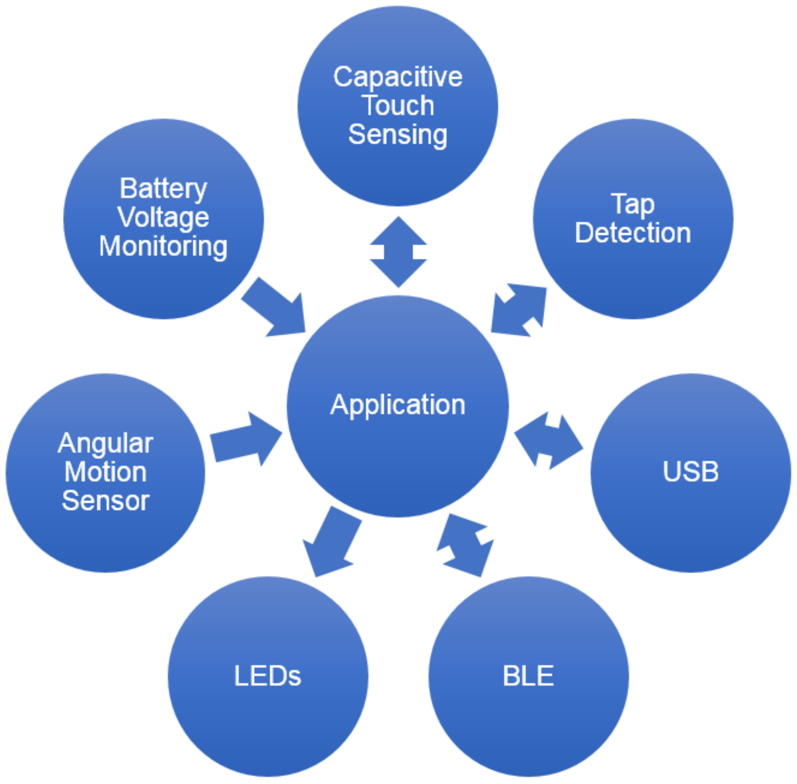
3. Firmware Design

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

File Name	Content
`main.c`	• `main()` function • Peripherals initialization • Firmware task scheduling
`touch.c` and `touch.h`	• Touch sensor configuration and measurement
`led.c` and `led.h`	• Driver for IS31FL3728 device • Predictive Backlighting • Turn ON/OFF backlighting LEDs based on touch status
`tch_process.c` and `tch_process.h`	• Process touch sensor state and resolve preferences
`gesture.c` and `gesture.h`	• Detect single and double tap
`bmg160.c` and `bmg160.h`	• Driver for BMG160 device Gyroscope data processing
`i2c.c` and `i2c.h`	• Low level I^{2}C driver for IS31FL3728 and BMG160
`bt_app.c` and `bt_app.h`	• Bluetooth initialization • Send response to pairing request
`usb_app.c` and `usb_app.h`	• USB initialization • Update system status based on various USB events
`sys_assit.c` and `sys_assit.h`	• System status initialization and updates • Battery voltage monitoring • Mode switching • Status LED update
`hid_app.c` and `hid_app.h`	• Fill the circular buffer with HID key codes • Send HID data via USB/BLE
`ble_hid.c` and `ble_hid.h`	• HID profile initialization • Update system status based on various BLE events

3.2. ASF Driver and Clock Configuration

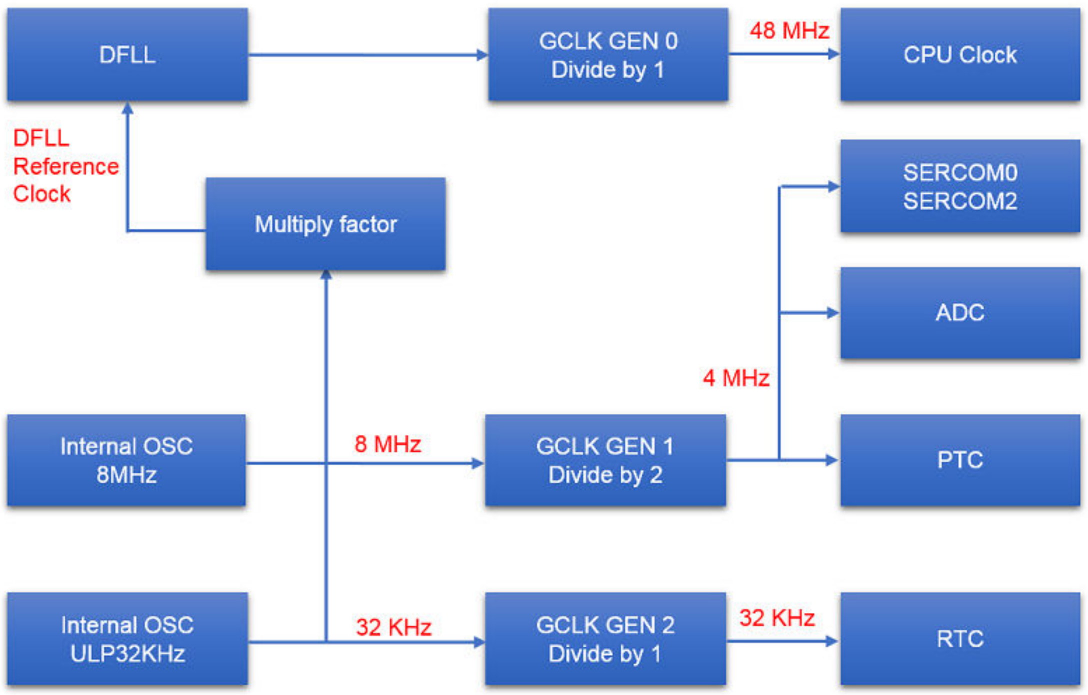
The application uses drivers from Atmel Software Framework (ASF) to interact with various peripherals. The following table provides the list of ASF drivers.

Table 3-2 ASF Drivers

ASF Driver	Purpose
ADC	Battery voltage monitoring
Delay	Application
EXTINT	To switch between keyboard and mouse mode
GPIOs	ATBTLC1000, LED driver, charging circuitry, and LED indication
SERCOM - I2C Master	Gyroscope and LED Driver
SERCOM - USART	ATBTLC1000
USB composite device	USB keyboard and mouse

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

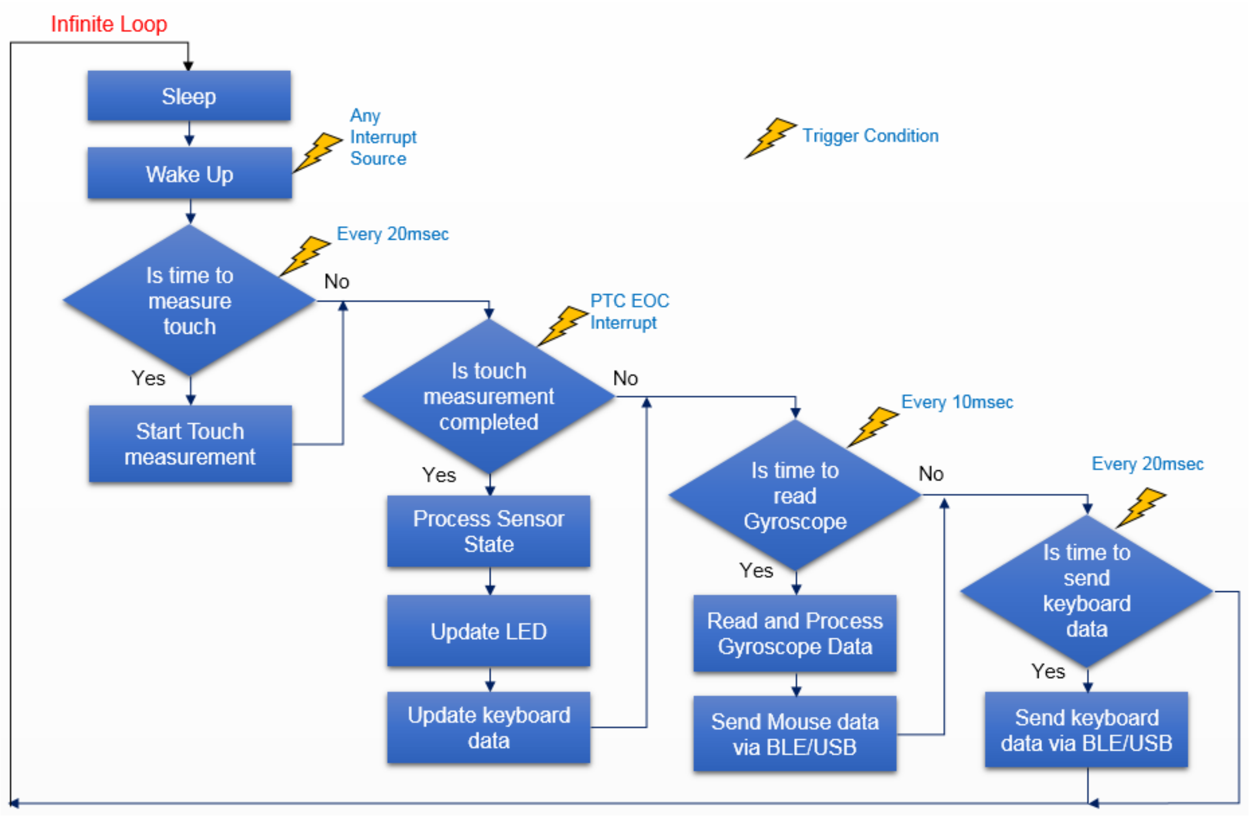
Scheduler monitors system time and invokes various tasks based on their scheduled time. The following table lists major tasks invoked by the scheduler.

Table 3-3 Task Details

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

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 QTouch 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

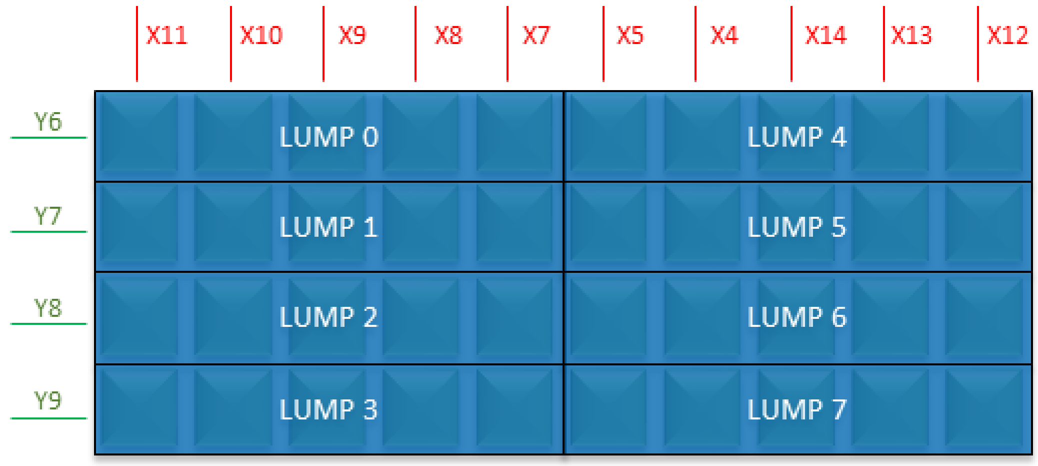
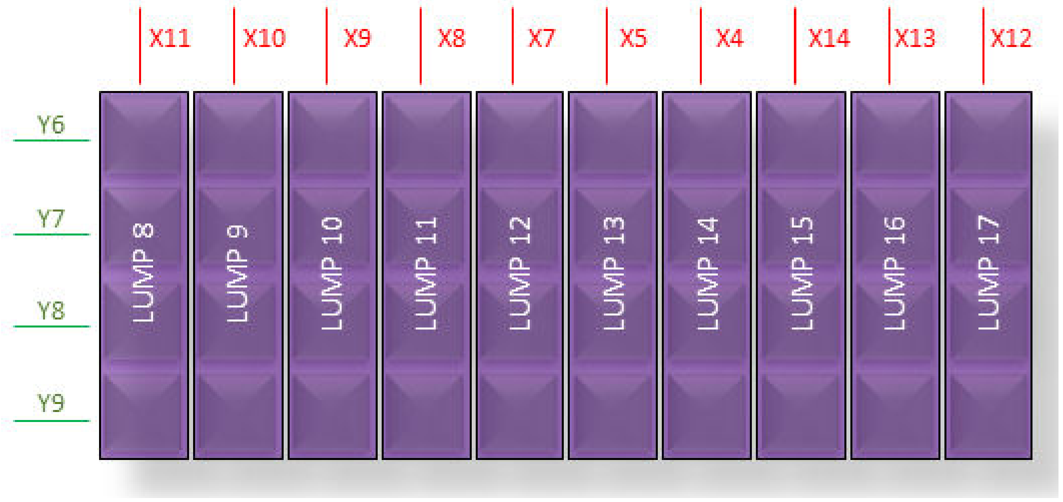


Figure 3-5 Vertical Lump Configuration

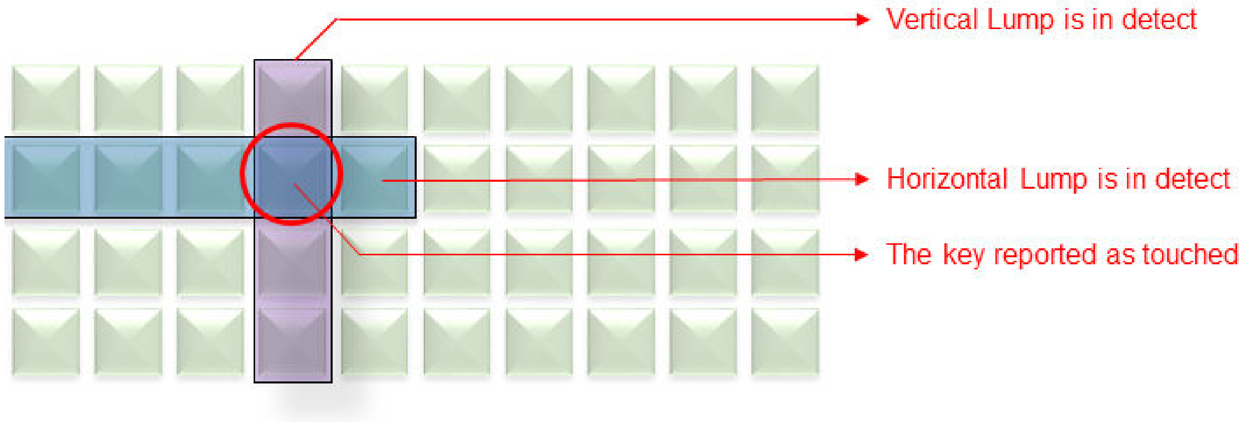


This lump configuration does not result in PTC saturation and increases proximity range for predictive backlighting. Using the lump feature, the total number of touch sensors are reduced to 18.

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.

Figure 3-6 Touch Sensor Post Processing

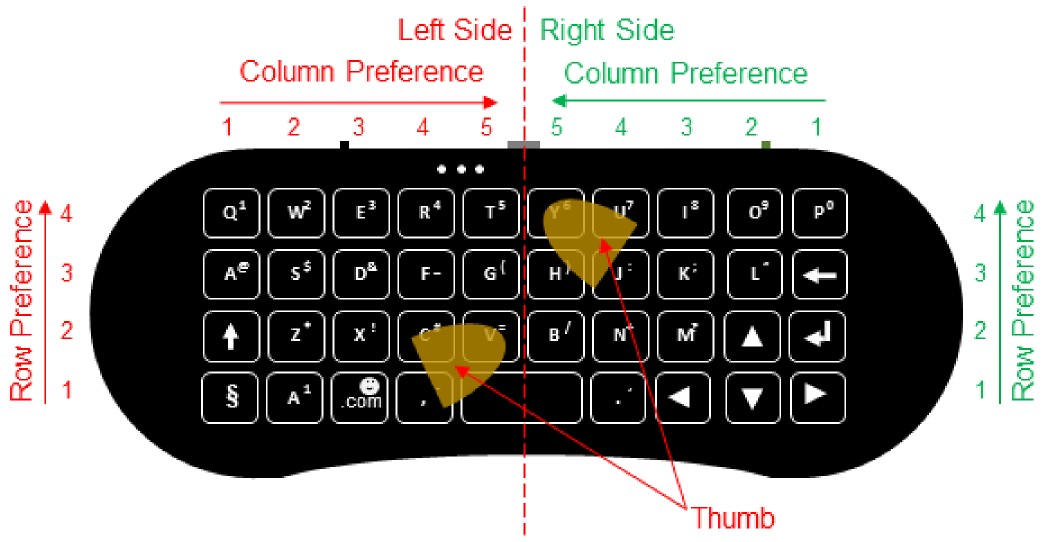


Vertical Lump is in detect
Horizontal Lump is in detect
The key reported as touched

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.

Figure 3-7 Lazy Touch Resolution



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 Re-burst

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.

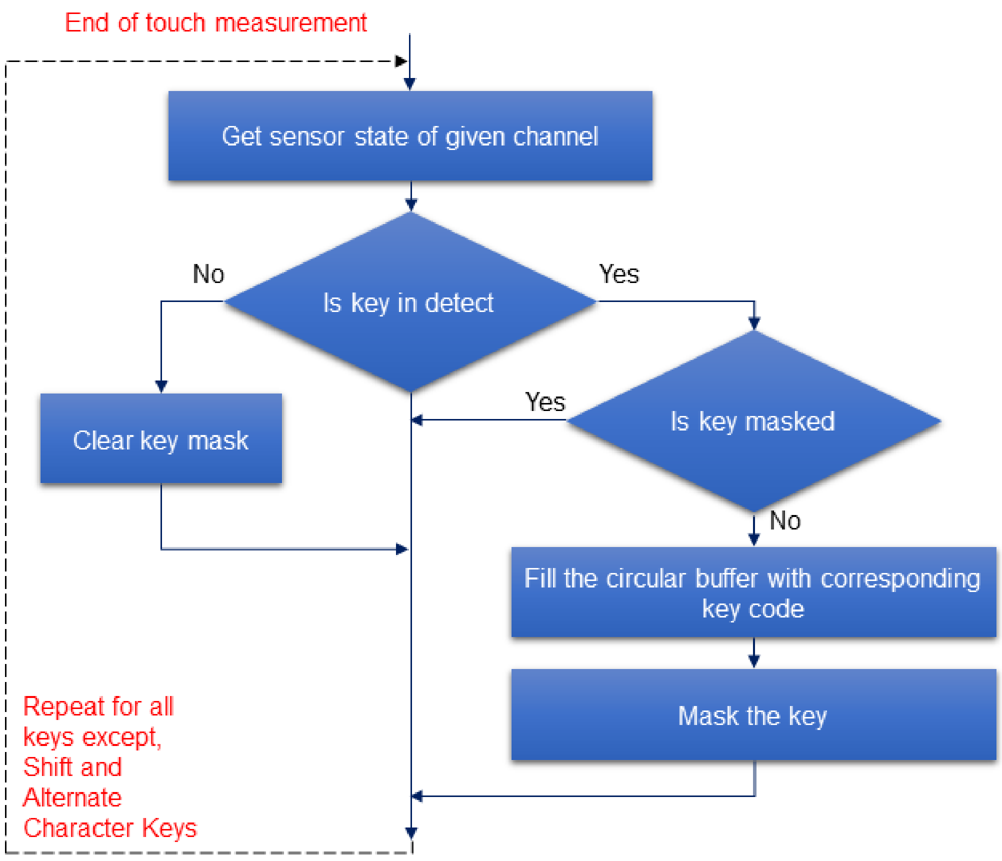
The key preference is illustrated in the above figure. Higher number means higher preference. On the left side, preference increases from left to right for column. On the right side, preference increases from right to left for column. Both left and right side preference is higher for upper rows compared 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

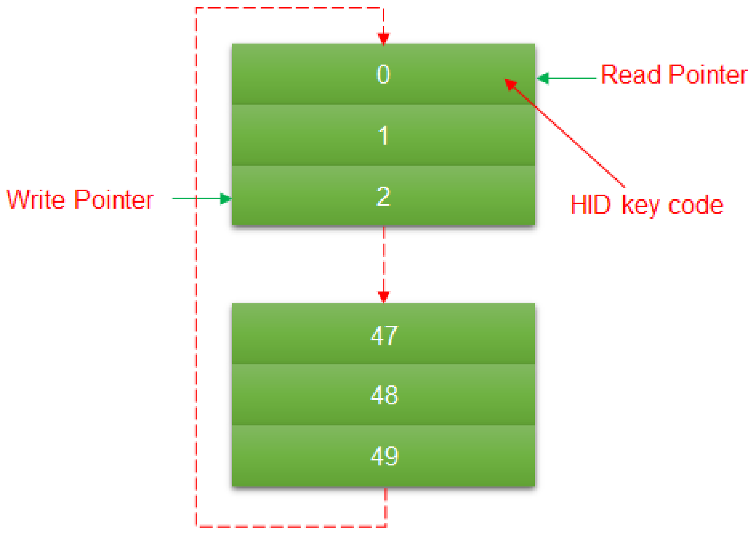

End of touch measurement
Get sensor state of given channel
No
Is key in detect
Yes
Clear key mask
Yes
Is key masked
No
Fill the circular buffer with corresponding key code
Mask the key
Repeat for all keys except, Shift and Alternate Character Keys

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.

The key codes are inserted in the circular buffer only if the keyboard is connected to a USB or BLE host. This ensures that no existing data is flushed out immediately after keyboard is connected. The size of the circular buffer is configured to 50.

Figure 3-9 Fill Circular Buffer



3.5.3. Angular Motion Sensor

Initilization

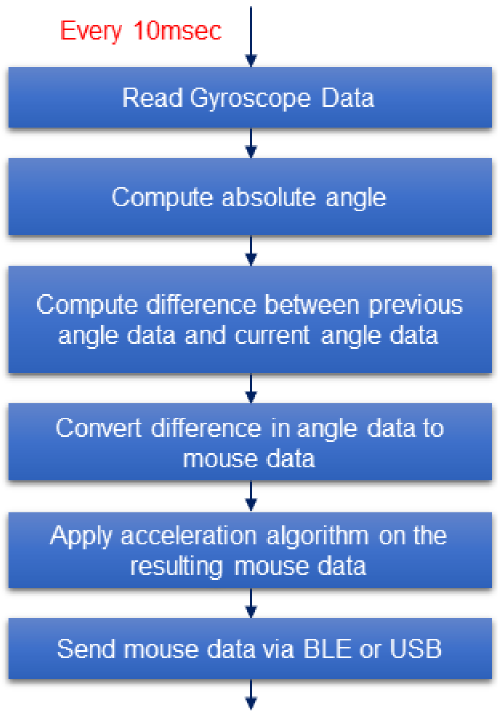
The BMG160 gyroscope supports a maximum range of ±2000 degree/second and has inbuilt low-pass filter with configurable cutoff frequency. The BMG160 is configured as follows.

- **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


Every 10msec
Read Gyroscope Data
Compute absolute angle
Compute difference between previous angle data and current angle data
Convert difference in angle data to mouse data
Apply acceleration algorithm on the resulting mouse data
Send mouse data via BLE or USB

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.

$$Absolute\ Angle = Absolute\ Angle\ + \left(\frac{Gyroscope\ Data}{16.4} * \ Sample\ Interval\right)$$

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.

$$Absolute\ Angle = Absolute\ Angle + \left(\frac{Gyroscope\ Data\ -\ Offset\ Error}{16.4} * Sample\ Interval\right)$$

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.

$$AngularDisplacement = AbsoluteAngle^{N-1} - AbsoluteAngle^{N}$$

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

$$Pixel\ Value = Angular\ Displacement * \ Pixel\ Factor$$

- 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.

$$Pixel\,Value = Pixel\,Value * \frac{abs(Pixel\,Value) + Acceleration\,Coefficient1}{Acceleration\,Coefficient1 + Acceleration\,Coefficient2}$$

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

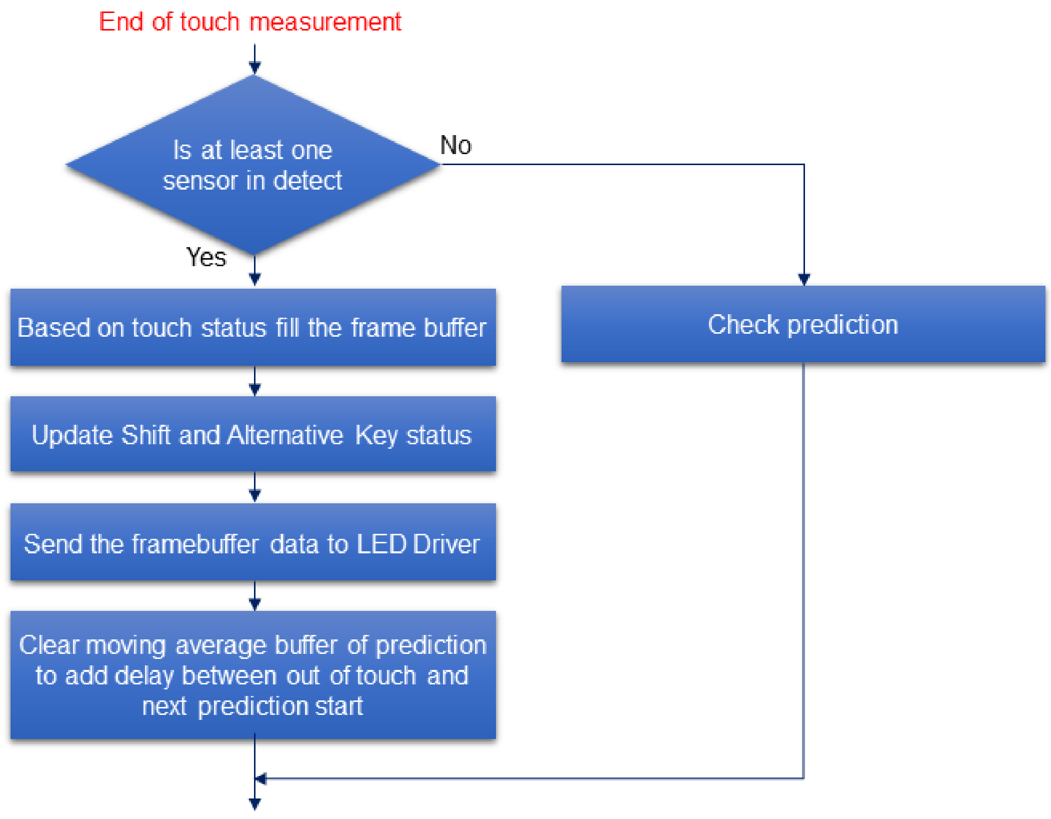
Initialization

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.

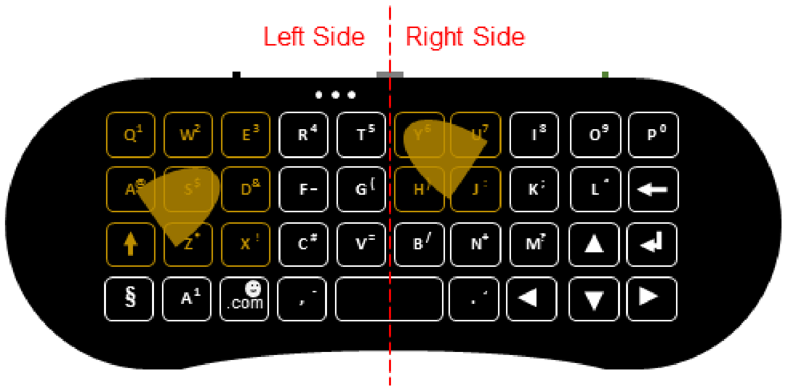
Figure 3-11 LED Indication Flow


End of touch measurement
Is at least one sensor in detect
No
Yes
Based on touch status fill the frame buffer
Check prediction
Update Shift and Alternative Key status
Send the framebuffer data to LED Driver
Clear moving average buffer of prediction to add delay between out of touch and next prediction start

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.

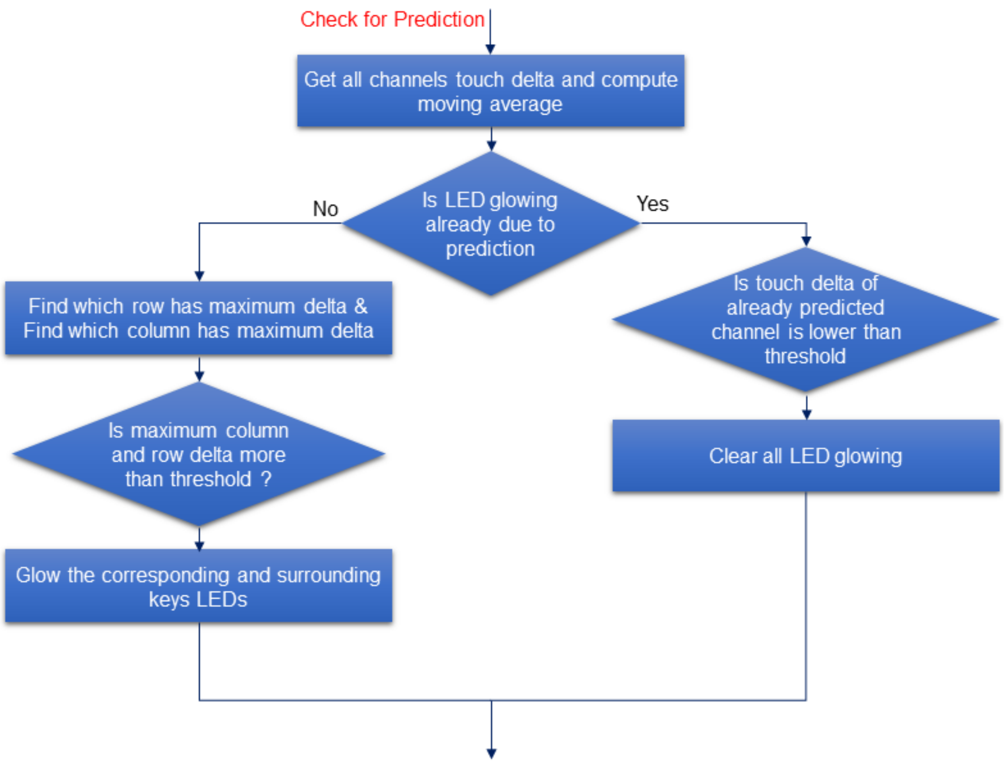
Figure 3-12 LED Indication for Prediction



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.

Figure 3-13 LED Prediction Flow


Check for Prediction
Get all channels touch delta and compute moving average
Is LED glowing already due to prediction
No
Yes
Find which row has maximum delta & Find which column has maximum delta
Is maximum column and row delta more than threshold ?
Glow the corresponding and surrounding keys LEDs
Is touch delta of already predicted channel is lower than threshold
Clear all LED glowing

3.5.5. Bluetooth

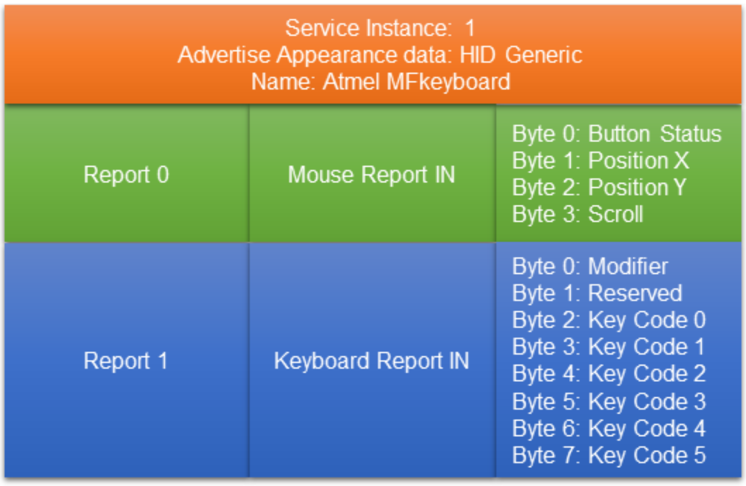
ATBTLC1000 provides Bluetooth Smart Link Controller that includes RF, Link Layer, GAP, GATT, and SMP in a single SoC. It provides the host microcontroller with methods to perform standard Bluetooth Smart GAP, GATT server, and client operations as well as security management with peer devices.

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



Service Instance: 1
Advertise Appearance data: HID Generic
Name: Atmel MFkeyboard

Report 0	Mouse Report IN	Byte 0: Button Status Byte 1: Position X Byte 2: Position Y Byte 3: Scroll
Report 1	Keyboard Report IN	Byte 0: Modifier Byte 1: Reserved Byte 2: Key Code 0 Byte 3: Key Code 1 Byte 4: Key Code 2 Byte 5: Key Code 3 Byte 6: Key Code 4 Byte 7: Key Code 5

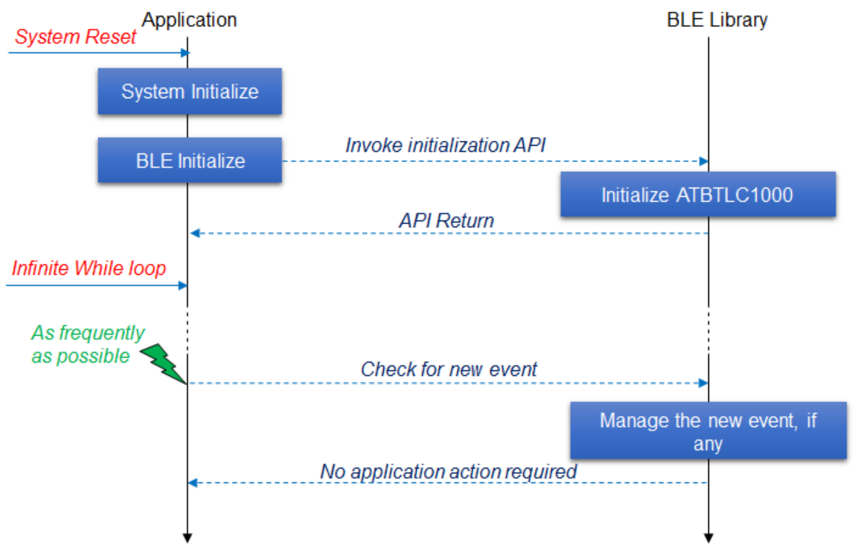
Data Transactions

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.

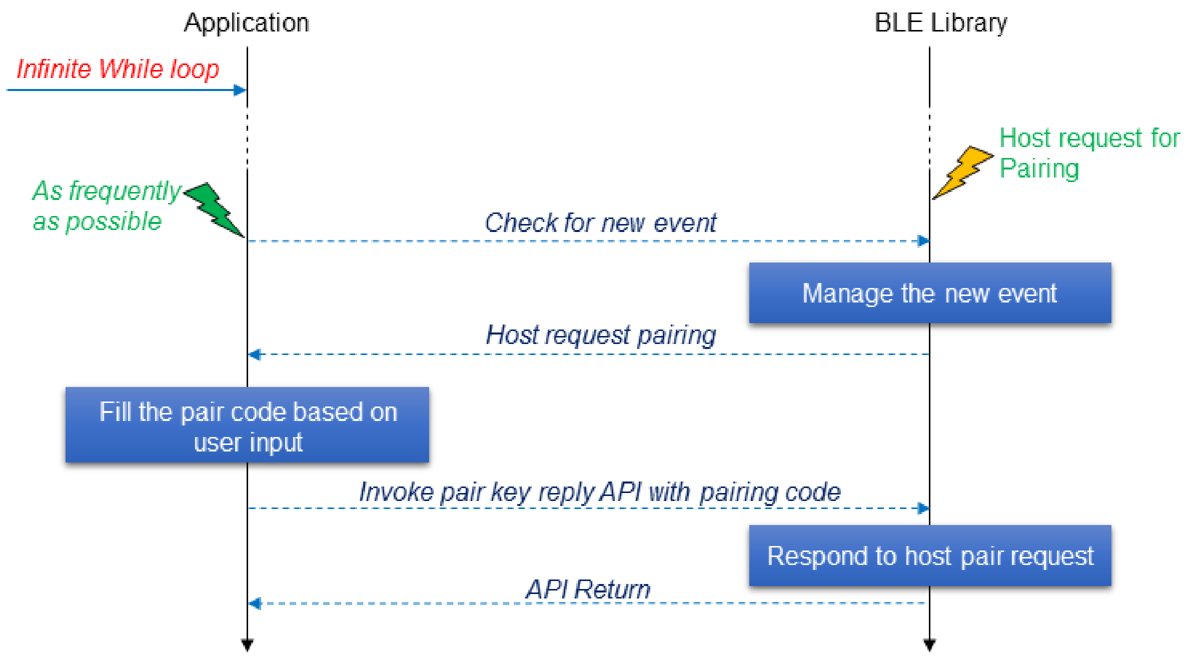
Figure 3-15 BLE Initialize



Pair Key Request

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

Figure 3-16 Pair Key Request

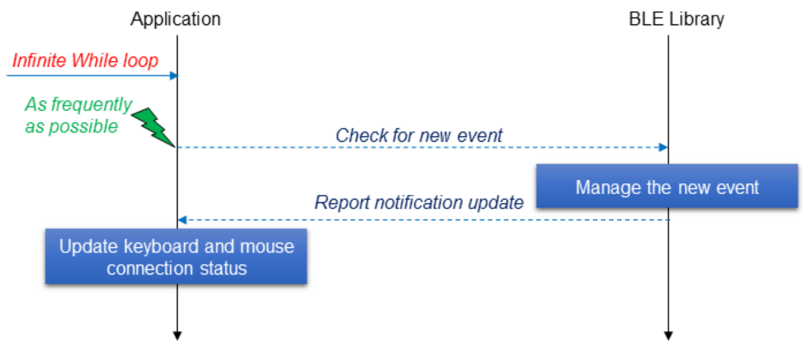

Application
BLE Library
Infinite While loop
Host request for Pairing
As frequently as possible
Check for new event
Manage the new event
Host request pairing
Fill the pair code based on user input
Invoke pair key reply API with pairing code
Respond to host pair request
API Return

The application monitors pairing code entered by the user and sends pairing code using ‘pair key reply’ API. BLE library respond to pairing request from the BLE host.

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.

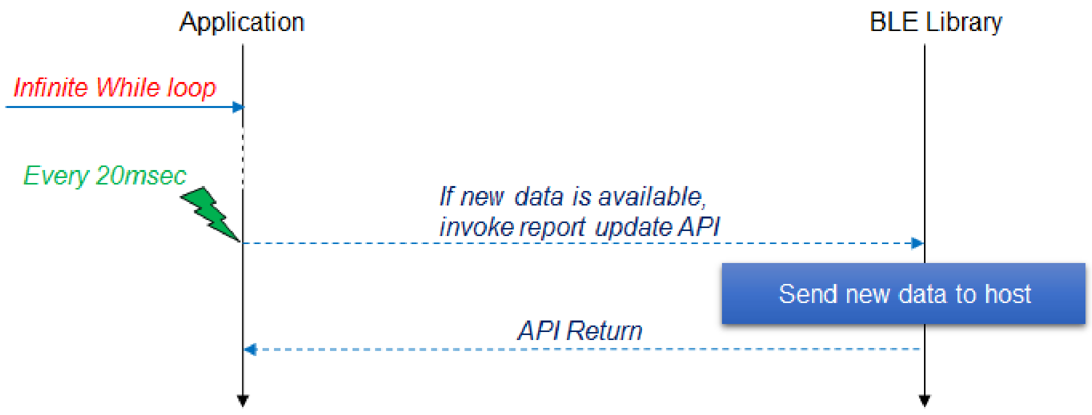
Figure 3-17 Report Notification


Application
BLE Library
Infinite While loop
As frequently as possible
Check for new event
Manage the new event
Report notification update
Update keyboard and mouse connection status

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 ATBTLC1000.

Figure 3-18 Send Data



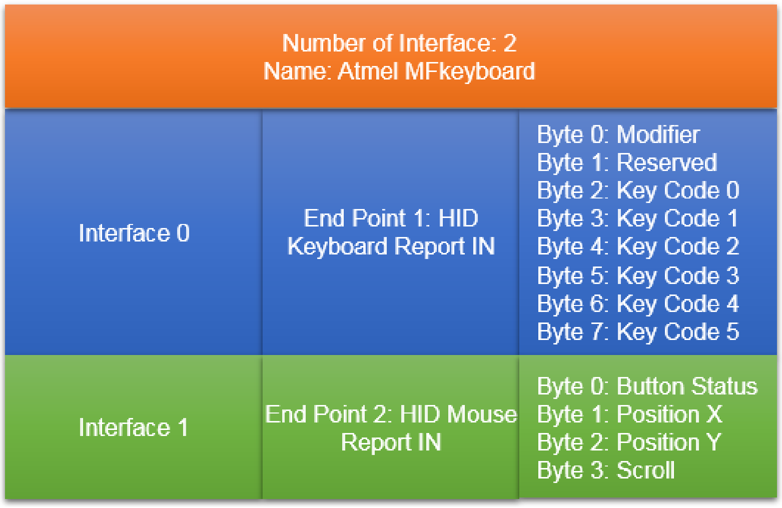
3.5.6. USB

ASF supports USB host and device stack for Atmel MCU, and provides an easy way to build a USB application. Keyboard application uses ASF driver to interact with USB peripheral.

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.

Figure 3-19 USB Configuration



Number of Interface: 2 Name: Atmel MFkeyboard		
Interface 0	End Point 1: HID Keyboard Report IN	Byte 0: Modifier Byte 1: Reserved Byte 2: Key Code 0 Byte 3: Key Code 1 Byte 4: Key Code 2 Byte 5: Key Code 3 Byte 6: Key Code 4 Byte 7: Key Code 5
Interface 1	End Point 2: HID Mouse Report IN	Byte 0: Button Status Byte 1: Position X Byte 2: Position Y Byte 3: Scroll

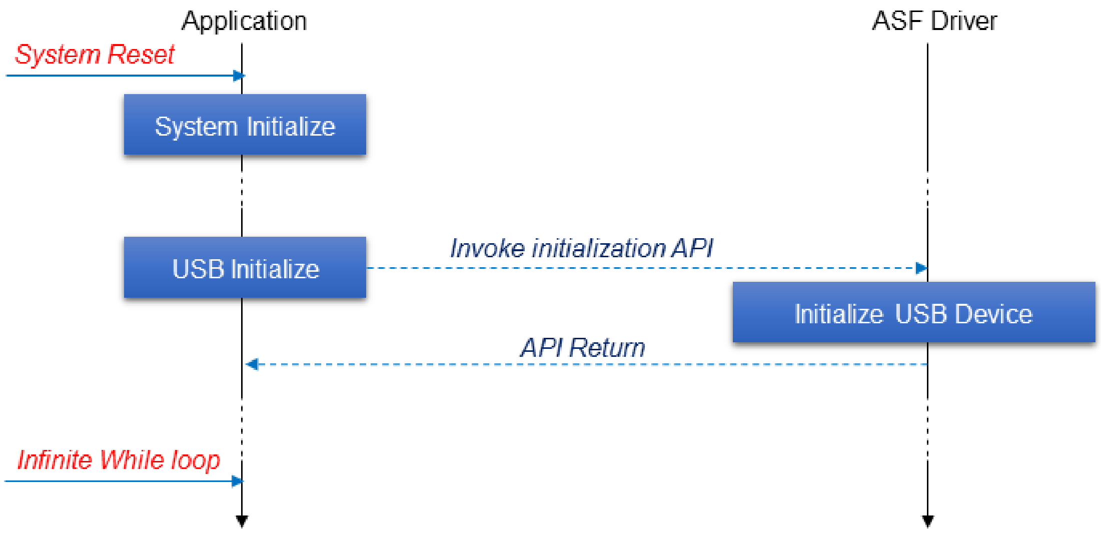
Data Transaction

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.

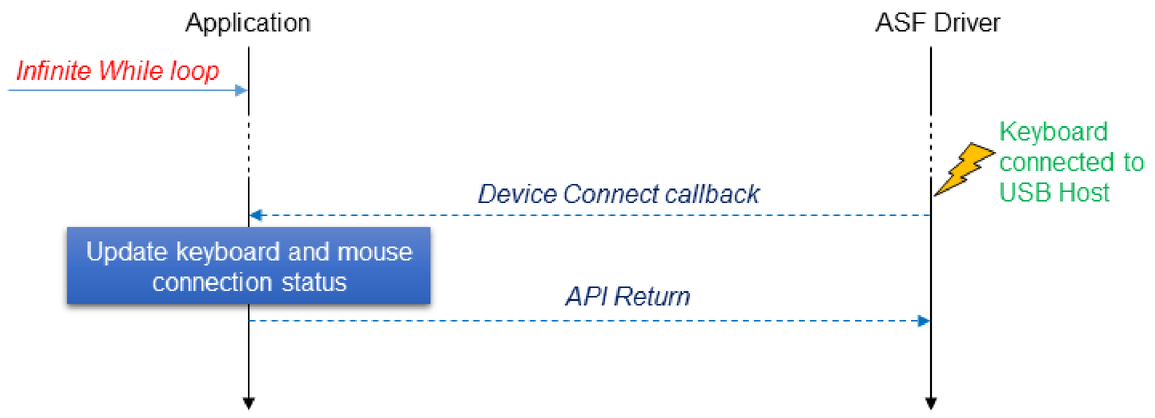
Figure 3-20 USB Initialization



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.

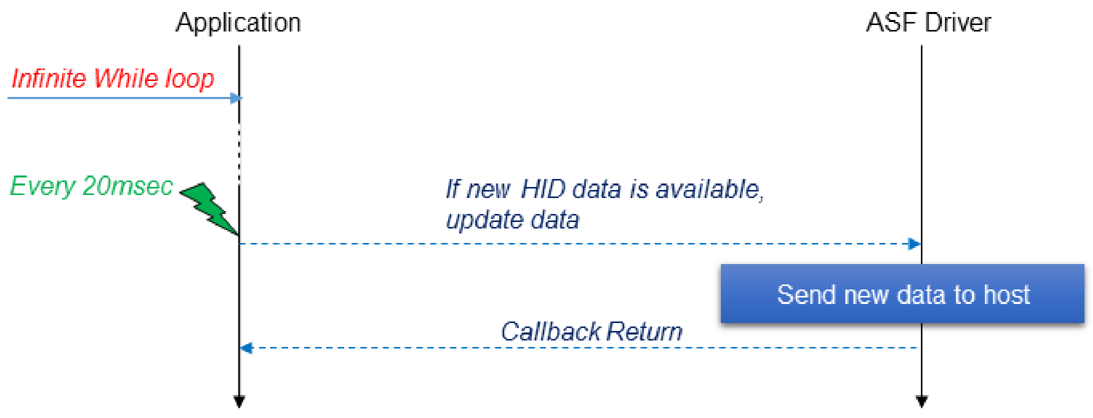
Figure 3-21 USB Host Connection


Application
ASF Driver
Infinite While loop
Keyboard connected to USB Host
Device Connect callback
Update keyboard and mouse connection status
API Return

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.

Figure 3-22 Send Data



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 `0xFFFFFFFF`

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

Module	Active Mode		Low Power Mode [µA]
	Condition	Current Consumption [mA]	
SAM D21	Without USB	3.2	8
	With USB	7.2	
ATBTLC1000	During Advertisement	0.28	32
	Post Pairing	1.4	
LED Driver	No LED Glowing	0.0006	0.36
	While LED Glowing	3 + 3mA/LED	
BMG160		2.853	85

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

Firmware Features	Bluetooth [mA]	USB [mA]
All features enabled	14.6	17.8
Without predictive backlighting	10.5	14.2
Without LEDs	6.7	10.2
Low power mode (after inactive period of 2 minutes)	0.14	0.14

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

[1] QTouch Library SAM Peripheral Touch Controller User Guide

http://www.atmel.com/Images/Atmel-42195-QTouch-Library-Peripheral-Touch-Controller_User-Guide.pdf

[2] SAM D21 USB

http://www.atmel.com/Images/Atmel-42261-SAM-D21-USB_Application-Note_AT06475.pdf

[3] SAM-BA Boot loader for SAM D21

http://www.atmel.com/Images/Atmel-42366-SAM-BA-Bootloader-for-SAM-D21_ApplicationNote_AT07175.pdf

[4] HID Over GATT Profile

https://developer.bluetooth.org/TechnologyOverview/Pages/HOGP.aspx

[5] ATBTLC1000 BluSDK Release Package

http://www.atmel.com/System/BaseForm.aspx?target=tcm:26-72436

[6] BSW Sensor Design Guide

http://www.atmel.com/Images/doc10752.pdf

7. Revision History

Doc Rev.	Date	Comments
42579B	11/2015	Non-technical enhancements to the images.
42579A	11/2015	Initial document release.

Enabling Unlimited Possibilities®

Atmel Corporation 1600 Technology Drive, San Jose, CA 95110 USA **T:** (+1)(408) 441.0311 **F:** (+1)(408) 436.4200 | **www.atmel.com**