Electrical System

SteamVR™ Tracking

Introduction

The SteamVR™ Tracking electrical system encapsulates all the functionality required to enable pose tracking, and the basic peripheral requirements of controllers and HMDs. The fundamental requirement of the electrical system is the ability to detect and timestamp up to 32 optical sensors, which receive reference signals emitted from base stations. Once the timestamps are generated, may be transferred to the SteamVR™ computer using either a USB or 2.4 GHz wireless link. In addition to the sensors, a stream of IMU data is also recorded generated and transmitted to the computer. Outside of this core functionality, some basic features of controllers and HMDs have also been implemented.

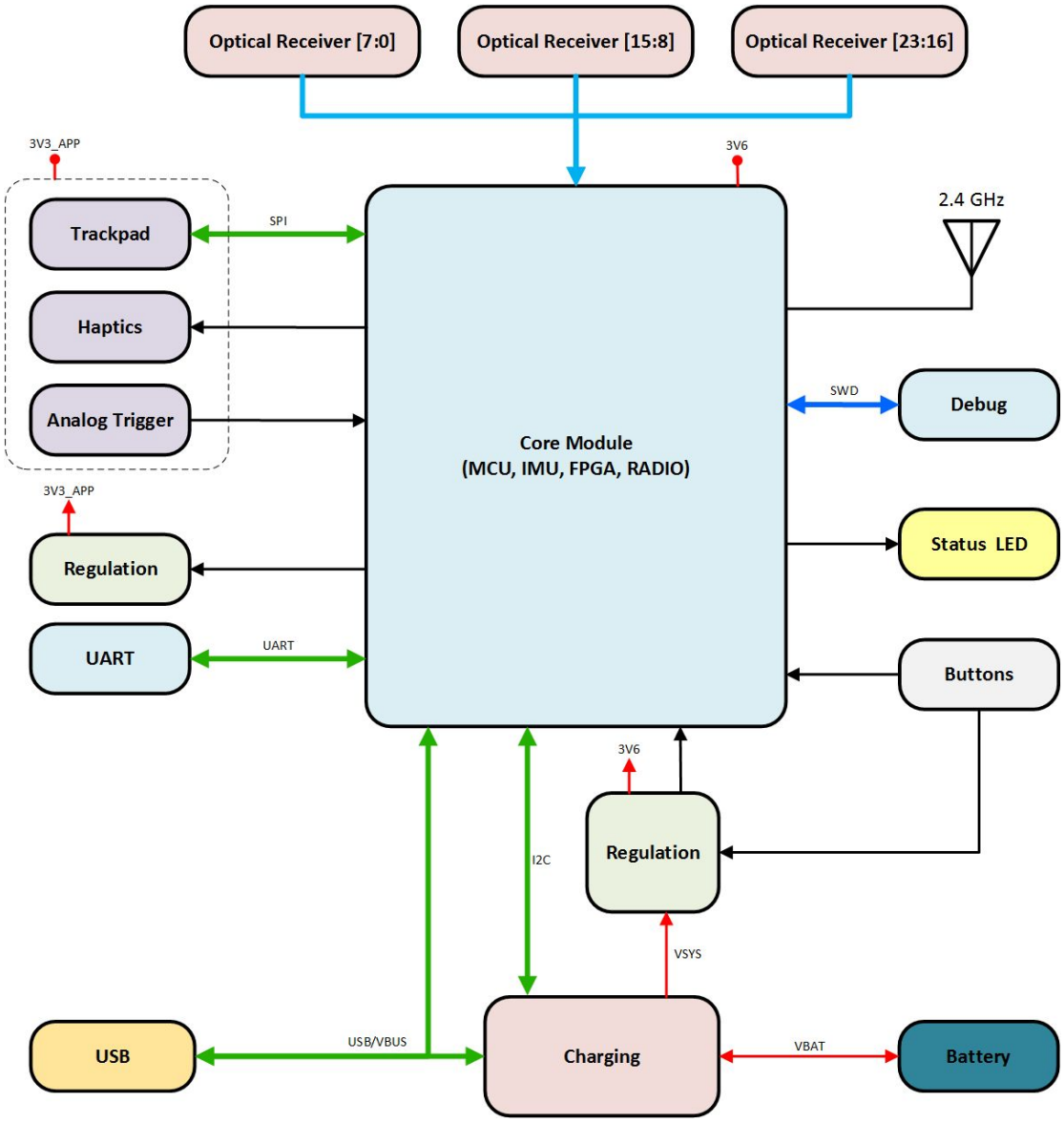
Controllers and HMDs have different peripheral requirements, some of which are integrated into the SteamVR™ Tracking electrical system. Controllers require battery power and user controls, such as buttons, trackpad, LEDs and haptic feedback. The electrical system implements battery charging and fuel gauge circuitry to manage a Lithium ion battery pack. It also implements basic controls, including an RGB status LED, push buttons, a Hall effect sensor for magnetic controls, capacitive trackpad, and haptic feedback. The flexible input and output peripherals on the system's microcontroller also enables replacing these controls with a variety of other controls and peripherals. HMDs require a serial flash memory to store display calibration data. Although HMDs do not have as many user accessible inputs and outputs, those peripherals may be replaced by display bridges and other video peripherals.

This document outlines the electrical architecture of the SteamVR™ Tracking reference design, and highlights recommendations for design and usage. The part numbers and signal names in this document correspond to the reference design schematics in the SteamVR™ Tracking HDK.

Block Diagrams

Controller

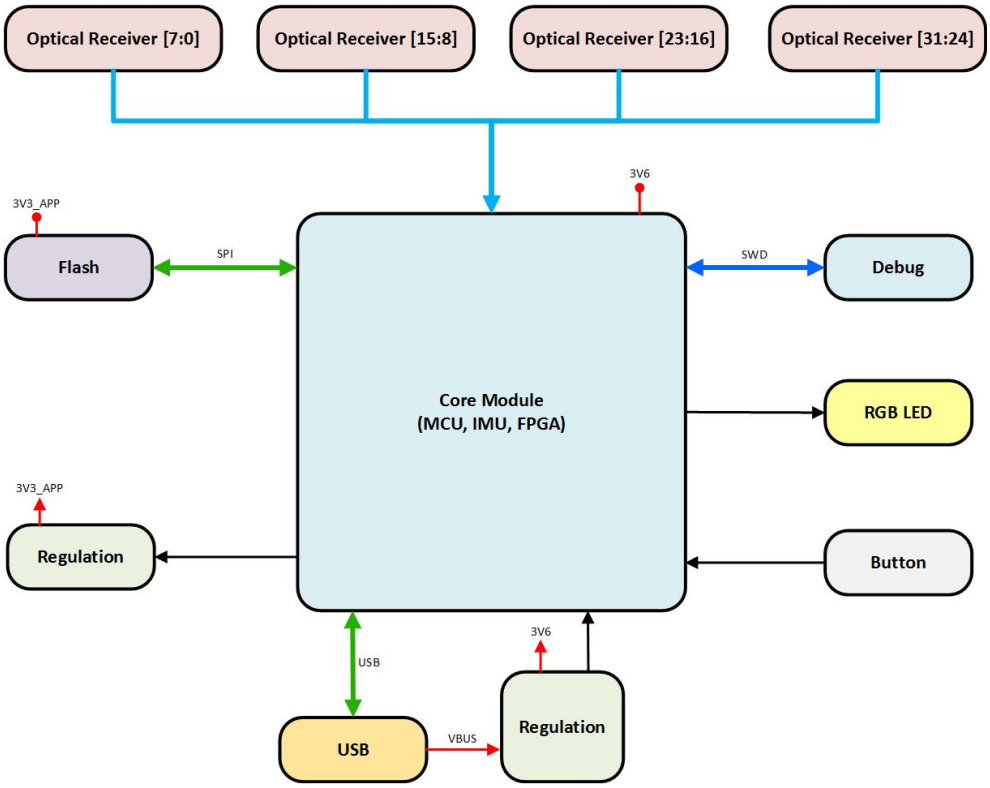
A high level block diagram for a controller is shown below.



Controllers typically contain fewer than 32 sensors due to their limited size. Also, controllers usually support wireless operation. Operating wirelessly requires a radio in the core and an antenna for wireless communications. Wireless operation also means battery power, which adds the requirement for a battery, charging circuitry, and a fuel gauge. The battery charges from USB. A controller also needs user input and output peripherals. The input controls supported by the SteamVR™ Tracking reference design include five buttons, a capacitive trackpad, and a Hall-effect sensor for an analog trigger. Use feedback is delivered using an RGB LED for status and haptic feedback. Communication to the SteamVR™ computer is accomplished over USB or 2.4 GHz wireless link. Finally, there are several communication interfaces to support system development and manufacturing, including single wire debug ports for the radio and microcontroller and a UART connection for debug and manufacturing test. This block diagram does not show every possible feature set that may be designed into a controller, but does show the fundamental set of controls that application developers currently use when creating SteamVR™ applications.

HMD

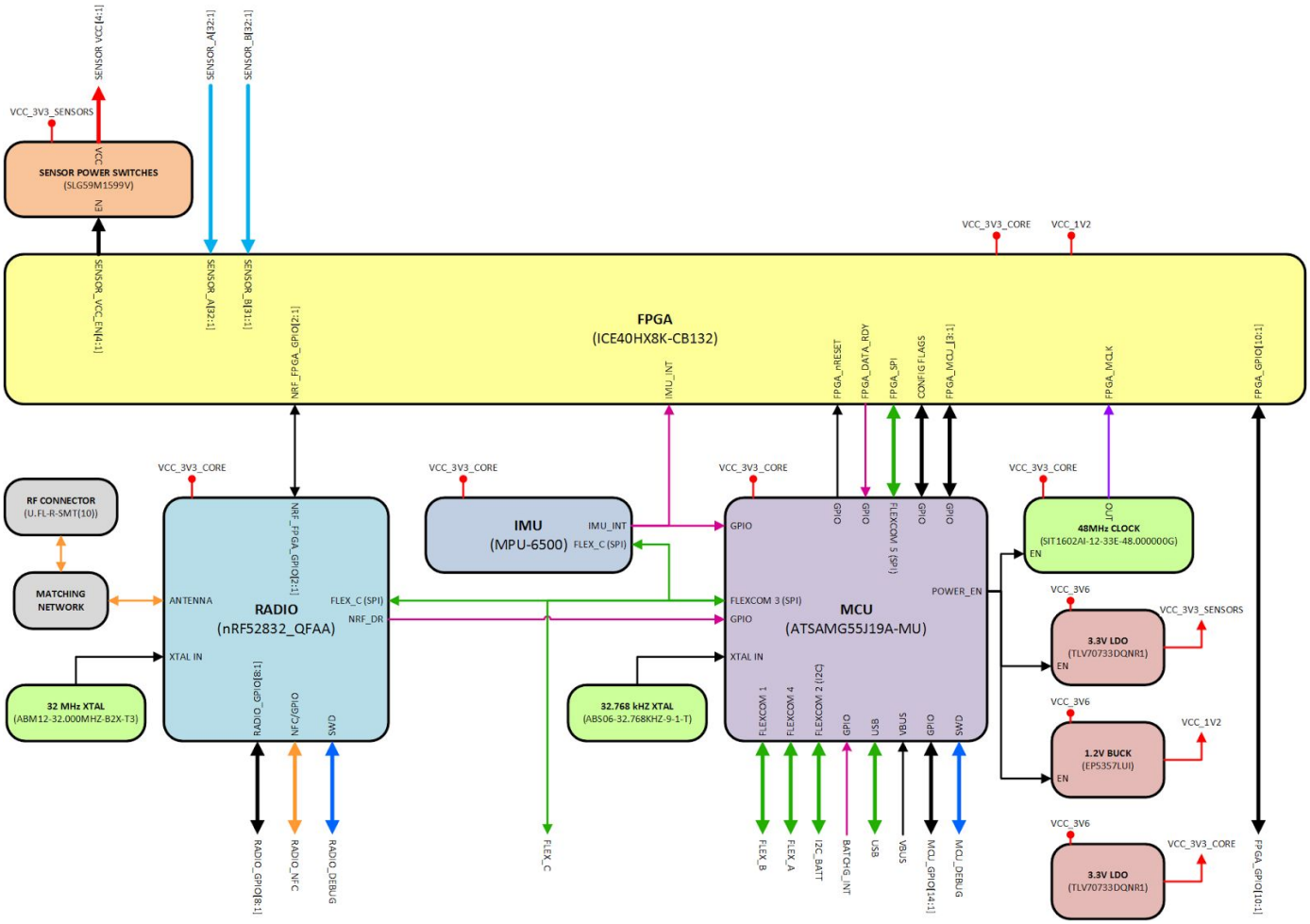
A high level block diagram for SteamVR™ Tracking in an HMD is shown below.



HMDs typically contain close to 32 sensors due to their larger size and the importance of very reliable tracking. Unlike controllers, HMDs do not require wireless operation and eliminate the radio, battery, and their associated circuitry as a result. Typically, the user controls on an HMD are also reduced to a status LED and single button. Communication to the SteamVR™ computer is accomplished over USB. As with controllers, there are several communication interfaces to support system development and manufacturing, including a single wire debug port for the microcontroller and a UART connection for debug and manufacturing test. This block diagram shows the requirements for the tracking system, but omits the peripherals required for video streaming and display. The specific requirements of the HMD dictate these peripherals, which may be controlled using the MCU in the tracking core, or another MCU added to the system.

Core Functions

A block diagram of the core functionality is shown below.



The FPGA timestamps signals arriving from the optical sensors and measurements taken by the IMU using capture registers driven by the 48 MHz clock. Those timestamps are read by the MCU and transmitted to SteamVR™ via either the radio or the USB interface. Note that the radio is only required in wireless devices, like controllers.

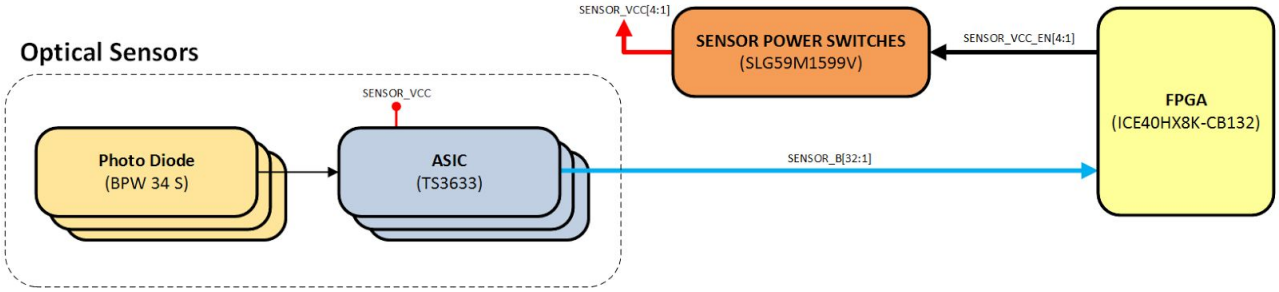
The core includes regulators to generate supply rails for the MCU, radio, FPGA, IMU, and optical sensors. The MCU has the ability to shut down power to the optical sensors and the FPGA core. When the FPGA core is powered down, the 48 MHz oscillator is also disabled. Through the FPGA, the MCU is also able to control four PFET switches to sequence power to the optical sensors.

A variety of busses and GPIO are available from the MCU to connect peripheral devices for different applications. The FPGA also offers ten GPIO signals for application specific features.

Core Functions

Optical Sensors

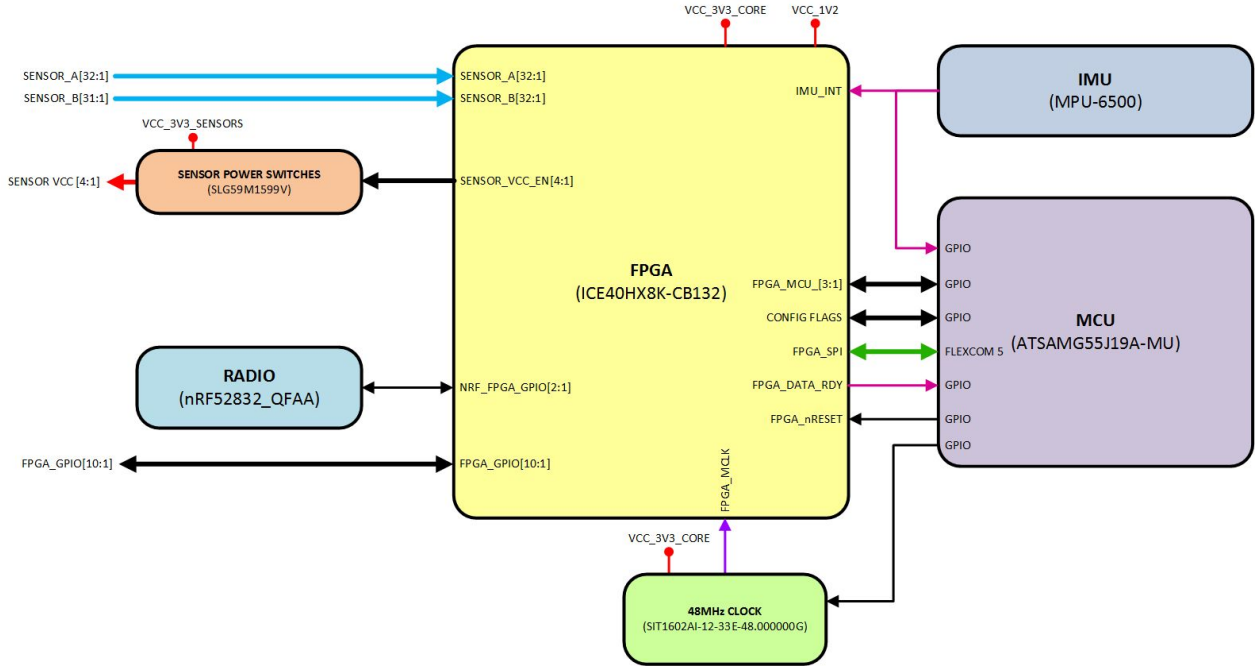
The optical sensors are comprised of a photodiode connected to a custom ASIC that implements a transimpedance amplifier (TIA) and envelope detector. The photodiode converts incident IR light into a current. That current is transformed into a voltage using the TIA of the ASIC. The modulation frequency is filtered out of the reference signal in the ASIC and the resulting envelope is transmitted to the FPGA for timestamping.



Up to 32 optical sensors are supported in the system. They receive power from one of four 3.3V power rails, SENSOR_VCC[4:1]. The optical sensors transmit the envelope of incident IR light to the FPGA via SENSOR_B[32:1].

FPGA

The FPGA's primary function is timestamping photodetector and IMU signals on the 48 MHz system clock.



Sensor outputs reach the FPGA as signals SENSOR_B[32:1]. Signals SENSOR_A[32:1] are reserved for future use. IMU interrupts are timestamped using the signal IMU_INT.

The 32 bit capture registers in the FPGA are driven by the 48 MHz clock delivered via FPGA_MCLK. The FPGA also controls four PFET switches to sequence power to the optical receivers using SENSOR_VCC_EN[4:1].

FPGA_GPIO[10:1] provide ten GPIO signals available from the FPGA to connect auxiliary signals for timestamping, expand GPIO, or implement other functions.

The MCU communicates with the FPGA over SPI, both to receive timestamp information and configure the FPGA after system reset.

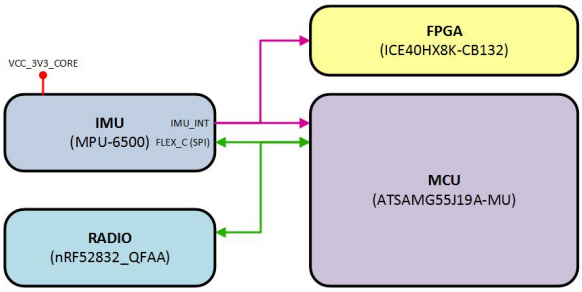
Two GPIO signals between the FPGA and the radio, NRF_FPGA_GPIO[2:1] are reserved for future use.

Design Notes

The power supplies for the FPGA are the 3.3V supply for the I/O ring, and 1.2V for the core. Although the 1.2V rail is controlled by the MCU, the 3.3V rail remains powered to avoid connecting powered I/O pads on the MCU and radio to unpowered I/O on the FPGA. When using FPGA GPIO signals to drive other peripherals that may be connected to other power sources, remember that the I/O ring for the FPGA is powered whenever power is applied to the system.

IMU

The IMU measures the translational and rotational acceleration of a tracked object at 1000 Hz in HMDs and 250 Hz in controllers.



The measurements are read by the MCU using the SPI bus, FLEX_C. The IMU also shares FLEX_C with the radio.

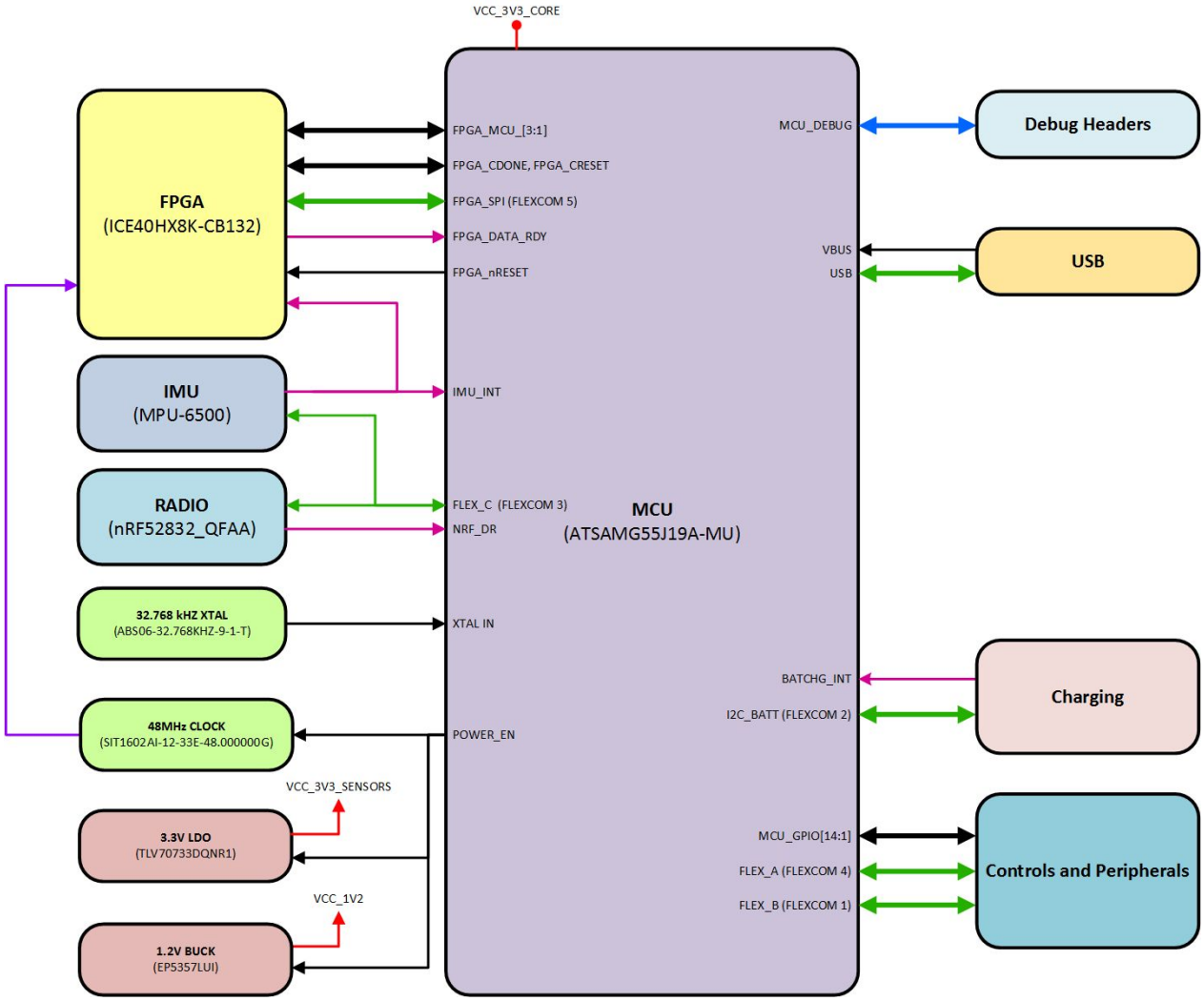
The IMU_INT signal is shared between the FPGA (for timestamping) and the MCU (as a data ready interrupt).

Design Notes

When laying out the MPU-6500 take care to place a copper keep-out under the package. Although the QFN package has a thermal pad, it is not recommended to connect that pad to the PCB. Mechanical stresses associated with this pad may affect measurement.

MCU

The MCU is the central processor for the electrical system. It controls peripheral power and the boot process, stores the FPGA image and configures the FPGA after reset, routes timestamps from the FPGA to the radio or USB connection, routes IMU measurements over the radio or USB, manages the battery peripherals, and drives the user input and output peripherals.



The MCU uses the same FPGA_SPI interface for configuration and communications. During configuration, the MCU controls CRESET and monitors CDONE while sending the FPGA bitstream to the device using FPGA_SPI. After configuration, the MCU uses FPGA_SPI to control functions within the FPGA and read timestamp information generated by the FPGA.

The MCU also reads acceleration data from the IMU using the SPI bus FLEX_C in conjunction with IMU_INT. The radio shares the FLEX_C SPI bus with the IMU and interrupts the MCU to transfer data using NRF_DR.

When the system needs to go into a lower power state, the MCU can power down sensors and the FPGA core using the POWER_EN signal.

In battery powered applications, the MCU manages the battery through the charging and fuel gauge peripherals connected to I2C_BATT. The peripherals may interrupt the MCU using BATCHG_INT.

The MCU includes a USB 2.0 Full Speed peripheral, which connects to the host computer running SteamVR™.

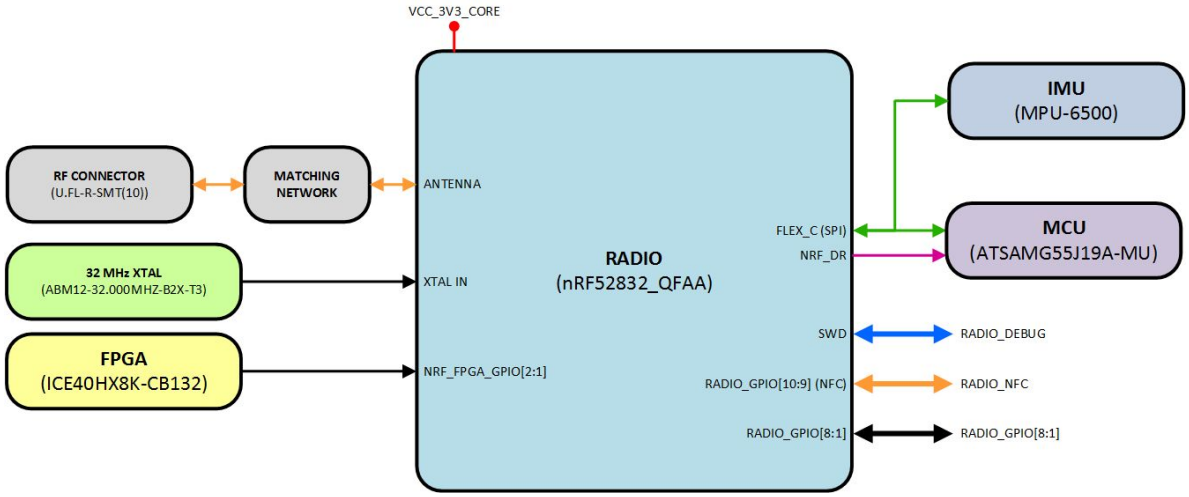
Controls and peripherals for HMDs and controllers are connected to MCU_GPIO[4:1], FLEX_A, and FLEX_B. These pins may be reconfigured in a variety of ways to implement SPI, I2C, UART, and GPIO interfaces.

Design Notes

When battery charging is not required, the signals used to manage the battery circuitry may also be reclaimed for other peripherals.

Radio

The radio provides a 2.4 GHz, 1 Mb, wireless link between the tracked object and SteamVR™ computer. The radio includes custom firmware that communicates using the Nordic Enhanced ShockBurst protocol to a USB connected dongle running complementary firmware. The wireless dongle may be plugged into a USB port or hub on the SteamVR™ computer, or integrated into an HMD.



The radio communicates to the MCU over and SPI bus, FLEX_C. It signals the MCU when data is ready using the interrupt line NRF_DR.

The RF antenna signal passes through a passive matching network before connecting to the 2.4 GHz antenna. In the reference schematic, the antenna connection is left open as a U.FL connector to enable different antenna designs and placements.

Two GPIO connections between the radio and FPGA, NRF_FPGA_GPIO[2:1] are reserved for future use.

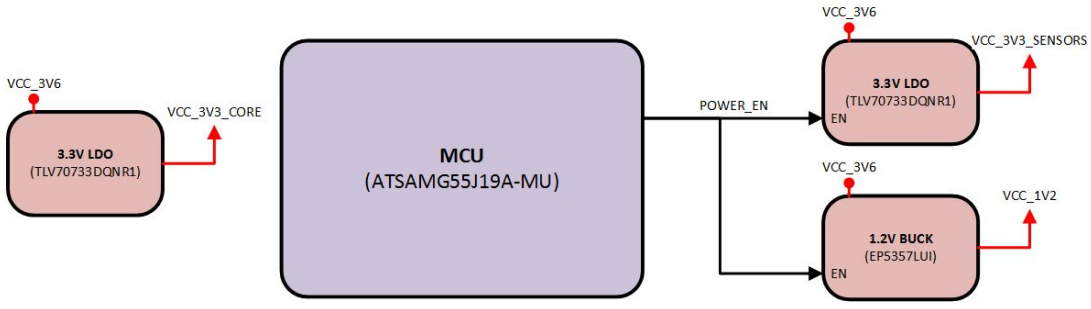
Ten GPIO connections RADIO_GPIO[10:1] are also reserved for future use. NFC is supported by the Nordic NRF52, but NFC is not used by SteamVR™ Tracking.

Design Notes

Take care to follow the layout recommendations of the Nordic radio, as specified in the datasheet.

Core Regulation

The regulators in the core provide the necessary supply voltages for the core functions.



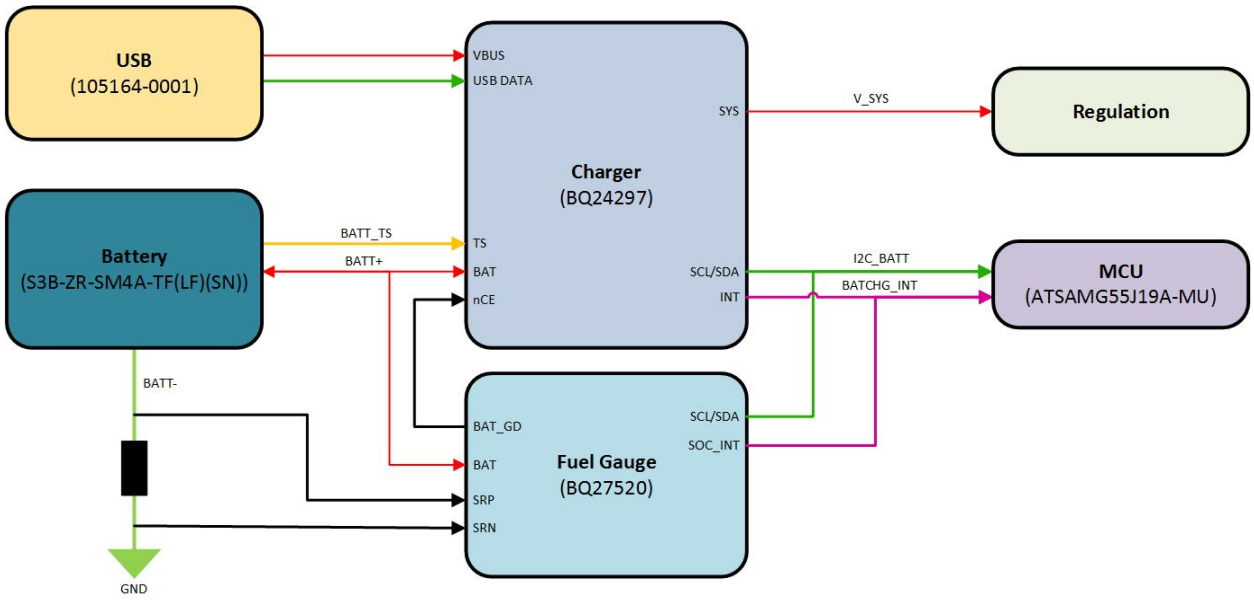
The core regulators are all powered from the 3.6V system rail. One 3.3V LDO creates the VCC_3V3_CORE rail, which powers the MCU, IMU, radio, and FPGA I/O. A buck regulator produces the VCC_1V2 rail that supplies power to the FPGA core. Another 3.3V LDO creates an analog supply voltage for the optical sensors,

which isolates the sensitive analog circuitry of the sensors from digital noise and allows the MCU to shut down power to the sensors if the system goes into a low power state. The core voltage for the FPGA is also disabled when the MCU powers down the core supplies.

Controller Functions

Charging

The charging peripherals manage the Lithium ion battery that powers the controller.



The charger powers the system either from the battery or VBUS. The BQ24297 has the ability to power the system while simultaneously charging the battery. The system voltage V_SYS, produced by the charger, is regulated. However, the charging IC maintains a voltage on V_SYS sufficient to charge the battery. Therefore the voltage on V_SYS may be between 3.5V and 4.35V DC depending on the state of the battery. For that reason, downstream regulation is required.

The charger also has the ability to monitor the USB data lines to detect the type of host port connected to the system. Depending on the type of port, the charger automatically selects the appropriate input current limit to comply with the USB specification.

The charger monitors the 10k thermistor built into the battery pack via the TS input. If the battery temperature falls outside of acceptable limits, the charger stops charging the battery.

The fuel gauge IC also monitors the battery. By measuring the battery voltage via BATT+ and battery current via a low-side current sense resistor connected to SRP and SRN, the fuel gauge calculates the energy stored in the battery. If the fuel gauge detects any problems in the battery, it has the ability to disable charging by raising the charge enable pin on the charger.

Both the charger and the fuel gauge are connected to the MCU by an I2C bus and interrupt signal. When either the charger or the fuel gauge request an interrupt, the MCU reads both to determine the source of the interrupt and any register values that have changed.

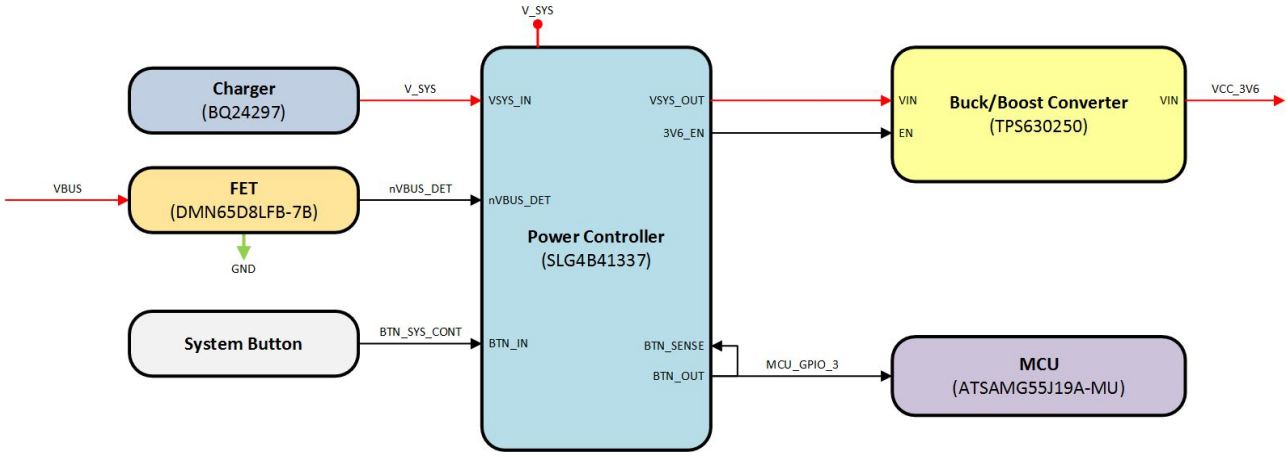
The MCU manages the input current limit of the charger over the I2C bus (I2C_BATT). Based on the configuration granted by the USB host, the input current limit is set to charge the battery as fast as possible within the USB specification.

The MCU also uses the I2C bus (I2C_BATT) to read the battery level from the fuel gauge.

Regulation

3.6V System Power

The regulation section provides control and regulation of the VCC_3V6 supply that powers the electrical system. It takes power from the charger and sends it to a buck boost regulator with a 3.6V output. The 3.6V output is regulated down by LDO and buck regulators throughout the system to power other functions. A Silego power monitor IC with integrated load switch and programmable logic watches the system button and VBUS to determine when to source power to the 3.6V regulator and power on the system.



The power controller is a programmable logic device with an integrated PFET load switch. The power controller is inserted in series with V_SYS as a power switch capable of isolating the battery from downstream devices. The power controller is able to run internal clocks only when external signals are asserted. This limits the leakage current of the battery to the leakage of the charger, fuels gauge, and power controller when the load switch is open.

The logic of the power controller is configured to enable power when VBUS is detected, or when the system button is depressed for 0.5 seconds. Both inputs are pulled up to V_SYS. The nVBUS_DET signal is pulled up through a strong pull-up that ensures a floating V_SYS rail is discharged below 0.9V through the pull-up when VBUS is applied. This guarantees that the power controller enters a power on reset state if no battery or USB power was present before VBUS is applied.

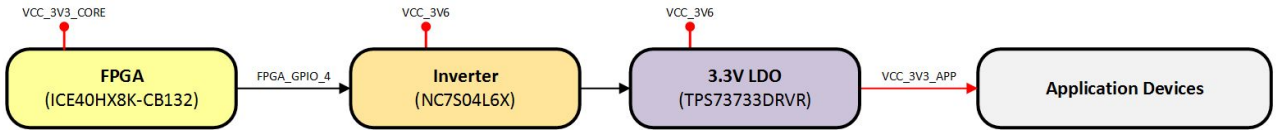
The power controller switches power off if VBUS is not present and the system button is depressed for 4 seconds. The MCU may also power off the system by driving MCU_GPIO_3 low for 0.5 seconds, when the system button is not pressed.

The input from the controller's system button, BTN_SYS_CONT, is passed through to the MCU via the BTN_OUT signal. BTN_OUT is an open drain output, pulled up using an internal pullup on the MCU. BTN_SENSE detects the voltage on the wire connected to BTN_OUT. If the system button is not depressed and BTN_OUT is not asserted, then the voltage to the MCU should be pulled high. If the MCU chooses, it may drive the signal low. The power controller detects that condition via BTN_SENSE and disconnects power if the condition persists for 0.5 seconds.

Whenever the internal load switch is closed, applying power to the input of the 3.6V regulator, the 3V6_EN signal is delayed by 15 ms before being asserted. The 15 ms delay ensures that the V_SYS rail is completely powered and the charger output capacitors and buck/boost input capacitor are fully charged before the 3.6V regulator attempts to draw current to power the rest of the system.

3.3V Application Power

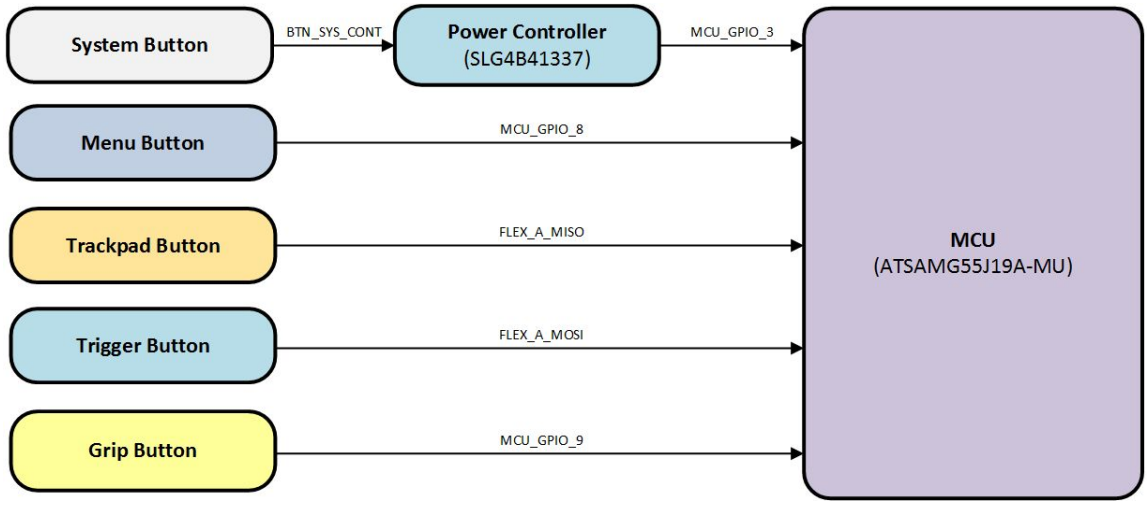
The application power rail isolates user controls from the core of the system and provides a means of powering down application specific peripherals that may not have low power states.



The FPGA_GPIO_4 signal connects to the enable input on the 3.3V LDO through an inverter. The inverter guarantees that the LDO is disabled by default, until the FPGA is configured and the MCU sets the FPGA_GPIO_4 output low.

Buttons

The default configuration of the electrical system supports five momentary, tact switches. These switches are assigned to SteamVR™ system, menu, trackpad, trigger, and grip button inputs.

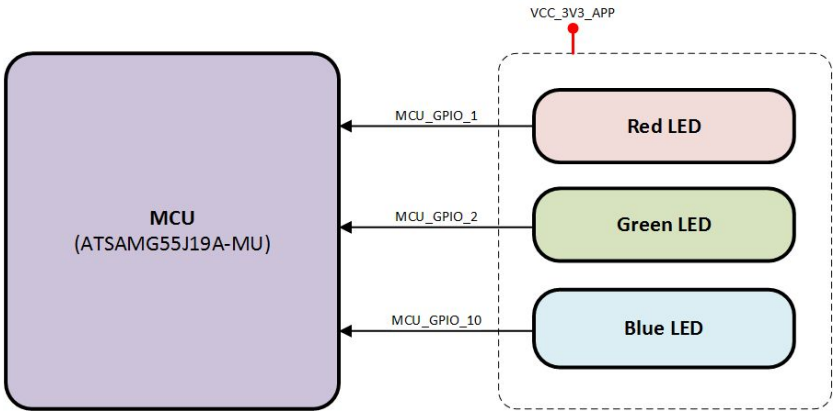


The system button passes through the power controller. The power controller uses the system button as a power switch. It powers up the system when depressed for 0.5 seconds and powers down the system when depressed for 4 seconds. Applying USB power always powers up the system. When USB power is removed, the system will return to the last state the system button defined.

All button inputs to the MCU are pulled up and debounced internally.

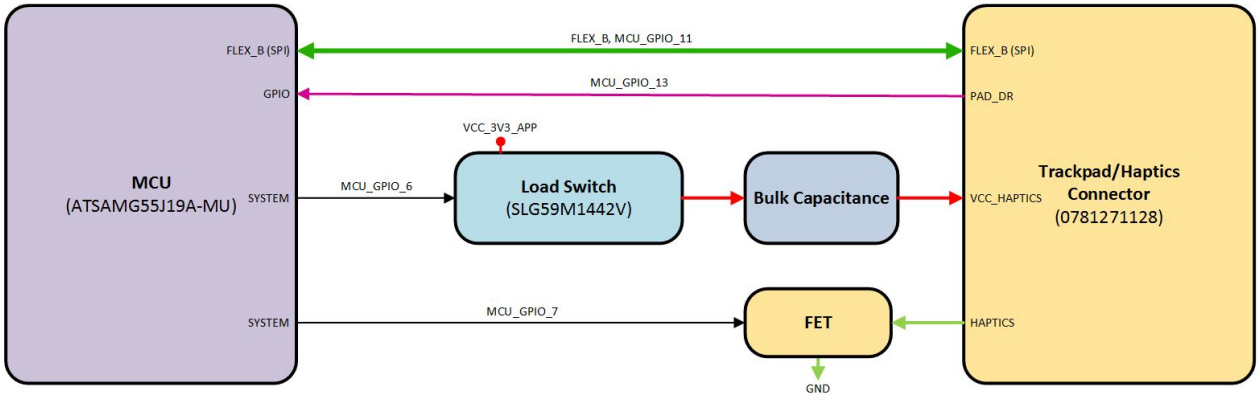
LEDs

An RGB LED, driven from the application power rail, is connected to the MCU for status indication.



Trackpad and Haptics

The trackpad and haptics are assembled as a module and connected to the MCU via SPI and GPIO. The trackpad peripheral is a capacitive touch device that locates a finger over the circular surface of the trackpad. The haptic feedback is an inductive electromechanical device.

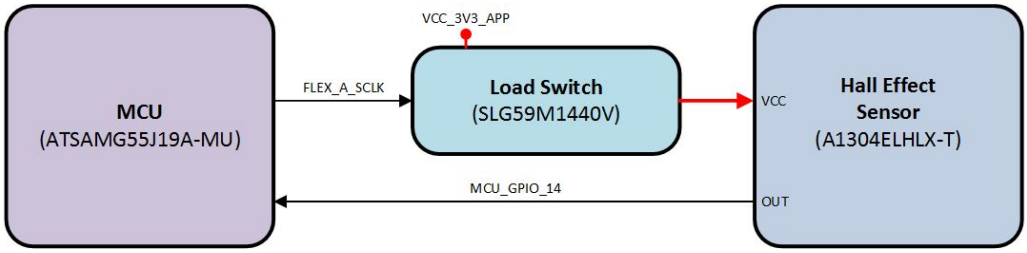


The trackpad module connects to the MCU through a low insertion force FPC socket. The trackpad communicates to the MCU over the FLEX_B SPI bus, using MCU_GPIO_11 as a chip select and MCU_GPIO_13 as a data ready interrupt to the MCU.

The haptics transducer is supplied by VCC_3V3_APP through a load switch that charges bulk capacitors located close to the haptics module. The bulk capacitors store charge to drive the haptics quickly when the low-side FET is activated by MCU_GPIO_7. To prevent overloading the system's power source as the bulk capacitors charge, the power rail for the haptics module is isolated by a PFET load switch with internal slew rate limiting.

Analog Trigger

If a controller's trigger contains a magnet, its position may be monitored by the MCU using a Hall-effect sensor.

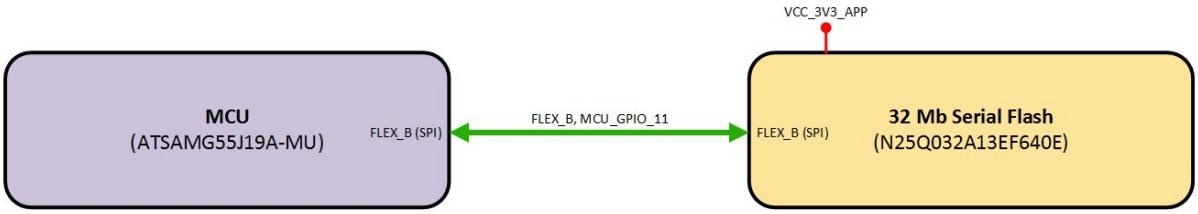


The Hall-effect sensor's power power consumption is reduced if it is only driven when sampling the output. The MCU does not use FLEX_A as a SPI bus, but rather as GPIO. The FLEX_A_SCLK signal enables power to the Hall effect sensor, and the output is sampled using the MCU's internal ADC. The Hall-effect sensor can latch up if VCC does not drop low enough before VCC is reapplied. The load switch includes an output load dump resistor to discharge the bypass capacitor on the Hall-effect sensor, which guarantees the sensor powers up correctly the next time the load switch is closed.

HMD Functions

Serial Flash

HMDs require an external serial flash to store calibration data for the display.



The serial flash connects to the MCU using the FLEX_B SPI bus. This is the same SPI bus used by the trackpad in controller applications.

Other HMD Peripherals

Other peripherals are required for HMDs, but are not included in the reference design. These devices should be chosen based on the unique requirements of the device.

Interfaces

The following table of interfaces outline the connections between devices in the electrical system.

Optical Sensors	
SENSOR_A[32:1]	Reserved for future use.
SENSOR_B[32:1]	Sensor envelope signal.
SENSOR_VCC[4:1]	Four power banks to supply eight sensors each.
SENSOR_VCC_EN[4:1]	Enable signals from the FPGA to each of the four sensor power rails.
FPGA GPIO	
FPGA_GPIO[10:5]	Available for application I/O expansion and auxiliary timestamp inputs.
FPGA_GPIO[4]	Available for application I/O expansion. Typically used for application power enable.
FPGA_GPIO[3:1]	Available for application I/O expansion and auxiliary timestamp inputs.
RADIO GPIO	
RADIO_GPIO[8:1]	Reserved for future use.
NFC_P, NFC_N	Reserved for future use.
NRF_FPGA_GPIO[2:1]	Reserved for future use.
FLEX_C (SPI)	
FLEX_C_SCLK	Serial clock for radio and IMU SPI interface.
FLEX_C_MOSI	MCU data output for radio and IMU SPI interface.
FLEX_C_MISO	MCU data input for radio and IMU SPI interface.
NRF_nCS	Radio chip select.
NRF_DR	Radio to MCU, data ready interrupt signal.
IMU_nCS	IMU chip select.
IMU_INT	IMU to MCU, data ready interrupt.
FPGA	
FPGA_SCLK	Serial clock for FPGA SPI interface.
FPGA_MOSI	MCU data output for FPGA SPI interface.

FPGA_MISO	MCU data input for FPGA SPI interface.
FPGA_nCS	FPGA chip select.
FPGA_DATA_RDY	FPGA to MCU, data ready interrupt.
FPGA_nRESET	Reset signal for FPGA configuration logic.
FPGA_CDONE	FPGA signal to the MCU indicating configuration status.
FPGA_CRESET	Forces the FPGA into configuration reset.
FPGA_MCU_1	Reserved.
FPGA_MCU_2	Reserved.
FPGA_MCU_3	Reserved.
MCU_GPIO	
MCU_GPIO_1	Red LED
MCU_GPIO_2	Green LED
MCU_GPIO_3	Controller: System button input. Power button.
MCU_GPIO_4	UART TX for debug and manufacturing test.
MCU_GPIO_5	UART RX for debug and manufacturing test.
MCU_GPIO_6	Controller: Haptics power enable.
MCU_GPIO_7	Controller: Haptics drive signal.
MCU_GPIO_8	Controller: Menu button, ISP button input. HMD: ISP button.
MCU_GPIO_9	Controller: Grip button input.
MCU_GPIO_10	Blue LED
MCU_GPIO_11	Controller: Trackpad chip select. HMD: Flash chip select.
MCU_GPIO_12	Unused
MCU_GPIO_13	Controller: Trackpad to MCU, data ready interrupt.
MCU_GPIO_14	Controller: Analog trigger ADC input.
FLEX_A_MISO	Controller: Configured as GPIO, trackpad button input.
FLEX_A_MOSI	Controller: Configured as GPIO, trigger button input.
FLEX_A_SCLK	Controller: Configured as GPIO, analog trigger enable output.
FLEX_B	
FLEX_B_SCLK	Controller: Trackpad SPI serial clock. HMD: Flash SPI serial clock.
FLEX_B_MOSI	Controller: Trackpad data input. HMD: Flash data input.
FLEX_B_MISO	Controller: Trackpad data output. HMD: Flash data output.
I2C_BATT	
SCL	I2C clock between MCU, charger and fuel gauge.
SDA	I2C data between MCU, charger and fuel gauge.
BATCHG_INT	Charger and fuel gauge to MCU, data ready interrupt.
USB	

USB_D_P	USB positive data signal. USB Full Speed.
USB_D_N	USB negative data signal. USb Full Speed.
USB_VBUS	USB 5V power.
MCU_DEBUG	
SWDIO	MCU single wire data.
SWCLK	MCU single wire clock.
nRST	MCU reset.
SWO	MCU single wire output.
RADIO_DEBUG	
SWDIO	Radio single wire data.
SWCLK	Radio single wire clock.
nRST	Radio reset.
SWO	Radio single wire output.