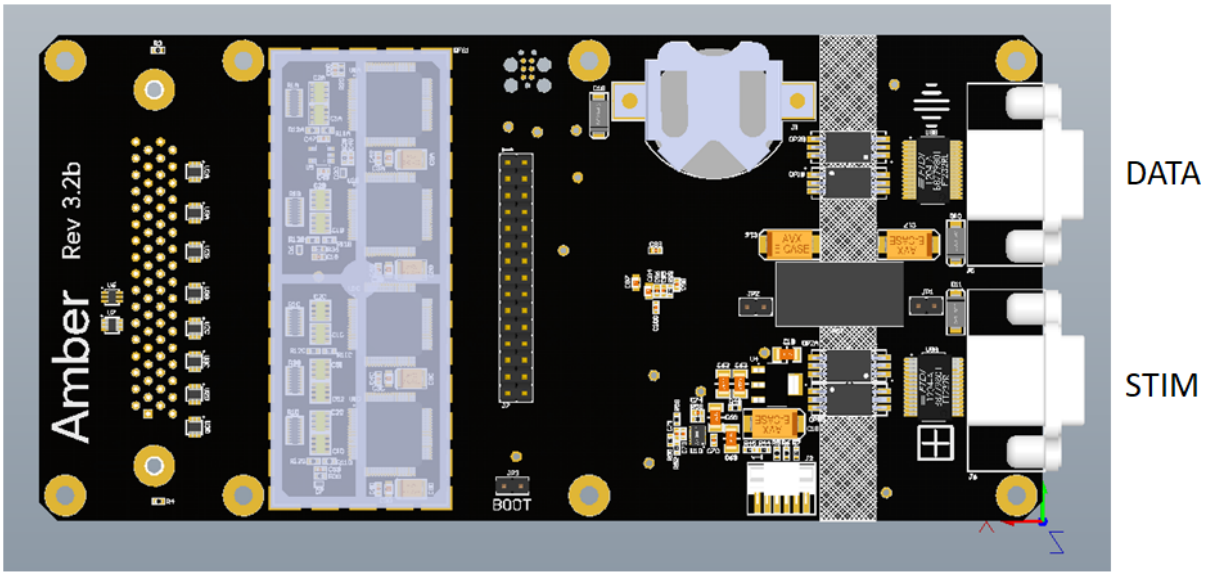
Draft: 2019-08
AMBER M20 EEG Acquisition System Design Document

1. The M20 Bioamplifier

M20 is a 32 channel EEG acquisition system. It is designed to be connected to a PC USB port for data communication.



1. Hardware Design

1.1. DAQ

The heart of the design is the ADS1299 8 channel bioamplifier from Texas Instruments. This single IC provides the front-end amplification and 24 bit conversion. It is particularly suitable for EEG measurements as it provides a ground bias generator and built-in lead-off detection.

4 of the ADS1299s are connected together in a parallel data configuration. All 4 together provide 32 EEG channels. U1A is the master ADS1299 and provides the conversion clock to the other 3 converters to keep things synchronized. U1A is also the master bias generator.

Although the ADS1299 can provide differential or single-ended inputs, the configuration of the board allows for only single-ended inputs on channels 1-24, and the option of single-ended or differential inputs on channels 25-32.

The sample rate is 250 samples per second.

Communication with the MCU is over 4 wire SPI running at a clock speed of 4 MHz. The ADS1299 provides a signal (#DRDY) that goes low when a conversion is ready to be read out of the chip. Only the master has this signal connected to the MCU and as all timing is referenced to it. Individual chip selects are asserted to read the data from each of the ADS1299s in turn. An alternate option would be to read the data in a daisy-chain configuration, but the parallel configuration was chosen for its speed and flexibility.

1.2. MCU

The MCU is a Kinetis (NXP/Freescale) KL27Z256 in a 64 pin QFP package. This is an Arm Cortex M0 processor running at 48 MHz with 256K of flash memory. The reasons for choosing this processor include the speed, cost, availability, and ease of development with the Kinetis tool chain.

Debugging is done through a dedicated JTAG port implemented with a special TAG connect cable. Production programming is done through the DATA USB port utilizing the built-in ROM bootloader and a special Python tool.

The MCU handles the data conversion from the ADS1299s and communication with the host PC. It is also responsible for monitoring the on-board power supplies.

4 input pins allow for identifying the hardware revision through pull-down resistors. 4 pins are also reserved for future expansion.

Although the MCU has low-power modes, in this design it is always running at full power.

1.3. Power Supplies

Power to the board comes from either one of the 2 USB connectors. USB voltage is diode-ORed and powers an isolated DC/DC converter (PS1). This provides the 4kv isolation and patient leakage current protection. Jumpers JP1 and JP2 must be installed for normal operation. These were implemented to provide power flexibility in the event a battery would need to be used.

The 5V output of the DC/DC converter is linear regulated down to 3.3V. A linear regulated was chosen to provide a lower noise floor than another DC/DC. The 3.3V primary powers the MCU and digital logic.

A charge pump based converter also runs off the 3.3V rail and provides the +/-2.5V rails to the ADS1299s. Normally charge pumps are rather noisy, but this particular IC (LM27762) has built-in linear regulators following the charge pumps and is tailored to these sorts of applications.

A separate 2.5V reference is also generated to power the MCU analog section and to provide a precise reference for power supply measurements.

1.4. USB Ports

Two USB ports connect the PCB to a PC. The DATA port is used to transfer the streaming EEG data. The STIM port allows control of the board and is used to set marks in the EEG data stream.

Each port uses an FTDI232 USB to serial IC on the PC side. The MCU side implements optical isolators for patient protection. These are rated at 4kv with a maximum data rate of 10Mbps.

1.5. Real Time Clock

A real time clock is implemented. It communicates over I2C with the MCU and has battery backup.

1.6. Micro SD

A micro SD slot is provided for future expansion. It is connected to a second SPI port on the MCU.

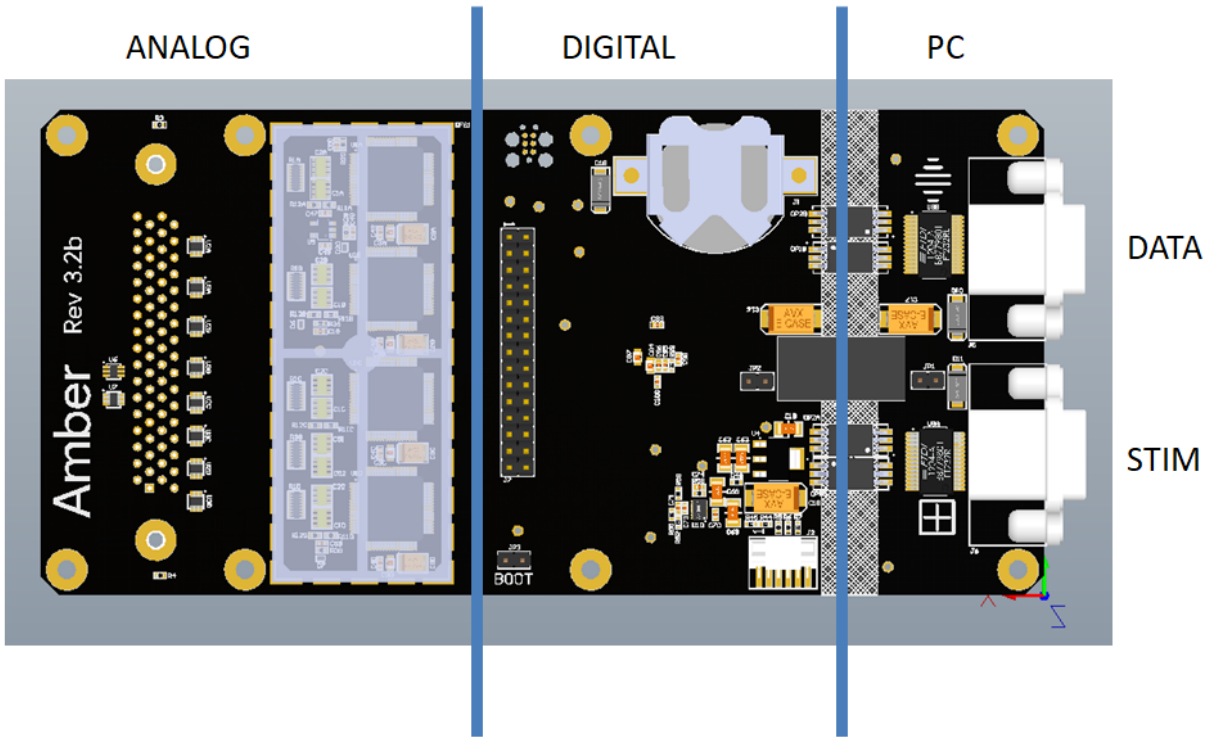
1.7. Expansion/Debug Header

Board test and expansion signals are brought to this header. They are labeled in the layout and provide an easy means of testing or debugging the board.

2. Layout

The PCB is designed as a 6 layer board. The layer stack is signal, ground plane, signal, signal, power connections, signal. Minimum line/space is 6/6 and minimum via size is 6/16. Board material is plain FR4 and finish is ENIG. This is standard for today's fabrication processes and allows the board to be manufactured at any number of board houses.

The board is separated into 3 distinct sections: PC non-isolated, digital, and analog.



2.1. PC

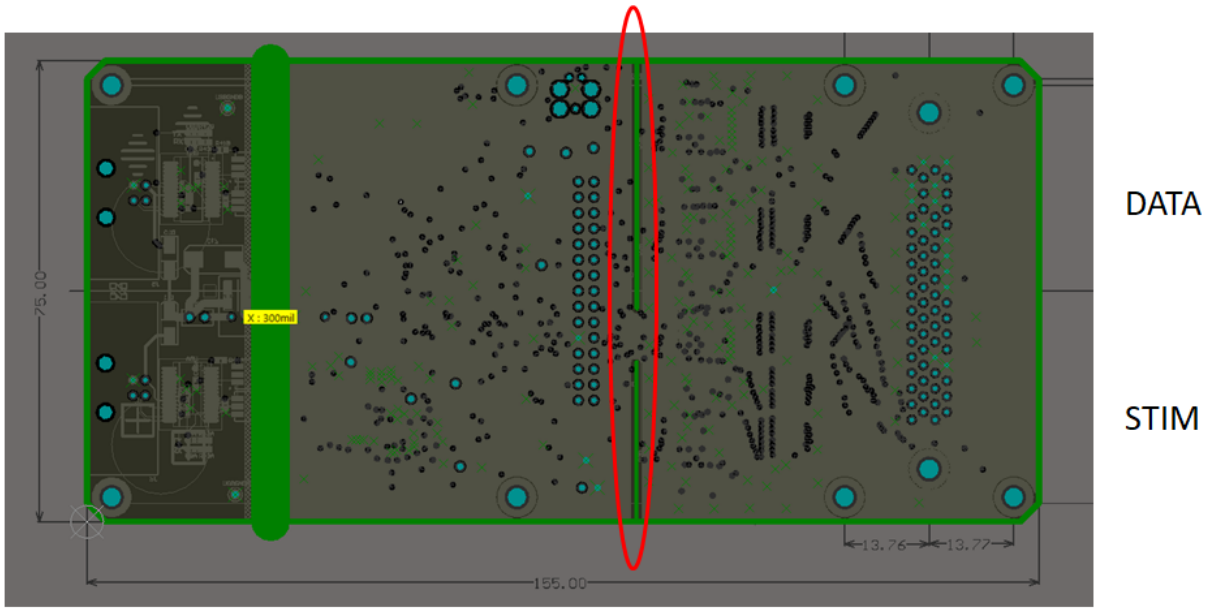
The PC side has the USB ports and half the optical isolators and DC/DC converter. Creepage and clearance is maintained at 8mm.

2.2. Digital

The digital section incorporates the power supplies and MCU. The expansion header is also included here.

2.3. Analog

The analog section protects the microvolt level EEG signals from the noisy digital sections. An RF shield is included that encapsulates the ADS1299 converters. A split ground plane separates the sensitive ground returns from the noisy digital grounds:



2.4. Test Points

Various individual test points are provided as well as the debug header for future automated testing.

3. Firmware

The firmware was developed in C using the Kinetis Design Studio. Some functions were developed with the help of Kinetis's Processor Expert which simplifies MCU configuration parameters. Some low-level drivers were hand written for the high performance and low latency needed processing the EEG data.

A main loop handles gathering EEG data, doing serial communication, and blinking LEDs. Some tasks are event driven such as EEG data ready and serial communication. These typically either buffer the data or raise flags for processing in the main loop.

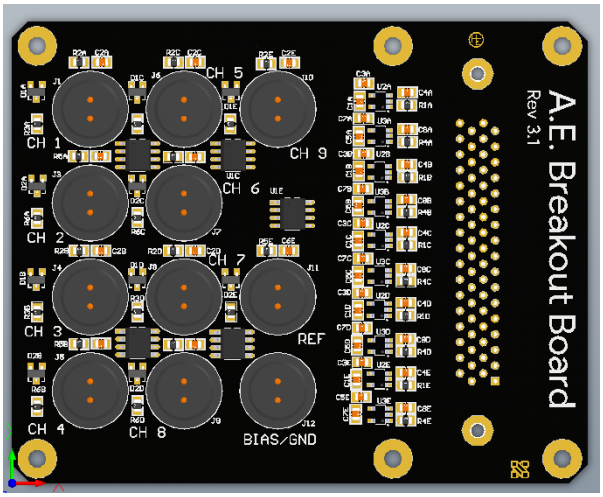
A text based command line interpreter (CLI) is implemented for easy control and communication. Any terminal program can be used for sending commands. Although the CLI will respond to commands from either port, only characters on the STIM port are echoed. CLI responses on the DATA port are multiplexed with the streaming EEG data by prefixing 'CLI:' to the response line. Commands to the CLI are terminated by a carriage return.

EEG data is continuously streamed on the DATA port in ASCII format. Each channel is comma separated and the EEG value is encoded as a signed 24 bit value. A frame of EEG data has the following format:

SEQUENCE NUMBER	CH 1	CH 2	…	CH32	MARK	CR/LF

The frame is prefixed by ‘DATA:’. The sequence number is an auto incrementing unsigned 32 bit value. The mark is a user-selected data value appended to the EEG stream. Each line is terminated by a carriage return/line feed.

M20 Active Sensor Breakout Board Design Document



1. Overview

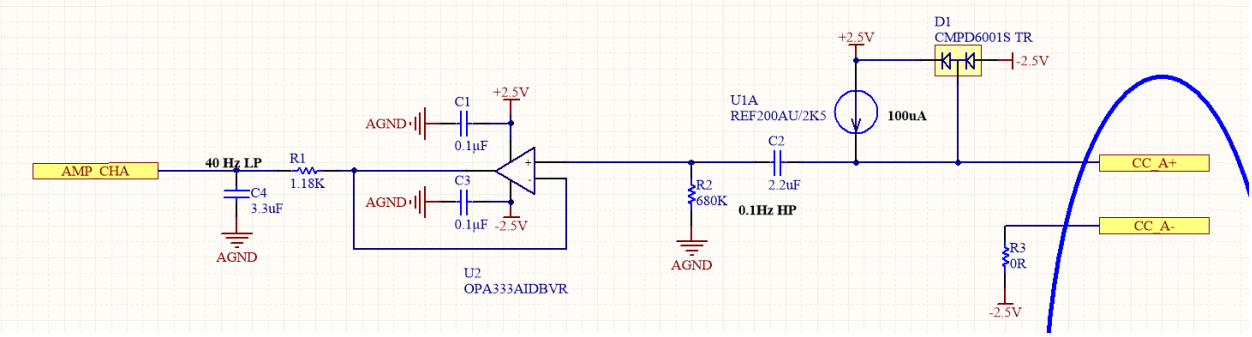
Conventional wet EEG sensors rely on electrolytic gels to penetrate hair and provide a clean conductive path between sensor and skin. The gel also 'fills-in' potential gaps formed between the sensor and skin during application and subject movement. The use of wet EEG sensors is time consuming and uncomfortable for the subject.

The M20 bioamplifier program remedies this by using dry electrodes. Dry electrodes have a higher impedance than wet electrodes. M20’s dry electrode solution combines sensors with mechanics and electronics which match the higher impedance. These are known as active sensors. A high impedance preamplifier circuit is integrated with the dry sensor to provide the bioamp with a clean signal. Typical active sensors require three wires, 2 for power and 1 for the signal. M20’s active sensors are powered by a current source requiring only 2 wires running to the electrode. The circuitry for the current source is integrated into the active electrodes adapter board that plugs into the bioamp.

The Active Electrode Breakout Board is used with the AMBER EEG Bioamplifer to make electrical connections with the test subject. It includes 9 channels of EEG plus reference and bias connections.

2. Design

2.1. Electrical



The electrical design of each EEG channels includes a 100uA current source. This powers the electrode, which is an operation-amplifier operating in a bootstrap mode. Microvolt signals from the operational amplifier electrode are high-pass filtered at 0.1Hz and buffered, then low-pass filtered at 40 Hz. This filtered and buffered signal is presented to the bioamplifier board.

The reference electrode is similarly processed as an EEG electrode.

The bias electrode is simply passed through without any electrical processing.

2.2. Mechanical

2.2.1. Electrodes

The electrodes connect to the breakout board through an industry-standard touchless connector from Plastics1. The touchless connector ensures that no electrical conductor can be accidently touched during the operation of the EEG system.

2.2.2. Main Connector

The connection to the AMBER bioamplifier is made through a 68 pin vertical SCSCI connector.

2.2.3. PCB

The PCB is a 4 layer standard FR-4 substrate. The size is 94x75mm.

Firmware Design

1. Overview

The firmware for the Amber project is responsible for taking EEG samples from the ADS1299 and formatting them for serial delivery to an external PC. It will also accept commands from a second serial port that is running a basic CLI (Command Line Interpreter). It is designed to run on an NXP MKL27Z256VLH4 processor.

2. Structure

All of the code runs in a single thread; therefore, each task must maintain its own state machine and relinquish orderly control to prevent the system from blocking. Each task is called from the main loop in the main.c module. The tasks are as follows:

1. Leds
2. EEG
3. Serial

2.1. Processor Expert

Processor Expert is an automatic code and configuration generator by NXP. It is used to speed up development of firmware for specific microcontrollers by abstracting to driver-level code. It is used extensively in this project. The modules will be described later in this document.

3. Modules

3.1. main.c

The entry point into the firmware is main.c. However, before the first line in main.c has been executed, C initialization routines have already been run to initialize static data structures. This occurs in the background and is invisible to the developer.

The first task is to initialize the Processor Expert(PE) modules. This includes setting the CPU clock and operating modes.

Following the PE initialization, the firmware initializes serial communication, LEDs, SPI communication (used with the ADS1299 converters), and EEG data structures. After initialization is complete, the main loop starts to service the LED, EEG, and serial tasks.

3.2. eeg.c

This module is responsible for high-level communication with the ADS1299. It includes functions for initializing the converters and reading and writing the ADS1299 registers. The EEG task checks if there is data to be read from the converters and calls the function to read that data.

3.3. spi.c

This module contains the low-level functions to communicate with the ADS1299 converters. It is responsible for setting the chip-select pins and performing low-level reads and writes of the converter registers and also the EEG data.

EEG data is read from each of the four ADS1299 converters in turn. Starting from converter 0, EEG data is clocked in from each of the 8 channels per converter, including a status register. Each channel's 24 bit EEG data is placed into a 32 bit data structure and pushed onto a queue. This queue is used by the serial task to transmit the data to the host PC. The queue should never be completely filled in normal operating conditions. In the case that the queue fills up, data will be overwritten.

Registers in the ADS1299 that need to be updated are written immediately after EEG is read to prevent timing issues and data loss.

3.4. leds.c

This module contains functions for setting the various board LEDs. The LED task is responsible for blinking the green status LED.

3.5. serial.c

This module contains the serial task and low-level functions to send data to the serial ports. Buffering and actually transmitting the data is handled by the UART PE module.

The serial task checks for new EEG data and packages it into a data string. This is transmitted by the PE module to the PC.

It also checks for new characters received on the data or mark serial ports. This is passed to the CLI and the response is sent to the serial port.

3.6. cli.c

This module handles the command interpreters on both the data and mark ports.

It takes serial input from the serial module and cleans and parses it. The sentence is parsed into a command string, and an argument string. The command string is fed into a large if-else-if structure to compare it to known commands and the arguments are evaluated. An output string is passed back to the serial module for transmission to the host.

3.7. Events.c

This module handles so-called events from the PE code. These are really post-processed interrupts from each PE module.

Significant interrupts are the UART interrupts and timer interrupt. The UART interrupts are called whenever a character is received from the host PC. This is transferred to the serial module and placed into a buffer to be parsed by the CLI.

The timer interrupt is triggered every millisecond and decrements various timer variables used by the modules.

3.8. version.c

This module reports the version and serial number of the firmware.

3.9. settings.h

This header file includes project-wide settings.

4. Processor Expert Modules

The following modules are part of the Processor Expert build system. These modules are compiled from PE GUI settings and provide driver-level access to the hardware.

4.1. CPU

This module allows low-level access to the CPU hardware. Oscillator, memory, and operating mode settings are provided by this module. This is where the master core clock is set to 48MHz.

4.2. Serial_LDD

This low-level driver module provides access to the serial UART. Two instances are instantiated for UART0 and UART1. Baud rate and other communication settings are configured with this module. The methods exposed include 'SendBlock' for sending data, and 'ReceiveBlock' for receiving data. Events include the data received interrupt.

4.3. TimerInt_LDD

This module configures the timer interrupt. This timer can be used for an embedded RTOS, but in this instance it generates an interrupt every millisecond to decrement timer variables used by various functions in the system.

The timer is auto-initialized and generates the timer interrupt event.

4.4. SPIMaster_LDD

This module interfaces with the SPI hardware on the microcontroller. Using this module, SPI pins, clock frequency, and phase are configured. Data to the ADS1299 converters are sent using this module.

This module provides the 'SendBlock' and 'ReceiveBlock' methods; however, these built-in functions are not used because they require a large latency which is unacceptable. The actual SPI subsystem is accessed through the registers for the lowest latency transmissions.

4.5. BitIO_LDD

Multiple instances of this module handle the GPIO operations of the MCU. This module allows the firmware to setup GPIO pin, direction, speed, and drive strength. This module is used to drive all the chip-select pins for ADS1299 access and LEDs.

4.6. Flash_LDD

This module handles FLASH memory reads and writes. This is primarily used to store the serial number of the device.

5. Building firmware

To build the firmware, Kinetis Design Studio 3.2 is used, although gcc should work for a complete project. KDS is required to work with the PE tools and modules. Also, KSDK1.3 will need to be downloaded and installed.

Install KDS 3.2, then select File-Import-Existing Projects into Workspace. Select the project directory and all files will be loaded into the workspace. Building is as simple as selecting 'build'.

6. Flashing

Flashing a board can either be accomplished with a JTAG debugger and KDS, or via USB and the compiled BIN file. For instructions, see the flash document.