Bootloader

This page contains information about how the bootloader functions with the microcontroller bootloader interface to allow for field-programming of the microcontrollers through UART.

Framing of Transmissions

All UART communications are sent through a framed transmission protocol. A frame consists of three distinct elements. A frame has a HEADER byte, called SOT (start of transmission) and ends with an EOT byte (end of transmission). All data in between consists of a single hex file record (hex file line) to be used for programming the microcontroller. The main purpose of the bootloader is to transmit each record from the hex file to the microcontroller for programming and interpretation. Currently, only a single record is transmitted within each frame. This could be modified to allow for multiple records in a frame to increase speed of bootloading, however it additionally complicates the program. Because the EOT and SOT bytes may not be unique, an additional byte is used to distinguish them. A DLE byte (delimiter byte) is used to indicate whether or not a byte is infact the SOT or EOT bytes. A DLE is placed before any bytes within the frame that are NOT the DLE, SOT, or EOT bytes. This way, it acts as a form of escape character. If the serial line sees the DLE and then an SOT byte, it knows that the SOT byte is actually just generic data for programming. To avoid errors, if a DLE is to be sent as data, it will also be preceeded by another DLE. This way, the serial line will know to interpret the DLE as raw data.

Programming with the Bootloader

The bootloader is quite easy to use as a class for programming. First, instantiate a bootlaoder object with the appropriate HEX file path, serial port path, and baud rate (Note: If not using a custom bootloader, use a baud rate of 115200, default). The Bootloader class provides a number of functions to interract with the microcontrollers memory. A call to the `loadHex()` function will initialize the hex file for transmission - this does not need to be done, as the file will be loaded automatically when the program function is called. Therefore, only a small number of functions are needed from the Bootloader class. These include `erase()`, `program()`, and `start()`. These functions are quite explanatory, but they will be discussed anyways. The erase function will erase all memory pages within the microcontroller that are not in use by the bootloader. The erase function must be called when loading a new program into the microcontroller to ensure that the memory space is not already occupied. If there is residual data there, the program will be copied on top of it, and will not function as intended. The program function can be used for programming the microcontroller with the specified hex file. This function will load the hex file and transmit it record by record until the microcontroller program is complete. The start function is what triggers the microcontroller to begin execution of the bootloaded program. Without a call to start, the microcontroller will remain in the bootloading mode and new files can be loaded onto it. Calling start will cause the microcontroller to jump to the start of the bootloaded program.