MooresCloud Holiday
ATmega328 Firmware Specification
DRAFT 7th January 2014

Commands are accepted on the serial interface, out-of-band from the SPI data flow. To keep the code and processing overhead low, commands use a postfix notation where the argument(s) precede the command.

Serial port is 115200, 8N1. Crystal is 20MHz for Holiday r2 as of Aug 2013.

Commands are mostly upper case letters. Arguments are decimal integers. Currently zero, one, or two arguments are allowed, and they are separated by commas. Commands do not need to be followed by a CR, LF, etc. CR or LF will be ignored, as will any control codes like BS. If echo is on, input is echoed (as typed), followed by CRLF.

Generally if no argument is provided, the setting will not be changed. If the command doesn't update a setting, the argument will default to 0. Commands will respond with the new or current setting (as integers). Responses are followed by CRLF.

| Command | Arg 1 | Arg 2 | Description |
|---|---|---|---|
| ? | | | Returns version e.g. "HolidayDuino07" |
| A | 0 to 3 | | Read analog input A0 to A3 on expansion header |
| B | | | Read button state as 6 bit value – bits 0-2 are current state, bits 4-6 are state captured at boot |
| C | 0 to 31 | | Read control byte from EEPROM first page (without interrupting any pattern program) |
| C | 0 to 31 | 0 to 255 | Write control byte to EEPROM first page |
| D | 0 to N | | Set default startup pattern 0=none – default 0D |
| E | 0 or 1 | | Disable or Enable echo – default 0E (not saved between boots) |
| G | 0 to 3 | | Set A0 to A3 on expansion header as output with GND/LOW state (not saved between boots) |
| H | 0 to 3 | | Set A0 to A3 on expansion header as output with HIGH state (not saved between boots) |
| I | 0 to 3 | | Read A0 to A3 as digital input |
| L | 0 to 250 | | Set length of LED string – default 50L (updates EEPROM, requires restart) |
| P | 0 to 32 | | Play predefined pattern program 0=stop – default 0P |
| Q | | | Read current pattern program # as 6 bit value (0 to 32, 0=none), bit 8 indicates user has pressed a button since start up |
| R | 0 to 15 | | Read register 0 to 15 (see EEPROM data structures document) |
| R | 0 o 15 | 0 to 255 | Write register 0 to 15 |
| T | 0 or 1 | | Enable test mode (for SPI verification) – this allows less than a full frame to be treated as good, and will echo back hex byte data over serial port |
| V | | | Calculate approximate supply voltage in mV |
| W | 0 to 65535 | | Set watchdog timer in milli-seconds 0=none – default 0W |
| [ | | | Lock I2C bus – Arduino becomes I2C slave, allowing Linux to access I2C EEPROM, also stop any pattern program |

| | | | |
|---|---|---|---|
| ] | | | Release I2C bus – Arduino allowed as master |

Setting LED length to 0 will disable the SPI interface & handshaking on the ATmega, potentially allowing connection of WS2801, LPD8806, or similar LED strips to the ATmega ICSP/SPI header (with appropriate changes on Linux side of SPI interface).

Watchdog timer is kicked by any SPI, I2C, or serial data (command or Linux console).

We might want to add checksum setting for SPI packet verification.

See EEPROM Data Structures document for further information on start up and pattern programs.

We might want an initial boot (console activity) time limit – e.g. if Linux console is in reboot loop, or if uSD card faulty/not present (imx outputs hex error code every X seconds).

SPI Handshaking – Linux to Arduino:

- Linux checks ACK/BUSY from Arduino (GPIO 23 is high) – if not, abort!
- Linux asserts SS (GPIO 19 made low)
- Linux waits for ACK/BUSY from Arduino (GPIO 23 goes low)
- Send bytes using MOSI & pulsing SCK high – 3 x #LED bytes in GRB order (to match WS2812 pixels), MSB first. Bitbanging is approx 3MHz, so we include an approx 18us inter byte delay to allow Arduino interrupt routine to process byte. If we can get hardware SPI working, we could try dropping to 500kHz, and avoid spinning CPU.
- Linux de-asserts SS (GPIO 19 made high)
- Arduino de-asserts ACK/BUSY (GPIO 23 goes high) immediately if incomplete frame received, otherwise only after frame delivered to LED pixels

Test firmware:

- Verify serial comms to imx
- Verify I2C EEPROM access
- Verify A2D test voltages
- Drive output LED
- Controlled by imx or need separate host connection?