EsPico and ES4 bootloader

Entering the bootloader

The bootloader can be entered either by using the `dlfw\n` command in firmware, or by driving `DWNLD` pin low and resetting the device (for example by driving the `RESET` pin low momentarily).

Communication settings

	EmStat Pico	EmStat4
Supported interfaces	UART	UART, USB (CDC)
Baudrate	230400	230400
Flow control	None	None

Protocol structure

The protocol is based on a request->response communication flow where the host requests (or sends) data and the EmStat responds. Data is transferred in ASCII style where numbers are formatted decimally or hexadecimally. The response from the EmStat will always start with the first letter of the command and end with a newline (`\n`) character.

The command format is as follows:
`[command string][payload (opt)]\n`

For example the reset/reboot command looks as follows:
`boot\n`

Commands

Summary

Command	String	Payload	Description
Version	`t`	-	Request the software version of the bootloader
Reset	`boot`	-	Reboot/Reset the device
Upload start	`startfw`	-	Start the firmware upload process
Upload data	`data`	Encrypted firmware block	Send block of firmware to the EmStat
Upload end	`endfw`	-	Finish firmware upload

Version command

The EmStat will respond with the bootloader version number.
Example:

```
tespbl11#Oct 18 2019 15:27:17\n
D*\n
```

Reset command

This will reset the device. If the uploaded firmware is valid and the DWNLD pin is high, uploaded firmware will run.

Upload start command

Tell the EmStat that we want to start uploading new firmware. The EmStat will respond with an empty line.

Upload data command

The upload data command is really where the magic happens. The `upload data` command has the following structure:

`"data"[block size (2 hex chars format)][Data block, hex format][Fletcher16 checksum 4 hex chars]\n`

If the fletcher checksum does not match the calculated one, then the bootloader will throw an error (`!000C\n`). The application can then choose to resend the data packet or restart the transfer (by sending a new `startfw` command).

The Fletcher16 calculation is implemented as follows (example code is in C#):

```
// returns 16 bits where bit 0-7: sum1 8-15: sum2
static uint16_t fletcher16( uint8_t *data, int count)
{
   uint16_t sum1 = 0;
   uint16_t sum2 = 0;
   int index;

   for( index = 0; index < count; ++index )
   {
      sum1 = (sum1 + data[index]) % 255;
      sum2 = (sum2 + sum1) % 255;
   }

   return (sum2 << 8) | sum1;
}
```

The following line illustrates how the `upload data` command looks like:

```
data803A752FF9C1F3390000ED950F6C018EC84D7228E2355D3C87DAD1B23C16652AE1EF779798D6
B727CE98F9F6173C298760B67EC883E97B2D89F3FF89AB99C51C2CE61FC5CC5280E9888AD579E360
8A3FE2C0F8F06210EA84E8504FEB892E284C76432164D4A90F935BE6B0E473D9E9319DE2F3A5E17A
F6ABD683DB0757EDC906A3E961\n
```

where `80` is the length in Hex (128 decimal) and `E961` the Fletcher16 checksum.

Upload end command

Indicate to the EmStat that the firmware is fully transferred. The firmware will run on the next reset, after validation. The EmStat will respond with an empty line.

Example program flow

The commands below show the normal program flow

1. Send `startfw` command and wait for newline response.
2. Send `data` command and wait for newline response.
3. Repeat 2. until firmware file is transferred.
4. Send `endfw` command and wait for newline response.
5. Send `boot` command to reset into firmware.

Error handling

If an error occurs it will be printed in the following format: "!0000\n" where 0000 is a hexadecimal presentation of the error code.