Smoothie error reporting system proposal

2017

OVERVIEW

This document aims at describing a new way for Smoothie to report errors to users.

GOALS

1. Unique error IDs makes reporting and documentation easier
2. Firmware redirects users to the wiki for detailed information about the error and how to troubleshoot it
3. Standardized error format makes interaction with host software easier to implement
4. Denser way of storing errors means they can be remembered and polled for, removing the need to rely on the host catching the serial message (useful for example over the web interface).
5. Errors taking less flash space would mean we would be able to add more errors (for example invalid configuration option formats, or invalid G-code parameters) to help users diagnose problems that were invisible before.

EXAMPLE SCENARIO

Currently, if a user has a temperature error, they will see the following error message over serial :

```
ERROR: MINTEMP or MAXTEMP triggered on T. Check your temperature sensors
```

Using the new system, the reported error would instead be :

```
ERROR: #12 [mintemp], go to http://smoothieware.org/error-12#T,-72,174
```

This link would take the user to a page describing in detail why the error occurs, and what to do to resolve the issue.

Additionally, parameters can be passed to the page, which can then be used by the page to describe and diagnose the specific problem this user is experiencing instead of staying general. For example here the error passes the parameters «T,-72,174» which can be displayed in the wiki page.

Users reporting errors often do not copy/paste the message, so we would go from :

```
<suXXor123> I have some weird temp error, fix it now
```

Which is far from ideally useful to :

```
<Great-job-smoothie> I have a #12 error
```

Which is more specific.

SPECIFICATIONS

Storage

After an error occurs, it is stored in RAM until it is cleaned, or the system is reset with the error fixed.

This system is only for errors, and not for warnings, meaning that it is aimed at informing users of problems they must absolutely resolve. (It is possible this could be extended in v2).

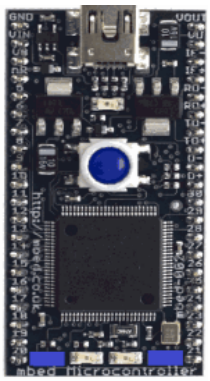
This means we can use a bit of RAM to store the errors, as it is expected that these errors will be solved before users actually use Smoothie, and therefore during normal use, no error is stored, so no RAM is used.

All in all, during normal use, this feature essentially only uses :

- The ram footprint of an empty std::string
- The flash footprint of a bit of code, and of a id->name lookup table for errors (for example 12 is “mintemp” etc), and some flash should be saved as this is more compact than the current system

The default place to store the error list is the Kernel module, where the ->error method would also be located. It’s possible when the feature is pushed as a pull request to github, the core dev team will ask it to be moved.

Signaling



When the error list is not empty, the firmware should do all it can to warn the user that there are errors to be read (via M637), by :

- Outputting a message whenever the board connects via USB or Ethernet explaining an error is present
- Blinking the 4 LEDs on the board in a specific way (for example the 4-LED pattern the mbed system uses)

Smoothie-side

The idea here would be that all errors use the same function call.

It would presumably be in Kernel, and it would look something like this :

```
char values [20];

sprintf (values, "%d,%d", current_temp, temp_delta);

THEKERNEL->error( 12, values, true, true );
```

Where the parameters are as follows :

- Uint16_t the error code
- Char [20] (or std::string, whatever) values, useful information about the error
- Bool halt, whether to enter HALT mode after this error, and display the HALT message
- Bool emergency, whether this error requires the machine to be turned off immediately (like for temperature errors)

The call above would result in the following message over serial :

```
ERROR: #12 [mintemp], go to http://smoothieware.org/error-12#-72,174

HALT asserted - reset or M999 required, turn machine off
IMMEDIATELY and resolve issue before use
```

Errors are stored in a string, for example :

```
THEKERNEL->errors ( std::string )
```

The format (proposition) they are stored as is, for example in the case of multiple errors :

```
#12,-72,17#36,xmin
```

The M636 (proposition) G-code can be used (for example by hosts) to get the raw error list :

```
> M636

< #12,-72,17#36,xmin
```

The M637 (proposition) G-code can be used (for example by users) to get a readable list of errors :

```
> M637

< ERROR: #12 [mintemp], go to http://smoothieware.org/error-12#-72,174
```

```
< ERROR: #36 [limitswitch], go to http://smoothieware.org/error-36#xmin

< Resolve all errors before you resume using Smoothie
```

The M638 (proposition) G-code can be used to empty the error list.

Wiki-side

Every error would be documented on the wiki, including :

- Possible causes
- Explanation of possible solutions
- Troubleshooting tree of possibilities
- Detailed analysis of parameters passed to the page

The new DokuWiki allows us to run JavaScript, which opens the possibility of actually parsing and using the parameters. A simple markup language would be defined to allow users to do this simply by editing the wiki.

Note: This document uses links to per-error wiki error pages, but there is also a wiki page that is supposed to link all errors in a table, which can be found here: http://smoothieware.org/error