Radio Telemetry Tracker C/C++ Style Guide

Nathan Hui, nthui@eng.ucsd.edu

June 10, 2020
v1.0a

1 Philosopy

In general, code should be written in a way that clearly expresses the local intent of the code, and allows for easy readability. The following are rules that should always be followed when writing code for the RTT project. Cases not covered by the below rules should be written in a way that is easy to understand and clearly expresses the intent behind the code.

The single most important rule when writing code is this: check the surrounding code and try to imitate it.

2 Code Formatting

1. Always use spaces. Tabs should be interpreted as 4 characters.
2. Always include spaces around binary operators (`+`, `-`, `*`, `/`, `&`, `|`, `>>`, `<<`, `>`, `<`, etc).
3. Include newlines between blocks of code where appropriate. Use functions to break up and reuse code to facilitate readability and debugging.
4. Always place braces on the new line, not indented:

```
while(true)
{
    if (true)
    {
        return;
    }
    else if (false)
    {
        return;
    }
}
```

5. Always retain braces for one-line bodies.
6. Always lace semicolons indicating an empty statement on their own line.

3 Code Namespace

1. Keep symbols as concise and informative as possible - avoid the servers ConnectionManager object that creates a server connection object.
2. Functions and members should be camelCase.

3. Local variables should be camelCase.

4. Avoid global variables. Where unavoidable, ensure the global variables do not unnecessarily pollute the namespace (e.g. prefix with the associated module or function).

5. Reserve the names `i`, `j`, `k`, `x`, `y`, `z` for array and loop indexes. Avoid using these if possible in favor of more descriptive names (e.g. `bufIdx`, `arrIdx`, `rowIdx`, `colIdx`).

6. Enum typedefs should be suffixed with `_e`:

```
typedef enum myEnum
{
    ...
} myEnum_e
```

7. Struct typedefs should be suffixed with `_t`:

```
typedef struct myStruct
{
    ...
} myStruct_t
```

8. Capitalize constants

9. Use `<stdint.h>` types where possible.

10. Use `uint8_t` for working with bytes.

11. Do not use macro magic to rename functions.

12. Header inclusion guards should be `__MODULE_H__`. Add a comment at the corresponding `#endif` to clarify which `#endif` belongs to the include guard.

3.1 C Specific Rules

1. Symbols in each module should be named `module_symbol`:

```
// in sdr_record.c
void sdr_record_InitBuffers(void);
// in localization.c
void localization_GenerateEstimate(void);
```

2. Declare all local variables at the beginning of the scope block.

3.2 C++ Specific Rules

1. Classes should be CamelCase

4 Documentation

1. Use Doxygen style comments (see https://www.doxygen.nl/manual/docblocks.html)

2. Functions should be documented both at definition and at declaration.

3. Use block comments to document the code throughout functions, particularly where the code is not immediately obvious.

4. Enums and structs should be thoroughly documented.

5. The top of every file should have the following comment block:

```
/**
 * @file [filename]
 *
 * @author [author], [email]
 *
 * @description
 * [description]
 *
 * This program is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program.  If not, see <http://www.gnu.org/licenses/>.
 *
 *
 * DATE      WHO DESCRIPTION
 * ----------------------------------------------------------------------------
 */
```

Every commit must be documented in the changelog at the top of the file. The author and name fields should be populated with the original author of the code. Subsequent edits should be denoted with initials in the changelog.