CanModule - ATLAS Branch 2022

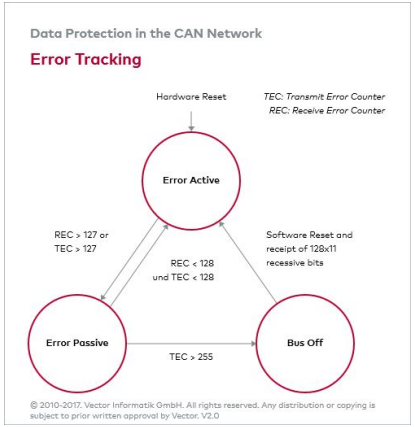
Work done by Piotr N.

Goal and Coverage

- CanModule to be used for CANopen NG server implementation
- Covers only SocketCAN interface under Linux wrt testing as it is only available option for ATLAS, many changes are agnostic to interface model though
- Branch https://github.com/quasar-team/CanModule/tree/pnikiel_canopen
- Difference with master currently: **636 additions** and **799 deletions**.

- Note: the documentation which had been added by Michael previously (not existing before) was extremely appreciated and helpful

Changes: high level description I



- General:
  - Major code cleanup
  - Major improvements of coding quality
- Introduced proper can controller state machine representation
  - Also added textual representation of can_state (needed in many applications,
- Error handling:
  - Coherent handling with respect to above state model (essential to implement FC3.1, FN1.3 of CanOpen NG document)
    - CanModule API originally proposed notification model (e.g. port down, bus off, reception of error frames etc): preserved
    - Improved handling to cover more and avoid leaking of notifications (e.g. recovery finished, leaking of certain state transitions)
    - Corrected wrong data passed previously via notifications
    - Added "query" model in addition to notifications (got merged-back from ATLAS Wiener server)
  - Uniformly converted to use standard exceptions:
    - runtime_error for plenty of possible runtime issues (port can't be opened, serious port issue, …)
    - logic_error for logic errors
    - **note** transitions into ERROR_ACTIVE or any other non-OK CAN state are considered part of life and as such not considered exceptional
  - Added error handling were it was missing or incomplete
- Aliasing rules of select()
  - Found by using CentOs 8's gcc that the SocketCAN use of select() violates the strict aliasing (__restrict__ flags) of select() -> fixed.
  - Could result in broken FDSET.
- Statistics module: removed misuse of stats module for error handling to avoid problems, cleaned up

Changes: high level description II

- removed this in multiple occurrences (*should be converted to std::chrono)*

```
    void setTimeSinceOpened() {
#ifdef _WIN32
        GetSystemTime(&m_dopen);
#else
        gettimeofday( &m_dopen, &m_tz);
#endif
    }
```

- removed nanosleep for std::chrono
- **not** to be done in the public **headers**, moved

```
using namespace std;
using namespace std::chrono;
```

- removed non-trivial implementation from headers
- Reduced code redundancy → applied write(), select() wrappers etc.

Changes: high level description III

- Messages longer than 8 bytes were just sent as 8 bytes, fixed now
- IDs longer than 11 bits were truncated to 11 bits, fixed now
- Rewrote algorithm which maps CAN ports to network interfaces
- Build system: fixed problems and cleaned up, enabled static linking to OPC UA servers

Remarks

- CanModule user API mostly unchanged, except:
  - Exceptions introduced
  - Inclusion of cleaned up headers
  - Port status relocation
  - Minor data type corrections
- Changes affect compatibility with non-SocketCan device modules and those need to be adapted
- Current “reconnection API” (in master) is incompatible with SocketCAN behavior and thus not agnostic to CAN interface model
- Rewrite of configuration parameter parser may be necessary as well (not yet done)
- Further improvements/changes likely to be necessary

- User documentation should be extended at some point covering:
  - Architecture description
  - What can block, what can't block, what may block, what by definition won't block?
  - Performance considerations?
  - Conditions on the callbacks?
  - Are functions reentrant?
  - What is the error handling model? Are exceptions used and/or supposed to be caught?