BTC embedded systems

AUTOSAR (Classic) Standard Library

What is it?

- Set of standardized library functions for **popular mathematical operations**
- Also called **AUTOSAR Math Routines**
- **Rationalize** the resources usage in the ECU (CPU/RAM/ROM)
- Specify **unique** function prototypes and their **intended** functionality
- Available since AUTOSAR Classic 4.0

BTC embedded systems

AUTOSAR (Classic) Standard Library

List of libraries

- **Mfx** Library of Mathematical Fixed-point calculations
- **Mfl** Library of Mathematical Floating-point calculations
- **Ifx** Library of Interpolation functions of Fixed-point
- **Ifl** Library of Interpolation functions of Floating-point
- **Bfx** Library of Bit handling
- **Efx** Library of Extended functions on Fixed point
- **Crc** Library of CRC routines
- **E2E** SW-C End-to-End Communication Protection Library

BTC | embedded systems

AUTOSAR (Classic) Standard Library

Key requirements for the functions

- Can be called from **any** A-SW and B-SW modules
- Shall be operational **before** the ECU Initialization and **during** Shutdown to be used by any related modules
- Shall **not** be calibratable/configurable, always return the same output(s) for the same input(s)
- Can be called **directly** from the caller module, not via RTE interface
- Must be **re-entrant** i.e. no write to a global variable, synchronous and process data only the context of the caller
- Can call another AUTOSAR library function but **not** a third-party one

BTC embedded systems

AUTOSAR (Classic) Standard Library

Implementation

- Implemented by the organization entity with a strong **software integration** role
- **One header-file** declaring all functions of a library (e.g. Mfx.h)
- Ideally routines from the same library group (e.g. Addition, Subtraction, Absolute, ...) in one file, bigger routine in its own file, c-files named with **library pre-fix** (e.g. **Mfx_**xxxx.c)
- Usage of macros #define to implement the function is **not recommended**
- Usage of **AUTOSAR data type** is **highly recommended** to enable platform portability
- Compliance to **MISRA** when implemented in C language

AUTOSAR (Classic) Standard Library

Example

Mfx.h

```
#include "Std_Types.h"

extern uint8 Mfx_Abs_s8_u8( sint8 argin );
extern sint8 Mfx_Abs_s8_s8( sint8 argin);
extern …
```

Mfx_Abs.c

```
#include  "Mfx.h"
//This routine computes the absolute value of a signed value.
uint8 Mfx_Abs_s8_u8( sint8 argin) {
    uint8 ret;
    if (argin < 0) {
        ret = -argin;
    } esle {
        ret = argin;
    }
   return ret;
}
```

BTC embedded systems

AUTOSAR (Classic) Standard Library

Can I integrate a non-standardized library in my AUTOSAR project?

Yes, but:

- **no redundant** operations with the standard functions
- shall **comply** with the AUTOSAR library requirements
- shall be mentioned in SWC Template via the property “**DependencyOnLibrary**”

BTC embedded systems

AUTOSAR (Classic) Standard Library

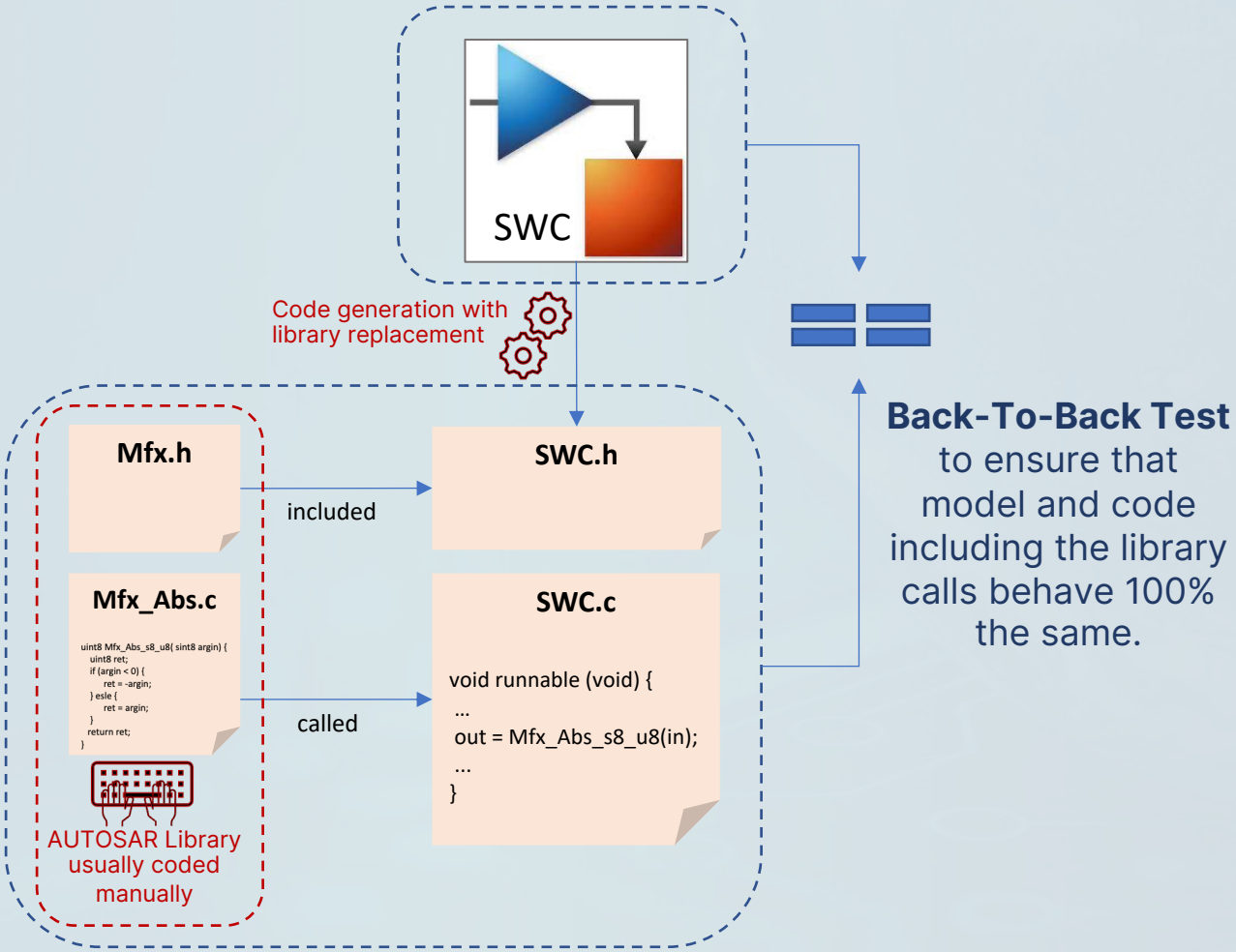
What about Model-based SW Development?

- Code generated from AUTOSAR SWC models can **automatically replace** math operations with calls to the AUTOSAR routines
- The SWC source code **includes** the dependencies to the respective library headers
- The C implementation of the functions needs to be provided for **SIL testing**
- Since the functions are implemented independently from the model, each function shall be **tested upfront** against the corresponding model block to ensure the same behavior

BTC embedded systems

AUTOSAR (Classic) Standard Library

What about Model-based SW Development?



Back-To-Back Test to ensure that model and code including the library calls behave 100% the same.