|  |  |
| --- | --- |
| Date of Review | **Review naming rule** |
| 11/02/2016 | code\_review\_ProtocolCanOpen |

|  |  |  |  |
| --- | --- | --- | --- |
| **Project** | **SW Release** | **SWPM / Dept.** | **SWPQM / Dept.** |
| **TANGELA** | **origin/TANG-33-PDO commit b2a186e** | **Vincent SCESA / Steery** |  |

Distribution List

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Participants / Roles:** | **Persons to be informed:** | | | | |
| **Name:** | **Role** | **Department** | **Name:** | **Role** | **Department** |
| Benoit Robin | Author | Steery Embedded system | see above | SWPM | see above |
| Sebastien Loisel | Reviewer | Steery Embedded system | see above | SWPQM | see above |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

|  |  |
| --- | --- |
| **Scope of Review :** | Full document is reviewed |
| Document is only partly reviewed  (list below the functions which have been reviewed)  Delta between Protocol\_CanOpen.c version 1.99 and 1.100 |

**Review Object(s) :**

|  |  |  |
| --- | --- | --- |
| **Document name** | **Version used for the review** | **version with review comments implemented** |
| Protocol\_CanOpen.c | Commit b2a186e |  |
| Library\_PrivateCanOpenSdoManage.c | Commit b2a186e |  |
| Library\_PrivateXtppCanopen.c | Commit b2a186e |  |
| Library\_PrivateXtppCommands.c | Commit b2a186e |  |

**Supporting Document(s):**

|  |  |
| --- | --- |
| **Document name** | **Version/date** |
| TV\_Protocol\_CanOpen.c |  |
|  |  |
|  |  |

Main Emphasis / Comments:

**Status Doc Review:**

Review finished: Yes  ; No  , Follow up / Date:

1. Module A
   1. Compilation errors/warning/info

List all compiler errors/warnings/info still present dealing with the C module, with completion date or justification.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No.** | **Reference** | **Comments / Actions** | **Classification** | **Responsible person  / Planned date for completion** | **Completion (Name / Date)**  **Optional:**  **CS reference** |
|  |  | none |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  | page | previous page | total | | |
| Sum of remarks |  |  |  | | |
| Sum of errors / risk |  |  |  | | |
| No.: Consecutive number Ref.: Reference to the review object / checklist Comments / : Findings, errors, risks, etc. Actions defined corrective actions | Classification: (E)rror/Risk / (R)emark Responsible person: Participant, responsible for completion  CS Reference: Optional Change Synergy Reference for unclosed items | | | | |

* 1. Quality report dealing with I B&S Coding Guideline.

**Are there remaining MISRA errors?**

No => Ok Ok  Nok

Insert Quality report (excel format) in which is present the analysis, risk and completion date, or justification, even if is empty

**Question C18: Are all message suppression documented?**

Yes => Ok Ok  Nok  N.R.[[1]](#footnote-2)

Please check if every comments like /\*lint …, //lint … and /\* lint … have a reference to the MISRA deviation list and if the deviation is documented.

* 1. Code metrics report.

Check the result of the Code metrics report (sheet Code Metrics (Cyclo) from step450\_all\_qycheck.xls) and justify the remaining directive (need comment, cross-review...). Insert the code metrics report file here.

**Question C16: Are there more than 700 effective lines of code in the module?**

Yes  No

**Question C17: Is there any function with more than 100 effective lines of code?**

Yes  No

**Question C19: Is there any function with a cyclomatic complexity number greater than 15?** Yes  No

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No.** | **Reference** | **Comments / Actions** | **Classification** | **Responsible person  / Planned date for completion** | **Completion (Name / Date)**  **Optional:**  **CS reference** |
|  |  | Protocol\_CanOpen.c |  |  |  |
|  |  |  |  |  |  |
|  | page | previous page | total | | |
| Sum of remarks |  |  |  | | |
| Sum of errors / risk |  |  |  | | |
| No.: Consecutive number Ref.: Reference to the review object / checklist Comments / : Findings, errors, risks, etc. Actions defined corrective actions | Classification: (E)rror/Risk / (R)emark Responsible person: Participant, responsible for completion  CS Reference: Optional Change Synergy Reference for unclosed items | | | | |

* 1. Results of Code Walkthrough or Intensive Inspection

Does the code match with the design specification? yes

Is the code readable, maintainable? yes

Do you detect possible code improvements? no

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No.** | **Reference** | **Comments / Actions** | **Clas-sific-ation** | **Responsible person  / Planned date for completion** | **Completion (Name / Date)**  **Optional:**  **CS reference** |
|  | **ProCan\_Process\_Frames** | Rename **ProCan\_Process\_Frames**  in **ProCan\_Receive\_Frames** |  |  |  |
|  | **ProCan\_SdoManage** | In **ProCan\_SdoManage\_Configure, set** gt\_CfgCanOpen\_SdoManage\_Status to *E\_NOT\_OK in else case* |  |  |  |
|  | **ProCan\_SdoManage\_StartUpload** | **check module is intialized before doing in action** |  |  |  |
|  | **ProCan\_SdoManage\_StartUpload** | Replace  **if** (*E\_OK* == t\_Start\_SdoUpload\_Status)  {  t\_Start\_SdoUpload\_Status = XtppWrapperCanOpen\_Write(&s\_canopen\_frame, &e\_can\_status);  } **else** {  /\* Internal driver error \*/  }  by  t\_Start\_SdoUpload\_Status |= XtppWrapperCanOpen\_Write(&s\_canopen\_frame, &e\_can\_status); |  |  |  |
|  | **Global** | replace *E\_OK* ==  by *E\_NOT\_OK* != |  |  |  |
|  | **ProCan\_SdoManage\_ProcessFrame** | Rename **ProCan\_SdoManage\_ProcessFrame**  in **ProCan\_SdoManage\_ReceiveFrame** |  |  |  |
|  | **Library\_PrivateXtppCanopen** | Replace PrivateXtppCom\_Set\_Sdo by PrivateXtppCom\_Send\_Sdo |  |  |  |
|  | **Library\_PrivateXtppCanopen** | Replace PrivateXtppCom\_Get\_Sdo by PrivateXtppCom\_Receive\_Sdo |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  | **General** | Replace all  } **else** {  by  }  **else**  { |  |  |  |
|  | **General** | Replace all  **if** ((*E\_OK* == a) &&  (NULL\_PTR != pe\_b) &&  (u8\_c < gu8\_c) &&  (*E\_SDO\_IDLE* == gas\_c ))  by  **if** ((*E\_OK* == a) &&  (NULL\_PTR != pe\_b) &&  (u8\_c < gu8\_c) &&  (*E\_SDO\_IDLE* == gas\_c )) |  |  |  |
|  | **General** | Replace  Std\_ReturnType **ProCan\_SdoManage\_StartUpload**(**const** uint8 u8\_canopen\_node\_idx, **const** uint16 u16\_canopen\_object\_idx, **const** uint8 u8\_canopen\_object\_subidx, te\_CanopenStatus \* **const** pe\_canopen\_exchange\_status)  by  Std\_ReturnType **ProCan\_SdoManage\_StartUpload**(  **const** uint8 u8\_canopen\_node\_idx,  **const** uint16 u16\_canopen\_object\_idx,  **const** uint8 u8\_canopen\_object\_subidx, te\_CanopenStatus \*  **const** pe\_canopen\_exchange\_status) |  |  |  |
|  | page | previous page | total | | |
| Sum of remarks |  |  | 12 | | |
| Sum of errors / risk |  |  | 4 | | |
| No.: Consecutive number Ref.: Reference to the review object / checklist Comments / : Findings, errors, risks, etc. ctions defined corrective actions | Classification: (E)rror/Risk / (R)emark Responsible person: Participant, responsible for completion  CS Reference: Optional Change Synergy Reference for unclosed items | | | | |

* 1. Result from code review check list

**Checklist for Coding Reviews**

The following checklist covers the remaining rules from the "SW C Coding Rules Method" which can not be covered by the PC-Lint or Code-check tools.

The checks have been grouped in two categories:

* **Required coding rules**
* **Advisory coding rules (these rules are mandatory only for safety critical modules)**
  + 1. Required coding rules

Comments must be added when the result is *Not OK*

| **No.** | **Category** | **Description** | **OK / NOK / N.R.** | **Comments / Actions** | **Responsible / Planned date** | **Completion**  **(Name / Date)** |
| --- | --- | --- | --- | --- | --- | --- |
| **C1** | Architecture | Is the module structured in a body (.c) and a header part (.h) based on the official templates (available on the SWP Sharepoint site), in order to comply with the coding architecture? ***Yes => OK*** *Standard comment header, mechanisms to avoid multiple inclusions, standard files to include, order of the declaring sections (ram/rom data, word/byte data, macros...), etc...* | **OK** |  |  |  |
| **C4** | Architecture | Are all automatic variables defined inside functions initialized in the definition?  ***Yes => OK*** | **NR** |  |  |  |
| **C5** | Maintenance / Readability | Are in the project logical source lines which exceed the maximum length accepted by the used compiler?  ***No => OK*** | **NR** |  |  |  |
| **C132.a** | Validity / Robustness | Are all shared variable used in different preemptive tasks (interrupt, preemptive tasks from the scheduler) or hardware register declared as *volatile*? (To avoid unexpected optimization) ***Yes => OK***  *If volatile is not used for good reason, put Nok and justify.* | **OK** |  |  |  |
| **C132.b** | Validity / Robustness | Is there any direct access to HW registers?  ***No=>OK*** | **NR** | **Basic Software !!!** |  |  |
| **C134** | Validity / Robustness | Is the exit condition of each loop robust? Does each loop waiting for an event have an alternate escape mechanism? (time-out, ...) ***Yes => OK*** *E.g. with while (\*ub8\_pt++ != END\_BYTE ) {...} an endless loop may occur if memory pointed by ub8\_pt is not as expected* | **OK** |  |  |  |
| **C135** | Maintenance / Readability | Is there any numeric constant in the source not used through a macro or an *enum*? ***No => OK***  *Deviation: '0' and '1' are allowed to initialize non-boolean variables (e.g., loop counters). '1' is also allowed in the context where it means "one more" or "one less" (e.g., len-1 or v[i+1]).* | **OK** | **Remark: 8, 16, 24 are allowed for shift operations. Some register masks also (0x80, ...).** |  |  |
| **C138** | Optimization | For each function, is the used stack size, through function parameters and local variables, acceptable?***Yes => OK*** | **OK** |  |  |  |
| **C141** | Optimization / Readability | Are all conditional expressions (“*?:*”) used with a good reason (code is faster, smaller or easier to read – reason explained in the source code)? ***Yes => OK***  Are there any sub-expressions longer than 1 line or not parenthesized? ***No => OK*** | **OK** |  |  |  |
| **C144** | Validity / Robustness | Are all critical inputs and data checked for valid boundary values (EEPROM data should be considered as critical) ?  ***Yes => OK*** | **OK** |  |  |  |
| **C146** | Validity / Robustness | Are the TRUE and FALSE macros used in comparisons? ***No => OK***  *E.g.* if (running == TRUE) *can fail if* running *has other value than 1. Use* if (running) *instead.* | **OK** |  |  |  |
| **M1.3** | Environment | If C code contains assembly language, is it for a whole C function? (and not only for a part of the C function)  ***Yes => OK***  *(Assembly wrapped inside a BSW macro is OK)* | **N.R** |  |  |  |
| **M3.4** | Documentation | Are all uses of *#pragma* directives explained by comments? ***Yes => OK*** | **N.R** |  |  |  |
| **M3.5** | Documentation | Is there any bit fields access of a larger data type which relies on the way that the bit fields are stored?  ***No => OK***  *For example splitting a word in high and low bytes based on struct unions. Only the usage of standard T\_FLAG8, T\_FLAG16 is allowed for this purposes.* | **OK** |  |  |  |
| **M8.7** | Validity / Robustness | Is there any object with global (file scope or global software) declaration un-justified? ***No => OK*** | **OK** |  |  |  |
| **M8.12** | Declarations and definitions | Do all external array variables have explicit size, by definitions or initialization? ***Yes => OK***  *External array definitions like T\_UBYTE a[]; are not allowed.* | **OK** |  |  |  |
| **M12.5** | Expressions | Are all operands of *&&* or *||* operatorsexpressions primary expressions? ***Yes => OK*** *(Primary expressions are identifiers, constants, string literals, or parenthesized expressions)* | **OK** |  |  |  |
| **M13.5** | Control statement expression | Are all expressions within the for statement concerning the loop control? ***Yes => OK*** | **OK** |  |  |  |
| **M13.6** | Control statement expression | In *for* loops, is the numeric variable used for iteration counting modified in the body of the loop? ***No => OK*** | **OK** |  |  |  |
| **M14.7** | Control flow | Is there any function with more than one exit point?  ***No => OK*** | **OK** |  |  |  |
| **M15.5** | Switch statements | Is there any switch statement with less than three case clauses? ***No => OK*** | **NR** |  |  |  |
| **M16.2** | Functions | Is there any recursivity (direct or indirect)? ***No => OK*** | **OK** |  |  |  |
| **M16.4** | Functions | Are parameter names used in the declaration and definition of a function identical? ***Yes => OK*** | **OK** |  |  |  |
| **M16.9** | Functions | Is there any function identifier without the & operator or parenthesized paramer list? ***No => OK***  *Ensure that all uses of function pointers (function names without parenthesis) are really intended.* | **OK** |  |  |  |
| **M17.1** | Pointers and arrays | Is there any pointer arithmetic applied to pointers which do not address and array or an array element?  ***No => OK***  *Pointer arithmetic is allowed only for array indexing purposes.* | **OK** |  |  |  |
| **M17.2** | Pointers and arrays | Is there any pointer substraction applied to pointers which do not address elements of the same array?***No => OK*** | **OK** |  |  |  |
| **M17.4** | Pointers and arrays | Is there used array indexing other than pointer arithmetic? ***No => OK*** | **OK** |  |  |  |
| **M18.2** | Structures and unions | Is there any assignment or memory copy operation of overlapping objects or memory areas/regions?  ***No => OK***  *E.g.: memcpy( &array[1], &array[4], 8 ); /\* destination and source overlap \*/* | **OK** |  |  |  |
| **M18.3** | Structures and unions | Is there any memory storage used to store some data at one time, and used to store a different type of data at another time? ***No => OK***  *E.g. The same variable used for the different purposes.* | **OK** |  |  |  |
| **M19.4** | Preprocessing directives | Is there any macro defined for other purposes than a braced initializer, a constant, a parenthesized expression, a type qualifier, storage class specifier or a do-while-zero construct? ***No => OK***  *For example to define type (instead of using typedef) or to replace C language keywords (e.g. #define IF if)* | **OK** |  |  |  |
| **M19.16** | Preprocessing directives | Are all preprocessor directives sintactically correct even the excluded ones? ***Yes => OK***  *Pay attention to the #else, #elif and #endif which might be misspelled and not reported by the compiler.* | **OK** |  |  |  |
| **M19.17** | Preprocessing directives | Is there any #else, #elif and #endif which do not reside in the same file as the corresponding #if or #ifdef directive?  ***No => OK*** | **OK** |  |  |  |
|  | Scheduler | If any variable is shared (write/read access) by more than one task (e.g. interrupt routine + background task), is it justified? ***No => OK***  If yes:  Is there a mechanism to avoid a read during a change of the variable? ***Yes => OK***  Is there a mechanism to avoid data modification during its treatment? ***Yes => OK*** | **OK** |  |  |  |
|  | Scheduler | In case of global variable (shared or not shared) used in reentrant function (reentrance raised by an ISR), is there a mechanism to avoid data modification during its treatment? ***Yes => OK*** | **OK** |  |  |  |
|  |  | *List not finished: to be completed from project's experience...* |  |  |  |  |

* + 1. Advisory coding rules

Comments must be added when the result is *Not OK*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No.** | **Category** | **Description** | **OK / NOK / N.R.** | **Comments / Actions** | **Responsible / Planned date** | **Completion**  **(Name / Date)** |
| **C2** | Arhitecture | Is included in .h or .c file, only all .h files that declare symbols used by your file? ***Yes => OK*** | **OK** |  |  |  |
| **C3** | Maintenance / Readability | Is there any function without a comment header indicating the purpose of the function and the in/out data?  ***No => OK*** | **OK** |  |  |  |
| **C130** | Maintenance / Readability | Are enums used when defining groups of related integer constants instead of a series of #defines?  ***Yes => OK***  Are state variables defined as enumerations?  ***Yes => OK***  Are assignment and/or comparison of the state variable made with/trough the enumeration constants?  ***Yes => OK***  Are case labels in a switch statement defined via enums (exception case with #defines)?  ***Yes => OK*** | **NR** |  |  |  |
| **C131.b** | Architecture / Robustness | Only for projects using SWP: do all global variables have initializers? ***Yes => OK***  *Do not confuse variable initializer (e.g., T\_UBYTE rsb\_Val = SYS\_OK;) with an assignment in a special Init function.*  *Please check all global variables definition for a default initializer.* | **OK** |  |  |  |
| **C137** | Maintenance / Readability | Are all complex declarations commented to show their use? ***Yes => OK*** | **OK** |  |  |  |
| **M2.4** | Maintenance / Readability | Is there any code section commented out? ***No => OK*** | **OK** |  |  |  |
| **M12.13** | Expressions | Is any increment (++) or decrement (--) operator used with other operators in an expression? ***No => OK*** | **OK** |  |  |  |
| **M17.5** | Pointers and arrays | Is there any pointer with more than 2 levels of indirection or arrays with more than 2 dimensions? ***No => OK*** | **OK** |  |  |  |
| **M19.7** | Preprocessing directives | Is there any function-like macro? ***No => OK***  *Functions are preferred instead.* | **OK** |  |  |  |
|  |  | *List not finished: to be completed from project's experience...* |  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **History of the review minutes template** | | | |
| **Version** | **Maturity/Date** | **Author** | **Description** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

1. Can be checked only if there is no message suppression, otherwise the Ok / Nok boxes must be used. [↑](#footnote-ref-2)