Advanced parameter

Project TFS path: $/Firmware-SVI II AP FF/FD-SW

Project changeset version: 37075

Starting point: MSB\_ExtendU8toU16

NO LONGER APPLICABLE. A complete rewrite followed.

Project TFS path: $/Firmware-SVI II AP FF/FD-SW

Project changeset version: 47046

Starting point: Use of advanced key

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Containing file | | FD-SW/target/appl/fdev/src/app\_ptb.c | Review Date | Reviewer |
| Function or data | | Read\_handler\_PTB | 15-05-19 | AK |
|  | | | | |
| TFS wit | 24455 | MN\_ASSERT is not a good means of checking the input parameters | | |
| Impact | | Critical bug | | |
| Resolution | |  | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Containing file | | FD-SW/target/appl/fdev/inc/mn\_advanced.h | Review Date | Reviewer |
| Function or data | | USIGNED\_CHAR\_NUMBER and prototypes | 15-05-19 | AK |
|  | | | | |
| TFS wit | 24456 | CONST\_ASSERT(USIGNED\_CHAR\_NUMBER==UCHAR\_MAX+1);  Could be good if the code relies on 8-bit byte | | |
| Impact | | Suggestion | | |
| Resolution | |  | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | | | |
| Containing file | | FD-SW/target/appl/fdev/src/mn\_advanced.c | Review Date | Reviewer |
| Function or data | | PRAMETER\_STANDARD\_DEFAULT | 15-05-19 | AK |
| TFS wit |  | Typo? | | |
| Impact | |  | | |
| Resolution | | Suggestion | | |
| TFS wit |  | Where are the defaults coming from? E.g. 0x180bfe00u? If it is a result of calculation, it is best to compute it at runtime using the exact same algorithm that is used for offline calculation | | |
| Impact | |  | | |
| Resolution | | Maintainability | | |
|  | | | | |
| Function or data | | mn\_GetBit | 15-05-19 | AK |
| TFS wit | 24459 | How is it better than util\_GetBit() of bitutils.h?  If demonstrably needed, the file is a bad home for a general utility. | | |
| Impact | | Reuse, quality | | |
| Resolution | |  | | |
| TFS wit | 24455 | Assert is a very poor choice for parameters checking; you have no control what comes in. Needs justification. | | |
| Impact | | quality | | |
| Resolution | |  | | |
| TFS wit |  | u8 uByte, uByteIndex;  Multiple variables in a single declaration | | |
| Impact | | MISRA, coding standard | | |
| Resolution | |  | | |
|  | | | | |
| Function or data | | mn\_GetByte | 15-05-19 | AK |
| TFS wit |  | How is it better than  MN\_INLINE mn\_GetByte(u8 byteIndex, const void \*Array)  { const u8 \*p = Array;  return p[byteIndex];  } ? | | |
| Impact | | quality | | |
| Resolution | |  | | |
| TFS wit | 24455 | MN\_ASSERT for a static function will prevent lint from checking; it is also a poor choice for input validation. Length is not checked anyway | | |
| Impact | | Quality, efficiency | | |
| Resolution | |  | | |
|  | | | | |
| Function or data | | ffcheck\_AlertEnableChange | 15-05-19 | AK |
| TFS wit |  | Why is it called from ffres\_restart\_factory\_defaults() (mn\_factory\_defaults.c), just before reset? What is the expected and actual effect? | | |
| Impact | | Suspected junk with no effect | | |
| Resolution | |  | | |
| TFS wit | 24459 | MN\_ASSERT(ptb\_param\_readonly[i].ptb\_index < USIGNED\_CHAR\_NUMBER);  What does this mean? Was it meant to be PARA\_ADV\_READ\_BYTES or something?  Anyway, assert is a bad choice for input vaidation | | |
| Impact | | Suspected junk with no effect | | |
| Resolution | |  | | |
| TFS wit |  | Why is there special handling of IP\_DRIVE\_CURRENT\_{HI,LO}\_ALERT? What are the magic numbers for subindex? Can this be part of the table? | | |
| Impact | | Quality; unclear/not reviewable | | |
| Resolution | |  | | |
| TFS wit |  | Calling fbs\_write\_param (should be \_loc?) in a loop is atrociously inefficient. It is less pronounced now that FRAM access is delayed; may be a disaster if/when we do it inline.  Should direct assignment to TB parameters be enough? What would be missing? | | |
| Impact | | Quality; efficiency | | |
| Resolution | |  | | |
|  | | | | |
| Function or data | | ffcheck\_WriteAdvanced | 15-05-19 | AK |
| TFS wit | 24460 | The function violates several patterns:   1. It is called from pre-check switch of the write handler but has side effects (actually, does all the work) 2. It modifies the object pointed to p\_write 3. It uses API not documented to allow writing a parameter on itself   I think a much better design is   1. Call it from execution switch of the write handler 2. Always call it to feed input data to sAdvancedBuffer 3. Write out sAdvancedBuffer and copy it to TB::advanced | | |
| Impact | | Very fragile design using undocumented API; I don’t know if it always works or will work as intended | | |
| Resolution | |  | | |
| TFS wit | 24460 | The use in appl\_ptb.c ignores an error in a partial write. It is above pay grade of the write handler who is not supposed to know the details and is the responsibility of ffcheck\_WriteAdvanced.  Also, why only errors in partial writes (0 != p\_write->subindex) are ignored? | | |
| Impact | | Modularity, quality | | |
| Resolution | |  | | |
|  | | | | |
| Function or data | | pFK\_1, pFK\_2 | 15-05-20 | AK |
| TFS wit |  | Why do they need to be const variables, not just #defines?  Any comment on chosen values? | | |
| Impact | | Quality, efficiency | | |
| Resolution | |  | | |
|  | | | | |
| Function or data | | sAdvancedBuffer, uAdvBufferFlag | 15-05-20 | AK |
| TFS wit |  | sAdvancedBuffer, uAdvBufferFlag can persist for arbitrarily long time and are not protected by CheckWord. | | |
| Impact | | Quality. It is true that FFP code generally lacks checksums but all new code shall use protection (and testing) | | |
| Resolution | |  | | |
|  | | | | |
| Function or data | | calcFrst, calcScnd | 15-05-20 | AK |
| TFS wit |  | A lot of code duplication. A single function that takes loop range as an argument would be much better.  Also, doing calculation on pairs is more efficient than choosing between even/odd numbers | | |
| Impact | | Quality, maintainability | | |
| Resolution | |  | | |
|  | | | | |
| Function or data | | PARA\_... and CAL\_... constants | 15-05-20 | AK |
| TFS wit | 24456 | Many are not independent, e.g. PARA\_BIT\_IN\_BYTE is CHAR\_BIT, PARA\_BIT\_IN\_U16 is 2U\* PARA\_BIT\_IN\_BYTE etc.  CONST\_ASSERTs of machine sizes (e.g. CHAR\_BIT==8) are welcome. | | |
| Impact | | Quality, maintainability | | |
| Resolution | |  | | |