Data Path: (Setpoint sent to APP) to XD\_FSTATE

Project TFS path: $/FW SVI FF Releases/Release1/FD-SW

Project changeset version: 34719

Starting point: FLOAT\_S sp object in mnipc\_SendSpAndMode <- FStateData <- fbs execution

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Containing file | | | FD-SW\target\appl\fdev\src\mntbspff.c | Review Date | Reviewer |
| Function or data | | | FLOAT\_S sp object in mnipc\_SendSpAndMode | 14-02-26 | AK |
|  | | | | | |
| TFS wit |  | | NO FINDINGS w.r.t. use of ff\_fstate\_ForceFstate\_SP | | |
| Impact | | |  | | |
| Resolution | | |  | | |
|  | | | | | |
| Containing file | | | FD-SW\target\appl\fdev\src\mn\_fstate.c | Review Date | Reviewer |
| Function or data | | | ff\_fstate\_ForceFstate\_SP | 14-04-01 | AK |
| TFS wit | |  | Detailed comment badly needed to explain why it doesn’t matter if p\_PTB->actual\_position.value is not GOOD and/or not yet received. (Hint: it is a bug! Last\_SP\_Set will NOT self-correct when GOOD actual\_position received because the flag LastSP\_Set is already set) | | |
| Impact | | | If fstate is activated, the valve may move to bogus position instead of staying where it is. | | |
| Resolution | | | DO NOT set LastSP\_Set unless status of actual\_position is GOOD. On a higher level though, it doesn’t matter because if actual\_position is not GOOD, APP is already in failsafe. | | |
|  | | | | | |
| Function or data | | | ff\_fstate\_Background | 14-02-26 | AK |
| TFS wit | |  | NO FINDINGS | | |
| Impact | | |  | | |
| Resolution | | |  | | |
|  | | | | | |
| Function or data | | | fstate\_TooEarlyToFailSetpoint | 14-02-26 | AK |
| TFS wit | | 20672 | Hardcoded STARTUP\_WRONG\_SP\_COUNT is too fragile to rely on APP configuration and time alone. | | |
| Impact | | | APP time slot waiting for setpoint may be missed, and the valve may de-energize for a short period. | | |
| Resolution | | | Instead, we should use the same logic as in e.g. tb\_PositionSp to wait for the second execution of the *corresponding* output block first, with timing as a backup in case the output block is in OOS. | | |
|  | | | | | |
| Function or data | | | ff\_fstate\_CheckTimeToActivate | 14-02-26 | AK |
| TFS wit | |  | NO FINDINGS | | |
| Impact | | |  | | |
| Resolution | | |  | | |
|  | | | | | |
| Function or data | | | ff\_fstate\_Execute\_fromFB | 14-02-26 | AK |
| TFS wit | | 20671 | NO SIGNIFICANT FINDINGS  But inefficient and redundant calculation. Pieces should be plugged directly in tb\_PositionSp, tb\_IntSp, tb\_BoolSp. As written, called whenever Execute\_PTB is called by whatever FB | | |
| Impact | | |  | | |
| Resolution | | |  | | |
|  | | | | | |
| Function or data | | | ff\_fstate\_StatusOK, FStateData.Active | 14-02-26 | AK |
| TFS wit | | 20670 | If “Initiate Failed State” is requested, the implementation merely stops updating the timestamp, waiting for setpoint timeout to expire. So if timeout is 5 hours, Failed State will not be activated for 5 hours since request. | | |
| Impact | | | Request to “Initiate Failed State” may take very long time to honor. | | |
| Resolution | | | This is left to the discretion of TB, so what is needed is clear documentation | | |
|  | | | | | |
| Function or data | | | ff\_fstate\_StoreLastSP, FStateData. LastSP | 14-02-27 | AK |
| TFS wit | |  | NO FINDINGS; call point OK | | |
| Impact | | |  | | |
| Resolution | | |  | | |
|  | | | | | |