Software Safety Tutorial
Student Handout
NSWCDD-PN-14-00391

SOFTWARE CRITICALITY
<table>
<thead>
<tr>
<th>2.a</th>
<th>2.b</th>
<th>2.c</th>
<th>2.d</th>
<th>2.e</th>
<th>2.f</th>
<th>2.g</th>
<th>2.h</th>
<th>2.i</th>
</tr>
</thead>
<tbody>
<tr>
<td>Safety-Function</td>
<td>Safety-Function (Portion)</td>
<td>CSCI</td>
<td>Software Function</td>
<td>Related Hazard Number from VIPPS Worksheet</td>
<td>Level of Mishap Severity for the Identified Hazard</td>
<td>Software Control Category</td>
<td>SwCl</td>
<td>Level of Rigor Tasks Required</td>
</tr>
<tr>
<td><strong>SSF #6:</strong> Verification of Gun activation / deactivate, and movement Commands</td>
<td>Verify Activate (Fire on fire cmd only if GSS ADDRESS in fire cmd matches)</td>
<td></td>
<td>Gun Controller</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

NSWCDD-PN-14-00391 is approved for Distribution Statement A: Approved for Public Release; distribution is unlimited.
<table>
<thead>
<tr>
<th>Hazard ID</th>
<th>Phase of Operation</th>
<th>State/Mode</th>
<th>Hazard Description</th>
<th>Causal Factors</th>
<th>Mishap</th>
<th>Effects</th>
<th>Hazard Controls</th>
<th>MRI</th>
<th>System</th>
<th>Subsystem</th>
<th>Component(s)</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>132</td>
<td>Operational</td>
<td>Tactical</td>
<td>Upon receipt of GSS Electronics overtemperature, the GCC fails to halt processing after 120 seconds (if C2 does not safe/deselect)</td>
<td>GCC fails to activate 120 second timer or terminate processing in response to 120 second timeout</td>
<td>Electronics overload, causing circuit board damage</td>
<td>Equipment damage</td>
<td>The GSS shall continuously monitor for a GSS over-temperature condition [electronics]. If a GSS over-temperature condition is detected and continues to exist 1.0 ± 0.026 seconds after initial detection, the GSS shall update the status register [indicating GSS Over-temperature] for transfer to the C2 within the ID 12, and continue operation until discontinued by the C2</td>
<td>3D</td>
<td>VIPPS</td>
<td>Gun Subsystem</td>
<td>Gun Control Computer</td>
<td>SSF 20</td>
</tr>
<tr>
<td>188</td>
<td>Operational</td>
<td>Tactical</td>
<td>Upon receipt of EF23 (Safe / Deselect), the GSS fails to remove power to the gun motor causing premature motor failure</td>
<td>Software failure in the Gun Control Computer</td>
<td>Premature motor failure</td>
<td>Equipment damage</td>
<td>None</td>
<td>3D</td>
<td>VIPPS</td>
<td>Gun Subsystem</td>
<td>Gun Control Computer</td>
<td>SSF 20</td>
</tr>
<tr>
<td>299</td>
<td>Operation</td>
<td>Tactical</td>
<td>Fire command processed by the wrong gun results in accidental firing by the (wrong) gun</td>
<td>The GCC software fails to validate firing commands intended for the gun, and processes commands intended for a different gun</td>
<td>Engage Friendly/non-hostile target</td>
<td>Personnel injury or death</td>
<td>[4 6.5 h] VIPPS [GSS] software shall prevent the lethal effector from being fired without a proper command and [C2] shall provide an alarm to the operator when an attempt is made to fire the lethal effector without a proper command. If the GSS receives an EF command with the GSS Address signal set to an illegal value, the GSS shall discard the command and transmit an Illegal Message (GSS-C2-ID04) to the C2 with the Malfunction Type signal set to 5 (Wrong GSS Address)</td>
<td>1D</td>
<td>VIPPS</td>
<td>Gun Subsystem</td>
<td>Gun Control Computer</td>
<td>SSF 5</td>
</tr>
<tr>
<td>310</td>
<td>Operation</td>
<td>Test</td>
<td>GSS fails to move at reduced rate of movement, moving the gun at “normal” rates when “reduced speed” is ordered, leading to accidental impact of gun with personnel</td>
<td>The GCC software fails to validate ‘rate of gun movement’ before commending gun to move at normal speed.</td>
<td>Inadvertent physical contact of gun with personnel</td>
<td>Personnel injury or death</td>
<td>Upon receipt of gun pointing data (EF 37), the GSS shall slew the gun barrel to the pointing location (relative azimuth and elevation) as designated in the message. The GSS shall move the gun at normal speed (6 m/s per degree) or reduced speed (80 m/s per degree) as specified in the message</td>
<td>1D</td>
<td>VIPPS</td>
<td>Gun Subsystem</td>
<td>Gun Control Computer</td>
<td>SSF 5</td>
</tr>
</tbody>
</table>

VIPPS is a Mock System used for training purposes only.
**VIPPS Functional Flow Diagram**

**Use for Column 2.g**

### OPERATOR / C2 LEVEL

- Activate
  - (Fire on fire cmd only if GSS ADDRESS” in fire cmd matches)

### GUN SUBSYSTEM LEVEL

- Operate Gun Subsystem
- Designate Gun Subsystem
- Position Gun Subsystem

### GUN SOFTWARE FUNCTIONAL LEVEL

- Process EF_ID Functional
  - **Process EF_ID FUNCTIONAL DESCRIPTION**
    - Process EF_ID function performs initial processing of gun command messages by verifying message transfer using a Cyclic Redundancy Check (CRC) then sending the message to the Validate Msg Function.

- Validate MSG Functional
  - **VALIDATE MSG FUNCTIONAL DESCRIPTION**
    - Validate MSG function checks that each message is valid by evaluating content for assigned Gun ID, boundary conditions, and message sequence based on last message received. If the message is validated, the appropriate action function is called based on message content (e.g., Select Gun, Deactivate Gun, Fire Command, Master Reset, BIT On, Activate Laser Aiming Device, De-Activate Laser Aiming Device or Move Gun).

### OTHER

- VIPPS is a Mock System used for training purposes only.
VIPPS Event Diagram
Use for Column 2.g

Process EF_ID verifies message content including a valid Gun ID

Fire gun with incorrect Gun Address

Wrong Gun Address passes verification checks (as valid Fire Command EF)

OR

Gun ID corrupted, valid, and undetected (C2 or GCC)

Gun ID set correctly for different gun

Gun ID set incorrectly for intended gun

OR

Gun ID set correctly for intended gun

CRC Failure (C2 or GCC)

Failure of operator

Failure of C2

AND

Failure of gun software to validate EF gun address against intended gun address for match

OR

Coding Error

Incorrect designation in gun specific datafile (e.g., wrong gun number in adaptation data)

Validate MSG ensures Gun ID correct for intended Gun

VIPPS is a Mock System used for training purposes only
## MIL-STD-882E Software Control Categories
### Use for Column 2.g

<table>
<thead>
<tr>
<th>Level</th>
<th>Name</th>
<th>Description</th>
<th>Or</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Autonomous (AT)</td>
<td>Software functionality that exercises autonomous control authority over potentially safety-significant hardware systems, subsystems, or components without the possibility of predetermined safe detection and intervention by a control entity to preclude the occurrence of a mishap or hazard. (This definition includes complex system/software functionality with multiple subsystems, interacting parallel processors, multiple interfaces, and safety-critical functions that are time critical.)</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Semi-Autonomous (SAT)</td>
<td>Software functionality that exercises control authority over potentially safety-significant hardware systems, subsystems, or components, allowing time for predetermined safe detection and intervention by independent safety mechanisms to mitigate or control the mishap or hazard. (This definition includes the control of moderately complex system/software functionality, no parallel processing, or few interfaces, but other safety systems/mechanisms can partially mitigate. System and software fault detection and annunciation notifies the control entity of the need for required safety actions.) Software item that displays safety-significant information requiring immediate operator entity to execute a predetermined action for mitigation or control over a mishap or hazard. Software exception, failure, fault, or delay will allow, or fail to prevent, mishap occurrence. (This definition assumes that the safety-critical display information may be time critical, but the time available does not exceed the time required for adequate control entity response and hazard control.)</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Redundant Fault Tolerant (RFT)</td>
<td>Software functionality that issues commands over safety significant hardware systems, subsystems, or components requiring a control entity to complete the command function. The system detection and functional reaction includes redundant, independent fault tolerant mechanisms for each defined hazardous condition. (This definition assumes that there is adequate fault detection, annunciation,, and system recovery to prevent the hazard occurrence if software fails, malfunctions, or degrades. There are redundant sources of safety-significant information, and mitigating functionality can respond within any time-critical period.) Software that generates information of a safety-critical nature used to make critical decisions. The system includes several redundant, independent fault tolerant mechanisms for each hazardous condition, detection, and display.</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Influential</td>
<td>Software generates information of a safety-related nature used to make decisions by the operator, but does not require operator action to avoid a mishap.</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>No Safety Impact (NSI)</td>
<td>Software functionality that does not possess command or control authority over safety-significant hardware systems, subsystems, or components and does not provide safety-significant information. Software does not provide safety-significant or time sensitive data or information that requires control entity interaction. Software does not transport or resolve communication of safety-significant or time sensitive data.</td>
<td></td>
</tr>
<tr>
<td>Level</td>
<td>Name</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>-------</td>
<td>-----------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
</tbody>
</table>
| 1     | Autonomous (AT)                   | • Autonomous CONTROL over safety hardware systems, and  
• No possibility of detection and intervention by control entity  
Software systems, or control entity to control or influence with multiple interacting parallel processors, interacting parallel processors, multiple interfaces, and safety-critical functions that are time critical. |
| 2     | Semi-Autonomous (SAT)             | • CONTROL over safety hardware systems, and  
• Time for detection and intervention by INDEPENDENT safety mechanism  
Software systems, or control entity to control or influence, no redundancy, no independent fault tolerant mechanisms. |
|       |                                   | • SW that displays safety-significant data for immediate predetermined operator action to prevent mishap, or  
• The SW fault or delay will allow, or fail to prevent, the mishap  
Software items that display safety-critical information requiring immediate operator action to execute a predetermined action for mitigation or control over a mishap or hazard. Software exception, failure, fault, or delay will allow, or fail to prevent, mishap occurrence. (This definition assumes that the safety-critical display information may be time critical, but the time available does not exceed the time required for adequate control entity response and hazard control.) |
| 3     | Redundant Fault Tolerant (RFT)    | • Issues commands over safety hardware system, and  
• Requires control entity to complete command function, and  
• Includes REDUNDANT, INDEPENDENT fault tolerant mechanisms |
|       |                                   | • Generates information of safety-critical nature to make decisions, and  
• System includes SEVERAL REDUNDANT, INDEPENDENT fault tolerant mechanisms |
| 4     | Influential                       | • Generates information of safety-related nature to make decisions by the operator, and  
• Does not require operator action to avoid mishap |
| 5     | No Safety Impact (NSI)            | Software functionality that does not possess command or control authority over safety-critical hardware systems, and does not provide safety-critical information. Software does not provide safety-critical or time-sensitive data or information that requires control entity interaction. Software does not transport or resolve communication of safety-significant or time-sensitive data.
MIL-STD-882E Software Criticality Index
Use for Column 2.h and 2.i

For Gun Software Function: Validate.MSG

<table>
<thead>
<tr>
<th>Severity Category</th>
<th>Software Control Category</th>
<th>Catastrophic (1)</th>
<th>Critical (2)</th>
<th>Marginal (3)</th>
<th>Negligible (4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SwCI 1</td>
<td>SwCI 1</td>
<td>SwCI 3</td>
<td>SwCI 4</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>SwCI 1</td>
<td>SwCI 2</td>
<td>SwCI 3</td>
<td>SwCI 4</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>SwCI 2</td>
<td>SwCI 3</td>
<td>SwCI 4</td>
<td>SwCI 4</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>SwCI 3</td>
<td>SwCI 4</td>
<td>SwCI 4</td>
<td>SwCI 4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>SwCI 5</td>
<td>SwCI 5</td>
<td>SwCI 5</td>
<td>SwCI 5</td>
<td></td>
</tr>
</tbody>
</table>

VIPPS is a Mock System used for training purposes only

For Gun Software Function: Validate_MSG

<table>
<thead>
<tr>
<th>SwCI</th>
<th>Level of Rigor</th>
</tr>
</thead>
<tbody>
<tr>
<td>SwCI 1</td>
<td>Program shall perform analysis of requirements, architecture, design, and code; and conduct in-depth safety-specific testing.</td>
</tr>
<tr>
<td>SwCI 2</td>
<td>Program shall perform analysis of requirements, architecture, and design; and conduct in-depth safety-specific testing.</td>
</tr>
<tr>
<td>SwCI 3</td>
<td>Program shall perform analysis of requirements and architecture, and conduct in-depth safety-specific testing.</td>
</tr>
<tr>
<td>SwCI 4</td>
<td>Program shall conduct safety-specific testing.</td>
</tr>
<tr>
<td>SwCI 5</td>
<td>Once assessed by safety engineering as Not Safety, then no safety specific analysis or verification is required.</td>
</tr>
</tbody>
</table>
SOFTWARE SAFETY TUTORIAL STUDENT HANDOUT

ARCHITECTURAL AND DESIGN ANALYSIS
Architecture and Design Analysis

- Task 1: Identify causal factors using SW generic requirements at the architecture level
  - Review AOP-52 definitions and JSSSEH Generic Requirements (E.8.5, E.3.13)
  - Review architectural data against current hazards and functional hazard analysis
  - Use the SSF-6 Interface and Function Architecture Diagram to identify and record causal factors

- Task 2: Identify causal factors using previous architecture hazard analysis at the design level
  - Compare SSF-6 Control Flow Analysis of Interrupt Design and PFS approved Control Flow Analysis of Interrupt (Concept of Execution) Architecture Diagram
  - Review design data against current hazards and previous architectural analysis and determine if additional risk is created and record causal factors identified
Task 1: JSSSEH Generic Requirements

- **E.8.5 Data Transfer Messages**
  - Data transfer messages shall be of a predetermined format and content. Each transfer shall contain a word or character string indicating the message length (if variable), the type of data, and the content of the message. At a minimum, parity checks and checksums shall be used for verification of correct data transfer. CRCs shall be used where practical. *No information from data transfer messages shall be used prior to verification of correct data transfer.*

- **E.3.13 Positive Feedback Mechanisms**
  - Software control of critical functions *shall have feedback mechanisms* that give positive indications of the function’s occurrence.

**Exercise #1**

Analyze architecture to ensure it supports verification of safety data.

Analyze architecture then design to ensure they support positive feedback for safety functions.
Task 1: SW Causal Factors
Identified by Architectural Analysis

SSF-6 Interface and Function Architecture Diagram

C2 Subsystem

Gun Subsystem

Validate command is in sequence

Validate command is intended for ADRS Gun

Validate Msg

Redundant No Fire Zone Controller

Receive EF commands

Process EF commands

Process EF_ID

VIPPS is a Mock System used for training purposes only
### Task 1: SW Causal Factors Identified by Architectural Analysis

<table>
<thead>
<tr>
<th>Mishap</th>
<th>Hazard</th>
<th>Causal Factor Description Based on JSSSEH Violations</th>
<th>Analysis Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Engage Friendly / non-hostile target</td>
<td>Fire command processed by the wrong gun results in accidental firing by the (wrong) gun</td>
<td></td>
<td>SSF-6 Interface and Function Architecture Diagram</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SSF-6 Interface and Function Architecture Diagram</td>
</tr>
</tbody>
</table>

VIPPS is a Mock System used for training purposes only.
Task 2: VIPPS PFS Approved Control
Flow Analysis of Interrupt Architecture

Concept of Execution

Interrupt is valid

Interrupt allowed in current code segment? N

Y

Stop main program task execution, save new messages as they come in

Process saved messages in the order received

N

Pause main program processing

Resume main program processing

Store Safety Significant Data

Perform Interrupt Processing

Restore Safety Significant Data

VIPPS is a Mock System used for training purposes only
Task 2: SSF-6 Control Flow Analysis of Interrupt Design

Interrupt is valid

Interrupt allowed in current code segment?

Y

Store SS Data Items:
- Deactivate_CMD_flag
- GunID
- EFmessage

Perform Interrupt Processing

Restore Safety Significant Data Items:
- GunID
- EFmessage

Resume main program Processing

N

Stop main program task execution

Pause main program processing

Resume main program execution
<table>
<thead>
<tr>
<th>Mishap</th>
<th>Hazard</th>
<th>Causal Factor Description Based on Design Evaluation</th>
<th>Analysis Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Engage Friendly / non-hostile target</td>
<td>Fire command processed by the wrong gun results in accidental firing by the (wrong) gun</td>
<td></td>
<td>Comparison of SSF-6 Control Flow Analysis of Interrupt Design and SSF-6 Control Flow Analysis of Interrupt Architecture</td>
</tr>
</tbody>
</table>

VIPPS is a Mock System used for training purposes only
SOFTWARE SAFETY TUTORIAL
STUDENT HANDOUT

CODE ANALYSIS
Task 1: Data Structure Analysis - For the given safety significant data items define the data types and usages using highlighted yellow code. Define any identified issues.

Task 2: Data Flow Analysis - Conduct data flow analysis to identify errors in the use of data that is accessed by multiple routines:

- Find “GunID” in each of the functions listed
- Document the value of the data item from each function
- After all values captured, determine if a safety concern exists and document rationale

Task 3: Use AOP-52 requirement, compliance assessment, and function to define compliance rationale. Define any safety issues.
Common Data Types

• Character (char):
  ➢ This is a single character, like X, £, 4, or *

• String:
  ➢ This is a “string” of characters of any length

• Integer (int):
  ➢ A whole number - whole meaning there are no digits after a decimal point. So 65 would be a valid integer; 65.78 would not.

• Floating-point number (float):
  ➢ A number that may have digits after the decimal place. 65.00 is technically a floating point number, even though it could be represented just as easily as an integer as 65. It takes more memory to store a float, which is why there is a distinction instead of just creating a “number” datatype.

• Boolean (bool):
  ➢ A variable to represent true or false (or it could also mean 0 or 1)
processEF_ID Function

//processEF_ID Function – is called by mainGun Function to initiate a CRC check and EF message verification after an EF command is received from C2

declare Local variables
Boolean CRC_pass,

// CRC function is called to ensure the integrity of the message bits in the received EF command.
If CRC_pass is equal to true, then
Send an ID02 back to C2,
Call validateMsgFunction,
Else
Send an ID04 back to C2;
//EndIf
//End of processEF_ID Function
//CRC Function – is called by processEF_ID Function to check the received EF command is a valid EF message or not.
//declare Local variables
Boolean CRC_pass,
Integer calculated_CRC, saved_CRC;

Get the values from the fields in the EFmessage to perform CRC calculation;

Calculate the CRC based on those values and store the result in calculate_CRC;
Get the pre-calculated CRC value from EFmessage and store it in the saved_CRC variable;

If calculate_CRC is equal to saved_CRC, then
   Call verifyGunIDFunction and store the return value in verifyGunID;
   If verifyGunID is equal to true, then
      Set CRC_pass to true;
   //EndIf
Else
   Set CRC_pass to false;
//EndIf

return the CRC_pass value to the called function;
//End of CRC Function

//Boolean type variable: 1(True) or 0(False) value
//Values used to perform the CRC calculation are stored in word 0 (bit 31-0) and half of word 1 (bit 31-16).
//The pre-calculated CRC value is stored in word 1 (bit15-0) //of each EF message.
//Call the verifyGunID Function to compare the gun IDs //between the config file and the received EF
//Set the CRC check to be true if and only if the CRC values //are same and the gun IDs are the same.
//Return either a 1 or 0 value to the processEF_ID function.
mainGun Function

//declare Global Variables – a global variable is a variable declared outside all functions.

Long EFmessage,
Integer GunID,
Float defaultAZ,
Integer currentAZ,
Integer defaultEL, currentEL,
Integer defaultGunSpeed, currentGunSpeed,
Boolean de_activate_CMD_flag,

Boolean verifyGunID,

//mainGun Function – where GCC program starts execution.

//Start the mainGun Function

Get the value of the gun address from Config File and store it in the GunID variable;
//Gun address is equal to 2 in the Configuration File

Get the value of the AZ position from Config File and store it in the defaultAZ variable;
//The value after decimal point will be truncated if a float number is going to be saved in an integer type variable.

Set the value of the currentAZ variable equal to the value of the defaultAZ variable;
.
.
.
Call processEF_IDFunction;

//End of mainGun Function
**Task 1: Data Structure Analysis Exercise**

STUDENT HANDOUT - Populate Table for the given safety significant data items using highlighted yellow code

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Data Type</th>
<th>Global/Local</th>
<th>Where is Variable used/referenced</th>
<th>Variable used/referenced consistently? If not, explain and provide any safety implications</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRC_pass</td>
<td></td>
<td></td>
<td>1. processEF_ID Function 2. CRC_Function</td>
<td></td>
</tr>
<tr>
<td>1. currentAZ</td>
<td>1.</td>
<td>1.</td>
<td>1. mainGun Function</td>
<td></td>
</tr>
<tr>
<td>2. defaultAZ</td>
<td>2.</td>
<td>2.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

VIPPS is a Mock System used for training purposes only

NSWCDD-PN-14-00391 is approved for Distribution Statement A: Approved for Public Release; distribution is unlimited.
verifyGunID Function

//verifyGunID Function – is called to check the value of Gun ID specified in the received EF match with the value of Gun ID specified in the config file.

//Start verifyGunID Function

//declare Local variable

Integer gunIDfrmMsg,

Boolean verifyGunID_flag,

Set GunID equal to 0;

Call Call getGunIDfrmMsgFunction and store the return value into gunIDfrmMsg variable;

If gunIDfrmMsg is equal to GunID, then

    Set verifyGunID_flag to True,
    Call FireGunFunction,
    Call validateMsgFunction;

Else

    Set verifyGunID_flag to False;
//EndIf

return the verifyGunID_flag value to the called function;
//End of verifyGunID Function
### Task 2: Data Flow Analysis Exercise

**STUDENT HANDOUT - Populate Table for GunID using highlighted yellow code**

<table>
<thead>
<tr>
<th>Data Item</th>
<th>Data Type</th>
<th>Global/Local</th>
<th>Where is Variable used/referenced</th>
<th>Value stored in Variable</th>
<th>Any safety concerns? [If yes, explain]</th>
</tr>
</thead>
<tbody>
<tr>
<td>GunID</td>
<td>Integer</td>
<td>Global</td>
<td>1. declare Global variable</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2. mainGun Function</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3. verifyGunID Function</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# Definition of Terms

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flags and Variables</td>
<td>Flags and variable names shall be unique. Flags and variables shall have a single purpose and shall be defined and initialized prior to use.</td>
</tr>
<tr>
<td>Execution Path</td>
<td>Safety Critical Computing System Functions (SCCSFs) shall have one and only one possible path leading to their execution.</td>
</tr>
<tr>
<td>Conditional Statements</td>
<td>Conditional statements shall have all possible conditions satisfied and be under full software control (i.e., there shall be no potential unresolved input to the conditional statement). Conditional statements shall be analyzed to ensure that the conditions are reasonable for the task and that all potential conditions are satisfied and not left to a default condition. All condition statements shall be annotated with their purpose and expected outcome for given conditions.</td>
</tr>
</tbody>
</table>
moveGun Function

//moveGun Function – which is called by validateMsg Function to command the gun motor to move gun to the specified AZ and EL positions with specified speed.

//Start moveGun Function which is passing three parameter variables i.e. AZ, EL, and gun speed

//declare Local variables
Integer commandedAZ,
Integer commandedEL,
Integer commandedGunSpeed,
Boolean moveGunStatus_flag = False,

Get the value of AZ position from the parameter (defaultAZ) and store it in the commandedAZ variable;
Get the value of EL position from the parameter (defaultEL) and store it in the commandedEL variable;

If the value of the commandedAZ variable is “not” equal to the value of the currentAZ variable, then
    Call moveAZ_MotorFunction (commandedAZ, commandedGunSpeed);

//EndIf

//End of moveGun Function
verifyGunID Function

//verifyGunID Function – is called to check the value of Gun ID specified in the received EF match with the value of Gun ID specified in the config file.

//Start verifyGunID Function

//declare Local variable

Integer gunIDfrmMsg,

Boolean verifyGunID_flag,

Call Call getGunIDfrmMsgFunction and store the return value into gunIDfrmMsg variable;

If gunIDfrmMsg is equal to GunID, then

Set verifyGunID_flag to True,

Call FireGunFunction,

Call validateMsgFunction;

Else

Set verifyGunID_flag to False;

//EndIf

return the verifyGunID_flag value to the called function;

//End of verifyGunID Function
## Task 3: AOP-52 Code Compliance Exercise

### STUDENT HANDOUT

<table>
<thead>
<tr>
<th>AOP–52 Requirement</th>
<th>Compliant</th>
<th>Function where code error is located</th>
<th>Rationale for non-compliance</th>
<th>Potential Safety Impact(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conditional Statements</td>
<td>No</td>
<td>validateMsgFunction</td>
<td>Select and power on the gun for ANY EF except for EF 21 - GSS Select</td>
<td>Will result in the Gun remaining active (selected and powered on) no matter what EF is received from C2, including an EF 23 - Safe Deselect (Power Off)</td>
</tr>
<tr>
<td>Flags and Variables</td>
<td>No</td>
<td>moveGunFunction</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Execution Path</td>
<td>No</td>
<td>verifyGunIDFunction</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
SOFTWARE SAFETY TUTORIAL STUDENT HANDOUT

TECHNOLOGY INSERTION
Task 1: Technology Insertion of Ethernet Protocol

- Task 1: Review reference material on Ethernet Protocol. Based on your review of the technology, define:
  - Any concerns about using the technology
  - Any recommendations concerning design selections or mitigations

<table>
<thead>
<tr>
<th>Technology of Concern</th>
<th>Concerns(s)</th>
<th>Recommendation(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ethernet Protocol</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Task 2: Technology Insertion of TCP Protocol

- Task 2: Review reference material on TCP Protocol. Based on your review of the technology, define:
  - Any concerns about using the technology
  - Any recommendations concerning design selections or mitigations

<table>
<thead>
<tr>
<th>Technology of Concern</th>
<th>Concerns(s)</th>
<th>Recommendation(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP Protocol</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>