Software Safety Tutorial

Student Handout

NSWCDD-PN-14-00391

SOFTWARE CRITICALITY

Software Criticality Analysis Worksheet

STUDENT HANDOUT - POPULATE 2.a - 2.i

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

Software Criticality Analysis Worksheet

Use for Columns 2.e and 2.f

Haz ID #	Phase of Operation	State/ Mode	Hazard Description	Causal Factors	Mishap	Effects	Hazard Controls	MRI	System	Subsystem	Component(s)	Comments
132	Operational	Tactical	Upon receipt detection of GSS Electronics overtemperature, the GCC fails to halt processing after 120 seconds (if C2 does not safe/deselect)	GCC fails to activate 120 second timer or terminate processing in response to 120 second timeout	Electronics overheat, causing circuit board damage	Equipment damage	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 + 0.025 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	3D	VIPPS	Gun Subsystem	Gun Control Computer	SSF 20
188	Operational	Tactical	Upon receipt of EF23 (Safe / Deselect), the GSS fails to remove power to the gun motor causing premature motor failure	Software Failure in the Gun Control Computer	Premature motor failure	Equipment damage	None	3D	VIPPS	Gun Subsystem	Gun Control Computer	SSF 20
299	Operation	Tactical	Fire command processed by the wrong gun results in accidental firing by the (wrong) gun	The GCC software fails to validate firing commands intended for the gun, and processes commands intended for a different gun	Engage Friendly/non-hostile target	Personnel injury or death	[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)	1D	VIPPS	Gun Subsystem	Gun Control Computer	SSF 6
310	Operation	Test	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	The GCC software fails to validate 'rate of gun movement' before commanding gun tpo move at normal speed.	Inadvertent physical contact of gun with personnel,	Personnel injury or death	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 ms per degree) or reduced speed (60 ms per degree) as specified in the message.	1D	VIPPS	Gun Subsystem	Gun Control Computer	SSF 6

VIPPS is a Mock System used for training purposes only

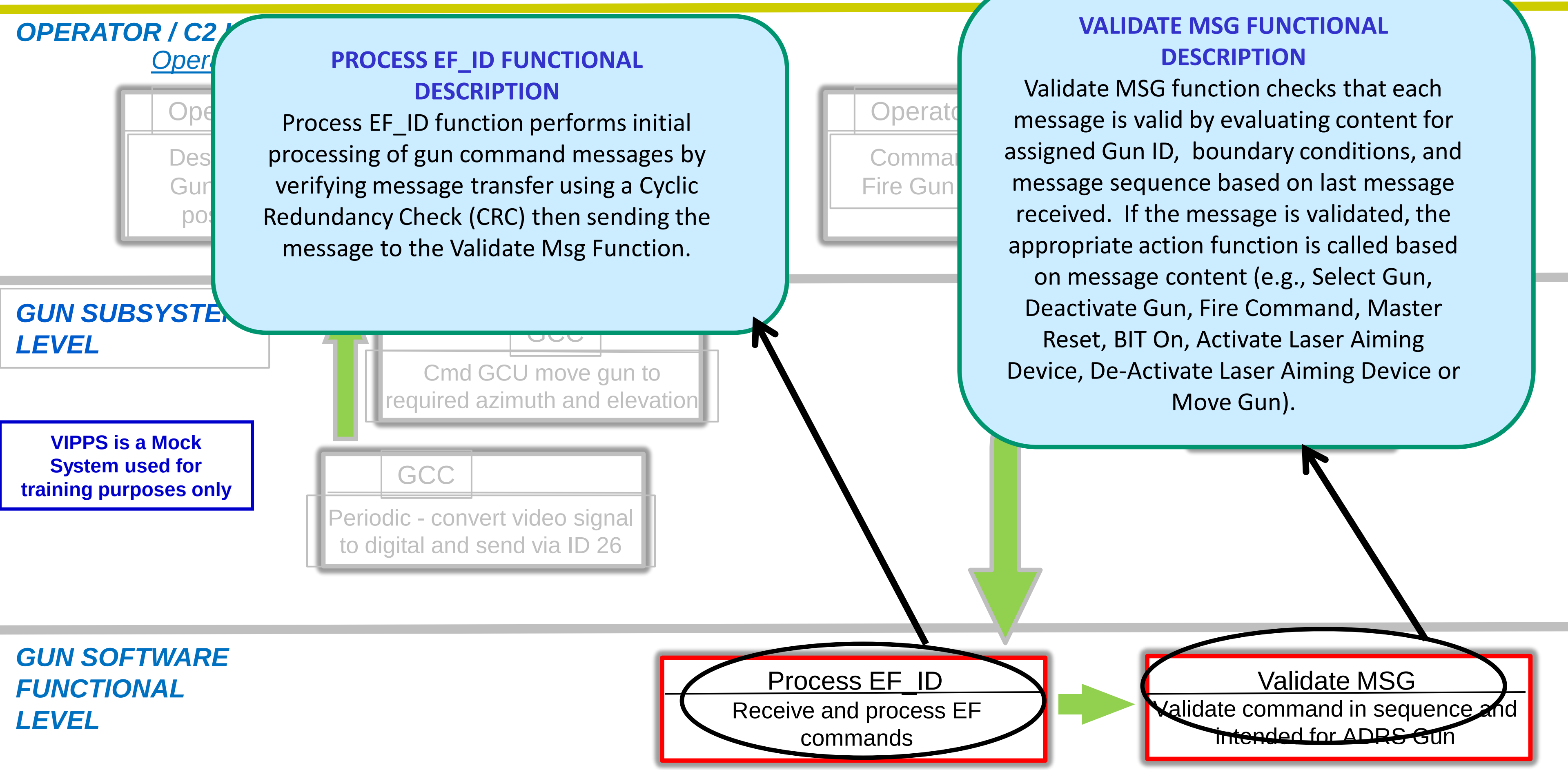
VIPPS Functional Flow Diagram

Use for Column 2.g

Activate

(Fire on fire cmd only if GSS ADDRESS" in fire cmd matches)

SSF-6: Verification of Gun Activation / Deactivate and Movement Commands



OPERATOR / C2 L
Oper
Ope
Des
Gun
pos
Operato
Comma
Fire Gun
GUN SUBSYSTE
LEVEL
GCC
Cmd GCU move gun to required azimuth and elevation
GCC
Periodic - convert video signal to digital and send via ID 26
GUN SOFTWARE FUNCTIONAL LEVEL

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 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).

Process EF_ID

Receive and process EF commands

Validate MSG

Validate command in sequence and intended for ADRS Gun

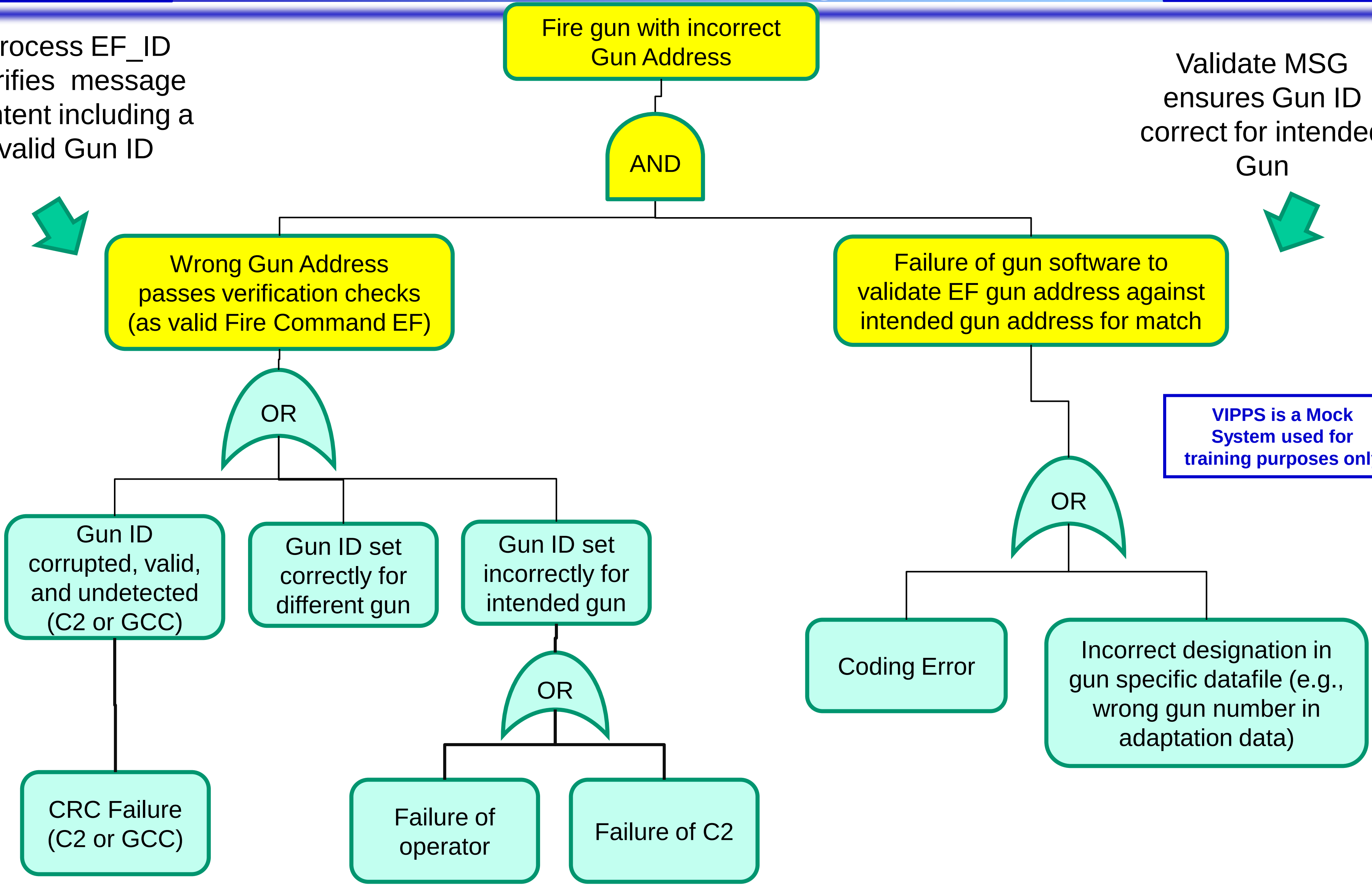
VIPPS is a Mock System used for training purposes only

VIPPS Event Diagram

Use for Column 2.g

Process EF_ID verifies message content including a valid Gun ID

Validate MSG ensures Gun ID correct for intended Gun


Fire gun with incorrect Gun Address
AND
Wrong Gun Address passes verification checks (as valid Fire Command EF)
Failure of gun software to validate EF gun address against intended gun address for match
OR
Gun ID corrupted, valid, and undetected (C2 or GCC)
Gun ID set correctly for different gun
Gun ID set incorrectly for intended gun
CRC Failure (C2 or GCC)
OR
Failure of operator
Failure of C2
OR
Coding Error
Incorrect designation in gun specific datafile (e.g., wrong gun number in adaptation data)

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.

MIL-STD-882E Software Control Categories

Use for Column 2.g

Level	Name	Description
1	Autonomous (AT)	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.)
2	Semi-Autonomous (SAT) or	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.)
3	Redundant Fault Tolerant (RFT) or	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 redutolerancendant 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.
4	Influential	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.
5	No Safety Impact (NSI)	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.

MIL-STD-882E Software Control Categories

Use for Column 2.g

Level	Name	Description
1	Autonomous (AT)	Sc … dware systems, su … control entity to or … tionality with multiple su … e critical.)
2	Semi-Autonomous (SAT)	S … ms, or mitigate or lity, no vare fault Software ite for m… atio occurrence not exceed the time required for adequate control entity response and hazard control.)
3	Redundant Fault Tolerant (RFT)	ts requiring a independent ult detection, There are redundar period.) Softw redunda
4	Influential	Software g … or action to
5	No Safety Impact (NSI)	Software fu subsystem … 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.

- Autonomous CONTROL over safety hardware systems, and
- No possibility of detection and intervention by control entity

- CONTROL over safety hardware systems, and
- Time for detection and intervention by INDEPENDENT safety mechanism

or

- 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

- Issues commands over safety hardware system, and
- Requires control entity to complete command function, and
- Includes REDUNDANT, INDEPENDENT fault tolerant mechanisms

or

- Generates information of safety-critical nature to make decisions, and
- System includes SEVERAL REDUNDANT, INDEPENDENT fault tolerant mechanisms

- Generates information of safety-related nature to make decisions by the operator, and
- Does not require operator action to avoid mishap

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

MIL-STD-882E Software Criticality Index

Use for Column 2.h and 2.i

For Gun Software Function : Validate_MSG

VIPPS is a Mock System used for training purposes only

Software Safety Criticality Matrix				
	Severity Category			
Software Control Category	Catastrophic (1)	Critical (2)	Marginal (3)	Negligible (4)
1	SwCI 1	SwCI 1	SwCI 3	SwCI 4
2	SwCI 1	SwCI 2	SwCI 3	SwCI 4
3	SwCI 2	SwCI 3	SwCI 4	SwCI 4
4	SwCI 3	SwCI 4	SwCI 4	SwCI 4
5	SwCI 5	SwCI 5	SwCI 5	SwCI 5

SwCI	**Level of Rigor**
SwCI 1	Program shall perform analysis of requirements, architecture, design, and code; and conduct in-depth safety-specific testing.
SwCI 2	Program shall perform analysis of requirements, architecture, and design; and conduct in-depth safety-specific testing.
SwCI 3	Program shall perform analysis of requirements and architecture, and conduct in-depth safety-specific testing.
SwCI 4	Program shall conduct safety-specific testing.
SwCI 5	Once assessed by safety engineering as Not Safety, then no safety specific analysis or verification is required.

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

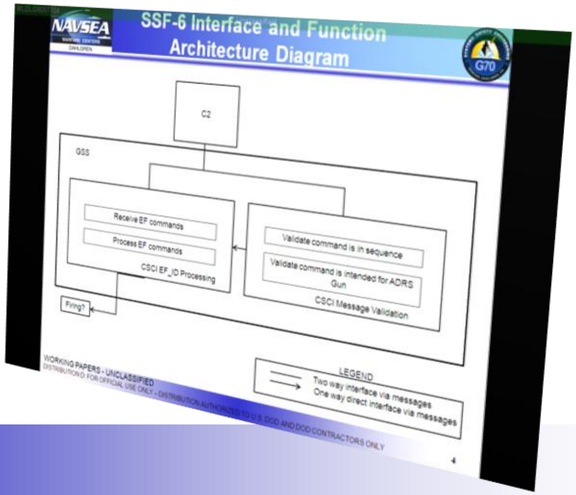
STUDENT REFERENCE

Analyze architecture to ensure it supports verification of safety data

- 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*.

Analyze architecture then design to ensure they support positive feedback for safety functions

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

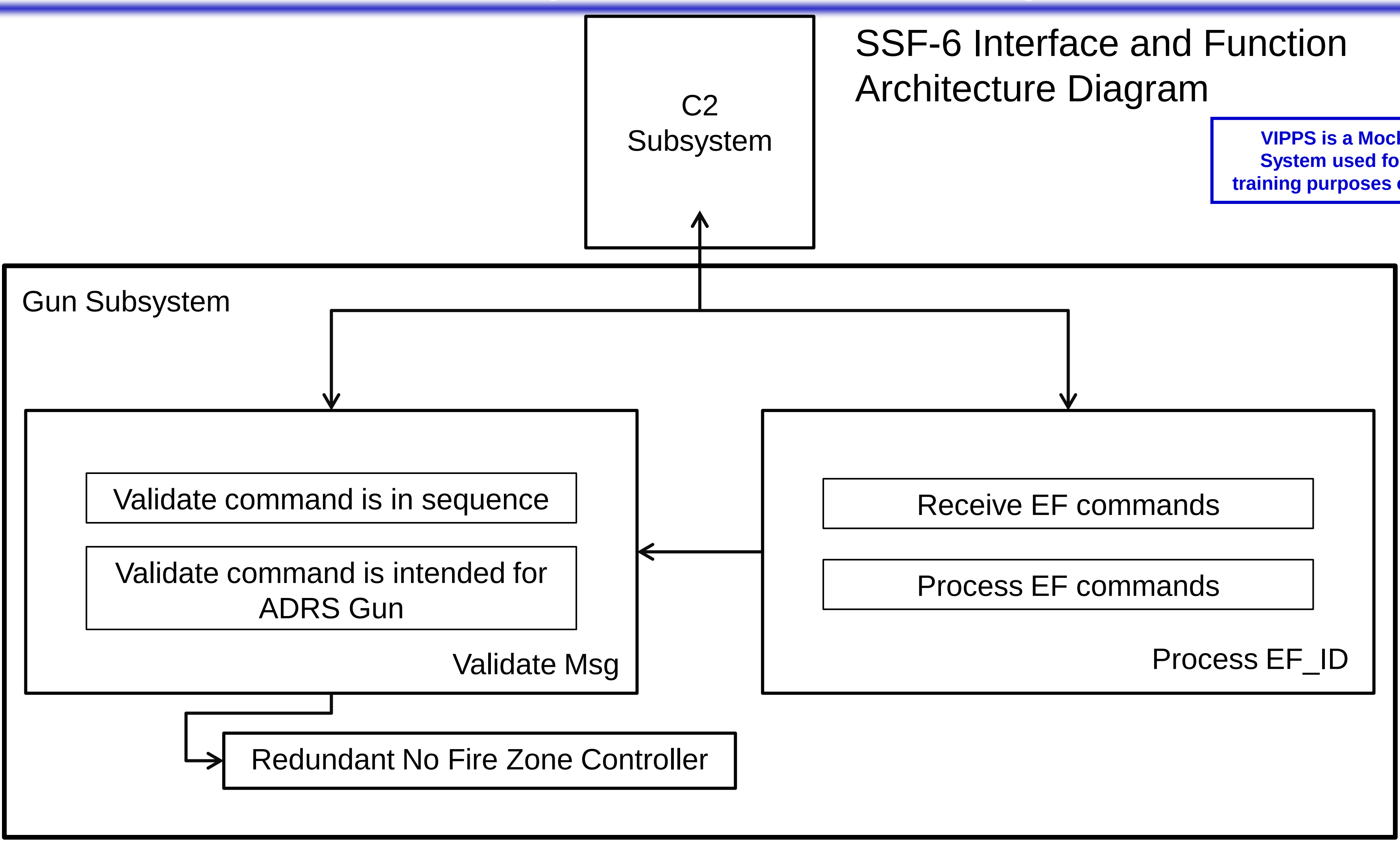

SSF-6 Interface and Function Architecture Diagram
C2
GSS
Receive EF commands
Process EF commands
CSCI EF_ID Processing
Validate command is in sequence
Validate command is intended for ADRS Gun
CSCI Message Validation
Firing?
LEGEND
Two way interface via messages
One way direct interface via messages
G70
4
WORKING PAPERS - UNCLASSIFIED
DISTRIBUTION D: FOR OFFICIAL USE ONLY – DISTRIBUTION AUTHORIZED TO U.S. DOD AND DOD CONTRACTORS ONLY

Exercise #1

Task 1: SW Causal Factors Identified by Architectural Analysis

SSF-6 Interface and Function Architecture Diagram

VIPPS is a Mock System used for training purposes only


C2 Subsystem
Gun Subsystem
Validate command is in sequence
Validate command is intended for ADRS Gun
Validate Msg
Receive EF commands
Process EF commands
Process EF_ID
Redundant No Fire Zone Controller

Task 1: SW Causal Factors Identified by Architectural Analysis

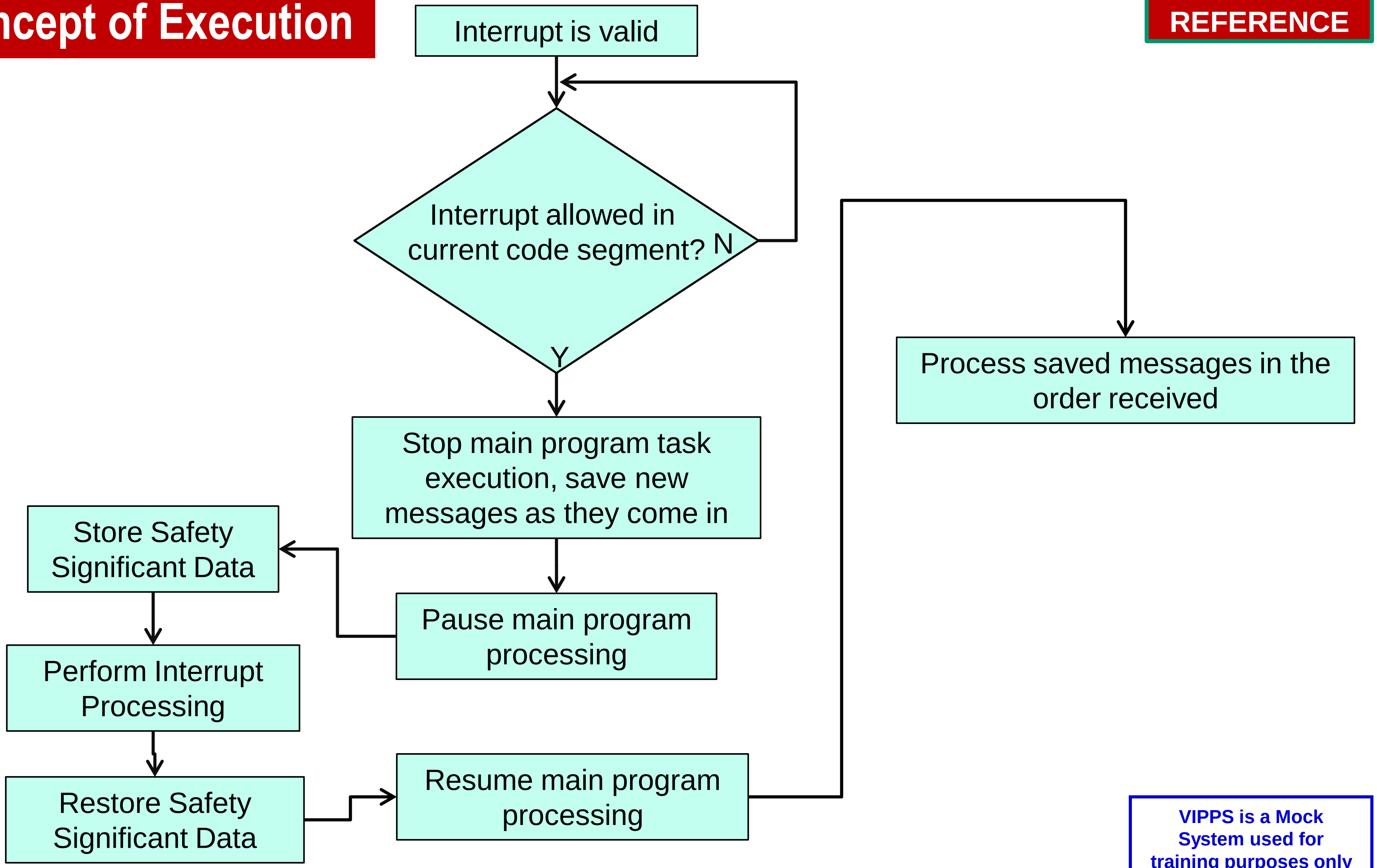
VIPPS is a Mock System used for training purposes only

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

Task 2: VIPPS PFS Approved Control Flow Analysis of Interrupt Architecture

STUDENT REFERENCE

Concept of Execution

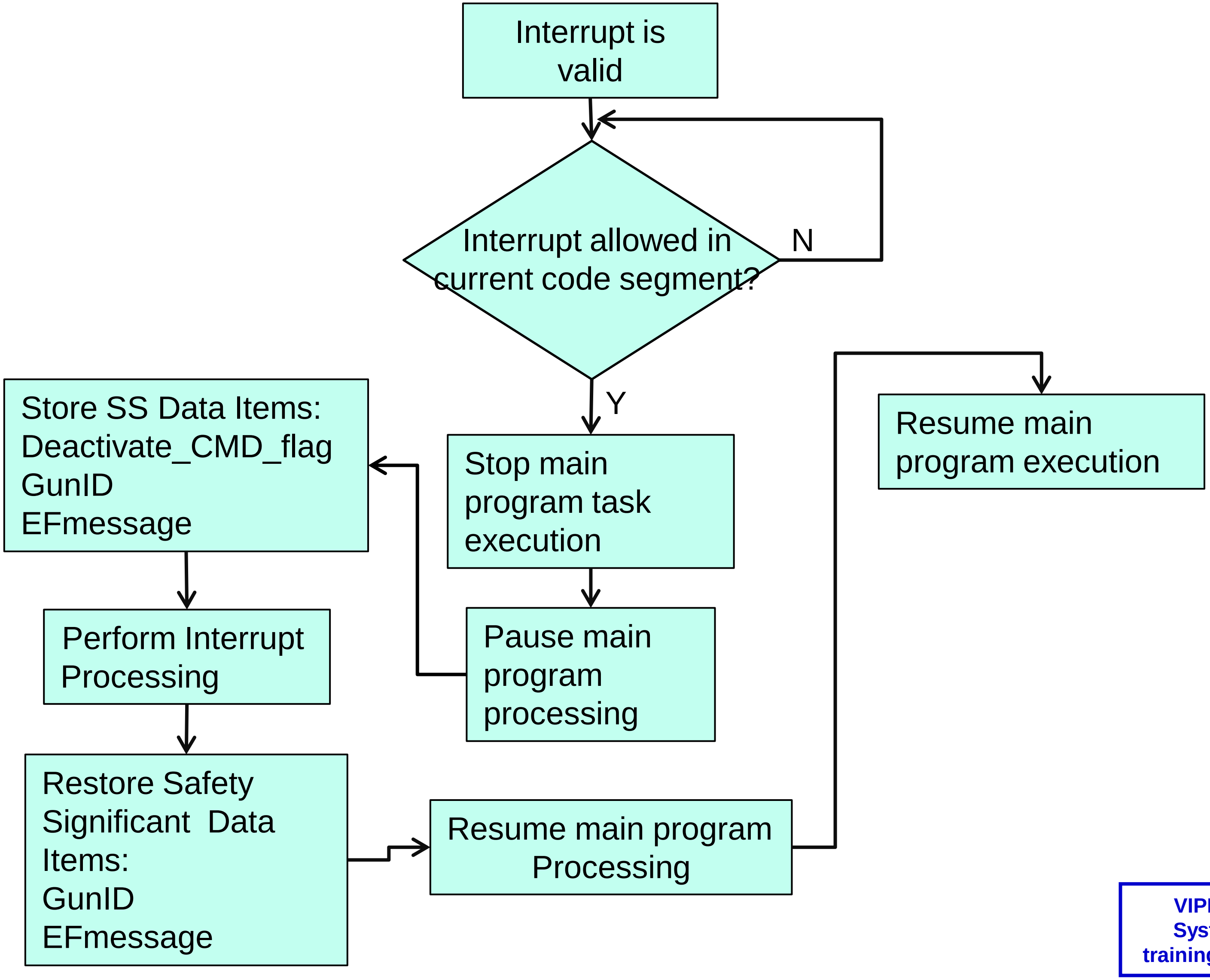


Interrupt is valid
Interrupt allowed in current code segment?
N
Y
Stop main program task execution, save new messages as they come in
Pause main program processing
Store Safety Significant Data
Perform Interrupt Processing
Restore Safety Significant Data
Resume main program processing
Process saved messages in the order received

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.

Task 2: SSF-6 Control Flow Analysis of Interrupt Design



Interrupt is valid
Interrupt allowed in current code segment?
N
Y
Stop main program task execution
Store SS Data Items:
Deactivate_CMD_flag
GunID
EFmessage
Pause main program processing
Perform Interrupt Processing
Restore Safety Significant Data Items:
GunID
EFmessage
Resume main program Processing
Resume main program execution

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.

Task 2: SW Causal Factors Identified by Design Analysis

Mishap	Hazard	Causal Factor Description Based on Design Evaluation	Analysis Source
Engage Friendly / non-hostile target	Fire command processed by the wrong gun results in accidental firing by the (wrong) gun		Comparison of SSF-6 Control Flow Analysis of Interrupt Design and SSF-6 Control Flow Analysis of Interrupt Architecture
			Comparison of SSF-6 Control Flow Analysis of Interrupt Design and SSF-6 Control Flow Analysis of Interrupt Architecture

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.

SOFTWARE SAFETY TUTORIAL STUDENT HANDOUT

CODE ANALYSIS

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

STUDENT REFERENCE

VIPPS is a Mock
System used for
training purposes only

```
*****************************************************************************************
//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                      //A local variable is one declared within the body of a function.
Boolean CRC_pass,                              //This is a Boolean type variable; used to indicate the result of the CRC check. If the check is 'pass', the
                                               //variable will be set to True (1). Otherwise the variable will be set to False (0).

Call CRC_Function and store the return value in 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                                          //CRC_pass='true' means the bits in the received EF command have not been corrupted.
        Send an ID02 back to C2,               //Send an Acknowledge (ID02) to C2
        Call validateMsgFunction,              //Call validate message function to validate command is in sequence and sent to the intended gun address
 Else                                          //CRC_pass='false' means the bits in the received EF command have been corrupted.
        Send an ID04 back to C2;               //GCC then transmits an Illegal Message (ID04) to the C2
//EndIf
//End of processEF_ID Function
```

CRC_Function

STUDENT REFERENCE

VIPPS is a Mock System used for training purposes only

```
*****************************************************************************************************
//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,                                                    //Boolean type variable: 1(True) or 0(False) value

Integer calculated_CRC, saved_CRC,

Get the values from the fields in the EFmessage to perform CRC calculation;   //Values used to perform the CRC calculation are stored in
                                                                     //word 0 (bit 31-0) and half of word 1 (bit 31-16).
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;   //The pre-calculated CRC value is stored in word 1 (bit15-0)
                                                                     //of each EF message.
If calculate_CRC is equal to saved_CRC,
then
      Call verifyGunIDFunction and store the return value in verifyGunID;   //Call the verifyGunID Function to compare the gun IDs
                                                                     //between the config file and the received EF
      If verifyGunID is equal to true,
      then
            Set CRC_pass to true;                                    //Set the CRC check to be true if and only if the CRC values
      //EndIf                                                        //are same and the gun IDs are the same.
Else
      Set CRC_pass to false;
//EndIf
return the CRC_pass value to the called function;                    //Return either a 1or 0 value to the processEF_ID function.
//End of CRC Function
```

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

mainGun Function

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

Long EFmessage,                                   //This is a 64-bit variable; used to store two 32-bit word EF data.
Integer GunID,                                    //This is an integer type variable; used to store the value of Gun ID
Float defaultAZ,                                  //This is a float type variable; used to store the default azimuth position.
Integer currentAZ,                                //This is an integer type variable used to store the current azimuth position
Integer defaultEL, currentEL,                     //Both are integer type variables; used to store the elevation position
Integer defaultGunSpeed, currentGunSpeed,         //Both are integer type variables; used to store the gun speed
Boolean de_activate_CMD_flag,                     //This is a Boolean type variable which has a either True (1) or False (0) value; used to indicate whether or
                                                  //not the deactivate command has been received
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;
Set the value of the currentAZ variable equal to the value of 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.
.
.
.
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

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

VIPPS is a Mock System used for training purposes only

verifyGunID Function

STUDENT REFERENCE

VIPPS is a Mock System used for training purposes only

```
*************************************************************************************************************************
//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                                              //A local variable is one declared within the body of a
                                                                      //function.
Integer gunIDfrmMsg,                                                  //An integer type variable used to store the value of gun
                                                                      //address from the received EF
Boolean verifyGunID_flag,                                             //A Boolean type variable used to indicate the pass or fail
                                                                      //condition of checking the values of gun addresses between
                                                                      //the received EF and the config file.

Set GunID equal to 0;                                                 //Reset the value of GunID to 0 (zero).  Recall that a GunID of
                                                                      //0 is valid and represents Gun 1.
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 the FireGunFunction to initiate sending an EF39 (Fire
                                                                      //Command)
        Call validateMsgFunction;                                     //Call the ValidateMsgFunction to validate content of the
                                                                      //current EF received from C2
Else
        Set verifyGunID_flag  to False;
//EndIf

return the verifyGunID_flag value to the called function;             //Return either 1 or 0 value of verifyGunID_flag to the
                                                                      //validateMsg Function.

//End of verifyGunID Function
```

Task 2: Data Flow Analysis Exercise

STUDENT HANDOUT - Populate Table for GunID using highlighted yellow code

VIPPS is a Mock System used for training purposes only

Data Item	Data type	Global/ Local	Where is Variable used/referenced	Value stored in Variable	Any safety concerns? [If yes, explain]
GunID	Integer	Global	1. declare Global variable		
			2. mainGun Function		
			3. verifyGunID Function		

AOP-52 Compliance Assessment Definition of Terms

Term	Definition
Flags and Variables	Flags and variable names shall be unique. Flags and variables shall have a single purpose and shall be defined and initialized prior to use.
Execution Path	Safety Critical Computing System Functions (SCCSFs) shall have one and only one possible path leading to their execution.
Conditional Statements	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.

moveGun Function

STUDENT REFERENCE

VIPPS is a Mock System used for training purposes only

```
**********************************************************************************************************
//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                                    //A local variable is one declared within the body of a function.
Integer commandedAZ,
Integer commandedEL,
Integer commandedGunSpeed,
Boolean moveGunStatus_flag = False,                          //A Boolean value ('True' or 'False') used to reflect if the gun is
                                                             //now pointing in the direction specified in the message; it is
                                                             //initialized at not pointing to the specified direction

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

If the value of the commandedAZ variable is "not" equal to the value of the currentAZ variable,
then
      Call moveAZ_MotorFunction (commandedAZ, commandedGunSpeed);   // This function commands  the gun azimuth movement

//EndIf
.
.
.
//End of moveGun Function
```

verifyGunID Function

STUDENT REFERENCE

VIPPS is a Mock System used for training purposes only

```
****************************************************************************************************
//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                                         //A local variable is one declared within the body of a
                                                                 //function.
Integer gunIDfrmMsg,                                             //An integer type variable used to store the value of gun
                                                                 //address from the received EF
Boolean verifyGunID_flag,                                        //A Boolean type variable used to indicate the pass or fail
                                                                 //condition of checking the values of gun addresses between
                                                                 //the received EF and the config file.


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 the FireGunFunction to initiate sending an EF39 (Fire
                                                                 //Command)
        Call validateMsgFunction;                                //Call the ValidateMsgFunction to validate content of the
                                                                 //current EF received from C2
Else
        Set verifyGunID_flag  to False;
//EndIf

return the verifyGunID_flag value to the called function;       //Return either 1 or 0 value of verifyGunID_flag to the
                                                                 //validateMsg Function.
//End of verifyGunID Function
```

Task 3: AOP-52 Code Compliance Exercise

STUDENT HANDOUT

VIPPS is a Mock System used for training purposes only

AOP–52 Requirement	Compliant	Function where code error is located	Rationale for non-compliance	Potential Safety Impact(s)
Conditional Statements	No	validateMsgFunction	Select and power on the gun for ANY EF except for EF 21 - GSS Select	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)
Flags and Variables	No	moveGunFunction		
Execution Path	No	verifyGunIDFunction		

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

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

Technology of Concern	Concerns(s)	Recommendation(s)
Ethernet Protocol		

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

Technology of Concern	Concerns(s)	Recommendation(s)
TCP Protocol		