Universidad Autónoma de Querétaro
Facultad de Informática
Maestría En Software Embebido

Especificación y Validación de Requerimientos para el Empaquetamiento de Datos en Software de Aviación de tipo Crítico

Opción de titulación
Tesis

Que como parte de los requisitos para obtener el Grado de
Maestría en Software Embebido

Presenta:
Francisco Javier Sánchez del Valle

Dirigido por:
M.S.I. Fausto Abraham Jacques García

M.S.I. Fausto Abraham Jacques García
Presidente — Firma

M.D.V.A. Leopoldo Rodríguez Salazar.
Secretario — Firma

Dr. Norberto Muñoz Gómez
Vocal — Firma

M.I. Oscar Morales González
Suplente — Firma

M.C. José Carlos Villela Tinoco
Suplente — Firma

MISD Juan Salvador Hernández Valerio
Directora de la Facultad

Dra. Ma. Guadalupe Flavia Loarca Piña
Director de Investigación y Posgrado

Centro Universitario
Querétaro, Qro. México
Fecha (28 de agosto de 2015)

RESUMEN

"Aeronautical Radio, Incorporated Standard 429" (A429) es un protocolo de comunicación que establece cómo se comunican los dispositivos y sistemas aviónicos en aviones comerciales. Las organizaciones que se dedican al desarrollo de software de aviación deben contar con la aprobación y certificación de la "Federal Aviation Administration" (FAA) para que dicho software pueda ser utilizado en vuelos comerciales. La FAA reconoce el documento DO-178B como guía para garantizar la seguridad de sistemas aviónicos. DO-178B especifica que existen 5 niveles de criticidad para el software de aviación, siendo el software categoría A el más crítico. Para el caso de software de categoría A, el DO-178B dictamina que para la validación del software es necesario realizar pruebas de rango normal y pruebas de robustez con el fin de asegurar que el software puede responder a condiciones normales y anormales. El software de nivel A de los sistemas de aviación modernos es extremadamente crítico, pues dependen vidas humanas de él. Por ello es de vital importancia reconocer que la validación de los requerimientos de alto nivel es una fase crucial en el ciclo de vida del software. Un requerimiento de alto nivel es una capacidad necesaria, cuantificable y verificable de alguna función, propiedad, característica o comportamiento que el software debe de tener para resolver un problema del mundo real. Existen diferentes tipos de palabras A429 para representar diferentes tipos de datos, palabras BNR ("Binary"), discretas y BCD ("Binary Coded Decimal"). La validación de software es un método para asegurar que el comportamiento del software es adecuado para un contexto de sistema. Un procedimiento de prueba es una especificación que contiene una serie de casos de prueba relacionados a requerimientos de alto nivel. Se establece una metodología para validar requerimientos de alto nivel relacionados con el empaquetamiento de datos del protocolo A429 a través de procedimientos de prueba en los que se especifique las pruebas necesarias para asegurar que los datos se transmiten correctamente y que además cumpla con los aspectos de certificación que dictamina el documento DO-178B.

(**Palabras clave**: A429, DO178-B, software de Aviación, validación de requerimientos de alto nivel)

SUMMARY

Aeronautical Radio, Incorporated Standard 429 (A429) is a communication protocol that establishes how devices and avionics systems communicate in an airplane network. The companies that develop avionics software must have the approval and certification of FAA (Federal Aviation Administration) in order to use the software in a commercial flight. FAA recognizes the document DO-178B as guide to warranty security of avionics systems. DO-178B specifies that there are 5 levels of criticism for aviation software, category A software is the most critic. For category A software, DO-178B stipulates that normal and robustness range tests are needed in order to validate software. Category A software in modern aviation systems is extremely critic, since human lives depend on it. So, it is important to recognize that validation of high level requirements is crucial in the software life cycle. A high level requirement is a necessary, quantifiable and verifiable characteristic of a function, property or behavior that the software must have to solve a problem in the real world. There are different A429 words to represent different data types, BNR (Binary), discrete and BCD (Binary Coded Decimal) words. Software validation is method to ensure that the software behavior is correct for the system context. A test procedure is a specification that contains a group of test cases related to high level requirements. A methodology is established to validate high level requirements related to the data packing process of A429 protocol though test procedures which specify the necessary tests to ensure that data is transmitted correctly and also accomplish the certification aspects that DO-178B specifies.

(**Key words**: A429, DO178-B, aviation software, high level requirement validation)

A las nuevas generaciones de ingenieros llenas de conocimiento.

AGRADECIMIENTOS

A los ingenieros Leopoldo Rodríguez Salazar, Norberto Muñoz Gómez, Oscar Morales Gonzales y José Carlos Villela Tinoco, sinodales de la tesis por brindar su tiempo y apoyo para revisar el texto y por sus comentarios y sugerencias para mejorarlo.

En especial, al Maestro Fausto Abraham Jacques García, catedrático de la Universidad Autónoma de Querétaro por brindar su conocimiento en la revisión del texto y sus atinados comentarios y sugerencias para mejorarlo.

A Mis padres Victoria del Valle Rodríguez y Jorge Sánchez Aguilar, así como a mi hermano Jorge Iván Sánchez del Valle, por apoyarme toda mi vida.

En particular a mi novia María Cassandra Ramos Garibay, por brindarme su amor, apoyo, confianza y comprensión.

TABLA DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

ÍNDICE DE FORMULAS

1. INTRODUCCIÓN

En 2006, se registraron 28 millones de vuelos programados, los cuales trasladaron a 2 billones de pasajeros alrededor del mundo y el crecimiento desde hace 30 años había sido del 5% anual en los últimos 30 años (Henckels, 2006). En 2013 cerca de 3 billones de personas fueron transportadas alrededor del mundo (IATA, 2013), lo cual equivale a un crecimiento del 7.14% anual de 2006 a 2013, lo cual es mayor a lo que se esperaba en 2006.

La siguiente gráfica (IATA, 2013) muestra el crecimiento exponencial tanto de tráfico de pasajeros como de tráfico de carga de 2007 a 2012:

Figura 1-1 Crecimiento de tráfico de pasajeros y tráfico de carga (2007-2012).

De forma similar la siguiente imagen muestra como se disminuyeron las eventualidades de accidentes aéreos de 2011 a 2012:

Figura 1-2 Eventualidades de accidentes aéreos de 2011 a 2012.

Actualmente la mayor parte de los sistemas de aviación funciona mediante software embebido. Las organizaciones que se dedican al desarrollo de software de aviación deben contar con la aprobación y certificación de la FAA para que dicho software pueda utilizarse en un vuelo comercial. La FAA es la autoridad nacional de aviación del departamento de transporte de los Estados Unidos, y una de sus funciones principales es promover la seguridad en la industria de aviación. La FAA reconoce el documento Radio Technical Comission For Aeronautics (RTCA) standard DO-178B como aceptable para garantizar la seguridad de sistemas aeronáuticos (Shultz, 2009).

ARINC 429 (A429) es un protocolo de comunicación que establece cómo los dispositivos y sistemas aviónicos se enlazan en aviones comerciales. Es un

protocolo que fue desarrollado en 1978 y el cual hoy en día se sigue utilizando en gran parte de los aviones a nivel global como lo expresa Fuch (2010).

Tal como lo expresa Fuch (2010) y Condor Engineering (2005) los protocolos de comunicación han ido evolucionando con el paso de tiempo. Actualmente existe el protocolo denominado Avionics Full Duplex Switched Ethernet (AFDX) el cual trabaja con base de Ethernet y sin embargo, es mucho más rápido y seguro que el protocolo A429, sin embargo el empaquetamiento de datos que dicta el protocolo A429 hoy en día se sigue utilizando para transmitir la misma información a través de AFDX.

El documento DO-178B dicta que es necesario cumplir con estándares de desarrollo y validación de software. Como lo comenta Messner (2007), este documento expresa en la sección denominada “Dessign Assurance Level” un nivel de software de acuerdo a la importancia y criticidad dentro de un sistema de navegación de la siguiente manera:

- Nivel A: Software del cual en caso de detectarse un error pueda causar una falla en el sistema del cual se desprendan resultados catastróficos como por ejemplo, que impidan que el avión viaje, fallas estructurales que puedan ocasionar la pérdida de vidas humanas.
- Nivel B: Software del cual se desprenda una falla severa en las condiciones del avión. Por ejemplo, reducir la capacidad del avión de tal manera que la tripulación no pueda ser capaz de pilotear con exactitud.
- Nivel C: Software del cual, en caso de detectarse un error, se desprenda una falla mayor en el sistema. Por ejemplo, que el error reduzca la capacidad del avión para enfrentarse a condiciones operacionales adversas, que disminuyen significativamente el trabajo de la cabina.
- Nivel D: Software del cual, en caso de detectarse un error, se desprenda una falla menor en el sistema. Un ejemplo puede ser un cambio del plan de vuelo o alguna molestia en los pasajeros. Desde el software de nivel D

hasta el software de nivel A se incrementa la carga de trabajo para la tripulación.

- Nivel E: Software del cual en caso de detectarse un error no contribuya a una falla mayor en el sistema y que no tenga efecto en la operación del vuelo del avión.

Para el caso de software de categoría A, el DO-178B (Directorate General Technical Airworthiness, 2008) dictamina que para la validación del software, es necesario realizar pruebas de rango normal y pruebas de robustez con el fin de asegurar que el software puede responder a condiciones anormales.

Hoy en día existen diversos trabajos relacionados al protocolo A429 y el DO-178B. El DO-178B dictamina las características y condiciones que el software de cumplir para certificarse, sin embargo no se especifica un método para realizarlo. La presente investigación propone una metodología para la verificación del software de empaquetamiento del protocolo A429 que cumpla con los lineamientos que dicta el documento DO-178B.

2. Revisión de Literatura

2.1 Definiciones Básicas de software

Con el fin de comprender la información contenida en la presente investigación es necesario definir algunos conceptos básicos de software.

2.1.1 Requerimiento de alto nivel.

Tal como lo comenta Kandt (2003), un requerimiento de alto nivel es una capacidad necesaria, cuantificable y verificable de alguna función, propiedad, característica o comportamiento que el software debe de tener para resolver un problema del mundo real. Los requerimientos de alto nivel son desarrollados en la fase de definición de requerimientos, la cual consiste en una serie de actividades estructuradas de la cual, se obtiene un documento de requerimientos que normalmente es denominado especificación de requerimientos del sistema

2.1.2 Tipos de requerimientos de alto nivel

Los requerimientos de alto nivel pueden ser de dos tipos (Westfall, 2006):

- Requerimientos funcionales: son aquellos que definen el comportamiento que el software debe tener con el fin de que los usuarios puedan cumplir con sus tareas, por lo tanto su fin es satisfacer los requerimientos del negocio.
- Requerimientos no funcionales: son aquellos requerimientos que definen características, propiedades o cualidades que el software debe poseer.

Para diferenciar un requerimiento funcional de un requerimiento no funcional se definen los siguientes ejemplos:

Requerimiento de alto nivel:

“La función división debe devolver el resultado de la división de la entrada A entre la entrada B.”

Requerimiento de bajo nivel:

“En caso de que la entrada B sea igual a cero, la función división debe devolver 0.”

Como se puede observar en el enunciado anterior, el requerimiento de alto nivel describe la funcionalidad como tal que la función debe realizar, mientras que el requerimiento de bajo nivel define una excepción para evitar una división entre 0. Esta excepción es una propiedad que toda división debe de tener mientras el software siga operando.

2.1.3 Diccionario de datos.

Un diccionario de datos se refiere a un conjunto de referencias hacia unidades de información que interactúan en el sistema. Cada definición debe de tener la descripción de cómo utilizar la unidad, cuál es su tipo de dato, su tamaño, etc. (Oracle, 2011).

Supongamos que se requiere realizar el diccionario de datos de una función que deba sumar dos parámetros. La siguiente tabla ejemplifica cuales deberían ser la información mínima que debe contener:

Tabla 2-1 Ejemplo de Diccionario de datos, función suma.

Formato genérico de diccionario de datos		
Función suma		
Entradas		
Item	Tipo de Dato	Descripción
Parametro_A	Int32	Primer parámetro para sumar.
Parametro_B	Int32	Segundo parámetro para sumar.
Salidas		
Item	Tipo de Dato	Descripción
Resultado_Suma	Int32	Resultado de la suma de Parametro_A con Parametro_B

2.1.4 Características de requerimientos de alto nivel.

Asimismo, los requerimientos deben cumplir con ciertas características con el fin de evitar problemas de cambio de requerimientos o ingeniería inversa durante el ciclo de vida del software (Westfall, 2006). Estas características son:

- Completos: Los requerimientos del documento incluyen toda la información necesaria. Todos los requerimientos funcionales y no funcionales, requerimientos de interfaz y el correspondiente diccionario de datos.
- Consistentes: No existen requerimientos que se contradigan entre sí.
- Modificables: Los requerimientos son escritos organizadamente de manera que puedan ser fácilmente modificados en caso de que exista un cambio futuro. Cada requerimiento puede ser cambiado sin requerir un cambio excesivo en otros requerimientos.
- No ambiguos: Cada requerimiento debe tener una sola interpretación y debe ser coherente y fácil de entender.

- Concisos: Los requerimientos deben tener un lenguaje corto, específico y orientado a la funcionalidad.
- Finitos: Los requerimientos no deben expresar adverbios de cantidad como “poco”, “todos”, “mucho”, “la mayor parte” etc.
- Medibles. Deben contener límites o valores medibles.
- Capaz de ser implementado: El requerimiento puede ser implementado utilizando tecnología disponible para el proyecto en particular.
- Validable: Existe la manera de comprobar que el requerimiento está correctamente implementado.
- Referenciado: Cada requerimiento debe tener una referencia para su localización de fuente y destino.

2.2 Bit, Byte, Palabras de 32 bits

Nick y Zelenski (2008) mencionan que casi todas las computadoras modernas basan su manejo de memoria siguiendo la arquitectura Von Neumann. Por lo cual, existen conceptos que son fundamentales describir para entender cómo se comunican las computadoras hoy en día:

- Bit es la unidad más pequeña de memoria en la que se puede representar un dato, tiene dos estados 1 y 0.
- Byte es una agrupación de 8 bits. Un grupo de N bits puede tener 2^n combinaciones o patrones. Por lo tanto un byte es equivalente a 2^8 lo cual equivale a 256 combinaciones diferentes.
- Palabra de 32 bits es una agrupación de 4 bytes lo cual equivale a 2^{32}(En decimal 4,294,967,296) combinaciones o patrones.

2.3 Representación hexadecimal de bytes.

Cualquier dato (enteros, reales, caracteres, cadenas, estructuras, videos, etc.) que se desee almacenar en una computadora se representa a través de bits. Sin embargo es complejo interpretar un número binario, por ello comúnmente se utiliza el sistema de numeración hexadecimal para poder representar un conjunto de bits o bytes de manera simplificada(Young, 2014). La siguiente tabla ejemplifica la equivalencia entre los sistemas de numeración binaria, decimal y hexadecimal:

Tabla 2-2 Tabla de equivalencias de sistema binario, hexadecimal y decimal.

Binario	Dec	Hex	Binario	Dec	Hex
0000	0	0	1000	8	8
0001	1	1	1001	9	9
0010	2	2	1010	10	A
0011	3	3	1011	11	B
0100	4	4	1100	12	C
0101	5	5	1101	13	D
0110	6	6	1110	14	E
0111	7	7	1111	15	F

2.4 Conceptos de validación de software.

La validación de software abarca diversos tópicos, por lo cual es necesario definir algunos conceptos básicos que ayuden a diferenciar el enfoque de la presente investigación.

La validación de software es un método para asegurar que el comportamiento del software es adecuado para un contexto de sistema (Directorate General Technical Airworthiness, 2008).

Los propósitos de la validación de software son los siguientes:

- Demostrar que el software satisface los requerimientos de alto nivel, también denominados requerimientos de software.
- Demostrar con un alto grado de confianza que los errores que puedan tener como resultado condiciones de falla inaceptables sean removidos.

Así mismo, se considera que la validación de software está completa cuando se ha probado el tamaño total del software, el cual es determinado por las siguientes partes:

- Cobertura de requerimientos: cuando todos los requerimientos de alto nivel o requerimientos de software son validados en cuanto a su rango normal, anormal y usos inesperados.
- Cobertura estructural: cuando cada elemento del software ha sido ejercitado durante la fase de validación de software.
- Cobertura arquitectural: los controles actuales y los vínculos de datos han sido utilizados durante la validación de software.

La presente investigación propone una metodología para la cobertura de requerimientos utilizando pruebas de rango normal y anormal hacia requerimientos de alto nivel que especifiquen como se debe empaquetar el protocolo A429 en un sistema determinado.

2.4.1 Condiciones de rango normal y condiciones de robustez.

DO-178B demanda validar cada variable utilizada en el software mediante pruebas de rango normal y condiciones de robustez (Directorate General Technical Airworthiness, 2008), por ello es necesario definir ambos conceptos:

- Condiciones de rango normal: validar los límites de cada variable, ya sea real, entera, booleana, etc. Se debe probar así mismo múltiples iteraciones de tiempo en elementos de software como filtros, integradores, retrasos (delays), etc.
- Condiciones de robustez: probar valores afuera de los límites de cada variable. Inicialización del sistema durante condiciones anormales. Modos de falla para entrada de datos. Valores fuera de rango para ciclos. Condiciones de protección contra desborde de memoria (overflow).

2.4.2 Caja negra y caja blanca.

Otra manera de ver el tipo de validación que se desea realizar en la presente investigación es la denominada prueba de caja negra, dado que está enfocada a la funcionalidad del software. A continuación se definen los conceptos de caja negra y caja blanca:

- Caja negra: pruebas enfocadas a la funcionalidad del software, a lo que se supone que debe hacer, sin infiltrarse en cómo está constituido ni en cómo trabaja. Por lo tanto las pruebas que se hacen a este nivel son en su mayoría basadas en requerimientos de alto nivel, sin mirar un diseño o una parte de código.

- Caja blanca: pruebas enfocadas a como está construido el software, normalmente las pruebas de caja de blanca se basan en un diseño de software o en el código como tal de la aplicación.

Para el DO178-B todos las pruebas de software son pruebas de caja negra, o en otras palabras, basadas en requerimientos (Directorate General Technical Airworthiness, 2008).

2.4.3 Procedimientos de prueba.

Un procedimiento de prueba es una especificación que contiene una serie de casos de prueba relacionados a cierta funcionalidad del software. Es necesario N procedimientos de prueba para cubrir cada uno de los requerimientos de software existentes.

Un procedimiento de prueba al menos debe contar al menos con la siguiente información:

- Nombre del procedimiento de prueba.

- Descripción general del procedimiento.

- Requerimientos que pretende validar.
- Nombre de la persona que realiza las pruebas.
- Fecha.
- Casos de prueba: se refieren a especificar los valores a las diferentes entradas de uno o más requerimientos de alto nivel, así como especificar la salida esperada de ese conjunto de entradas. Normalmente cuentan con un identificador que lo diferencia de los demás casos de prueba.

2.5 Protocolo A429.

A429 es un protocolo que existe desde 1978 y el cual hoy en día se sigue utilizando en gran parte de los aviones a nivel global como lo expresa Fuch (2010).

ARINC es una organización privada que nace en 1929 por cuatro aerolíneas en Estados Unidos con la finalidad de servir de protocolos de comunicación necesarios para la industria aeroespacial (AIRLINES ELECTRONIC ENGINEERING COMMITTEE, 2004).

2.5.1 Empaquetamiento de datos A429.

El empaquetamiento de datos del protocolo A429 se transmite en palabras de 32 bits de un dispositivo aviónico a otro(Avionics Interfaces Technologies, 2008).

2.5.2 Tipos de Palabras A429.

Existen diferentes tipos de palabras A429 para representar diferentes tipos de datos (AIRLINES ELECTRONIC ENGINEERING COMMITTEE, 2004):

- Binary Coded Decimal (BCD): En estas palabras los datos numéricos de representan cada 4 bits(Digital Electronics, 2003). Los datos que normalmente se representan con este tipo de palabras, son datos de

frecuencia, fecha y hora. La siguiente tabla representa el número 989 en formato BCD:

Tabla 2-3 Representación de datos BCD (Binary Coded Decimal).

	Campo BCD 1	Campo BCD 2	Campo BCD 3
Valor Decimal	9	8	9
Valor Binario	1001	1000	1001

- BNR (Binary): Estas palabras codifican los datos como un número binario. Los datos que normalmente se transmiten mediante esta representación son grados, velocidad y altura (AIRLINES ELECTRONIC ENGINEERING COMMITTEE, 2004).El bit más significativo representa el bit de signo, cuando es uno representa un valor negativo y cuando es 0 representa un valor positivo y transmitido como un complemento fraccional A2. El siguiente bit representa la mitad del valor máximo que se puede representar y así sucesivamente. Supóngase que cada bit prendido representa 0.125 y se quiere representar el número 10 y -10, lo primero que se debe hacer es dividir 10 entre el valor de cada bit para sacar la cantidad de bits necesarios, en este caso son 80. Las siguientes figuras ejemplifican como se representaría este valor:

  Para el 10 positivo:

Figura 2-1 Representación de datos BNR (Valores Positivos).

Valor Decimal	128	64	32	16	8	4	2	1
Valor Binario	0	1	0	1	0	0	0	0

Para el 10 negativo representado en complemento fraccional A2:

Figura 2-2 Representación de datos BNR (Valores Negativos).

Valor Decimal	128	64	32	16	8	4	2	1
Valor Binario	1	0	1	0	1	1	1	1

Discretas: Estas palabras pueden estar formadas por la combinación de datos BNR y/o BCD o bits individuales que representan condiciones específicas del equipo, como pueden ser Correcto/Incorrecto, Activado/Desactivado o Verdadero/Falso.

2.5.3 Estructura de palabras A429.

Las palabras A429 están compuestas de 5 campos principales:

- Bit de paridad: El bit más significante (Bit 32) es denominado bit de paridad. El protocolo A429 utiliza paridad impar para verificar la recepción de datos. En caso de que la suma de los bits 1 a 31 sea par, este se ajustará a 1 para que la suma total sea impar, y se ajustará a 0 cuando la suma de los bits 1 al 31 sea impar (Avionics Interfaces Technologies, 2008).

- "Sign/Status Matrix" (SSM), matriz de signo/status: Utiliza los bits 30 y 31 de la palabra A429 correspondiente. Dependiendo el tipo de palabra puede representar el signo o la dirección de los datos o el estado actual del equipo o sensor que produce el dato. Las siguientes tablas representan el significado de los bits 30 y 31 para los tipos de palabra BCD, BNR y discreto:

Tabla 2-4 Representación de SSM para palabras BCD.

Bit 31	Bit 30	Representación
0	0	Plus, North, East, Right, To, Above
0	1	No Computed Data (NCD)
1	0	Functional Test
1	1	Minus, South, West, Left From, Below

Tabla 2-5 Representación de SSM para palabras BNR.

Bit 31	Bit 30	Representación
0	0	Failure Warning
0	1	No Computed Data (NCD)
1	0	Functional Test
1	1	Normal Operation

Tabla 2-6 Representación de SSM para palabras discretas.

Bit 31	Bit 30	Representación
0	0	Verified Data, Normal Operation
0	1	No Computed Data (NCD)
1	0	Functional Test
1	1	Failure Warning

- Datos: Utiliza 19 bits para representar la información de la palabra. Los tipos de datos que se pueden representar son binarios (BNR), BCD o datos discretos.

- "Source/Destination Identifier" (SDI), identificador de fuente/destino: Utiliza los bits 9 y 10 para representar cual es la fuente de transmisión de datos o para identificar cual es el receptor destino de un conjunto de receptores en la red. La siguiente Figura ejemplifica el funcionamiento del SDI.

Tabla 2-7 Ejemplo de SDI en palabra A429.

Bit 10	Bit 9	Interpretación
0	0	Receptor 1
0	1	Receptor 2
1	0	Receptor 3
1	1	Receptor 4

- Label: Utiliza los primeros 8 bits de la palabra y este campo es conocido también como el identificador de información. El "label" es expresado como 3 dígitos octales programados para aceptar hasta 255 combinaciones. El bit más significativo del "label" reside en la posición menos significativa de la

palabra, es decir, se escribe de derecha a izquierda como se ejemplifica en la siguiente tabla con el número octal 321:

Figura 2-3 Representación de número octal de campo "Label" en palabra A429.

BIT	8	7	6	5	3	2	1
Valor Octal	4	2	1	4	1	2	1
Valor Binario	0	0	1	0	0	1	1
Label 321	1			2		3	

En la siguiente Figura se puede observar el orden en el que cada uno de los campos que conforman una palabra A429 de 32 bits:

Figura 2-4 Campos de palabra A429.

P	SSM		Datos																	SDI		Label ID							
32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1

2.6 DO-178B.

De acuerdo a Messner (2007), "Software Considerations in Airbone Systems and Equipment Certification" es el título del documento DO-178B, el cual es un documento que describe el proceso de desarrollo de software en sistemas de aviación.

La primer versión DO-178, fue creada por la RTCA en 1980 y actualmente es un estándar aceptado por las principales agencias de certificación como la FAA y la EASA para proporcionar la certificación de un software de aviación.

La siguiente tabla muestra las diferentes versiones del documento DO-178B:

Tabla 2-8 Versiones de DO178

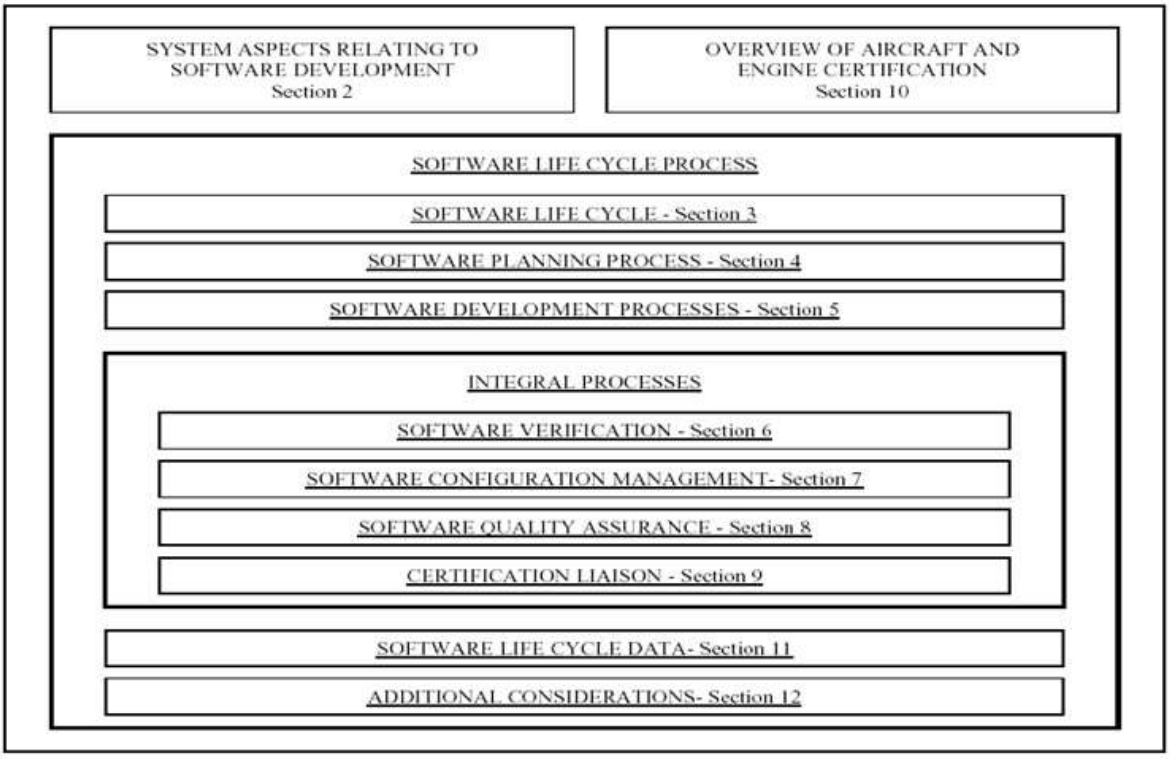
Nombre del documento	Año
DO-178	1980

DO-178A	1985
DO-178B	1992
DO-178C	2011

Es importante resaltar que la versión DO-178C no tiene cambios relevantes para el propósito de la presente investigación, por lo cual la metodología propuesta es compatible con ambos documentos.

El ciclo de vida de software para sistemas de aviación propuesto por el DO-178B se resume en la siguiente figura:

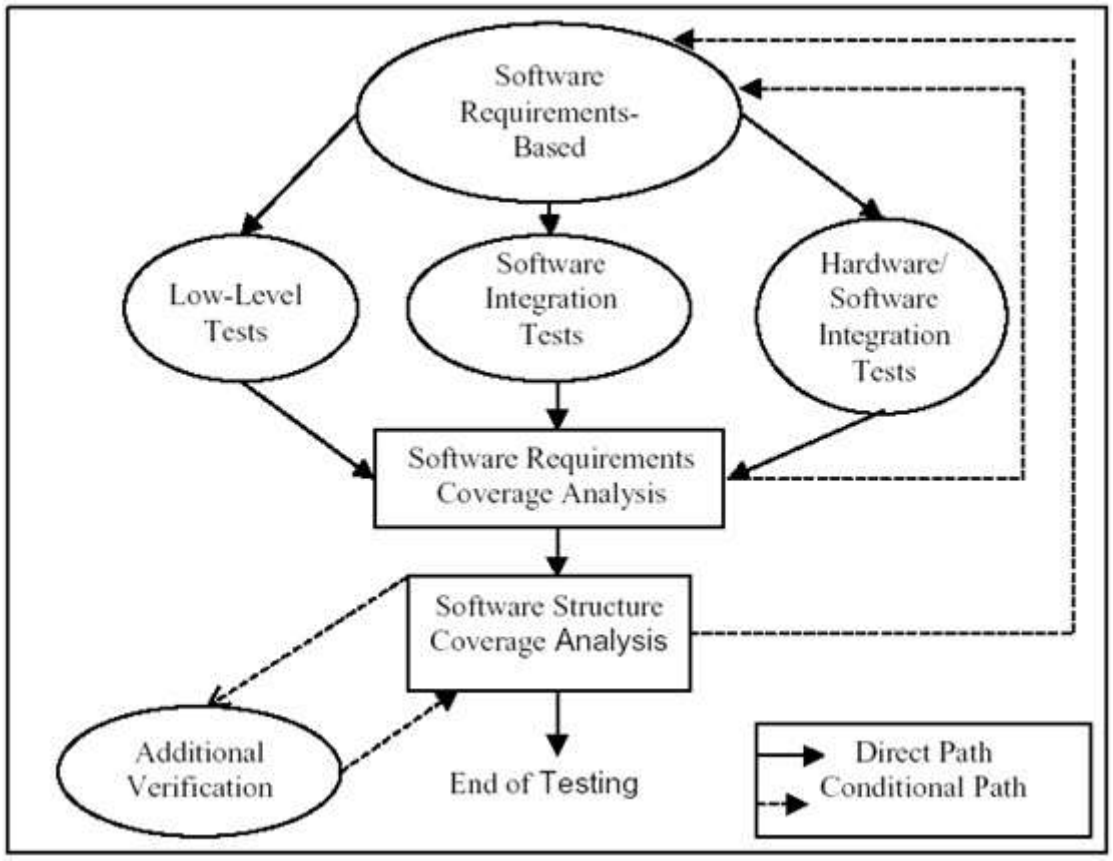
Figura 2-5 Ciclo de vida de software acorde a DO-178B.



Para los fines de la presente investigación, la sección 6 del documento DO-178B será la única utilizada y por lo tanto definida.

El proceso de validación de software acorde con el DO-178B (Messner, 2007) es un proceso donde se identifican y se reportan errores de software a resultado del proceso de desarrollo de software así como lo describe la siguiente figura:

Figura 2-6 Proceso de validación de software.



Como se puede observar en la figura 2-6, todas las pruebas de software se realizan a partir de los requerimientos de software. Posteriormente se realiza un proceso denominada análisis de cobertura estructural en el cual se asegura que cada uno de las posibles combinaciones del software se haya cubierto dependiendo de su nivel de criticidad. Sin embargo, para los fines de la presente investigación la metodología define solamente la primer parte del diagrama, es decir la creación de casos de prueba a partir de requerimientos de software.

Así mismo, las salidas del proceso del proceso de verificación de software serán las siguientes:

a) Procedimientos y casos de prueba de software. La presente investigación detalla la estructura que debe llevar tanto los procedimientos como los casos de prueba.
b) Resultados de verificación de software. Una vez que se ejecutan las pruebas realizada en los procedimientos se obtienen los resultados de cada caso de prueba, es decir se detalla si el software pasa o no la prueba. La presente investigación no ejecuta pruebas ante un software, dado que la metodología propuesta es genérica para cualquier lenguaje de programación.

3. Hipótesis, objetivos y justificación.

3.1 Hipótesis

Contar con una metodología confiable para desarrollar casos de prueba de empaquetamiento de datos puede ayudar al aumento de la seguridad en software de tipo catastrófico y reducir el re trabajo.

3.2 Objetivo general

Proponer una metodología de pruebas para la validación del empaquetamiento de datos para el protocolo A429 basado en requerimientos de alto nivel para software declarado por el DO-178B como Software de Nivel A (Catastrófico).

3.3 Objetivos particulares

- Establecer una plantilla genérica de procedimiento de prueba. Para el proceso de validación de requerimientos de alto nivel de software categoría A, el DO-178B demanda tener especificaciones o procedimientos de prueba

en donde se detallen las entradas y las salidas esperadas de cada uno de los casos de prueba. Para ello es recomendable tener una plantilla genérica con el fin de que todos los procedimientos de prueba tengan el mismo formato, esto facilitará el desarrollo de los casos de prueba así como la revisión de los mismos.

- Establecer Banco de datos y representar binariamente palabras A429. Para realizar casos de prueba para empaquetamiento de datos de palabras A429, es fundamental conocer el significado de cada uno de los 32 bits del protocolo A429 así como las diferencias que existen entre palabras BNR, BCD y discretas. Por ello la creación de una hoja de cálculo, que obtenga los valores hexadecimales o binarios de cada uno de los 32 bits del protocolo A429 es fundamental para obtener los resultados esperados de una manera sencilla y segura.
- Definir casos de prueba de rango normal y robustez para palabras BNR, BCD y discretas A429. Con el uso de la plantilla genérica de procedimientos de prueba y la hoja de cálculo para obtener los resultados de cada caso de prueba, se obtienen especificaciones de pruebas correspondientes a las palabras BNR, BCD y discretas A429.

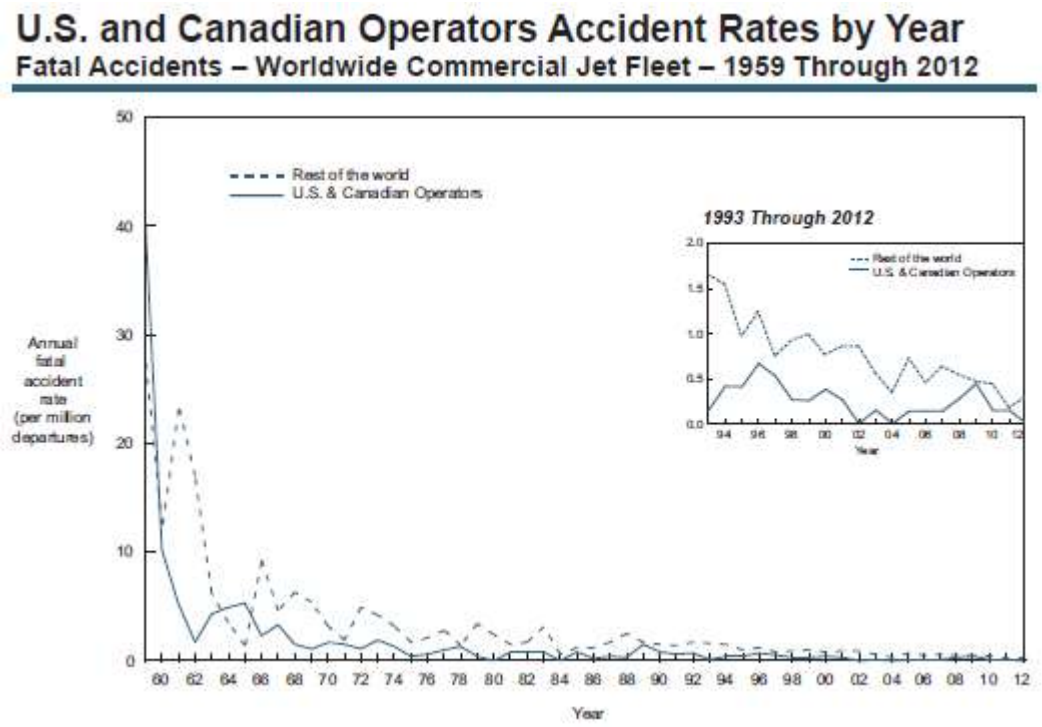
3.4 Justificación

Cada año millones de personas viajan a través del mundo, la industria aeroespacial invierte millones de dólares en la seguridad de los sistemas de aviación modernos debido a dos grandes razones:

- Garantizar la seguridad de los pasajeros.
- Cumplir con los lineamientos de la FAA, European Aviation Safety Agency (EASA) y/o otros organismos para que el software se pueda certificar y por ende volar.

La seguridad en los sistemas aviónicos ha sido incrementada considerablemente en los últimos años, la siguiente figura muestra los accidentes aéreos de la aerolínea BOIENG por millón de vuelos (BOIENG Commercial Airplanes, 2012):

Figura 3-1 Accidentes Aéreos Boeing de 1960 a 2012.



Así mismo, la mayor parte de los aviones comerciales viajan a través del denominado piloto automático, el cual funciona a través del sistema de manejo de vuelo (FMS por sus siglas en ingles Flight Managment System) el cual está encargado de administrar los recursos del avión, provee los comandos para que el piloto automático y la computadora de vuelo mantengan una ruta determinada con ciertos parámetros específicos de vuelo. Todo este software funciona a través de un conjunto de sensores, dispositivos conectados en red y una serie de funcionalidades y cálculos que le permiten al FMS administrar los recursos para que el piloto automático pueda funcionar y lograr aterrizajes en condiciones

anormales como lluvia, niebla o nieve, y así mismo predecir una posible colisión con algún objeto cercano (FAA, 2011).

Una falla en este tipo de software puede resultar en daños catastróficos donde vidas humanas pueden perderse. Por ejemplo, imagínese un desbordamiento en la variable que controle la altura, supóngase que la altura promedio del vuelo es de 10,000 pies, y repentinamente exista un desbordamiento de memoria que ocasione que los pies marquen 0 pies de altura, el resultado podría ocasionar un accidente fatal, en caso de que el software no se encuentre preparado para evitar este tipo de situaciones.

Ningún sistema es perfecto, pues el factor humano siempre está de por medio, sin embargo los sistemas son perfectibles y esto lo demuestra la figura 2-7, la cantidad de accidentes aéreos ha disminuido considerablemente a lo largo del tiempo.

Por ello la importancia de asegurar que el software es lo suficientemente robusto para garantizar seguridad de las personas que viajan diariamente a través del mundo. El software debe estar preparado, no solamente para condiciones normales de vuelo, sino también para condiciones anormales que puedan ocasionar una catástrofe.

Los requerimientos de alto nivel deben ser escritos de manera tal que se pueda saber cómo responde el software ante condiciones anormales, uno de los principales problemas es que normalmente los requerimientos de alto nivel no especifican el comportamiento del software ante condiciones como valores fuera de rango, ciclos infinitos, arreglos fuera de su rango, divisiones entre cero, etc. Sin embargo, la etapa de validación y verificación deben contemplar escenarios de prueba tales que puedan ocasionar errores en el sistema.

Existen condiciones anormales que se pueden forzar en el empaquetamiento de datos A429. Por ejemplo, acorde con Digital Electronics

(2003), el fin de los valores BCD es representar un número del 0 al 9 en un espacio de 4 bits, sin embargo, existen combinaciones que se consideran anormales dado que 4 bits representan 15 posibles combinaciones y sólo 10 son valores normales tal como lo demuestra la siguiente tabla:

Tabla 3-1 Representación binaria de campo BCD (Binary Coded Decimal).

Bits	Valor decimal
0000	1
0001	2
0010	3
0011	4
0100	5
0101	6
0110	7
0111	8
1000	9
1001	Inválido
1010	Inválido
1011	Inválido
1100	Inválido
1101	Inválido
1110	Inválido
1111	Inválido

En condiciones ideales, deberían existir requerimientos que especifiquen la manera en la que el software deba reaccionar ante una entrada inválida. En caso de que no exista un requerimiento que hable sobre este comportamiento, el proceso de validación y verificación debe demostrar que existe algo mal en el software. Siguiendo el mismo ejemplo, si se inyecta un valor de 15 a un valor BCD y el resultado obtenido es 15, entonces se está hablando de un comportamiento que no debería existir, puesto que el valor BCD solo debe mostrar valores del 0 al 9. De esta manera dicho caso de prueba debe de fallar al momento de ejecutarse.

Por ello es importante conocer los diferentes tipos de datos que se pueden encapsular en el protocolo A429 para poder realizar pruebas suficientes que comprueben que el software encapsula correctamente los datos. Si no existieran

requerimientos que hablen sobre comportamientos no esperados y además el desarrollador de pruebas no conociera el protocolo A429, entonces lo más probable es que los casos de prueba de robustez nunca se realicen, por lo cual existe una gran posibilidad que el software pueda fallar después de la etapa de validación y verificación de software.

Uno de los protocolos más usados para transmisión de datos en los sistemas de aviación es el A429, el cual ha existido desde 1978; mediante este protocolo viaja información que viene desde sensores, dispositivos electrónicos, computadoras, etc., con el fin de que toda la red interna del avión reciba y mande información hacia los demás dispositivos. El tipo de información que normalmente se manda a través de este protocolo, son datos como altitud, velocidades, fecha, hora, latitud, longitud, presión barométrica, etc.

Hoy en día existen diversos trabajos relacionados al protocolo A429 y el DO-178B. Tal como lo plantea Lauterbach (2012), es posible realizar pruebas de comunicación del protocolo a través de un osciloscopio para asegurar que el protocolo se transmite a la velocidad estipulada por la especificación A429 (AIRLINES ELECTRONIC ENGINEERING COMMITTEE, 2004) y cuantifica la calidad de la transmisión y recepción de paquetes funciona de manera adecuada, sin embargo, esto no comprueba que el empaquetamiento de datos funcione de acuerdo al protocolo.

Por otra parte, existen diversos artículos que hablan sobre cómo se debe empaquetar los datos del protocolo A429 tal como lo estipulan A. Martinec (2001), AIM GmbH (2001), Condor Engineering (2000) y la misma especificación del protocolo generada por ARINC (2004). En estos artículos se detalla cada parte, bit por bit, como se debe empaquetar y cómo funciona el protocolo, sin embargo en ninguna de estas estipula un procedimiento específico de prueba.

Así mismo existen artículos que acuerdan lo necesario para la verificación de software de categoría de acuerdo con el DO-178B. Tal es el caso como lo

plantea DGTA-ADF (2008) donde se plantea que para cualquier software de tipo crítico es necesario realizar pruebas de rango normal y anormal, estas últimas también conocidas como pruebas de robustez, con el fin de comprobar que cada requerimiento de alto nivel ha sido completamente probado. La problemática principal es que el DO-178B no menciona métodos específicos acerca de cómo se deben realizar las pruebas, únicamente dictamina lo necesario para certificación. Para poder interpretar y aplicar una metodología para validación de software de categoría A, es necesario aunar diversos temas como es el caso de las especificaciones de pruebas, requerimientos de alto nivel, pruebas de rango y de robustez, etc.

De una manera similar la FAA (2001) estipula que se debe asegurar que cada decisión en el código ha sido ejecutada y que cada variable por si sola produce una salida determina es necesario seguir la metodología “Modified Condition Decision Coverage” (MCDC) para software de nivel A. Sin embargo siguiendo esta metodología no es suficiente para poder decir que el software es suficientemente robusto dado que MCDC se enfoca únicamente en los diversos comportamientos que cada variable puede ocasionar y que todas las salidas lógicas hayan sido ejecutadas, por lo cual existe aún posibilidad que el software falle debido a pruebas de rango y robustez. Debido a que el protocolo A429 manda información de valores numéricos y discretos, pruebas de rango y robustez son necesarias para asegurar que el software empaqueta los datos de manera adecuada.

Por otra parte, Bentley (2005) define los conceptos básicos de validación de software, los tipos de pruebas que existen, en qué consisten y así mismo diversos roles de personas y terminologías necesarias para llevar a cabo pruebas de software.

Linda Westfall (2006), por otra parte menciona las características fundamentales que deben contener los requerimientos de alto nivel y además la

importancia que tiene una etapa de levantamiento de requerimientos en la planeación de un proyecto de software. Siguiendo la metodología para escribir requerimientos que propone Linda Westfall se reduce enormemente la cantidad de errores y re trabajo por ambigüedad de requerimientos.

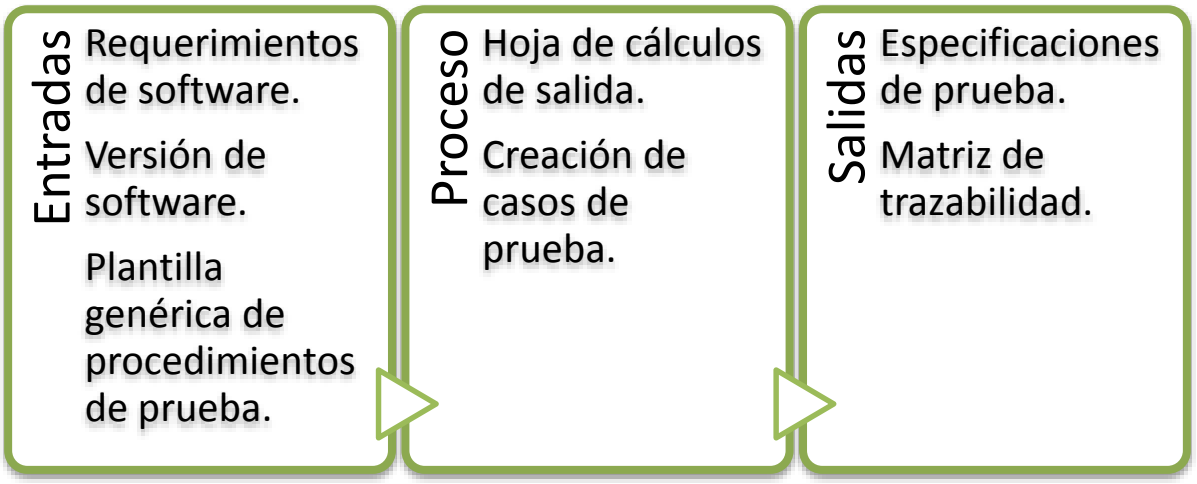
La presente investigación pretende proponer una metodología que prueba el empaquetamiento de datos A429 mediante pruebas de rango normal y anormal basado en requerimientos de alto nivel.

4. Metodología

4.1 Metodología para desarrollo de especificaciones de prueba de software.

La siguiente figura de muestra la metodología propuesta para el desarrollo de especificaciones de pruebas con base a requerimientos de alto nivel:

Figura 4-1 Metodología para desarrollo de especificaciones de prueba de software.



A continuación se define cada uno de los pasos de la metodología:

a) Requerimientos de software: conjunto de requerimientos que reflejan la funcionalidad del software, debe existir una versión asignada a un

conjunto de requerimientos, debido a que los requerimientos se actualizan constantemente y los procedimientos de prueba deben especificar la versión específica de requerimientos que se están probando.

b) Versión de software: una versión específica de software surge al momento de implementar un conjunto de requerimientos de alto nivel. Esta versión de software es muy importante, dado que después de la creación de las especificaciones de software, los casos de prueba se deben ejecutar con el software corriendo, para saber si existen errores en el software y así mismo para que quede evidencia de que cada requerimiento se ha implementado correctamente.
c) Plantilla genérica de procedimientos de prueba: como se define en el capítulo 4.2, la plantilla genérica es utilizado como base para la creación de procedimientos de prueba.
d) Hoja de cálculos: de manera similar en el capítulo 4.3, se define la creación de la hoja de resultados, que funcionara para obtener las salidas de los casos de prueba relacionados a cualquier requerimiento de alto nivel que describa la funcionalidad de alguna palabra A429.
e) Creación de casos de prueba: el ingeniero de software encargado de validar un grupo de requerimientos de alto nivel con ayuda de la hoja de cálculos, debe utilizar la plantilla genérica de procedimientos de prueba para especificar las entradas y salidas de cada caso de prueba.
f) Especificaciones de prueba: una vez que detalla cada caso de prueba y agrega la información de la cabecera de la plantilla genérica de especificaciones de prueba, el ingeniero de software ha finalizado y el resultado obtenido es una especificación de prueba.
g) Matriz de trazabilidad: Una vez finalizada la especiación de prueba, es necesario especificar los casos de prueba que cubren un conjunto de requerimiento de alto nivel. Para ello se utiliza una matriz de trazabilidad donde es necesario ingresar los identificadores de casos de prueba para

cada uno de los identificadores de los requerimientos de alto nivel. El capítulo 4.4.4 brinda ejemplos de cómo se define esta matriz.

4.2 Definición de plantilla para procedimiento de prueba.

Antes de empezar a definir los casos de prueba necesarios para cada tipo de palabra A429 (BCD, BNR, Discreta), es necesario definir un archivo común donde se especifique una trazabilidad entre los casos de prueba y los requerimientos de alto nivel, es decir especificar qué casos de prueba cubren determinado o determinados requerimientos de alto nivel. A este documento se le denomina plantilla para procedimiento de prueba.

La presente investigación relaciona un procedimiento de prueba para cada conjunto de requerimientos de alto nivel específicos de una “label” de palabra A429, esto no quiere decir que no se pueda probar más de una “label” en un mismo procedimiento de prueba, sin embargo es recomendable lograr la independencia de cada palabra, para evitar procedimientos muy grandes, los cuales pueden presentar dificultades como tiempo de desarrollo y/o revisión excesivos.

Para empezar se define el encabezado del procedimiento de prueba, la siguiente tabla ejemplifica las partes que debe de llevar:

Tabla 4-1 Plantilla genérico de procedimiento de prueba.

Plantilla genérica para casos de prueba A429	
Nombre de procedimiento de prueba	
Versión	
Autor	
Fecha	
HLR_ID	
Descripción	
Versión de Sofwate	
Versión de Documento de requerimientos	

A continuación se explica cómo llenar cada una de las partes del encabezado del procedimiento de prueba:

- Nombre de procedimiento de prueba: es un identificador con el cual se distingue un procedimiento de prueba de otro. Debería existir una manera común de llamar al procedimiento de prueba, esto varía dependiendo los estándares de cada proyecto o por definición del arquitecto de software. Para los fines de la presente investigación, el identificador será formado iniciando por la palabra “Procedimiento”, seguido de un guión bajo “_” y al último el nombre del “label” de la palabra ARINC que se esté probando. En caso de que existen “labels” con el mismo identificador, se agregará un número del 1 al 9 para evitar que el identificador se repita. Por ejemplo si el procedimiento de prueba desea probar la “label” 125, el nombre que se usará será “Procedimiento_Label125”.
- Versión: normalmente todos los proyectos utilizan un sistema de manejo de versiones, con el cual se pretende tener control de las diferentes versiones que se van generando de cada archivo. En el caso de los procedimientos de prueba, estos pueden irse actualizando en caso de que después de un proceso de revisión se encuentre que el procedimiento carece de información suficiente para cubrir completamente los requerimientos

relacionados; también pueden cambiar debido a modificaciones en los requerimientos o cambios en el software. Para llevar control de la versión se especifica un carácter numérico iniciando desde 1 hasta n, donde n es la versión final del documento.

- Autor: el nombre de la persona que realiza las pruebas, el formato en el que debe de estar realmente carece de importancia, para fines prácticos en la presente investigación se utilizará el formato primer nombre más primer apellido. Por ejemplo: Francisco Sánchez.
- Fecha: la fecha en que se terminó de realizar el procedimiento de prueba, esto sirve para tener un registro de cuando se generó la versión del documento. El formato así como el formato de autor, carece de importancia siempre y cuando sea un formato entendible. Para fines de la presente investigación el formato que se utilizará es dd/mm/aaaa. Por ejemplo: 28/05/2014.
- HLR_ID: identificador del requerimiento de alto nivel, normalmente cuando se redactan los requerimientos de alto nivel, estos deben llevar un identificador que los distinga de los demás, con el fin de tener una trazabilidad entre los casos de prueba y los requerimientos de alto nivel que lo cubren, es necesario especificar dichos identificadores. El formato del identificador varía dependiendo la nomenclatura descrita por el arquitecto de software. Para fines de la presente investigación el formato será: HLR_xxxx, donde xxxx es un carácter numérico que va desde el 1 hasta n, donde n es el último requerimiento creado. Por ejemplo: HLR_0001.
- Descripción: una breve descripción de lo que se pretende validar con el conjunto de casos de prueba pertenecientes al procedimiento de pruebas. Se recomienda especificar el tipo de “label” y el tipo de pruebas que se esté validando. Por ejemplo: “Este procedimiento pretende validar pruebas de rango normal y anormal para la “label” BCD 150”. En esta descripción se puede especificar detalles que sean relevantes a los casos de prueba o que simplemente el autor considere importante notar.

- Versión de software: es un identificador el cual indica el nivel de desarrollo de software, normalmente este número incrementa y se compone de al menos dos números separados por un punto. Ejemplo: Versión 1.0, Versión 1.1, etc.
- Versión de documento de requerimientos: los requerimientos sufren cambios conforme se avanza en el desarrollo de software, estos cambios pueden ser actualizaciones, eliminaciones o agregaciones de requerimientos. Por ello es necesario especificar la versión del documento de requerimientos el cual obtiene un punto de partida para empezar a probar los requerimientos.

La siguiente tabla contiene un ejemplo llenado de un encabezado de procedimiento de prueba terminado:

Tabla 4-2 Ejemplo de encabezado de procedimiento de prueba.

Plantilla genérica para casos de prueba A429	
Nombre de procedimiento de prueba	Procedimiento_Label125
Versión	1
Autor	Francisco Sánchez
Fecha	28/05/2014
HLR_ID	HLR_0001, HLR_0002
Descripción	Este procedimiento cubre las pruebas de rango normal y anormal de la label BCD 125.
Versión de Sofwate	Versión 1.0
Versión de Documento de requerimientos	2

La siguiente parte del plantilla, son los casos de prueba, la tabla 4-3 ilustra los campos necesarios para la implementación de los casos de prueba:

Tabla 4-3 Plantilla de casos de prueba.

	Entradas				Salidas	Descripción de caso de prueba
	Entrada A	Entrada A	..	Entrada N	Label_XXX	
CP1						
CP2						
....						
CPn						

El plantilla de los casos de prueba es una matriz que relaciona el nombre de las entradas y salidas que especifican los requerimientos de alto nivel con los identificadores de los casos de prueba.

A continuación se definen cada una de las partes del plantilla de casos de prueba:

- Entradas: cada una de las entradas especificadas en los requerimientos de alto nivel y/o diccionario de datos.
- Salidas: cada una de las salidas especificadas en los requerimientos de alto nivel y/o diccionario de datos.
- Identificador de casos de prueba: el identificador sirve para distinguir un caso de prueba de otro, un procedimiento de prueba puede tener un o más casos de prueba relacionados, por lo cual es importante saber cómo distinguir uno de otro. El formato que debe de llevar varía dependiendo de la nomenclatura especificada por el arquitecto de software, sin embargo para fines de esta investigación el formato utilizado será CP seguido de un número, empezando desde el número 1 hasta n, donde n es el último caso de prueba relacionado al procedimiento de prueba.
- Descripción de caso de prueba: la finalidad de este campo es especificar brevemente la finalidad del caso de prueba. Se puede detallar información como que tipo de prueba es la realizada, ya sea rango normal o anormal, y así mismo cualquier información que el desarrollador considere importante comentar. Por ejemplo: “Caso de prueba que cubre el rango mínimo de la

variable “Label_ID”, Caso de prueba que cubre un rango anormal para la variable “Label_ID”.

La siguiente tabla ejemplifica cómo se debe llenar una plantilla de casos de prueba:

Tabla 4-4 Ejemplo de plantilla de casos de prueba.

	Entradas				Salidas	Descripción de caso de prueba
	Label_ID	Longitud	SDI	SSM	Label_010	
CP1	10	-180	0	Normal	0x60000410	Rango mínimo soportado
CP2	10	-90	0	Normal	0xE002BC10	Valor entre 0 y valor mínimo
CP3	10	0	0	Normal	0x70000010	Valor es igual a 0
CP4	10	90	0	Normal	0x7F000010	Valor entre 0 y valor máximo
CP5	10	180	0	Normal	0x7FFFFC10	Rango máximo soportado

4.3 Definición de hoja de resultados hexadecimales.

Para obtener las salidas de cada una de las palabras ARINC, independientemente de su tipo (BNR, BCD o discretas), se puede establecer una hoja de cálculo donde se pueda obtener rápidamente el valor hexadecimal de la palabra manipulando cada una de sus partes como un valor independiente.

Esta hoja, facilitará la manera en cómo se obtienen los resultados, pues en vez de calcular cada combinación de manera manual, se puede obtener automáticamente a través de simples fórmulas de Excel ®.

La hoja de Excel propuesta consta de tres partes principales:

- Encabezado: la finalidad de esta parte de la hoja de cálculo es que el usuario proporcione información para que se calculen los resultados en hexadecimal. La información proporcionada puede ser el identificador del “label”, del SSM, del SDI y los datos.

- Representación binaria: la representación binaria sirve como apoyo visual al usuario de cómo se está formando la palabra A429 determinada y así mismo realiza algunas funciones como el cálculo del bit de paridad y la inversión de los bits en el caso del “label”, pues es bit más significativo del “label” es el primer bit de la palabra.
- Resultados: Aquí se detalla la salida hexadecimal de la palabra con los campos especificados en el encabezado.

4.3.1 Hoja de resultados, Encabezado

La siguiente figura especifica el formato del encabezado de la hoja de resultados:

Tabla 4-5 Encabezado de hoja de cálculo.

Label	271	Valor de label en octal (Rango 0-377)	Representación Binaria	10111001
SDI	0	Valor de SDI en entero (Rango 0-2)		00
SSM	0	Valor de SSM en entero (Rango 0-2)		00

A continuación se define cada una de las partes del encabezado de la hoja de resultados:

- Entradas del usuario: en la parte superior izquierda existen 3 celdas marcadas en azul, aquí el usuario específico el valor que desea ingresar. Los campos básicos son “Label”, SDI y SSM. Aunque en algunas ocasiones se puede ingresar también los datos, en caso en que el usuario desee obtener un resultado en específico. La parte de en medio son simplemente instrucciones que le dicen al usuario que tipo de dato es el que tiene que ingresar para evitar errores en los cálculos.
- Representación binaria: en la parte superior derecha existe la representación binaria que el usuario ingresa, para el caso de la “label” se utiliza la función:

Fórmula 4-1 Celda de "label".

$$OCT.A.BIN(Celda\_entrada\_label, 8)$$

Y para los demás campos se utiliza la función:

Fórmula 4-2 Decimal a Binario

$$DEC.A.BIN(Entrada\_n, 2)$$

Estos datos solamente sirven de referencia para la siguiente parte de la hoja, representación binaria.

4.3.2 Hoja de resultados, Representación binaria

La siguiente figura especifica el formato de la representación binaria en la hoja de resultados:

Figura 4-2 Representación binaria, hoja de resultados.

P	SSM		Datos																			SDI		Label ID							
32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	1	0	0	0	0

A continuación se define cada una de las partes que conforma la representación binaria en la hoja de resultados:

- Encabezados. Los encabezados representan el nombre de cada uno de los campos que conforman los 32 bits de cada palabra A429. En el caso del bit 32 (Bit de paridad) existe una función de Excel que autocompleta con 0 o 1 con el fin de que toda la palabra sea impar, esta fórmula es:

Fórmula 4-3 Cálculo de paridad.

$$SI(RESIDUO(SUMA(BIT31{:}BIT1),2) = 1,0,1)$$

Así mismo usando las representaciones binarias definidas en el encabezado se utiliza la función EXTRAE, la cual obtiene los caracteres especificados de una celda determinada, estas funciones son colocadas en

los bits correspondientes al SSM, SDI y “Label” ID, en este último los bits se colocan de manera inversa, pues el bit más significativo del campo “Label” es el primer bit de la palabra acorde con la especificación A429 (AIRLINES ELECTRONIC ENGINEERING COMMITTEE, 2004).

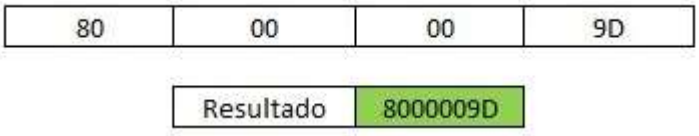
- Numeración de bits: el segundo renglón es simplemente una numeración de los 32 bits de la palabra A429 leída de derecha a izquierda.
- Representación binaria: el tercer renglón especifica la combinación de los 32 bits necesarios para representar la entrada del usuario.

4.3.3 Hoja de resultados, Resultados

La última parte de la hoja de cálculos, son los resultados en hexadecimal, la cual convierte a hexadecimal los resultados binarios obtenidos en la representación binaria de la palabra.

La siguiente figura especifica los campos que los resultados deben llevar:

Figura 4-3 Hoja de cálculos, Resultados.



La hoja de resultados para las celdas ubicadas en la parte superior utiliza la fórmula:

Fórmula 4-4 Binario a hexadecimal.

$$BIN.A.HEX(CONCATENAR(B8, C8, D8, E8, F8, G8, H8, I8),2)$$

Esta fórmula obtiene de byte en byte los bits especificados en la representación binaria, esto debido a que Excel no puede representar valores muy altos en hexadecimal en una sola fórmula.

La celda marcada en verde, es la celda del resultado esperado, esta simplemente es una concatenación de los resultados de los 4 campos de arriba.

Dependiendo de cómo el software este implementado, los resultados pueden especificarse en hexadecimal o en binario, para fines de reducir el tamaño de información y de facilitar la lectura, se decidió tomar el lenguaje hexadecimal como sistema de numeración para obtener las salidas de cada palabra A429.

La siguiente figura muestra las tres partes de la hoja de cálculos unida:

Figura 4-4 Hoja de cálculo y obtención de datos hexadecimales.

Label	271	Valor de label en octal (Rango 0-377)	Representación Binaria	10111001
SDI	0	Valor de SDI en entero (Rango 0-2)		00
SSM	0	Valor de SSM en entero (Rango 0-2)		00

P	SSM		Datos																	SDI		Label ID							
32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	1	0	1

80	00	00	9D

Resultado	8000009D

4.4 Definición de casos de prueba para palabras A429 BNR

4.4.1 Requerimientos de alto nivel y diccionario de datos base para palabras BNR.

Para ejemplificar los casos de prueba para una palabra A429 BNR, se utilizará una palabra estandarizada por la especificación A429 (AIRLINES ELECTRONIC ENGINEERING COMMITTEE, 2004), la palabra BNR que contiene el parámetro de navegación: “Groundspeed” el cual se define cómo la velocidad del avión sobre la superficie terrestre. Acorde con la especificación esta palabra cuenta con las siguientes características:

- Valor de “label” en octal 312.

- Rango de datos de 0 a 4095.875 nudos.
- La resolución de datos es de 0.125. Lo que significa que por cada bit prendido en el campo de datos equivaldrá a 0.125.
- El rango de datos utilizará únicamente del bit 14 al 29 para representar el valor, los bits 11 al 13 serán usados como “padding” y siempre deberán ser igual a 0.

Suponiendo que la etapa de revisión de requerimientos de alto nivel ha sido terminada, se proponen los siguientes requerimientos que especifican la funcionalidad del empaquetamiento de datos para la palabra “GroundSpeed”:

Tabla 4-6 Requerimientos de alto nivel, palabra BNR, GroundSpeed.

HLR-ID	Descripción
HLR-0001	El software debe asignar el bit 32 de Label_312_Ground_Speed si la suma de los números es impar.
HLR-0002	El software debe asignar 1 el bit 32 de Label_312_Ground_Speed si la suma de los números es par.
HLR-0003	El software debe asignar los bits 30 y 31 (SSM) de Label_312_Ground_Speed a la entrada SSM_valor.
HLR-0004	En caso que SSM_Valor sea igual a 0 o 1, el software deberá asignar los bits 14 a 29 de Label_312_Ground_Speed a 0.
HLR-0005	El software debe asignar los bits 14 a 29 de Label_312_Ground_Speed a la representación BNR de la variable Ground_Speed_Valor.
HLR-0006	El software debe asignar los bits 11 al 13 de Label_312_Ground_Speed a cero.
HLR-0007	El software debe asignar los bits 9 y 10 (SDI) de Label_312_Ground_Speed a la entrada SDI_valor.
HLR-0008	El software debe asignar los bits 1 al 8 de Label_312_Ground_Speed a la variable Label_Ground_Speed_Const convertida a formato Label A429.
HLR-0009	En caso de que se detecte que la variable SSM_Valor o SSM_SDI no se encuentren entre el rango 0 a 3, el software deberá asignar los bits 9 al 31 a 0.

Y así mismo se propone el siguiente diccionario de datos para el empaquetamiento de datos de la palabra A429 GroundSpeed:

Tabla 4-7 Diccionario de datos, palabra BNR, GroundSpeed.

Empaquetamiento de datos, label 312, ground speed.		
Entradas		
Item	Tipo de Dato	Descripción
SSM_Valor	Int32	Valor del SSM
Ground_Speed_Valor	Int32	Valor de ground speed en nudos
Label_Ground_Speed_Const	Constante	Constante Octal= 312
SDI_Valor	Int32	Valor del SDI
Salidas		
Item	Tipo de Dato	Descripción
Label_312_Ground_Speed	ARINC429_Word	Palabra ARINC429 con label 312, ground speed.

Tanto la tabla de diccionario de datos como la tabla de requerimientos de alto nivel, serán las entras para el proceso de validación, pues en la tabla de requerimientos se especifica el comportamiento del software y en el diccionario de datos se especifican las entradas y salidas que el software utiliza para formar la palabra A429 GroundSpeed.

Es importante resaltar que los primeros 8 requerimientos definen una funcionalidad del software, mientras que el requerimiento 9 y 10, definen una funcionalidad anormal, dado que es un mecanismo de defensa para cuando las entradas se encuentran fuera de rango.

4.4.2 Casos de prueba, rango normal.

Una vez que tenemos definidas las entradas para el proceso de validación, los requerimientos de alto nivel y el diccionario de datos correspondientes, se puede empezar a realizar el procedimiento de prueba que cubra la funcionalidad que dictamina los requerimientos de alto nivel.

Este proceso constará de dos etapas principales:

- Completar el documento de procedimiento de prueba, especificando las entradas y salidas de cada uno de los casos de prueba, así como completar el encabezado del procedimiento de prueba.

- Utilizar la hoja de cálculos para obtener las salidas de cada uno de los casos de prueba.

Para cubrir con la funcionalidad de los requerimientos de alto nivel de la “label” BNR 312 GroundSpeed, se definen el siguiente procedimiento de prueba:

Tabla 4-8 Procedimiento de prueba, palabra BNR 312, GroundSpeed.

Empaquetamiento de datos, palabra BNR, Ground Speed.					
Nombre de procedimiento de prueba	Procedimiento_Label312				
Versión	1				
Autor	Francisco Sánchez				
Fecha	28/05/2014				
HLR_ID	HLR_0001, HLR_0002, HLR_0003, HLR_0004, HLR_0005, HLR_0006, HLR_0007, HLR_0008, HLR_0009, HLR_0010				
Descripción	Este procedimiento pretende cubrir pruebas de rango normal y anormal para la label BNR 312, Ground Speed.				
Versión de software	Versión 2.0				
Versión de Documento de requerimientos	3				
	Entradas			Salidas	Descripción de caso de prueba
Caso de prueba	Ground_Speed_valor	SDI_valor	SSM_valor	Label_312_Ground_Speed	
CP1	4095.875	0	3	0x6FFFE053	Rango máximo de Ground_Speed_valor. Valida cuando SSM es 3. Valida bits 11 al 13 en cero. Valida rango mínimo de SDI
CP2	2048	3	2	0x48000353	Valida cuando el SSM es 2 Valida Rango intermedio de Ground_Speed_Valor Valida valor máximo de SDI Valida cuando el bit de paridad es 0.
CP3	4095.875	3	1	0xA0000353	Valida cuando el SSM es 1
CP4	4095.875	3	0	0x80000353	Valida cuando el SSM es 0
CP5	0	0	3	0xE0000053	Rango mínimo de Ground_Speed_valor Valida reperesentación de label octal 312. Valida cuando el bit de paridad es 1.
CP6	10	3	4	0x80000353	Valor fuera de rango de la variable SSM
CP7	-1024	3	3	0x80000353	Valor fuera de rango de la variable Ground_Speed_Valor
CP8	10	4	3	0x80000053	Valor fuera de rango de la variable SDI

Como se puede observar, el procedimiento de prueba consta de únicamente 8 casos de prueba para cubrir 10 requerimientos de alto nivel, esto debido a que se puede probar más de un requerimiento a la vez si se desarrollan casos de prueba que sean observables para cubrir más de un requerimiento.

A continuación se analiza cada uno de los casos de prueba con su respectiva representación binaria con el fin de explicar cómo es que se cubren todos los casos de prueba, tanto rangos normales como rangos anormales.

Caso de prueba 1: Las entradas a este caso de prueba son:

- Ground_Speed_Valor = 4095.875 (valor máximo)
- SDI_valor = 0 (valor mínimo)
- SSM_Valor = 3 (valor máximo)

La representación binaria de este caso de prueba es el siguiente:

Figura 4-5 Representación binaria, caso de prueba 1, palabra BNR.

P	SSM		Datos																		SDI		Label ID							
32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	1	0	1	0	0	1	1

6F	FF	E0	53

Resultado	6FFFE053

Con este caso de prueba se cubre la siguiente funcionalidad:

- Rango máximo de Ground_Speed_Valor, se puede notar que el valor 4095.875 es representado con los bits 14 al 28 prendidos y el bit 29 apagado, cuando el bit 29 está apagado significa que el signo es positivo, por lo tanto se cubre correctamente.
- Rango máximo de la variable SSM_Valor: se puede notar el bit 30 y 31 prendidos, lo cual equivale a un 3, de acuerdo al protocolo A429 cuando el SSM es 3 equivale al estado “Normal Operation”, el cual significa que la señal está siendo mandado sin ningún tipo de falla.
- Bits 11 al 13 se encuentren apagados o asignados a cero. En la representación binaria de muestra que los bits 11 al 13 están asignados a cero.

- Valor mínimo de la variable SDI_Valor. Los bits 9 y 10 esta apagados por lo cual se cubre el rango mínimo.

  El caso de prueba 2 consta de las siguientes entradas:

- Ground_Speed_Valor = 2048 (valor intermedio entre máximo y mínimo)
- SDI_Valor = 3 (valor máximo)
- SSM_Valor = 2 (valor equivalente a Functional Test)

  La representación binaria de este caso de prueba es el siguiente:

Figura 4-6 Caso de prueba 2, palabra BNR.

P	SSM		Datos																		SDI		Label ID							
32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0	1	0	0	1	1

48	00	03	53

Resultado	48000353

La funcionalidad que cubre el caso de prueba 2 es la siguiente:

- Valor de SSM_Valor cuando representa Functional Test o 2, según el protocolo ARINC este es un valor válido para empaquetamiento de datos, en algunos sistemas este valor es irrelevante, pues el software nunca lo utiliza, sin embargo en ocasiones es necesario comprobar que los datos se empaqueten correctamente en este estado. Los bits 30 y 31 se encuentran representados a la combinación equivalente a 2.
- Rango intermedio de Ground_Speed_Valor, la representación BNR de un entero se prueba con sus valores máximos, mínimos y algún rango intermedio para asegurar que soporte cualquier tipo de valor en el rango completo. En este caso el valor que se prueba es 2048 y se puede notar que el bit 28 esta prendido, por lo cual se cubre correctamente.

- Valor máximo de SDI_Valor. Se puede notar que el bit 9 y 10 se encuentran prendidos, por lo cual equivale a un 3.
- Valida que el bit 32 o bit de paridad es cero cuando la suma de los bits 1 a 31 son impar. Por lo tanto no es necesario ajustar la paridad de la palabra.

El caso de prueba 3 y 4 cubren el HLR-0004, el cual especifica que en caso de que el SSM sea 1 o 0, se asignarán los valores a 0 sin importar sus entradas, esto es debido a que acorde con la especificación A429, estos valores representan “No computed data” y “Failure Warning” respectivamente, en otras palabras son estados inválidos para la palabra y por ende el software debe estar preparado para seguir transmitiendo los datos a manera de falla.

Las entradas para los casos de prueba 3 y 4 son las siguientes:

- Ground_Speed_Valor = 4095.875
- SDI_valor = 3
- SSM_Valor = 1 para caso de prueba 3 y 2 para caso de prueba 4.

Pese a que las entradas de los bits correspondientes a la representación numérica BNR contienen información diferente a 0, la salida representará dichos valores en 0, pues el requerimiento así lo estipula. Se puede apreciar a detalle en las siguientes figuras:

Figura 4-7 Caso de prueba 3, palabra BNR, SSM = NCD.

P	SSM		Datos																		SDI		Label ID							
32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0	1	0	0	1	1

A0	00	03	53

Resultado	A0000353

Figura 4-8 Caso de prueba 4, palabra BNR, SSM=FW.

P	SSM		Datos																		SDI		Label ID							
32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0	1	0	0	1	1

80	00	03	53

Resultado	80000353

Caso de prueba 5: Las entradas de este caso de prueba son las siguientes:

- Ground_Speed_Valor = 0(Valor mínimo)
- SDI_valor = 0
- SSM_Valor = 3

La funcionalidad que se cubre con este caso de prueba es la siguiente:

- Rango mínimo de la variable Ground_Speed_Valor, se puede observar que los bits 14 a 29 se encuentran asignados a 0, lo cual representa el valor mínimo 0.
- Valida la representación octal 312 de la palabra GroundSpeed, realmente cualquier caso de prueba validar esto, debido a que en todos los casos de prueba los primeros 8 bits son el mismo.
- Valida cuando el bit de paridad o bit 32 es 1 cuando la suma de los bits 1 al 31 es par.

La representación binaria del caso de prueba 5 es la siguiente:

Figura 4-9 Caso de prueba 5, palabra BNR.

P	SSM		Datos																		SDI		Label ID							
32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	1	1

E0	00	00	53

Resultado	E0000053

4.4.3 Casos de prueba de robustez

Todo caso de prueba de robustez debe ser basado en un requerimiento de alto nivel, en este caso los requerimientos de alto nivel HLR-0008, HLR-0009 y HLR-0010 especifican como se debe comportar el software en caso de que reciba un valor fuera de rango, en este caso los bits 11 al 31 se deben colocar en 0 a excepción de cuando el SDI es el que esta fuera de rango, para ello los bits 9 y 10 deberán ser cero, esto debido a que si el SDI está fuera de rango entonces el software no sabría que dispositivo enviar la información por ello debe viajar con el valor por default que en este caso es 0.

Los casos de prueba 6, 7 y 8 cubren estos escenarios, las entradas fuera de rango de estos casos de prueba son las siguientes:

- Caso de prueba 6: SSM = 4 (Rango normal de 0 a 3).
- Caso de prueba 7: Ground_Speed_valor = -1024 (Rango normal de 0 a 4095.875).
- Caso de prueba 8: SDI = 4 (Rango normal de 0 a 4).

La representación binaria de los casos 6 y 7 son exactamente las mismas pues todos los bits se asignan a 0 a excepción de los bits 1 al 10 correspondientes al “label” y al SDI, en caso del caso de prueba 8 será muy similar a excepción de los bits 9 y 10 pues acorde con el requerimiento también deberán ser expresados

en 0. En las siguientes figuras se puede apreciar más a detalle la representación de cada uno de los bits:

Figura 4-10 Caso de prueba 6 y 7, Pruebas de robustez.

P	SSM		Datos																		SDI		Label ID							
32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0	1	0	0	1	1

80	00	03	53

Resultado	80000353

Figura 4-11 Caso de prueba 8, prueba de robustez, palabra BNR.

P	SSM		Datos																		SDI		Label ID							
32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	1	1

80	00	00	53

Resultado	80000053

4.4.4 Matriz de trazabilidad entre Requerimientos de alto nivel y casos de prueba.

Una vez que se han terminado de realizar los casos de prueba, una manera de asegurar que los requerimientos de alto nivel se han cubierto es mediante una matriz que especifique que casos de prueba cubren determinados requerimientos de alto nivel.

Para ello simplemente relacionamos en una tabla especificando el requerimiento en la columna 1 y los casos de prueba en la columna 2 como representa la siguiente figura:

Tabla 4-9 Matriz de trazabilidad, Palabra BNR.

HLR	Caso de prueba
HLR-0001	Procedimiento_Label312.CP2
HLR-0002	Procedimiento_Label312.CP5
HLR-0003	Procedimiento_Label312.CP1, Procedimiento_Label312.CP2, Procedimiento_Label312.CP3, Procedimiento_Label312.CP4
HLR-0004	Procedimiento_Label312.CP3, Procedimiento_Label312.CP4
HLR-0005	Procedimiento_Label312.CP1, Procedimiento_Label312.CP2, Procedimiento_Label312.CP5
HLR-0006	Procedimiento_Label312.CP1
HLR-0007	Procedimiento_Label312.CP1, Procedimiento_Label312.CP2
HLR-0008	Procedimiento_Label312.CP6
HLR-0009	Procedimiento_Label312.CP7
HLR-0010	Procedimiento_Label312.CP8

Como se puede observar todos los requerimientos de alto nivel han sido completados por el procedimiento de prueba, con esto se da concluido la generación del procedimiento de prueba y con ello podrá pasar a una etapa de revisión y de ejecución del procedimiento para validar que efectivamente el software se comporta conforme a los requerimientos de alto nivel.

4.5 Definición de casos de prueba para palabras A429 BCD

La definición de casos de prueba para palabras A429 BCD es muy similar a los casos de prueba para palabras BNR. Antes de empezar con la definición de requerimientos de alto nivel y diccionario de datos, es necesario mencionar las principales diferencias entre las palabras BCD y BNR:

- El SSM para las palabras BCD solamente cuenta con estado inválido “No Computed Data” (NCD) que equivale a uno, a diferencia de las palabras BNR que cuentan con dos valores inválidos.
- El formato en el que se representan los datos es BCD a diferencia de las palabras BNR que se representan binariamente en complemento A2.

Para ejemplificar los casos de prueba para una palabra A429 BCD, se utilizará una palabra estandarizada por la especificación A429 (AIRLINES ELECTRONIC ENGINEERING COMMITTEE, 2004), la palabra BCD que contiene el parámetro de navegación “Universal Time Constant” (UTC), la cual contiene el tiempo estandarizado en un formato dado para el “Global Positioning System” (GPS). Acorde con la especificación, esta palabra cuenta con las siguientes características:

- Valor de “label” en octal 125.
- Rango de datos de 00:00.0 a 23:59.9.
- Cada dígito del reloj se representa binariamente en un campo BCD por separado, por ende existen 5 campos BCD entre los bits 11 al 29.

Para comprender más a detalle cómo se representan los datos de tiempo en la “label” 125, la figura siguiente ejemplifica la hora 12:30.5.

Figura 4-12 Ejemplo representación BCD, UTC.

P	SSM		Datos																		SDI		Label ID							
32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
1	0	0	0	0	1	0	0	1	0	0	0	1	1	0	0	0	0	0	1	0	1	0	0	1	0	1	0	1	0	1	0
			1			2				3				0				5													

Como se puede observar existe un cuarto renglón en la sección de datos, el cual obtiene los valores binarios de cada campo BCD y lo transforma a decimal. Para ello se utiliza la función de Excel de conversión de binario a decimal donde los números de celda son cada uno de los binarios que contiene el campo BCD, tal como lo especifica la siguiente fórmula:

Fórmula 4-5 Concatenar y convertir binarios a decimal.

$$BIN.A.DEC(CONCATENAR(Celda1, Celda2, Celda3, Celda4))$$

Acorde con la especificación el tiempo se representa de la siguiente manera:

- Bits 11 a 14: Décimas de segundo.
- Bits 15 a 18: Unidades de minuto.
- Bits 19 a 22: Decenas de minuto.
- Bits 23 a 26: Unidades de hora.
- Bits 27 a 29: Decenas de hora.

4.5.1 Requerimientos de alto nivel y diccionario de datos base para palabras BCD.

Suponiendo que la etapa de revisión de requerimientos de alto nivel ha sido terminada, se proponen los siguientes requerimientos que especifican la funcionalidad del empaquetamiento de datos para la palabra “UTC”:

Tabla 4-10 Requerimientos de alto nivel, palabra BCD, UTC.

HLR-ID	Descripción
HLR-0100	El software debe asignar 0 el bit 32 de Label_125_UTC si la suma de los bits 0 a 31 es impar.
HLR-0101	El software debe asignar 1 el bit 32 de Label_125_UTC si la suma de los bits 0 a 31 es par.
HLR-0102	El software debe asignar los bits 30 y 31 (SSM) de Label_125_UTC a la entrada SSM_valor.
HLR-0103	En caso que SSM_Valor sea igual a 1, el software deberá asignar los bits 11 a 29 de Label_125_UTC a 0.
HLR-0104	El software debe asignar los bits 11 a 29 de Label_125_UTC a la representación BCD del tiempo del sistema acorde con la siguiente información: - Bits 11 a 14: Décimas de segundo. - Bits 15 a 18: Unidades de minuto. - Bits 19 a 22: Decenas de minuto. - Bits 23 a 26: Unidades de hora. - Bits 27 a 29: Decenas de hora.
HLR-0105	El software debe asignar los bits 9 y 10 (SDI) de Label_125_UTC a la entrada SDI_valor.
HLR-0106	El software debe asignar los bits 1 al 8 de Label_125_UTC a la variable Label_UTC_Const convertida a formato Label A429.
HLR-0107	En caso de que se detecte que la variable SSM_Valor o SSM_SDI no se encuentren entre el rango 0 a 3, el software deberá asignar los bits 9 al 29 a 0 y el SSM a 1.
HLR-0108	En caso de que se detecte que cada alguna de las variables de tiempo este fuera de rango permitido, el software deberá asignar los bits 9 al 29 a 0 y el SSM a 1. Los rangos permitidos son los siguientes: - Decima_Segundo: Rango 0 a 9. - Unidad_Minuto: Rango 0 a 9. - Decena_Minuto: Rango 0 a 5. - Unidad_Hora: Rango 0 a 9. - Decena_Hora: Rango 0 a 2. El rango máximo expresado por ende es 23:59.9

Como se puede observar las diferencias entre palabras BNR y BCD son especificadas en los requerimientos de alto nivel. En primera instancia cuando se requiere mandar una falla, el SSM se cambia a 1 (NCD) y los bits de datos a 0 y así mismo los requerimientos relevantes a la representación binaria de los datos cambian, pues ahora existen 5 variables para los datos, una para representar cada campo BCD de la palabra.

De la misma manera se propone el siguiente diccionario de datos para el empaquetamiento de datos de la palabra A429 UTC:

Tabla 4-11 Diccionario de datos, palabra BCD, UTC.

Empaquetamiento de datos, label 125, UTC.		
Entradas		
Item	Tipo de Dato	Descripción
SSM_Valor	Int32	Valor del SSM
Decima_Segundo	Int32	Decimas de segundo del sistema
Unidad_Minuto	Int32	Unidades de minuto del sistema
Decena_Minuto	Int32	Decenas de minuto del sistema
Unidad_Hora	Int32	Unidades de hora del sistema
Decena_Hora	Int32	Decenas de hora del sistema
Label_UTC_Const	Constante	Constante Octal= 125
SDI_Valor	Int32	Valor del SDI
Salidas		
Item	Tipo de Dato	Descripción
Label_125_UTC	ARINC429_Word	Palabra ARINC429 con label 312, ground speed.

Para cubrir con la funcionalidad de los requerimientos de alto nivel de la “label” BCD 125 UTC, se definen el siguiente procedimiento de prueba:

Tabla 4-12 Procedimiento de prueba, palabra BCD 125, UTC.

Casos de prueba	Entradas							Salidas	Descripción de caso de prueba
	Decena_Hora	Unidad_Hora	Decena_Minuto	Unidad_Minuto	Decima_segundo	SDI_valor	SSM_valor	Label_125_UTC	
CP1	0	9	5	9	0	0	0	0x825640AA	Rangos normales de variables de tiempo. SDI y SSM rango 0 Bit de paridad = 1
CP2	2	3	0	5	5	1	2	0x48C155AA	Rangos normales de variables de tiempo. Rango SDI 1 y SSM rango 2 Bit de paridad = 0
CP3	1	0	3	0	9	2	3	0x640C26AA	Rangos normales de variables de tiempo. Rango SDI 2 y SSM rango 3
CP4	2	3	5	9	9	3	0	0x08D667AA	Rangos máximos normales de las variables de tiempo. Rango SDI 3
CP5	2	3	5	9	9	0	1	0xA00000AA	SSM = NCD
CP6	3	3	5	9	9	0	0	0xA00000AA	Campo 1 BCD inválido
CP7	2	10	5	9	9	0	0	0xA00000AA	Campo 2 BCD inválido Label 125 en el formato correcto.
CP8	2	3	10	9	9	0	0	0xA00000AA	Campo 3 BCD inválido
CP9	2	3	5	10	9	0	0	0xA00000AA	Campo 4 BCD inválido
CP10	2	3	5	9	10	0	0	0xA00000AA	Campo 5 BCD inválido
CP11	2	9	5	9	9	0	0	0xA00000AA	Combinación de tiempo inválido.
CP12	2	3	5	9	9	4	0	0xA00000AA	SDI fuera de rango.
CP13	2	3	5	9	9	0	4	0xA00000AA	SSM fuera de rango.

4.5.2 Casos de prueba, rango normal.

A continuación se analiza cada uno de los casos de prueba con su respectiva representación binaria con el fin de explicar cómo es que se cubren todos los ellos, tanto rangos normales como rangos anormales.

Los rangos normales se verifican en los primeros 3 casos de prueba de manera independiente y en el caso de prueba 4, el máximo de todas las variables de tiempo (23:59.9).

En la siguiente tabla se puede observar las combinaciones de los primeros casos de prueba 1 a 4, que pretenden probar el rango normal de las variables relacionados a los requerimientos de alto nivel:

Tabla 4-13 Detalle de casos de prueba 1 a 4, palabra BCD.

	Entradas						
Caso de prueba	Decena_Hora	Unidad_Hora	Decena_Minuto	Unidad_Minuto	Decima_segundo	SDI_valor	SSM_valor
CP1	0 (MIN)	9 (MAX)	5 (MAX)	9 (MAX)	0 (MIN)	0 (MIN)	0 (MIN)
CP2	2 (MAX)	3 (MID)	0 (MIN)	5 (MID)	5 (MID)	1	2 (MID)
CP3	1 (MID)	0 (MIN)	3 (MID)	0 (MIN)	9 (MAX)	2 (MID)	3 (MAX)
CP4	2	3	5	9	9	3 (MAX)	0

Como se puede observar cada una de las variables cubren su valor Mínimo (MIN), su valor máximo (MAX) y al menos un valor intermedio (MID). Si se redactan casos de prueba de manera independiente se puede probar más de un rango a la vez en un solo caso de prueba. En este caso, el rango de 7 variables diferentes fue probado en tan solo 4 casos de prueba.

El caso de prueba 4, valida que la salida máxima que el tiempo puede registrar (23:59.9) esto puede ser visto como un caso de robustez o un caso de rango normal, pues se está estresando al máximo el resultado final. Es altamente recomendable realizar este tipo de pruebas, pues tanto entradas como salidas deben de estresarse al máximo para cubrir completamente los requerimientos de alto nivel.

Para los requerimientos relacionados con el bit de paridad y la estructura del “label”, se puede especificar cualquiera de los casos que lo cubra de igual manera como se especificó en los casos de prueba de la palabra BNR. En este caso, acorde con la Tabla 3-7 los casos de prueba relacionados con el bit de paridad se cubren en los caso de prueba 1 y 2 respectivamente y el formato de

“label” en el caso de prueba número 7. En el caso del formato de “label”, el caso de prueba podría ser cualquiera de los 13, pues el valor siempre es asignado a una constante, lo importante es al menos seleccionar uno de los casos.

A continuación se detalla la representación binaria de los casos de prueba 1 a 4 para la palabra BCD 125 UTC:

Figura 4-13 Representación binaria, caso de prueba 1, palabra BCD.

P	SSM		Datos																			SDI		Label ID							
32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
1	0	0	0	0	0	1	0	0	1	0	1	0	1	1	0	0	1	0	0	0	0	0	0	1	0	1	0	1	0	1	0
			0			9				5				9				0													

82	56	40	AA

Resultado	825640AA

Figura 4-14 Representación binaria, caso de prueba 2, palabra BCD.

P	SSM		Datos																			SDI		Label ID							
32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
0	1	0	0	1	0	0	0	1	1	0	0	0	0	0	1	0	1	0	1	0	1	0	1	1	0	1	0	1	0	1	0
			2			3				0				5				5													

48	C1	55	AA

Resultado	48C155AA

Figura 4-15 Representación binaria, caso de prueba 3, palabra BCD.

P	SSM		Datos																			SDI		Label ID							
32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
0	1	1	0	0	1	0	0	0	0	0	0	1	1	0	0	0	0	1	0	0	1	1	0	1	0	1	0	1	0	1	0
			1			0				3				0				9													

64	0C	26	AA

Resultado	640C26AA

Figura 4-16 Representación binaria, caso de prueba 4, palabra BCD.

P	SSM		Datos																		SDI		Label ID							
32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
0	0	0	0	1	0	0	0	1	1	0	1	0	1	1	0	0	1	1	0	0	1	1	1	1	0	1	0	1	0	1	0
			2			3				5				9				9													

08	D6	67	AA

Resultado	08D667AA

4.5.3 Casos de prueba de robustez

Los casos de prueba 6 al 13, validan las condiciones anormales, para las cuales el software debe estar preparado. A continuación se detalla las condiciones anormales de cada uno de los casos de prueba.

- Caso de prueba 6: El campo BCD relacionado con las decenas de horas es igualado a 3, siendo que el rango normal es de 0 a 2, sin embargo la representación BCD de este campo podría soportar hasta 7 combinaciones de bits.
- Caso de prueba 7, 8, 9 y 10: El campo BCD relacionado a las variables de tiempo unidades de hora, decenas de minuto, unidades de minuto y décimas de segundo son igualados a 10, siendo que el rango normal es de 0 a 5 para las decenas de minuto y de 0 a 9 para el resto, sin embargo la representación BCD de estos campo podría soportar hasta 15 combinaciones de bits debido a que existen 4 bits relacionados a cada campo BCD.
- Caso de prueba 11: Existe algunas combinaciones, las cuales representan un tiempo inválido aunque los caracteres no excedan el rango normal permitido, este es el caso de cuando las horas son definidas entre el 24 y el 29. Como se puede apreciar en la tabla 3-7 el valor elegido para las horas es el 29, el cual no excede los rangos normales permitidos para cada uno

de sus campos BCD, sin embargo la combinación de ambos proporciona una hora inválida, pues la hora solo se puede representar entre 00 y 23.

- Caso de prueba 12 y 13: Prueba cuando el SDI y el SSM se encuentran fuera del rango normal respectivamente.

Para los casos de prueba 6 al 13, la salida esperada es la misma, pues es acorde con los requerimientos de alto nivel, el SSM se deberá asignar a 1 y los bits de datos a 0, así como lo muestra la siguiente figura:

Figura 4-17 Representación binaria, casos inválidos, palabra BCD.

4.5.4 Matriz de trazabilidad entre Requerimientos de alto nivel y casos de prueba.

Así como se hizo anteriormente para las palabras BNR, es necesario asegurar que todos los requerimientos de alto nivel han sido cubiertos, para ello relacionamos una tabla los requerimientos de alto nivel en la columna 1 y en los respectivos casos de prueba que lo cubren en la columna 2:

Tabla 4-14 Matriz de trazabilidad, Palabra BCD.

HLR	Caso de prueba
HLR-0100	Procedimiento_Label125.CP2
HLR-0101	Procedimiento_Label125.CP1
HLR-0102	Procedimiento_Label125.CP1, Procedimiento_Label125.CP2, Procedimiento_Label125.CP3, Procedimiento_Label125.CP5
HLR-0103	Procedimiento_Label125.CP5
HLR-0104	Procedimiento_Label125.CP1, Procedimiento_Label125.CP2, Procedimiento_Label125.CP3, Procedimiento_Label125.CP4
HLR-0105	Procedimiento_Label125.CP1, Procedimiento_Label125.CP2, Procedimiento_Label125.CP3, Procedimiento_Label125.CP4
HLR-0106	Procedimiento_Label125.CP12, Procedimiento_Label125.CP13
HLR-0107	Procedimiento_Label125.CP6
HLR-0108	Procedimiento_Label125.CP6, Procedimiento_Label125.CP7, Procedimiento_Label125.CP8, Procedimiento_Label125.CP9, Procedimiento_Label125.CP10, Procedimiento_Label125.CP11

4.6 Definición de casos de prueba para palabras A429 Discretas

Las palabras discretas al igual que las palabras BCD y BNR, transmiten información en un formato de 32 bits, sin embargo las palabras discretas no representan valores numéricos, sino su finalidad es representar el estado de diversos sensores, equipos, botones, etc. del avión.

Para ejemplificar los casos de prueba para una palabra A429 discreta, se utilizará una palabra estandarizada por la especificación A429 (AIRLINES ELECTRONIC ENGINEERING COMMITTEE, 2004), la palabra discreta Flight Control Computer (FCC) 271, la cual contiene la información relacionado a

diferentes acciones de vuelo. La siguiente tabla define cada uno de los 32 bits de la palabra FCC 271:

Tabla 4-15 Definición de palabra discreta FCC 271.

Bit No.	Función	Descripción	
1 a 8	Label	Octal 271	
9	Altitude Hold Mode	1 = Requested	0 = Not Requested
10	Altitude Select Mode	1 = Requested	0 = Not Requested
11	Vertical Speed Select Mode	1 = Requested	0 = Not Requested
12	Vertical Speed Hold Mode	1 = Requested	0 = Not Requested
13	Horizontal Navigation	1 = Requested	0 = Not Requested
14	Vertical Navigation	1 = Requested	0 = Not Requested
15	Land Command	1 = Requested	0 = Not Requested
16	LOC Approach Command	1 = Requested	0 = Not Requested
17	Back Course Approach Command	1 = Requested	0 = Not Requested
18	CWS #1	1 = Requested	0 = Not Requested
19	CWS #2	1 = Requested	0 = Not Requested
20	CWS #3	1 = Requested	0 = Not Requested
21	Pitch Upper Mode Cancel	1 = Requested	0 = Not Requested
22	Roll Upper Mode Cancel	1 = Requested	0 = Not Requested
23	Heading Hold	1 = Requested	0 = Not Requested
24 a 29	Spare	0	
30 a 31	SSM		
32	Paridad		

Como se puede apreciar existen algunas similitudes con las palabras A429 BNR y BCD:

- Bits 1 al 8 se refieren a los bits que representan el número de “label” en octal.
- Bits 30 y 31 representan el SSM.
- Bit 32 representan el bit de paridad.

Así mismo existen algunas diferencias:

- Los bits 9 y 10 que normalmente son usados para representar el SDI, ahora forman parte de la representación de datos de la palabra, esto no quiere

decir que todas las palabras discretas se comporten de este modo, esta palabra en particular así está constituida por el estándar, dado que la palabra no contiene SDI, será transmitida a todos los equipos conectados a la red.

- Los bits de datos no representan números, por ende no tiene signo de bit o bits relacionados a complemento A2 o campos BCD, etc. Los bits 9 al 23 representan si cada una de las acciones de vuelo ha sido requerida o no. Este tipo de bits normalmente son especificados a partir de un tipo de variable booleana.
- Otra diferencia que no se menciona en la tabla, es que el SSM utiliza un formato discreto como lo especifica la tabla 2-5, y el cual así como las palabras BNR contienen dos estados inválidos.

4.6.1 Requerimientos de alto nivel y diccionario de datos base para palabras Discretas.

Suponiendo que la etapa de revisión de requerimientos de alto nivel ha sido terminada, se proponen los siguientes requerimientos que especifican la funcionalidad del empaquetamiento de datos para la palabra discreta FCC 271:

Tabla 4-16 Requerimientos de alto nivel, palabra discreta FCC 211.

HLR-ID	Descripción
HLR-0200	El software debe asignar 0 el bit 32 de Label_271_FCC si la suma de los bits 0 a 31 es impar.
HLR-0201	El software debe asignar 1 el bit 32 de Label_271_FCC si la suma de los bits 0 a 31 es par.
HLR-0202	El software debe asignar los bits 30 y 31 (SSM) de Label_271_FCC a la entrada SSM_valor.
HLR-0203	En caso que SSM_Valor sea igual a 1 o 3, el software deberá asignar los bits 9 a 23 de Label_271_FCC a 0.
HLR-0204	El software debe asignar los bits 9 a 23 de Label_271_FCC acorde con la siguiente información: - Bit 9= Bit9Bool - Bit 10= Bit10Bool - Bit 11= Bit11Bool - Bit 12= Bit12Bool - Bit 13= Bit13Bool - Bit 14= Bit14Bool - Bit 15= Bit15Bool - Bit 16= Bit16Bool - Bit 17= Bit17Bool - Bit 18= Bit18Bool - Bit 19= Bit19Bool - Bit 20= Bit20Bool - Bit 21= Bit21Bool - Bit 22= Bit22Bool - Bit 23= Bit23Bool
HLR-0205	El software debe asignar los bits 24 a 29 a 0.
HLR-0206	El software debe asignar los bits 1 al 8 de Label_271_FCC a la variable Label_271_Const convertida a formato Label A429.
HLR-0207	En caso de que se detecte que la variable SSM_Valor no se encuentren entre el rango 0 a 3, el software deberá asignar los bits 9 al 29 a 0 y el SSM a 1.

Así mismo se propone el siguiente diccionario de datos:

Tabla 4-17 Diccionario de datos, palabra discreta, FCC 271.

Empaquetamiento de datos, label discreta FCC 271.		
Entradas		
Item	Tipo de Dato	Descripción
SSM_Valor	Int32	Valor del SSM
Bit9Bool	Boolean	Booleano del bit 9
Bit10Bool	Boolean	Booleano del bit 10
Bit11Bool	Boolean	Booleano del bit 11
Bit12Bool	Boolean	Booleano del bit 12
Bit13Bool	Boolean	Booleano del bit 13
Bit14Bool	Boolean	Booleano del bit 14
Bit15Bool	Boolean	Booleano del bit 15
Bit16Bool	Boolean	Booleano del bit 16
Bit17Bool	Boolean	Booleano del bit 17
Bit18Bool	Boolean	Booleano del bit 18
Bit19Bool	Boolean	Booleano del bit 19
Bit20Bool	Boolean	Booleano del bit 20
Bit21Bool	Boolean	Booleano del bit 21
Bit22Bool	Boolean	Booleano del bit 22
Bit23Bool	Boolean	Booleano del bit 23
Label_271_Const	Constante	Constante Octal= 271
Salidas		
Item	Tipo de Dato	Descripción
Label_271_FCC	ARINC429_Word	Palabra ARINC429 con label 271.

Al igual que los requerimientos de alto nivel y diccionario de datos para las palabras BNR y BCD, para el proceso de validación ambas tablas servirán como entradas para desarrollar los procedimientos de prueba.

Es importante destacar que las entradas correspondientes a los bits de datos (9 a 23) son representados por palabras booleanas, por lo cual no se pueden definir casos de robustez para ellos, puesto que un booleano por definición solo puede tomar los valores de 0 o 1.

4.6.2 Casos de prueba, rango normal.

Para cubrir con la funcionalidad de la “label” discreta FCC 271, se definen los siguientes casos de prueba:

Tabla 4-18 Procedimiento de prueba, palabra discreta, FCC 271.

	Entradas																Salidas	Descripción de caso de prueba
Casos de prueba	Bit 9B ool	Bit 10 Bool	Bit 11 Bool	Bit 12 Bool	Bit 13 Bool	Bit 14 Bool	Bit 15 Bool	Bit 16 Bool	Bit 17 Bool	Bit 18 Bool	Bit 19 Bool	Bit 20 Bool	Bit 21 Bool	Bit 22 Bool	Bit 23 Bool	SSM_ valor	Label_FCC_ 271	
CP1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0x0000019D	Aislación en TRUE de bit 9
CP2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0xC000029D	Aislación en TRUE de bit 10
CP3	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0x0000049D	Aislación en TRUE de bit 11
CP4	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0x0000089D	Aislación en TRUE de bit 12
CP5	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0x0000109D	Aislación en TRUE de bit 13
CP6	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0x0000209D	Aislación en TRUE de bit 14
CP7	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0x0000409D	Aislación en TRUE de bit 15
CP8	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0x0000809D	Aislación en TRUE de bit 16
CP9	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0x0001009D	Aislación en TRUE de bit 17
CP10	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0x0002009D	Aislación en TRUE de bit 18
CP11	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0x0004009D	Aislación en TRUE de bit 19
CP12	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0x0008009D	Aislación en TRUE de bit 20
CP13	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0x0010009D	Aislación en TRUE de bit 21
CP14	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0x0020009D	Aislación en TRUE de bit 22
CP15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0x0040009D	Aislación en TRUE de bit 23
CP16	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0x807FFE9D	Aislación en FALSE de bit 9
CP17	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0x807FFD9D	Aislación en FALSE de bit 10
CP18	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0x807FFB9D	Aislación en FALSE de bit 11
CP19	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	0	0x807FF79D	Aislación en FALSE de bit 12
CP20	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	0	0x807FEF9D	Aislación en FALSE de bit 13
CP21	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	0	0x807FDF9D	Aislación en FALSE de bit 14
CP22	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	0	0x807FBF9D	Aislación en FALSE de bit 15
CP23	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	0	0x807F7F9D	Aislación en FALSE de bit 16
CP24	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	0	0x807EFF9D	Aislación en FALSE de bit 17
CP25	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	0	0x807DFF9D	Aislación en FALSE de bit 18
CP26	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	0	0x807BFF9D	Aislación en FALSE de bit 19
CP27	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	0	0x8077FF9D	Aislación en FALSE de bit 20
CP28	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	0	0x806FFF9D	Aislación en FALSE de bit 21
CP29	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0	0x805FFF9D	Aislación en FALSE de bit 22
CP30	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0x803FFF9D	Aislación en FALSE de bit 23
CP31	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0x007FFF9D	Todos los bits en TRUE Bit de paridad = 0 Bits 24 a 29 = 0
CP32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0x8000009D	Todos los bits en FALSE Bit de paridad = 1 Constante label 271
CP33	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0x2000009D	SSM = 1
CP34	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	3	0xE000009D	SSM = 3
CP35	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	4	0x2000009D	SSM fuera de rango = 4

Para validar cada bit de las palabras discretas es necesario validar cada bit de manera independiente y aislada, por ello cuando un bit se quiere validar en TRUE todos los demás bits necesitan estar apagados, con el fin de asegurar que esa variable de manera independiente causó que únicamente el bit correspondiente se prendiera.

En la tabla 3-13 se puede observar en los casos de prueba del 1 al 15, los bits que se están validando de color gris. El valor que se prueba es el valor TRUE, por lo cual todos los demás bits necesitan estar en 0.

Así mismo se puede observar en los casos de prueba 16 a 30, los bits en FALSE que se están validando de color gris, por ello todos los demás bits se encuentran en 1.

Para clarificar como se deben aislar los bits, las siguientes figuras, muestran la representación binaria de los casos de prueba 1 y 16, correspondientes al aislamiento del bit 9 en TRUE y en FALSE respectivamente:

Figura 4-18 Representación binaria, caso de prueba 1, palabra discreta.

P	SSM		Datos																				Label ID							
32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	1	1	0	1

00	00	01	9D

Resultado	0000019D

Figura 4-19 Representación binaria, caso de prueba 16, palabra discreta.

P	SSM		Datos																				Label ID							
32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0	0	1	1	1	0	1

80	7F	FE	9D

Resultado	807FFE9D

4.6.3 Casos de prueba de robustez

Los casos de prueba 31 y 32, verifican que el software sea capaz de procesar la información cuando todos los bits de datos se encuentran prendidos y

apagados respectivamente. La representación binaria de los casos de prueba 31 y 32, son las siguientes:

Figura 4-201 Representación binaria, caso de prueba 31, palabra discreta.

P	SSM		Datos																				Label ID							
32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	0	1

00	7F	FF	9D

Resultado	007FFF9D

Figura 4-21 Representación binaria, caso de prueba 32, palabra discreta.

P	SSM		Datos																				Label ID							
32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	1	0	1

80	00	00	9D

Resultado	8000009D

Los casos de prueba 33 y 34 validan que al momento de que el SSM sea 1 (NCD) o 3 (FW) los bits de datos se coloquen a 0, pues son estados inválidos para la palabra. De la misma manera el caso de prueba 35 valida cuando la variable SSM se encuentra fuera de rango, el comportamiento será el mismo que los casos de prueba 33 y 34.

La siguiente figura, muestra la representación binaria de la palabra discreta con SSM inválido:

Figura 4-22 Representación binaria, SSM inválido, palabra discreta.

P	SSM		Datos																				Label ID							
32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	1	0	1

A0	00	00	9D

Resultado	A000009D

4.6.4 Matriz de trazabilidad entre Requerimientos de alto nivel y casos de prueba

Por último, se define la matriz de trazabilidad entre los requerimientos de alto nivel y los casos de prueba relacionados al igual que las palabras BNR y BCD. La siguiente figura, muestra la matriz de trazabilidad para la palabra discreta FCC 271:

Tabla 4-19 Matriz de trazabilidad, palabra discreta, FCC 271.

HLR-ID	Caso de prueba
HLR-0200	Procedimiento_Label271.CP0031
HLR-0201	Procedimiento_Label271.CP0032
HLR-0202	Procedimiento_Label271.CP0001, Procedimiento_Label271.CP0002, Procedimiento_Label271.CP0033, Procedimiento_Label271.CP0034
HLR-0203	Procedimiento_Label271.CP0033, Procedimiento_Label271.CP0034
HLR-0204	Procedimiento_Label271.CP0001 a Procedimiento_Label271.CP0030
HLR-0205	Procedimiento_Label271.CP0031
HLR-0206	Procedimiento_Label271.CP0032
HLR-0207	Procedimiento_Label271.CP0034

5. Resultados y discusión.

5.1 Plantilla genérica de procedimiento de prueba.

La plantilla genérica de procedimiento como se expresó anteriormente, sirve para especificar las entradas y salidas de los casos de prueba que cubren una cantidad determinada de requerimientos de alto nivel

Con tal fin se define la siguiente plantilla genérica:

Figura 5-1 Resultados, Plantilla genérico de procedimiento de prueba.

Template genérico para casos de prueba ARINC429						
Nombre de procedimiento de prueba	Procedimiento_Labelxxx					
Versión	n					
Autor	Francisco Sánchez					
Fecha	dd/mm/yyyy					
HLR_ID						
Descripción						
	Entradas				Salidas	Descripción de caso de prueba
	Entrada A	Entrada B	....	Entrada n	Label_XXX	
CP1						
CP2						
CP3						
....						
CPn						

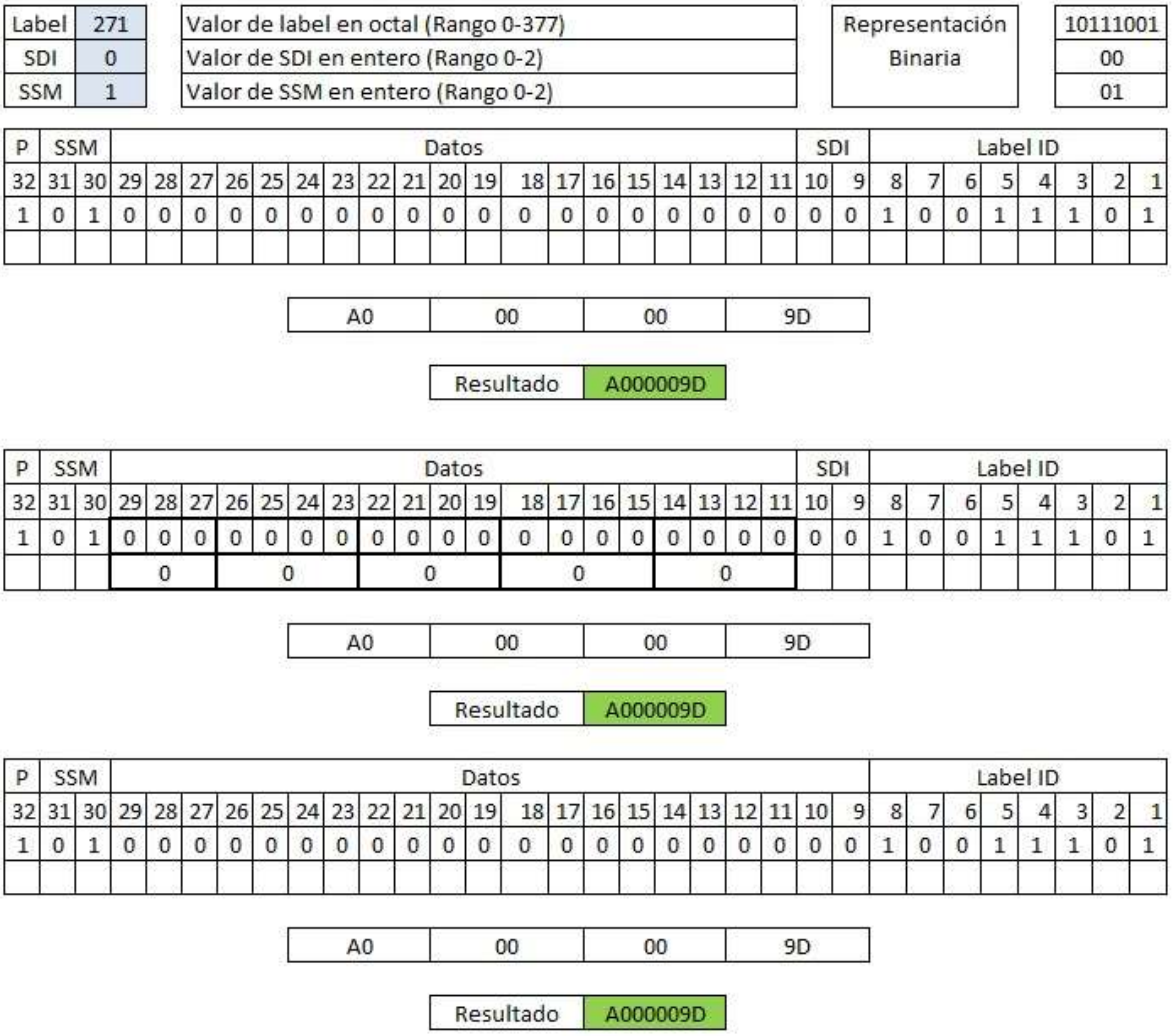
Este plantilla será la base o el esqueleto para especificar todas las pruebas necesarias para cubrir los requerimientos de alto nivel de palabras A429, sin importar si se trata de palabras BNR, BCD o discretas.

5.2 Hoja de cálculo y representación binaria para palabras A429.

Para realizar casos de prueba para empaquetamiento de datos de palabras A429, es fundamental conocer el significado de cada uno de los 32 bits del protocolo A429 así como las diferencias que existen entre palabras BNR, BCD y discretas.

Para ello la siguiente figura, muestra la hoja de cálculo utilizada en la presente investigación.

Figura 5-2 Hoja de cálculo de salidas binario/hexadecimal para palabras A429.



Label	271	Valor de label en octal (Rango 0-377)	Representación Binaria	10111001
SDI	0	Valor de SDI en entero (Rango 0-2)		00
SSM	1	Valor de SSM en entero (Rango 0-2)		01

P	SSM		Datos																		SDI		Label ID							
32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	1	0	1

A0	00	00	9D

Resultado	A000009D

P	SSM		Datos																		SDI		Label ID							
32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	1	0	1
			0			0			0			0			0			0													

A0	00	00	9D

Resultado	A000009D

P	SSM		Datos																				Label ID							
32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	1	0	1

A0	00	00	9D

Resultado	A000009D

En la figura 5-2 se puede observar las representaciones binarias respectivas para las palabras BNR, BCD y discretas respectivamente.

Estas representaciones binarias pueden ser modificadas fácilmente dependiendo del tipo de palabra que se quiera validar, ya que existen palabras que combinan campos discretos, BNR y/o BCD entre los bits 11 al 29.

5.3 Casos de prueba para palabras BNR A429.

Para ejemplificar la metodología de validación para las palabras A429 BNR, se utilizó la palabra estandarizada de “GroundSpeed”. A partir de la funcionalidad definida en los requerimientos de alto nivel, se utiliza el plantilla genérico de procedimientos de prueba y la hoja de cálculo para obtener los resultados de cada caso de prueba.

La siguiente figura muestra los casos de prueba necesarios para cubrir la funcionalidad de los requerimientos de alto nivel de la palabra BNR “GroundSpeed”:

Figura 5-3 Resultados, Ejemplo de Procedimiento de prueba para palabras BNR.

Empaquetamiento de datos, palabra BNR, Ground Speed.	
Nombre de procedimiento de prueba	Procedimiento_Label312
Versión	1
Autor	Francisco Sánchez
Fecha	28/05/2014
HLR_ID	HLR_0001, HLR_0002, HLR_0003, HLR_0004, HLR_0005, HLR_0006, HLR_0007, HLR_0008, HLR_0009, HLR_0010
Descripción	Este procedimiento pretende cubrir pruebas de rango normal y anormal para la label BNR 312, Ground Speed.
Versión de software	Versión 2.0
Versión de Documento de requerimientos	3

	Entradas			Salidas	Descripción de caso de prueba
Caso de prueba	Ground_Speed_valor	SDI_valor	SSM_valor	Label_312_Ground_Speed	
CP1	4095.875	0	3	0x6FFFE053	Rango máximo de Ground_Speed_valor. Valida cuando SSM es 3. Valida bits 11 al 13 en cero.
CP2	2048	3	2	0x48000353	Valida cuando el SSM es 2 Valida Rango intermedio de Ground_Speed_Valor Valida valor máximo de SDI Valida cuando el bit de paridad
CP3	4095.875	3	1	0xA0000353	Valida cuando el SSM es 1
CP4	4095.875	3	0	0x80000353	Valida cuando el SSM es 0
CP5	0	0	3	0xE0000053	Rango mínimo de Ground_Speed_valor Valida reperesentación de
CP6	10	3	4	0x80000353	Valor fuera de rango de la
CP7	-1024	3	3	0x80000353	Valor fuera de rango de la variable Ground_Speed_Valor
CP8	10	4	3	0x80000053	Valor fuera de rango de la

5.4 Casos de prueba para palabras BCD A429.

Para ejemplificar la metodología de validación para las palabras A429 BCD, se utilizó la palabra estandarizada de UTC. A partir de la funcionalidad definida en los requerimientos de alto nivel, se utiliza el plantilla genérico de procedimientos de prueba y la hoja de cálculo para obtener los resultados de cada caso de prueba.

La siguiente figura muestra los casos de prueba necesarios para cubrir la funcionalidad de los requerimientos de alto nivel de la palabra BCD UTC (Universal Time Constant):

Figura 5-4 Resultados, Ejemplo de Procedimiento de prueba para palabras BCD.

Nombre de procedimiento de prueba	Procedimiento_Label125								
Versión	1								
Autor	Francisco Sánchez								
Fecha	30/05/2014								
HLR_ID	HLR_0100, HLR_0101, HLR_0102, HLR_0103, HLR_0104, HLR_0105, HLR_0106, HLR_0107, HLR_0108.								
Descripción	Este procedimiento pretende cubrir pruebas de rango normal y anormal para la label BCD 125, UTC.								
Versión de Sofwate	2								
Versión de Documento de requerimientos	6								
Casos de prueba	Entradas							Salidas	Descripción de caso de prueba
	Decena_Hora	Unidad_Hora	Decena_Minuto	Unidad_Minuto	Decima_segundo	SDI_valor	SSM_valor	Label_125_UTC	
CP1	0	9	5	9	0	0	0	0x825640AA	Rangos normales de variables de tiempo. SDI y SSM rango 0 Bit de paridad = 1
CP2	2	3	0	5	5	1	2	0x48C155AA	Rangos normales de variables de tiempo. Rango SDI 1 y SSM rango 2 Bit de paridad = 0
CP3	1	0	3	0	9	2	3	0x640C26AA	Rangos normales de variables de tiempo. Rango SDI 2 y SSM rango 3
CP4	2	3	5	9	9	3	0	0x08D667AA	Rangos máximos normales de las variables de tiempo. Rango SDI 3
CP5	2	3	5	9	9	0	1	0xA00000AA	SSM = NCD
CP6	3	3	5	9	9	0	0	0xA00000AA	Campo 1 BCD inválido
CP7	2	10	5	9	9	0	0	0xA00000AA	Campo 2 BCD inválido Label 125 en el formato correcto.
CP8	2	3	10	9	9	0	0	0xA00000AA	Campo 3 BCD inválido
CP9	2	3	5	10	9	0	0	0xA00000AA	Campo 4 BCD inválido
CP10	2	3	5	9	10	0	0	0xA00000AA	Campo 5 BCD inválido
CP11	2	9	5	9	9	0	0	0xA00000AA	Combinación de tiempo inválido.
CP12	2	3	5	9	9	4	0	0xA00000AA	SDI fuera de rango.
CP13	2	3	5	9	9	0	4	0xA00000AA	SSM fuera de rango.

5.5 Casos de prueba para palabras Discretas A429.

Para ejemplificar la metodología de validación para las palabras A429 discretas, se utilizó la palabra estandarizada de FCC 271. A partir de la funcionalidad definida en los requerimientos de alto nivel, se utiliza el plantilla genérico de procedimientos de prueba y la hoja de cálculo para obtener los resultados de cada caso de prueba.

Las siguientes figuras muestran los casos de prueba necesarios para cubrir la funcionalidad de los requerimientos de alto nivel de la palabra discreta FCC 271:

Figura 5-5 Resultados, Procedimiento de prueba, palabra discreta, parte 1.

Empaquetamiento de datos, palabra discreta, FCC 271.																		
Nombre de procedimiento de prueba	Procedimiento_Label271																	
Versión	1																	
Autor	Francisco Sánchez																	
Fecha	41789																	
HLR_ID	HLR_0200, HLR_0201, HLR_0202, HLR_0203, HLR_0204, HLR_0205, HLR_0206, HLR_0207.																	
Descripción	Este procedimiento pretende cubrir pruebas de rango normal y anormal para la label discreta FCC 271.																	
Versión de Sofwate	Versión 1.0																	
Versión de Documento de requerimientos	Versión 4																	
	Entradas																Salidas	Descripción de caso de prueba
Casos de prueba	Bit 9B ool	Bit 10 Bo ol	Bit 11 Bo ol	Bit 12 Bo ol	Bit 13 Bo ol	Bit 14 Bo ol	Bit 15 Bo ol	Bit 16 Bo ol	Bit 17 Bo ol	Bit 18 Bo ol	Bit 19 Bo ol	Bit 20 Bo ol	Bit 21 Bo ol	Bit 22 Bo ol	Bit 23 Bo ol	SSM_ valor	Label_FCC_ 271	
CP1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0x0000019D	Aislación en TRUE de bit 9
CP2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0xC000029D	Aislación en TRUE de bit 10
CP3	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0x0000049D	Aislación en TRUE de bit 11
CP4	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0x0000089D	Aislación en TRUE de bit 12
CP5	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0x0000109D	Aislación en TRUE de bit 13
CP6	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0x0000209D	Aislación en TRUE de bit 14
CP7	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0x0000409D	Aislación en TRUE de bit 15
CP8	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0x0000809D	Aislación en TRUE de bit 16
CP9	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0x0001009D	Aislación en TRUE de bit 17
CP10	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0x0002009D	Aislación en TRUE de bit 18
CP11	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0x0004009D	Aislación en TRUE de bit 19
CP12	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0x0008009D	Aislación en TRUE de bit 20
CP13	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0x0010009D	Aislación en TRUE de bit 21
CP14	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0x0020009D	Aislación en TRUE de bit 22
CP15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0x0040009D	Aislación en TRUE de bit 23

.

Figura 5-6 Resultados, Procedimiento de prueba, palabra discreta, parte 2.

CP16	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0x807FFE9D	Aislación en FALSE de bit 9
CP17	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0x807FFD9D	Aislación en FALSE de bit 10
CP18	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0x807FFB9D	Aislación en FALSE de bit 11
CP19	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	0	0x807FF79D	Aislación en FALSE de bit 12
CP20	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	0	0x807FEF9D	Aislación en FALSE de bit 13
CP21	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	0	0x807FDF9D	Aislación en FALSE de bit 14
CP22	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	0	0x807FBF9D	Aislación en FALSE de bit 15
CP23	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	0	0x807F7F9D	Aislación en FALSE de bit 16
CP24	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	0	0x807EFF9D	Aislación en FALSE de bit 17
CP25	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	0	0x807DFF9D	Aislación en FALSE de bit 18
CP26	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	0	0x807BFF9D	Aislación en FALSE de bit 19
CP27	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	0	0x8077FF9D	Aislación en FALSE de bit 20
CP28	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	0	0x806FFF9D	Aislación en FALSE de bit 21
CP29	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0	0x805FFF9D	Aislación en FALSE de bit 22
CP30	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0x803FFF9D	Aislación en FALSE de bit 23
CP31	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0x007FFF9D	Todos los bits en TRUE Bit de paridad = 0 Bits 24 a 29 = 0
CP32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0x8000009D	Todos los bits en FALSE Bit de paridad = 1 Constante label 271
CP33	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0x2000009D	SSM = 1
CP34	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	3	0xE000009D	SSM = 3
CP35	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	4	0x2000009D	SSM fuera de rango = 4

6. Literatura citada

A., M. D. 2001. The Avionics Handbook Chapter 2 ARINC 429.

AIM GmbH. 2001. ARINC429 Specification Tutorial. Freibug, Germany.

AIRLINES ELECTRONIC ENGINEERING COMMITTEE. 2004. ARINC Specification 429 Part 1-17. Annapolis, Maryland.

Avionics Interfaces Technologies. 2008. ARINC 429 Protocol Tutorial. Omaha, NE.

Bentley, J. E. 2005. Software Testing Fundamentals Concepts, Roles, and Terminology.

BOIENG Commercial Airplanes. 2012. Statistical Summary of Commercial Jet Airplane Accidentes (1959-2012).

Ceritification Authorities Software Team (CAST). 2006. Position Paper CAST-26 Verification Independence.

Condor Engineering, Inc. 2000. ARINC Protocol tutorial. Santa Barbara, CA.

Condor Engineering, Inc. 2005. AFDX/ARINC 664 Tutorial (1500-049). Santa Barbara, CA.

Digital Electronics. 2003. Binary Coded Decimal (B.C.D).

Directorate General Technical Airworthiness. 2008. Software Testing and Test Coverage.

FAA.2011. Automated Flight Control.

Federal Aviation Administration (FAA). 2001. An Investigation of Three Forms of the Modified Condition Decision Coverage (MCDC) Criterion.

Fuch, C. 2010. The Evolution of Avionics Networks from ARINC 429 to AFDX.

Henckels, E. 2006. Airline Industry Overview.

IATA. 2013. IATA 2013 Annual Review.

Kandt, R. K. 2003. Software Requirements Engineering: Practices and Techniques.

Lauterbach, M. 2012. Testing & Debugging Avionics Systems that Use ARINC 429 or MIL-STD-1553 Data Busses.

Lei B, L. X. 2009. Robustness Testing for Software Components. Macao, China.

Messner, S. 2007. An Overview of RTCA DO-178B.

Nick, P., & Zelenksi, J. 2008. Computer Memory: Bits and Bytes.

Oracle. 2011. Development Tools: Data Dictionary Guide.

Shultz, T. 2009. Meeting DO-178B software verification guidelines with Coverity Integrity Center. San Francisco, CA.

Westfall, L. 2006. Software Requirements Engineering: What, Why, Who, When, and How .

Young, B. 2014. Computer Organization and Architecture, Bits and Bytes.

7. APÉNDICE

7.1 Acrónimos

AFDX	Avionics Full-Duplex Switched Ethernet
ARINC	Aeronautical Radio Incorporated.
BCD	BinaryCoded Decimal
BNR	Binary
EASA	European Aviation Safety Agency
FAA	Federal AviationAdministration
FCC	Flight Control Computer
FMS	Flight ManagmentSystem
FT	Functional Test
FW	FailureWarning
GPS	Global Positioning System
HLR	High LevelRequirement
IATA	International Air TransportAssociation
INT32	32 bits Integer
LSB	LeastSignificant Bit
MSB	MostSignificant Bit
NCD	No Computed Data
NO	Normal Operation
P	Parity bit
RTCA	Radio Technical Comission For Aeronautics
SDI	Source/DestinationIdentifier
SSM	Signal/Status Matrix
US	UnitedStates
UTC	Universal Time Constant