Marcos Jesús Barrios Lorenzo, 6/04/2023

Desarrollo del paquete del módulo Command and Control

En este documento se describen las decisiones tomadas en el diseño del paquete del cubesat TEIDESAT-I para el módulo software que permite el control remoto del satélite: Command and Control module.

Se toma como referencia la definición del protocolo CSP, diseñado por un grupo de estudiantes y pensado específicamente para satélites cubesat. También se tienen en cuenta las recomendaciones de dos estándar de la European Cooperation for Space Standardization (ECSS):

- ECSS-E-ST-70-11C para entender el contexto de operación espacial
- ECSS-E-70-41A para disponer de un paquete de referencia aparte del de CSP y también saber qué funciones son necesarias (servicios) en un cubesat típicamente.

Por otro lado, también se sigue la [tesis del proyecto Upsat], que documenta el desarrollo de su módulo de control remoto y de su ordenador de abordo (OBC), quienes lograron lanzar su propio Cubesat en 2017.

Se tienen en cuenta ambos paquetes ECSS y CSP. El primero porque es altamente personalizable y dispone de muy buena documentación formal. El segundo porque es el mejor si el objetivo es que sea lo más simple y eficiente posible.

Bit offset	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	Priority		Source					Destination					Destination Port						Source Port											Reserved				HMAC	XTEA	RDP	CRC
32	Data (0 – 65535 bytes)																															

Paquete CSP

Packet Header (48 Bits)							Packet Data Field (Variable)			
Packet ID				Packet Sequence Control		Packet Length	Data Field Header (Optional) (see Note 1)	Application Data	Spare	Packet Error Control (see Note 2)
Version Number (=0)	Type (=1)	Data Field Header Flag	Applica-tion Process ID	Sequence Flags	Sequence Count					
3	1	1	11	2	14					
16				16		16	Variable	Variable	Variable	16

Paquete ECSS

CCSDS Secondary Header Flag	TC Packet PUS Version Number	Ack	Service Type	Service Subtype	Source ID	Spare
Boolean (1 bit)	Enumerated (3 bits)	Enumerated(4 bits)	Enumerated (8 bits)	Enumerated (8 bits)	Enumerated (n bits)	Fixed BitString (n bits)

campo Data Field Header del paquete ECSS

ECSS vs CSP

La principal desventaja de ECSS es que su tamaño es mucho mayor que el de CSP, pero es posible que no suponga un problema si el hardware es capaz de procesar los paquetes ECSS sin problema si los componentes de nuestro cubesat son al menos tan buenos como los de Upsat.

En primer lugar, en cuanto a capacidad de memoria, en el cubesat de Upsat los componentes OBS, COMMS, ADCS tienen una memoria flash de 1 MB y 192 KBytes de SRAM, el resto no tiene memoria (EPS no necesita, la Unidad de ciencia tampoco, el componente de adquisición de imagen tampoco). Nosotros tenemos más del doble de tamaño de memoria, aunque no alcanzan los tamaños típicos de una CPU moderna; disponemos de 2 MBytes de Flash y 1.4 MBytes de SRAM en el triskel Cortex m7 para el OBS.

En segundo lugar, en cuanto a capacidad de cómputo el prototipo actual de cubesat TEIDESAT-I se basa en un triskel cuya OBC es un Cortex m7 de hasta 280 MHZ, y como se trata de microcontroladores cuya arquitectura es sencilla sencilla, entonces se puede asumir que la frecuencia de procesamiento es el principal factor de rendimiento y, por lo tanto, también se puede afirmar con seguridad que disponemos de mayor capacidad de procesamiento que el cubesat de Upsat, el cual trabaja a 32MHz.

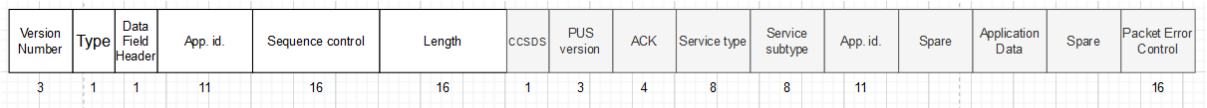
Por otro lado, en la tesis de Upsat se menciona que en general los estándar de ECSS están más orientados a microprocesador que a microcontroladores. Y a pesar de ello, ellos utilizaron prácticamente todos los campos, **lo que sugiere que en nuestro caso también es viable trabajar con el tamaño de paquete ECSS típico, por lo que podemos descartar entonces el paquete CSP a favor de ECSS**.

A continuación se analiza la utilidad de los campos estándar ECSS para decidir de cuál podemos prescindir.

Cambios realizados al paquete ECSS

Para empezar, tanto Upsat como CSP utilizan un solo paquete porque simplifica el código fuente. Es por ello que sería un desaprovechamiento el no tratar de hacer lo mismo, sobre todo teniendo en cuenta lo mucho que se parecen ambos paquetes de telecomando y de telemetría.

A partir de ahora se describen los cambios acompañados del estado actual del paquete, partiendo del paquete original:



En la figura se pueden observar en blanco los campos del encabezado principal, mientras que en gris están los de la parte de datos.

Nota: Destaca que el campo de packet error control no pertenece al encabezado principal, ni al encabezado de datos ni a Application Data, es un campo especial.

Anotación sobre la eliminación de bits

A bajo nivel, si el tamaño de un paquete no coincide con el tamaño de la palabra de memoria de la CPU, entonces se producirá un efecto de padding. Esto da lugar a que en ciertos casos, reducir el número de bits del paquete no importa porque van a ser reintroducidos antes de su procesamiento (nótese los campos “Spare”, que tienen por objetivo representar este proceso).

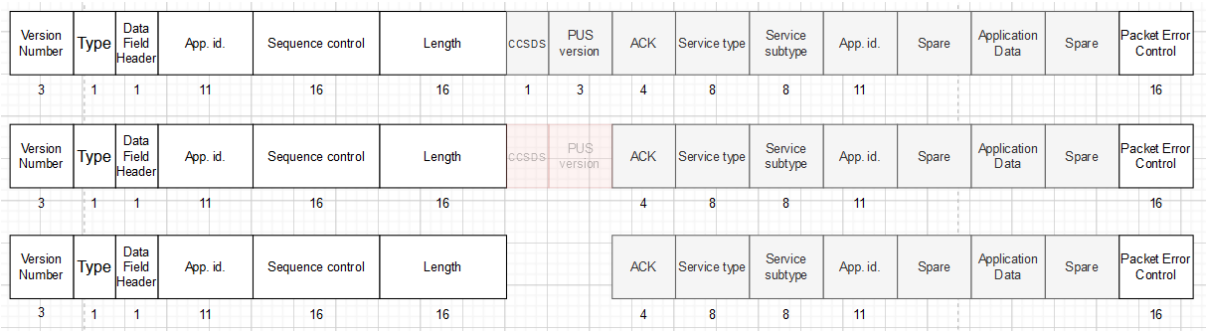
Sin embargo, como depende de muchos factores, (el número de bits de la arquitectura, etc), se ignora ese hecho y se asume que eliminar un campo tiene efectos positivos en el rendimiento del sistema, pero no se tratará de reducir la cantidad de bits de un campo respecto a su tamaño original.

Campos de versión

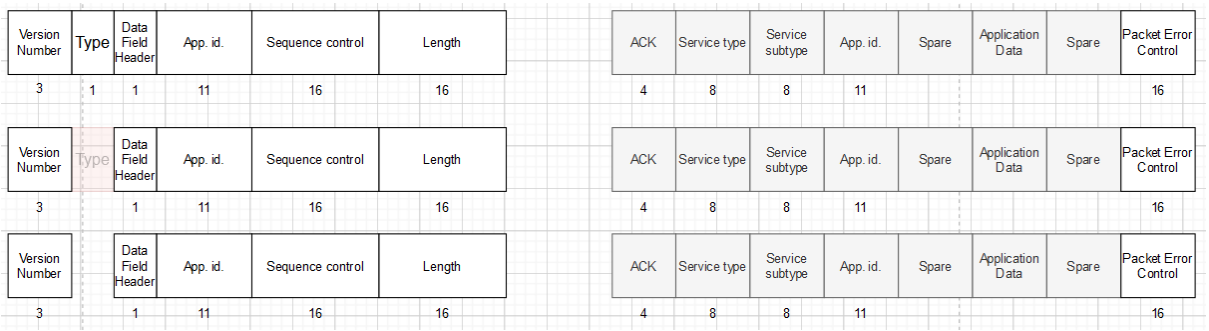
Se pueden eliminar dos campos porque representan información inútil en nuestro caso:

- **CCSDS:** Como no estamos siguiendo de forma completamente estricta el estándar ECSS, resulta innecesario tener un bit que indique si seguimos pautas o no.
- **PUS Version Number:** Indica la versión del estándar ECSS usado. Al usar el estándar ECSS-E-70-41A su valor será siempre igual a uno, por lo que es redundante.

Se eliminan dicho campos:



Tampoco es necesario utilizar el flag **Type** porque es utilizado para especificar si es un telecomando o si es de telemetría, pero en nuestro caso ambos son lo mismo porque solo tenemos un paquete, así que se elimina ese flag por ser redundante:



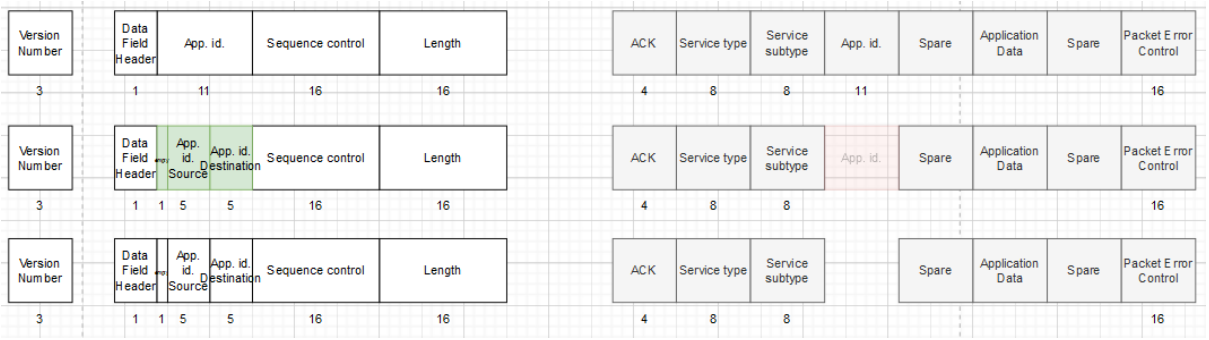
Por otro lado, no necesitamos dos campos de **Application Identifier** porque no vamos a tener tantos módulos. Con 11 bits se pueden direccionar 2056 (2^11) módulos distintos, lo cual es excesivo. Se propone entonces utilizar un solo campo que contiene dos subcampos de 5 bits para origen y destino (ya que deben ser simétricos en tamaño), lo cual da para 32 módulos (2^5). Se deja 1 bit sin utilizar.

Nota

Se utiliza tanto una dirección origen como destino, a pesar de que el estándar da libertad para solo utilizar uno u otro dependiendo de si el paquete es de telecomando (destino) o si es de telemetría (solo origen), ya que el segundo campo de Application Identifier está en el encabezado de datos, el cual es opcional. La razón es que si solo necesitamos cinco bits de direccionamiento, entonces podemos incluir ambos en un solo campo de Application Identifier.

Nota

Se valoró el solo utilizar un origen o un destino porque en el estándar ECSS un telecomando tiene como App. id.(Application Identifier) el destino en el encabezado principal, dejando el origen como opcional ya que está en el encabezado de datos, el cual es opcional. Esto tiene sentido cuando a un módulo le da igual el origen y ejecuta todos los comandos que reciba. De la misma forma, si en la estación se obtiene un paquete de telemetría, interesa saber más el origen de los datos que contiene, que el destino, porque va implícito. Sin embargo, esto puede conllevar limitaciones en la comunicación. Por ejemplo, si se ha recibido un paquete de telemetría y se quiere enviar un telecomando en consecuencia, no estaría del todo claro a qué lugar enviarlo, a no ser que se diseñe el sistema tal que un servicio solo está implementado en un solo módulo, en cuyo caso se puede discernir mediante el campo de Service Type. Es por ello que se descarta el concepto.



Campos “Service type” y “Service subtype”

El campo “service type” permite establecer qué paquete procesar antes, la prioridad. Es algo esencial de cara a contingencias, porque puede pasar que el cubesat o la estación se saturen de paquetes recibidos.

Por otro lado, el campo de “service subtype” es útil para obtener información extra del cualquier servicio. Gracias a éste, se puede obtener por ejemplo el código de error de la ejecución de un telecomando. De nuevo resulta esencial para obtener información extra sobre lo que está pasando en el cubesat, por lo que también se mantiene.

Campo “Sequence Control”

El campo de control de secuencia tiene por principal objetivo la identificación única de un paquete. Consiste en un contador que se resetea al llegar al valor máximo y que cada subtipo del servicio tiene. Es necesario por ejemplo para el servicio de verificación de telecomandos, el cual permite confirmar la ejecución de un paquete.

Potencialmente se puede reducir su tamaño en relación a la cantidad de paquetes que el sistema puede manejar en un momento dado, pero como mucho uno o dos bits, así que el impacto es mínimo por el ya mencionado padding.

Además, cabe mencionar que resulta difícil estimar cuántos paquetes se enviarán en un momento dado, y ya que la filosofía de construcción de un cubesat debería basarse en datos reales y no únicamente en especulación teórica, (sobre todo teniendo en cuenta que una de sus ventajas es su bajo riesgo de cara a la experimentación), entonces es complicado modificar su tamaño sin causar problemas.

Se mantiene el campo porque es importante de cara a contingencias.

Campo “Length”

Tanto el CSP como ECSS definen un tamaño de hasta 2^16 bytes para el paquete entero incluyendo los datos. Es por ello que el campo de Length, es de hasta 16 bits, porque a pesar de que Upsat sugiere que un paquete típico en la comunicación estación-cubesat puede transmitir tamaños de no más de 210 bytes (o 223 bytes en el caso de CSP) de cara a una comunicación eficiente, dentro del propio cubesat se podría encontrar el servicio de transferencia de paquetes grandes, que permite manejar archivos grandes de forma interna.

Por ejemplo, Upsat permite hasta 2050 bytes porque utilizan scripts de Unidad de Ciencia.

Nosotros también necesitamos poder realizar transferencias grandes, ya que hay imágenes involucradas que pueden llegar a ser de tamaño considerable.

Como no solo se transmiten paquetes del cubesat a estación, sino también de forma intermodular dentro del cubesat, entonces resulta razonable que el campo length tenga 16 bits, pues es posible que se trabaje directamente con paquetes en vez de archivos.

Destaca que CSP no lo utiliza. ¿Qué ventajas tiene incluir un campo de longitud?

- Transferencia eficiente de datos: asignación de suficiente memoria en el receptor. Contrato de duración de transferencia que ata las manos al emisor a la hora de enviar datos extra o no, problema que puede causar retrasos en el receptor.
- Detección de errores: Puede comprobarse para detectar errores en el paquete, por la suma de comprobación de errores, aunque en nuestro caso es redundante porque ya hay campos específicos para ello al final del paquete.

Y si en CSP no se utiliza el Length, entonces se emplean técnicas de sincronización para tanto la detección de comienzo como el final del paquete, mientras que en el caso de un paquete ECSS, como sí que lo contiene, entonces se puede hacer la sincronización detectando solo el comienzo del paquete. Sin embargo, parece difícil hacerlo mientras se está escuchando la transmisión, porque implicaría procesar el paquete al vuelo y no parece que de tiempo suficiente.

No está claro qué técnica es mejor, es necesario un análisis en profundidad, por lo que se mantiene el campo por ser la técnica elegida por ECSS.

Campo “ACK”

Al enviar un telecomando, se espera recibir el mismo paquete como confirmación de que se ha ejecutado con éxito, como parte del servicio de verificación de comandos, el cual es esencial. Éste campo configura si se espera recibir un paquete de confirmación o no en las distintas fases de procesamiento del paquete en el cubesat dependiendo de si el correspondiente bit está en 0 (no se confirma) o en 1(sí que se confirma).

Cuatro bits para configurar si se quiere recibir confirmación en distintas fases:

- Confirmar cuando se haya aceptado el paquete.
- Confirmar cuando se haya comenzado la ejecución del telecomando.
- Confirmar cuando se haya progresado. El proceso en ejecución decide cuándo enviar uno según se ejecuta.
- Confirmar cuando se haya completado la ejecución.

Pero se podría simplificar el campo ACK, igual que han hecho los de Upsat, porque tener distintos estados de verificación de un telecomando complica el código.

Tiene su utilidad el manejo al completo de todos los bits, ya que dichos estados pueden ser útiles por ejemplo en un hipotético caso de uso de manejo de múltiples hilos; el servicio sabría que no hace falta reenviar el paquete pero que está a la espera de que el cubesat complete la ejecución, lo que sigue un cambio de hilo de mientras para aprovechar el tiempo. Sin embargo en nuestro caso no está justificado optimizar tanto porque las tareas a

completar por el cubesat tienden a ser relativamente sencillas y tampoco es un proyecto que requiera mucha capacidad de cómputo.

Por lo anterior, se mantiene el campo y se sigue el ejemplo de Upsat de limitar el uso del ACK a solo dos valores, 0 y 1.

Campo de “Error checking”

Contiene información que los algoritmos de control de errores pueden utilizar para comprobar si el paquete ha sido corrompido o no.

En el Cubesat Space Protocol se utilizan los siguientes flags:

- HMAC (Hash-based Message Authentication Code)
- XTEA (eXtended Tiny Encryption Algorithm)
- RDP (Reliable Data Protocol, equivalente a TCP)
- CRC (Comprobación de Redundancia Cíclica)

Cada uno indica si se utiliza ese mecanismo o no, al ser flags de un solo bit.

En ECSS se disponen de hasta 16 bits. De nuevo siguiendo como ejemplo la tesis de Upsat, se observa que ellos usaban 8 bits, ya que el algoritmo de control de errores elegido es el CRC-8, pero no lo justifican.

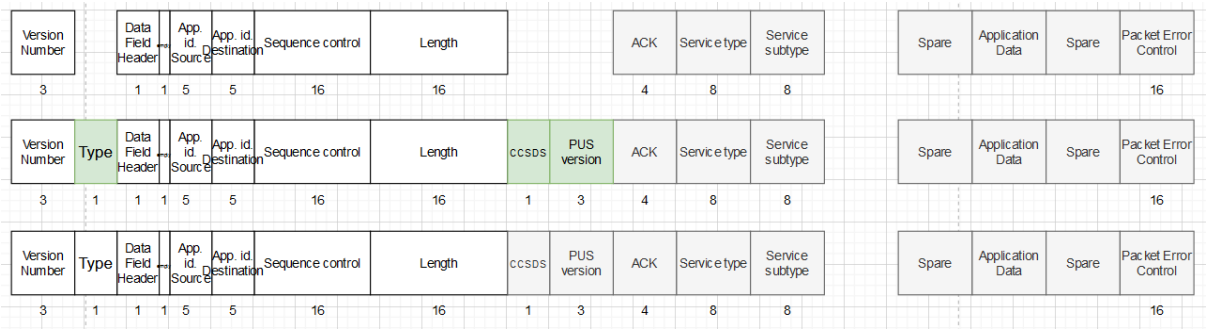
Una alternativa es utilizar CRC-16, pero si los de Upsat se manejaron con 8 bits, entonces seguramente lo hicieron para optimizar el tiempo de procesamiento de un paquete. Podríamos utilizar la versión de 16 bits, pero es preferible seguir el ejemplo.

Se mantiene el campo por ser esencial de cara a contingencias.

Reintroducción de los campos de Tipo, PUS, CCSDS eliminados previamente

El campo de Length indica el número de bytes del encabezado principal. Llegados a este punto el número de bits es 47. Como se trabaja por bytes entonces a nivel de implementación se van a tener 48 bits independientemente de si lo usamos o no. Es por ello que se reintroduce el campo de Tipo, de 1 bit, y se deja siempre en el mismo valor.

La ausencia de los campos CCSDS y PUS Version causa que el encabezado de datos se quede en 20 bits, por lo que se pueden reintroducir, ya que no supone diferencia a nivel de rendimiento. El resultado es:



Campos opcionales presentes solo en paquetes de telemetría

Algunos campos solo están en el paquete ECSS de telemetría, mientras que la evolución anterior parte del paquete de telecomando, por lo que no se mencionaron los siguientes dos campos.

Campo Time

En el estándar ECSS se observa un campo Time perteneciente al encabezado de datos, el cual no está presente en el paquete de telecomando. Se utiliza para incluir el tiempo actual en el cubesat cuando ha sido deducido a partir de un proceso de comprobación y/o márgenes de tiempo. Sin embargo, también está el servicio de gestión de temporización, el cual genera periódicamente paquetes con Application Identifier igual a cero, cuyo propósito es justamente eso: indicar qué tiempo tiene el cubesat.

Se puede incluir el campo o no dependiendo de si podemos permitirnos menor precisión en la temporización general de todos los paquetes y sincronizar cada cierto tiempo mediante el servicio de gestión de temporización.

En nuestro caso, como la arquitectura es sencilla, podríamos subsistir adecuadamente con menor precisión porque sabremos fácilmente cuando han sucedido las cosas, ya que no estaríamos realizando una enorme cantidad de cálculos al tener pocos experimentos. Es por ello que no se incluye el campo Time.

Campo Sub-Count

Al igual que el campo Time, el paquete de telemetría tiene un campo Sub-count que no está en el de telecomando. Es usado para obtener información extra cuando se pierde un paquete. El Sequence Control Count se usa para saber si falta un paquete, mientras que el Sub-count se usa para saber la naturaleza de ese paquete que falta, por lo que cambia menos que el campo Sequence Control count.

No lo incluyo porque la arquitectura es sencilla, es improbable que no hayan suficientes funciones como para justificar un campo de información extra.

Introducción de flags de configuración para funciones importantes en el encabezado principal

Hay un servicio que me ha recordado una cuestión; el servicio de distribución de comandos, que como se describió anteriormente, es un mecanismo para el envío de comandos de bajo nivel directamente al hardware.

La cuestión es que el software puede dejar de funcionar por cualquier error insalvable, pero a bajo nivel se puede salvar el cubesat gracias a los comandos de registro, que se ejecutan sin necesidad de una capa superior de software.

Se puede hacer algo similar pero a nivel de paquete; introducir flags que configuren directamente el software sin tener que ejecutar funciones software para esa misma configuración.

Miembros del equipo han sugerido que sería interesante introducir un campo para controlar desde el OBS el voltaje de unos LEDS, mediante un campo de 5 bits que representa su intensidad. Sin embargo, finalmente se decide en contra de ello porque puede ser incluido en la parte de datos, y agregar ese campo al encabezado principal no está justificado porque es una función que si bien es uno de los pocos parámetros modificables en el sistema dado su simplicidad, no es relevante en todos los contextos.

Finalmente, no se introduce ningún campo extra en el encabezado principal para controlar la configuración importante.