2 EL PROTOCOLO OCP

2.1 Introducción

El protocolo implementado en el servo amplificador por el fabricante se conoce como OCP (Orbit Control Protocol). Está diseñando para la comunicación entre diferentes dispositivos de control de Orbit, y dispositivos de control de usuarios, o equipos *host*. La comunicación entre dispositivos se basa en la utilización de los objetos de comunicación (COB, Communication objects).

La capa de aplicación de este protocolo está basado en el estándar industrial CANopen, aunque no de manera completa. En capa física, el protocolo está implementado tanto en bus CAN como en UART full dúplex. [2]

La red de área de controlador (CAN, Controller Area Network) es un protocolo de comunicación serie que soporta control en tiempo real distribuido. Tiene aplicaciones desde redes de alta velocidad hasta dispositivos simples, desde sistemas de automoción hasta aplicaciones de sensorización. Permite tasas de hasta 1 Mbps. [1]

Algunas características de CAN son la priorización de mensajes, garantización de los tiempos de latencia, flexibilidad en la configuración, detección de errores, mensajes multicast… haciendo de CAN un protocolo seguro y fiable en términos de comunicaciones [1]. Estas características hacen de CAN la base apropiada de OCP.

Figura 10: Trama CAN

En la figura 2 se presenta la trama de datos de CAN, en la que se basa la trama definida en OCP. Los campos SOF y EOF son el carater 0x7e, al igual que en HDLC, patrón de bits utilizado para delimitar las tramas. Los campos de control y arbitración proporcionan mecanismos para el control del enlace. En los datos se incluye la información, mientras que CRC y ACK aportan métodos para la protección de los datos transportados y de los procesos usuarios de estos mensajes.

2.2 Implementación del protocolo

En el esquema de implementación del protocolo se ha de distinguir entre dos tipos de dispositivos: dispositivos maestros y dispositivos esclavos.

El dispositivo maestro de OCP será el equipo controlador, aquel que realice peticiones a los nodos para que respondan a sus comandos. Por otra parte el esclavo responde a estos comandos requeridos por el maestro. En el caso del control de la DSA, es la DSA la que actúa de dispositivo esclavo, siendo el host el maestro.

Como en los escenarios habituales en este tipo de arquitectura maestro esclavo, los esclavos tienen un identificador único, que en este caso estará en el rango (1,127), y solo atienden a las peticiones que van a su identificador o a la dirección de difusión (0, en este caso). Además, marcan sus tramas con su identificador. El maestro no necesita, sin embargo, ninguna identificación. [2]

Como ya se ha comentado en la introducción de CAN, el protocolo sigue una estructura similar a la del clásico HDLC. Delimitando la trama al inicio y al final de la misma encontramos el carácter 0x7e, de manera similar a la figura 2. Antes del carácter final de trama (EOF) se encuentra el código de redundancia cíclico. Este código es el CRC-CCITT x16+x12+x5+1. Entre el byte SOF y el CRC están tanto los datos (de longitud variable) como los 2 bytes de control. Esto se puede apreciar en detalle en la figura 3.

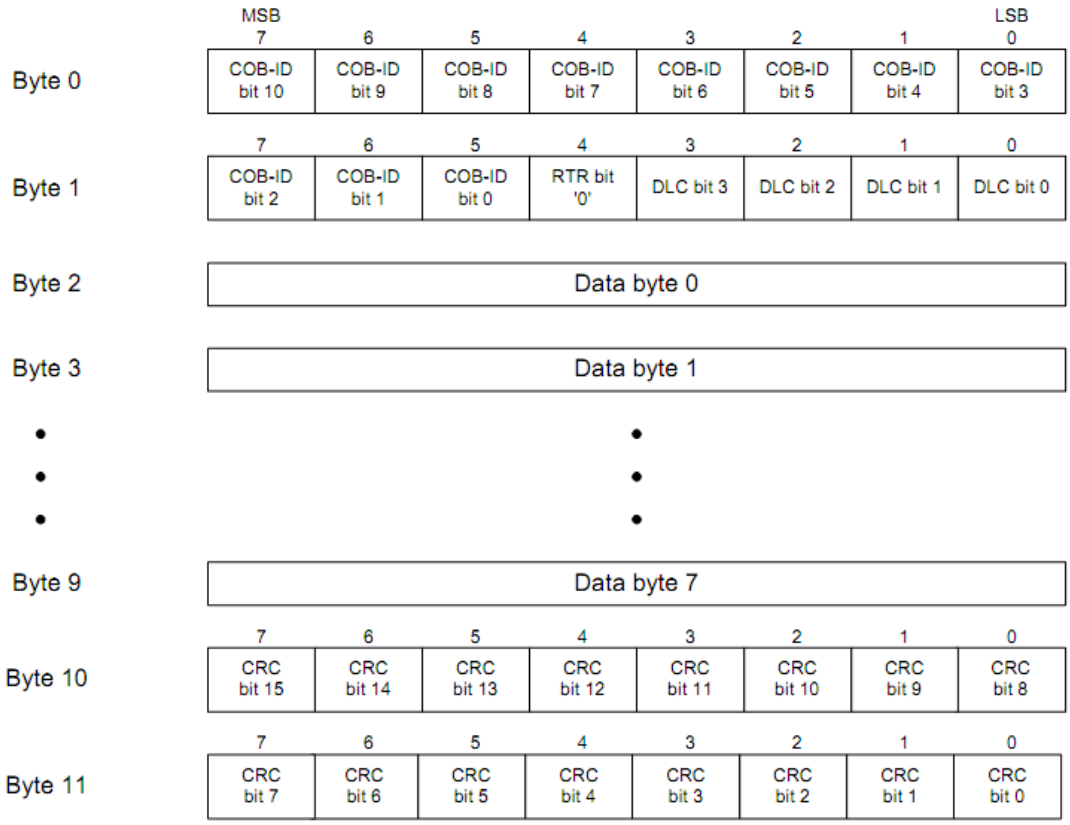

Figura 11 Detalle de las tramas OCP (EOF y SOF excluídos) [2]

La carga de usuario de la trama HDLC, representada en la figura 3 sería ya la implementación del protocolo OCP en sí misma. Los dos primeros bytes

después del inicial 0x7e son la cabecera de protocolo. En ella, los 11 bits más significativos forman el campo denominado *identificador de objeto de comunicación*, COB ID (Communication Object identifier).

Los 5 bits restantes en la cabecera son el bit RTR (heredado de CAN; en OCP siempre toma el valor 0) y 4 bits que codifican la longitud, en bytes, de los datos de usuario (la longitud de los datos de usuario es variable). Esta longitud no incluye, como en HDLC, ni los caracteres delimitadores de trama (0x7e iniciales y finales), ni el CRC, ni la cabecera ni el byte stuffing (método heredado de HDLC usado para eliminar el carácter 0x7e en los datos si existiese)

2.2.1 El diccionario de objetos (OD)

Un diccionario de objetos (OD, object dictionary) es un sistema de nombramientos que proporciona un identificador único a cada objeto de datos que puede ser enviado por el bus. Se identifica a cada objeto por un índice y un subíndice, caso de ser una estructura. Un maestro podrá manipular un objeto del esclavo refiriéndose a su identificador si tiene permisos para realizar dicha acción.[2]

En el caso del protocolo tratado en este proyecto, OCP, el Diccionario de Objetos presenta seis columnas. En la tabla 2 se presentan estas seis columnas, así como una descripción de su funcionalidad.

Tabla 2: Diccionario de objetos

Índice (Hexadecimal)	Objeto	Nombre	Tipo	Atributo de acceso	M/O
Posición del objeto en el diccionario.	Tipo de formato del objeto	Descripción textual del objeto	Tipo de objeto	Permisos sobre el objeto	Mandatory u Optional

El índice es utilizado para referirse al objeto. De esta manera, al incluirse este valor en una trama, maestro y esclavo se ponen de acuerdo en qué objeto está siendo utilizado. También existirá un subíndice para el caso en que el tipo de formato del objeto sea una estructura.

La columna objeto contiene el tipo de formato de objeto, referido al tipo de dato que es el objeto. De esta forma podemos encontrar los tipos referidos en la tabla 3.

Tabla 3: Tipos de formato de objeto en el OD

Objeto	Comentarios	Código
NULL	Entrada del diccionario sin campo de datos	0

Objeto	Comentarios	Código
DEFTYPE	Definiciones de tipo, como booleano, enteros sin signos de 16 bits…	5
DEFSTRUCT	Define un nuevo tipo de RECORD	6
VAR	Valores únicos como sin signo de 8 bits, booleanos, flotantes, enteros 16 bits…	7
RECORD	Campo de datos múltiples donde los campos de datos pueden ser de diferentes tipos de VAR	9

El campo Tipo de objeto localiza genéricamente al objeto dentro de la globalidad del protocolo. Por otra parte, los atributos de acceso pueden ser:

-RW: Acceso en lectura y escritura
-WO: Acceso en escritura
-RO: Acceso en lectura
-Const: Acceso en lectura en un objeto de valor constante

La columna M/O define si el objeto es tipo Mandatory u Optional de acuerdo con el protocolo CAN [1], si es que el objeto en cuestión pertenece a dicho estándar (hay que recordar que OCP no cumple completamente dicho estándar. Sólo lo usa como referencia). Si el objeto no estuviera definido en CAN, este campo estaría vacío [2].

En la siguiente tabla se hace un resumen de los tipos de datos soportados por el OD. Con estos tipos se conformarán los diferentes campos de datos de usuario de las diferentes tramas.

Tabla 4: Tipos soportados de datos [1]

Índice	Objeto	Nombre
2	DEFTYPE	INTEGER8
3	DEFTYPE	INTEGER16
4	DEFTYPE	INTEGER32
5	DEFTYPE	UNSIGNED8
6	DEFTYPE	UNSIGNED16
7	DEFTYPE	UNSIGNED32
8	DEFTYPE	REAL32
20	DEFSTRUCT	PDO_COMMUNICATION_PARAMETER
21	DEFSTRUCT	PDO_MAPPING
22	DEFSTRUCT	SDO_PARAMETER
23	DEFSTRUCT	SDO_PARAMETER

El protocolo OCP soporta también una serie de datos complejos, en forma de estructura. Estos datos son de tipo RECORD, y definen los parámetros a utilizar en los COB (PDO y SDO).

2.2.2 Los objetos de comunicación: PDO y SDO

Las unidades de datos transportadas por la capa física son llamadas objetos de comunicación (COB, Communication Objects). Existen dos tipos de COB:

-Objetos de datos de servicio (SDO, Service data objetcs).

Los mensajes SDO son usados para manipular objetos del diccionario de objetos de acuerdo a los identificadores de dichos objetos. El esclavo recibe dichos SDO, los cuales especifican en el mensaje con qué objetos se va a tratar.

-Objetos de datos de proceso (PDO, process data objects)

En este caso la manipulación de objetos del OD sin una referencia explícita. Esto es posible si hay algún convenio a priori (convenio por defecto) acerca del objeto del OD referenciado. Este convenio es conocida como “mapeo de PDO”, lo cual es un objeto de del OD por sí mismo y puede ser manipulado usando una SDO.

La transferencia de datos en tiempo real es conseguida a través de los objetos PDO. No hay ninguna cabecera de protocolo adicional asociada a la transferencia de PDO, lo que significa que en el campo de datos sólo habrá datos de usuario.

Los PDO corresponden a entradas en el OD del dispositivo, y proveen la interfaz a los objetos de aplicación. El tipo de datos y el mapeo de los objetos de aplicación dentro de un PDO son determinados, como ya se ha comentado, por una estructura de mapeo dentro del OD del dispositivo.

Existen dos posibles usos de los PDO. El primero es para transmitir datos (TPDO), mientras que el segundo es para recepción (RPDO). Es conveniente aclarar que el sentido “recepción” y “envío” es desde el punto de vista del esclavo. El RPDO de un dispositivo puede ser escrito por cualquier otro dispositivo de la red en cualquier momento. Los TPDO se transmiten a una tasa predefinida. Por último, es importante apuntar que el servicio de PDO es un servicio no confirmado, lo que implica que el hecho de enviar una TPDO no es garantía de que el comando deseado haya sido realmente ejecutado.[2]

Con los SDO, se ofrece el servicio de acceso a las entradas del OD de un dispositivo. Nuevamente en el campo de datos del SDO sólo hay datos para transferir. La transferencia está basada en un modelo Maestro/Esclavo. Es el Maestro quien toma la iniciativa. Por otra parte, el dueño del OD sobre el que se está actuando es el esclavo. En CAN, tanto maestro como esclavo pueden

decidir abortar la transferencia [1]. Sin embargo, al no permitir OCP el envío de datos segmentados en varios mensajes, sólo el esclavo podrá abortar el envío de SDO.

En el caso de los SDO, a diferencia del PDO, se trata de un servicio confirmado. Se describen en OCP tres sub protocolos, para descargar, subir o abortar una SDO. En la siguiente tabla se puede ver un resumen de cada caso.

Tabla 5: Protocolos de manejo de SDO

Protocolo de descarga	Protocolo de subida	Protocolo de aborto
El maestro envía al esclavo los datos para descargar; el esclavo confirma la recepción	El maestro envía al esclavo la petición de datos deseados, el esclavo confirma a través de los propios datos	El esclavo responde a la petición de subida o bajada con un "Abort Transfer Request" (Petición de aborto de transferencia)

2.2.3 Implementación del protocolo

El protocolo se implementa mediante una serie de PDOs y SDOs predefinidos o no. Los predefinidos están configurados por defecto y mapeados a objetos del OD.

Como se ha comentado anteriormente, existen dos tipos de objetos PDO: El RPDO y el TPDO. Desde el punto de vista del pedestal, RPDO serán las PDO recibidas, mientras que TPDO serán las PDO transmitidas. En ambos tipos existen cuatro posibles objetos, esto es, RPDO1 a RPDO4, y de similar forma para TPDO. Sin embargo, sólo RPDO1 y TPDO1 están preconfigurados, el resto están disponibles para ser configuradas. Sin embargo, en el marco de este proyecto no se estimó necesario tener más RPDO o TPDO, por lo que únicamente se utilizarán en la práctica RPDO1 y TPDO1.

En el caso de RPDO1, dicho servicio tiene su campo de datos mapeado a tres objetos del diccionario de objetos, correspondientes al campo de modo de operación, comando de posición y comando de velocidad. Dichos objetos quedan de la siguiente manera:

Tabla 6: Objetos del servicio RPDO1

Orden en el campo de datos	Entrada del Diccionario de objetos	Descripción del objeto
1	0x2005	Comando de Velocidad
2	0x2003	Comando de Posición
3	0x2001	Comando del modo de operación

Por otra parte, el servicio TPDO1 tiene mapeados cuatro objetos, los cuales son los estados de posición y velocidad, el estado del comando de operación, y la palabra de estado.

Tabla 7: Objetos del servicio TPDO1

Orden en el campo de datos	Entrada del Diccionario de objetos	Descripción del objeto
1	0x2009	Palabra de Estado
2	0x2004	Estad de Velocidad
3	0x2002	Estado de Posición
4	0x2000	Comando del modo de operación

Finalmente, en el servicio SDO hay dos escenarios posibles. En el primero de ellos, el cliente le pide al pedestal unos datos determinados, correspondiente al diccionario de objetos. El pedestal responderá con dichos datos (la recepción por parte del cliente de estos datos sirve de asentimiento).

En el segundo, el cliente desea entregar unos datos al pedestal, para configurar algún objeto, o con cualquier otro propósito. En este caso el pedestal confirmará la correcta recepción de los datos.

2.3 Buil-in-Test (BIT)

El bit es un grupo de tests que funcionan en segundo plano durante el funcionamiento normal del DSA. Los informes del BIT son implementados a través del protocolo OCP, descrito anteriormente. Una *palabra de estado actual* de 32 bits (correspondiente a la entrada 0x2160 subíndice 0x1 del diccionario de objetos) y una *palabra seguro* (correspondiente al objeto 0x2160, subíndice 0x2) represantan los resultados de cada test.

Cada bit en estas palabras represantan el resultado de un test específico. Cuando existe un fallo el bit correspondiente es puesto a uno en ambas palabras. Cuando desaparece el error, el bit correspondiente en *la palabra de estado actual* es puesto a cero. Esto no es así para la *palabra seguro*, que mantendrá dicho bit a uno hasta que un comando externo borre todas las banderas pasadas de dicha palabra.

Con este último mecanismo aseguramos que un error que se produzca durante un corto espacio de tiempo nunca sea detectado, siendo potencialmente muy peligroso, no solo ya para el buen funcionamiento del sistema, sino para la seguridad general de la instalación. La *palabra de estado actual* es de solo lectura, mientras que la *palabra seguro* puede ser reiniciada a cero por un comando externo.