CAN EN EL dsPIC30F4013

Autores:

Bonillo Picó, Jorge

Doménech Juárez, Hugo

Navarro Valverde, Sergio

Parra Camacho, Juan Sebastián

INDICE

BLOQUE 1: Que es CAN

BLOQUE 2: CAN en el dsPIC30F4013

BLOQUE 1: QUE ES CAN

1.- INTRODUCCIÓN

CAN (Controller Area Network) es un protocolo de comunicaciones desarrollado por la firma alemana Robert Bosch GmbH, basado en una topología bus , en 1986 fue presentado oficialmente por el grupo Bosch en la “Sociedad de Ingeniería de la Automoción”, principalmente para aplicaciones de automoción.

1.1 Los inicios

El ingeniero Uwe Kiencke inició el desarrollo del nuevo protocolo en 1983.

Este protocolo de bus serie se ideó principalmente para aportar mayor seguridad y fiabilidad así como interconectar los dispositivos electrónicos internos de un vehículo con la finalidad de sustituir o eliminar el cableado.

En su desarrollo se involucran ingenieros de “Mercedes-Benz” e Intel como principal vendedor de semiconductores.

Tras la presentación oficial en 1986, surgieron los primeros chips controladores de CAN, los cuales no estaban regidos por ninguna normalización, por ello existía muchas incompatibilidades entre fabricantes.

A mediados de 1987, “Intel” presentó el primer chip de controlador CAN: el 82526. Poco tiempo después, “Semiconductores Philips” presentaría el 82C200.

Intel adopto el concepto de FullCan y Philps el BasicCan, el primero requería menos carga de CPU que el de su competencia , pero era limitado con respecto al número de los mensajes que podían ser recibidos. En cambio el BasicCAN elegida por Philips, requería menos silicio, lo que abarataba aun más su coste.

El desarrollo de los controladores actuales se ha basado en el concepto de los ya obsoletos ‘BasicCAN’ y ‘FullCAN’ .

Los esfuerzos por estandarizar y mejorar el protocolo CAN aplicado a la automoción, dio lugar en Marzo de 1992 a la fundación de la (CiA) “CAN en la Automoción” formada por usuarios internacionales y fabricantes, usuarios internacionales y grupos de fabricantes fundaron oficialmente la organización'.

La primera publicación técnica, que trataba acerca de la capa física, fue emitida solo unas semanas después: ‘CiA’ recomendaba utilizar solamente transceptores CAN que cumplieran la normativa ISO 11898. A día de hoy, son los transceptores RS485 los utilizados más comúnmente.

Otra de las primeras tareas de la CIA fue la especificación de una capa de aplicación de la CAN. La CAL 'CAN Application Layer ' se creó utilizando el material procedente de Philips Medical Systems y STZP, junto con la ayuda de otros miembros de la CIA.

También ha desarrollado normalización CAN para los camiones. La creación de redes entre el camión y el remolque se ha estandarizado como ISO 11992 . Este protocolo se basa en J1939 y se está utilizando en Europa desde 2006.

En 1993 un consorcio Europeo liderado por Bosch ,desarrolló un prototipo conocido como CANopen para las redes internas en las cadenas de producción. En 1995, se publicó el perfil totalmente revisado de las comunicaciones de CANopen, que en el plazo de cinco años ya se había convertido en la red integrada estandarizada más importante de Europa, puesto que ofrecía una gran flexibilidad y un gran número de opciones de configuración, así como diversos perfiles de dispositivo, interfaz y uso.

Actualmente, CANopen se sigue utilizando especialmente en Europa: máquinas de moldeado por inyección en Italia, máquinas expendedoras en Inglaterra, grúas en Francia, control de maquinaria en Austria, o maquinaria de fabricación de relojes en Suiza.

En el año 2000, surgió el TTCAN (Time-tiggered communication on CAN), un nuevo protocolo basado en CAN, fruto de una larga investigación entre académicos y expertos de la industria de los semiconductores. Esta extensión permitía tanto la transmisión de mensajes simultáneos, como la implementación de un bucle cerrado para el control del bus. Y gracias a que el protocolo del bus no fue modificado, este tipo de mensajes podían seguir siendo enviados con el mismo sistema físico de bus.

1.2 Estandarización

La especificación del protocolo CAN está estandarizado por La Organización Internacional para la Estandarización (ISO, International Organization for Standarization).

La ISO define dos tipos de redes CAN:

A) Una red de alta velocidad (hasta 1 Mbps), bajo el estándar ISO 11898-2, destinada para controlar el motor e interconectar las unidades de control electrónico (ECU), sensores, sistemas antideslizantes, etc.

B) Una red de baja velocidad tolerante a fallos (menor o igual a 125 Kbps), bajo el estándar ISO 11898-3, dedicada a la comunicación de los dispositivos electrónicos internos de un automóvil como son control de puertas, techo corredizo, luces y asientos.

C) También destacan los estándares basados en el US-protocol J1939, como el ISO 11992 (referente a la interfaz para camiones y remolques) e ISO 11783 (referente a la maquinaria agrícola y forestal).

1.3 Características básicas del bus CAN

En general podría decirse que el bus CAN es idóneo en aplicaciones que precisen control distribuido en tiempo real (interconexión de controladores) en redes sin un gran flujo de datos.

También destacan los siguientes aspectos:

Acceso al medio: Multidifusión (multicast), permite que todos los nodos puedan acceder al bus de forma simultánea con sincronización de tiempos. Funcionamiento en modo Broadcast y comunicación Multimaestro, todos los módulos CAN reciben y envían de información.

Velocidad flexible: ISO define dos tipos de redes CAN: una red de alta velocidad (de hasta 1 Mbps) definida por la ISO 11898-2, y una red de baja velocidad tolerante a fallos (menor o igual a 125 Kbps) definida por la ISO 11898-3.

Desconexión autónoma de nodos defectuosos: Si un nodo de red cae, sea cual sea la causa, la red puede seguir funcionado, ya que es capaz de desconectarlo o aislarlo del resto. De forma contraria, también se pueden añadir nodos al bus sin afectar al resto del sistema, y sin necesidad de reprogramación.

Fiabilidad de transmisión dado que el bus CAN posee una eficiente detección de errores (lograda a través de cinco mecanismos de detección, 3 al nivel de mensaje y 2 a nivel de bit) y puede funcionar con condiciones extremas de ruido e interferencias. Los errores además pueden ser señalizados

Longitud: las redes CAN pueden alcanzar 1000 metros de longitud ampliables con repetidores y su velocidad depende de la longitud de la red.

2.- TRANSMISIÓN DE DATOS

2.1. Arquitectura de capas

El bus CAN está dividido en 3 capas, la física y enlace de datos por una parte, que está sujeta al modelo estándar de comunicaciones OSI y la capa de aplicación, implementada a nivel software mediante diversas soluciones.

Las estandarizaciones ISO (International Standard Organization), a diferencia de las normas BOSCH, especifican también el medio de comunicación. Por lo tanto una implementación CAN a partir de las especificaciones de BOSCH no siempre será compatible con las normas ISO.

2.1.1. Capa física

La capa física de CAN, es responsable de la transferencia de datos entre los distintos nodos que componen la red. Define aspectos como niveles de señal, codificación, sincronización y tiempos en que los bits se transfieren al bus.

En la especificación original de CAN, la capa física no fue definida, permitiendo diferentes opciones para la elección del medio y niveles eléctricos de transmisión. Las características de la señales eléctricas en el bus, fueron establecidas más tarde por el ISO 11898 para las aplicaciones de alta velocidad y, por el estándar ISO 11519 para las aplicaciones de baja velocidad.

Respecto al estándar 11519, los nodos conectados en este bus interpretan dos niveles lógicos denominados: Dominante y Recesivo. El dominante tiene una tensión diferencial de 2V con CAN_H = 3.5V y CAN_L = 1.5V (nominales). Mientras que la tensión diferencial del recesivo es de 5V con CAN_H = 0V y CAN_L = 5V (nominales).

Este bus de baja velocidad requiere dos resistencias en cada transceptor: RTH para la señal CAN_H y RTL para la señal CAN_L. Esta configuración permite al transceptor de bus de baja velocidad (fault-tolerant) detectar fallas en la red. La suma de todas las resistencias en paralelo, debe estar en el rango de 100-500.

Por otra parte, en el estándar 11898 los nodos interpretan los niveles lógicos Dominante, con la tensión diferencial (CAN_H - CAN_L) es del orden de 2.0 V con CAN_H = 3.5V y CAN_L = 1.5V (nominales); recesiva, con la tensión diferencial (CAN_H - CAN_L) es del orden de 0V y con CAN_H = CAN_L = 2.5V (nominales).

NOTA: El par de cables trenzados (CAN_H y CAN_L) constituyen una transmisión de línea. Si dicha transmisión de línea no está configurada con los valores correctos, cada trama transferida causa una reflexión que puede originar fallos de comunicación. Como la comunicación en el bus CAN fluye en ambos sentidos, ambos extremos de red deben de estar cerrados mediante una resistencia de 120 Ω. Ambas resistencias deberían poder disipar 0.25 w. de potencia.

2.1.2. Capa de enlace de datos

La capa de enlace de datos es responsable del acceso al medio y el control lógico y está dividida a su vez en dos niveles, el LLC y el MAC. El primero (Logical Link Control), gestiona el filtrado de mensajes, controla la saturación y su administración. El MAC (Medium Acces

Control), gestiona la trama de los mensajes, el tiempo de acceso al medio y reconocimiento de mensajes, sus errores y su señalización, así como la identificación del momento para transmitir o emitir en el bus.

2.2. Estructura de un nodo CAN

Dentro de un nodo CAN, se pueden distinguir una serie de módulos interconectados entre ellos: un bus de direcciones, datos y un control (paralelo) enlazando el controlador central, la memoria de los datos y el programa (donde está almacenado el software de aplicación y el controlador de red de alto nivel), los dispositivos de entrada y salida y, la interfaz de comunicación.

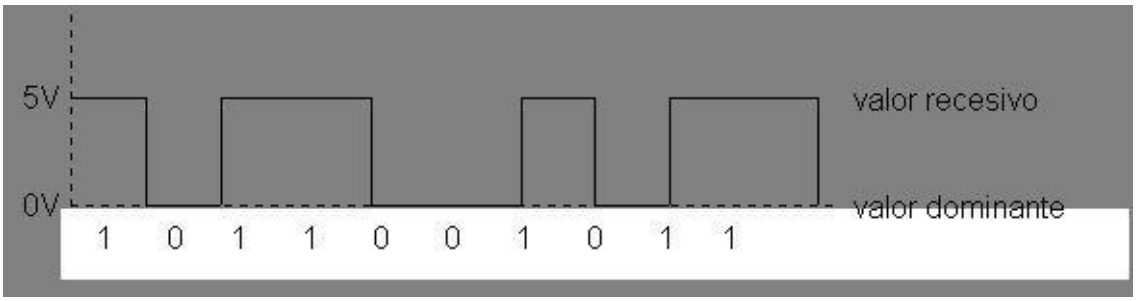
Desde el punto de vista del controlador, la interfaz de comunicación se puede ver como un conjunto de buzones, donde cada uno de estos sirve como registro lógico de interfaz entre el controlador local y los nodos remotos. Si un nodo quiere comunicarse, tiene que dar de alta los correspondientes buzones de recepción y transmisión antes de hacer ninguna operación.

Teniendo en cuenta esta arquitectura, el programador se inicializa especificando los parámetros de los registros de control de interfaz de comunicación, como las características del controlador de red o la velocidad de transmisión. A continuación, se actualizan todos los buzones y en cada uno se especifica si es receptor o transmisor y, su estado inicial, inicializando su parte de datos del búfer. Posteriormente, para transmitir un mensaje es necesario poner los datos en el búfer de datos correspondiente al buzón de transmisión y activar el flanco de transmisión. Por último, la interfaz de red intenta comunicar los datos a través de la red. El estado de la transferencia se puede comprobar en el estatus de estado de cada buzón y una vez configurado el software de aplicación, los nodos actúan autónomamente.

2.3 Acceso al medio

Se usa la técnica conocida como CSMA/CD+CR (Carrier Sense Multiple Access/Collision Detection + Collision Resolution). Cada nodo está escuchando en el bus y llega un momento que todos los nodos tienen la misma oportunidad de transmitir. En caso de que dos nodos comiencen a transmitir al unísono se detectará la colisión. Además añade la característica de resolución de colisión. En la técnica CSMA/CD, utilizada en redes Ethernet ante colisión de varias tramas, todas se pierden. CAN resuelve la colisión con la supervivencia de una de las tramas que chocan en el bus, eligiendo a la de mayor prioridad. Todo paquete contiene una etiqueta identificativa, aunque no se guarda ni el emisor ni el receptor del mensaje.

La resolución de colisión se basa en una topología eléctrica que aplica una función lógica determinista a cada bit, que se resuelve con la prioridad del nivel definido como bit de tipo dominante. Definiendo el bit *dominante* como equivalente al valor lógico '0' y bit *recesivo al* nivel lógico '1' se trata de una función AND de todos los bits transmitidos simultáneamente. Cada transmisor escucha continuamente el valor presente en el bus, y se retira cuando ese valor no coincide con el que dicho transmisor ha forzado. Mientras hay coincidencia la transmisión continua, finalmente el mensaje con identificador de máxima prioridad sobrevive. Los demás nodos reintentarán la transmisión lo antes posible.



Se ha de tener en cuenta que la especificación CAN de Bosch no establece cómo se ha de traducir cada nivel de bit (dominante o recesivo) a variable física. Cuando se utiliza par trenzado según ISO 11898 el nivel dominante es una tensión diferencial positiva en el bus, el nivel recesivo es ausencia de tensión, o cierto valor negativo, (los transceptores no generan corriente sobre las resistencias de carga del bus). Esta técnica aporta la posibilidad de fijar con determinismo la latencia en la transmisión de mensajes entre nodos y el funcionamiento en modo multimaestro sin necesidad de gestión del arbitraje, es decir control de acceso al medio, desde las capas de software de protocolo.

2.4. Detección de Errores

Una de las características más importantes y útiles de CAN es su alta fiabilidad, incluso en entornos de ruido extremos, el protocolo CAN proporciona una gran variedad de mecanismos para detectar errores en las tramas. Esta detección de error es utilizada para retransmitir la trama hasta que sea recibida con éxito.

Otro tipo de error es el de aislamiento, y se produce cuando un dispositivo no funciona correctamente y un alto por ciento de sus tramas son erróneas. Este error de aislamiento impide que un dispositivo condicione el funcionamiento del resto de nodos implicados en la red.

Los errores relacionados con la transmisión de paquetes son el error de bit, que detecta cualquier fallo en los bits que contienen la información mientras transmiten; el error de relleno (Stuff Error), para evitar la confusión en la coincidencia de unos bits de información con los delimitadores de trama; el error de CRC, donde se hace una división polinómica de dos partes del mensaje; el error de forma, cuando un campo de tamaño fijo recibe un bit alterado; y el error de reconocimiento, cuando ningún nodo cambia a dominante el bit de reconocimiento.

Si un nodo advierte alguno de los fallos mencionados, iniciará la transmisión de una trama de error. Se especifican diversos fallos en la línea física de comunicación, como por ejemplo línea desconectada, problemas con la terminación de los cables o líneas cortocircuitadas. Sin embargo, no especificará como reaccionar en caso de que se produzca alguno de estos errores.

2.4.1 Aislamiento de nodos defectuosos

Para evitar que un nodo en problemas condicione el funcionamiento del resto de la red, se han incorporado a la especificación CAN medidas de aislamiento de nodos defectuosos que son gestionadas por los controladores. Un nodo pude encontrarse en uno de los tres estados siguientes en relación a la gestión de errores. Error activo, pasivo o anulado, que detecta un nodo que está participando en la comunicación y envía la trama correspondiente de error.

2.5. Formato de codificación y sincronización de datos

La codificación de bits se realiza por el método NRZ (Non-Return-to Zero) que se caracteriza por que el nivel de señal puede permanecer constante durante largos periodos de tiempo y se tiene que determinar que no se sobrepasa (sirnconización por Bit Timing).
Se requiere poco ancho de banda para transmitir, pero no puede garantizar la sincronización de la trama transmitida. Para resolverlo se emplea la técnica del "bit stuffing", donde cada 5 iguales se inserta uno de signo contrario para no perder la sincronización; este será eliminado por el receptor una vez sincronizado.

2.5.1. Tipo de tramas

El protocolo CAN está basado en mensajes, con las siguientes tramas para la gestión de transferencia de mensajes. La trama de datos (contiene información que se quiere transmitir), la trama de información remota (para solicitar la transmisión de una trama de datos con la información de un identificador), trama de error, trama de sobrecarga (cuando un nodo necesita tiempo extra para procesar los datos), el espacio entre tramas (tiempo de espera entre emisión y recepción) y el bus en reposo (los intervalos de inactividad que mantiene el nivel recesivo del bus).

2.5.2. Trama de datos

Es la utilizada por un nodo normalmente para poner información en el bus. Puede incluir entre 0 y 8 bytes de información útil. Los mensajes de datos consisten en celdas que envían datos y añaden información definida por las especificaciones CAN:

- *Inicio de trama (SOF):* El inicio de trama es una celda de un sólo bit siempre dominante que indica el inicio del mensaje, sirve para la sincronización con otros nodos.
- *Celda de Arbitraje (Arbitration Field):* Es la celda que concede prioridad a unos mensajes o a otros
- *Celda de control (Control Field):* El campo de control está formado por dos bits reservados para uso futuro y cuatro bits adicionales que indican el número de bytes de datos. En realidad el primero de estos bits (IDE) se utiliza para indicar si la trama es de CAN Estándar (IDE dominante) o Extendido (IDE recesivo). El segundo bit (RB0) es siempre recesivo. Los cuatro bits de código de longitud (DLC) indican en binario el número de bytes de datos en el mensaje (0 a 8).
- *Celda de Datos (Data Field):* Es el campo de datos de 0 a 8 bytes.
- *CRC:* Código de redundancia cíclica: Tras comprobar este código se podrá comprobar si se han producido errores.
- *Celda de reconocimiento (ACK):* es un campo de 2 bits que indica si el mensaje ha sido recibido correctamente. El nodo transmisor pone este bit como recesivo y cualquier nodo que reciba el mensaje lo pone como dominante para indicar que el mensaje ha sido recibido.
- *Fin de trama (EOF):* Consiste en 7 bits recesivos sucesivos e indica el final de la trama.
- *Espaciado* entre *tramas* (IFS): Consta de un mínimo de 3 bits recesivos.

2.5.3. Trama remota (Remote Frame)

Los nodos tienen habilidad para requerir información a otros nodos. Un nodo pide una información a los otros y el nodo que tiene dicha información envía una comunicación con la respuesta que puede ser recibida además por otros nodos si están interesados.
En este tipo de mensajes se envía una trama con el identificador del nodo requerido, a diferencia con los mensajes de datos, el bit RTR toma valor recesivo y no hay campo de datos. En caso de que se envíe un mensaje de datos y de petición remota con el mismo identificador, el de datos ganará el acceso al bus puesto que el RTR lleva valor dominante.

2.5.4. Trama de error

Las tramas de error son generadas por cualquier nodo que detecta un error y consiste en dos campos, Indicador de error ("Error Flag") y Delimitador de error (“Error Delimeter”) que permite reiniciar la comunicación tras el error con 8 bits recesivos consecutivos. El indicador de error es distinto según el estado de error del nodo que detecta el error.

En un nodo Activo, el indicador de error interrumpe la comunicación con 6 bits dominantes sucesivos. Esta secuencia rompe la regla de relleno de bits y provocará la generación de tramas de error en otros nodos. Por tanto el indicador de error puede extenderse entre 6 y 12 bits dominantes sucesivos. Finalmente se recibe el campo de delimitación de error formado por los 8 bits recesivos. Entonces la comunicación se reinicia y el nodo que había sido interrumpido reintenta la transmisión del mensaje. Si un nodo en estado de error "Pasivo" detecta un error, el nodo transmite un "Indicador de error pasivo" seguido, de nuevo, por el campo delimitador de error.

El indicador de error de un nodo pasivo consiste en 6 bits recesivos seguido y, por tanto, la trama de error para un nodo pasivo es una secuencia de 14 bits recesivos. De aquí se deduce que la transmisión de una trama de error de tipo pasivo no afectará a ningún nodo en la red, excepto cuando el error es detectado por el propio nodo que está transmitiendo. En ese caso los demás nodos detectarán una violación de las reglas de relleno y transmitirán a su vez tramas de error.

Tras señalar un error por medio de la trama de error apropiada cada nodo transmite bits recesivos hasta que recibe un bit también recesivo, luego transmite 7 bits recesivos consecutivos antes de finalizar el tratamiento de error.

Otro método para la detección de errores es el chequeo de la trama; el campo CRC contiene información para hacer una división polinómica para detectar errores de bits corruptos. El campo ACK en el caso del emisor será recesivo y el receptor deberá sobrescribirlo como dominante y el primero comprobará, mediante la monitorización, que el mensaje ha sido escuchado. Si no sucede así, la trama se considerará corrupta.

2.5.5. Trama de sobrecarga

Tiene el mismo formato que una trama de error activo pero se transmite durante el espacio entre tramas, y este lo hace durante la transmisión del mensaje. Este espacio ha de contar al menos con 3 bits recesivos (intermisión); a continuación cualquier nodo en estado de error activo puede iniciar una transmisión. Si en cambio el nodo está en estado pasivo la

deberá esperar una secuencia adicional de 8 bits recesivos, para así asegurar la transmisión de los nodos en estado activo frente a los de estado pasivo.

Consta de 2 campos, el indicador de sobrecarga y el delimitador. El primero consta de 6 bits dominantes, ampliable a 12 generados por otros nodos. Y el delimitador es de 8 bits recesivos. Hay dos motivos para generar una trama de sobrecarga, para retrasar el mensaje cuando las condiciones del nodo no son las adecuadas para recibir, pudiendo enviar hasta dos consecutivas, y para iniciar la transmisión de una trama de sobrecarga cuando se detecta los 3 bits de intermisión. Por tanto una trama de sobrecarga dará lugar a más, hasta un máximo de 12 bits dominantes de indicador de sobrecarga.

2.6. Especificaciones

La descripción de ISO sobre la especificación de los formatos CAN 2.0A y 2.0B está orientada a los requisitos de fabricación de los controladores:

- El formato CAN 2.0A de trama estándar define la cabecera con 11 bits para el identificador y sus controladores únicamente transmiten y reciben mensajes en este formato, produciéndose un error si se trata del extendido.
- El formato CAN 2.0B de trama extendida define la cabecera con 29 bits para el identificador y tiene dos tipos de controladores. Los pasivos transmiten y reciben en formato estándar, aunque admiten la recepción de los extendidos. Los controlados 2.0B activos transmites y reciben mensajes en ambos formatos.

2.7. Implementaciones

Existen tres tipos de arquitecturas en microcontroladores: Stand-Alone CAN controller, Integrated CAN Controller y Single-Chip CAN Node.

• Stand-Alone CAN Controller

Es la arquitectura más simple, para llevar a cabo una comunicación en una red CAN será necesario:

1. Un microcontrolador como puede ser el 16F877, con el que se ha venido trabajando a lo largo de todo este proyecto.

2. Un controlador CAN que introduzca el protocolo CAN, filtro de mensajes,… y todo el interface necesario para las comunicaciones. Un ejemplo de controlador CAN es el MCP2510 cuya DATA SHEET se puede encontrar en la página web de microchip.

3. Un transceiver CAN, esto es, un transmisor/receptor que puede ser por ejemplo el MCP2551 desarrollado por microchip.

Así pues, si de alguna manera se pretende trabajar en una red CAN con una placa de pruebas, en un principio no sería factible a no ser que de alguna manera se le acoplasen el controlador y el transceptor CAN correspondiente. Microchip ha creado una placa de pruebas específica para este tipo de comunicaciones, la cual puede conseguirse en su página web.

• Integrated CAN Controller

Este tipo de arquitectura consiste en un microcontrolador que incluya, no sólo sus características propias sino además un módulo CAN con las características de un

microcontrolador CAN. El transceiver se sitúa de manera separada. Este es el caso de nuestro microcontrolador dSPIC30F4013. El módulo del transceptor actúa como un controlador de línea, balanceando la señal, esto es, adecuando los niveles de tensión de esta al exterior.

• Single-chip CAN Node

Se trata de un chip que incluye en su interior los tres elementos necesarios para llevar a cabo las comunicaciones en un entorno CAN.

3.- APLICACIONES

Casi todos los vehículos fabricados en Europa, por no decir todos, incorporan al menos un nodo CAN en su sistema electrónico, e incluso los hay con 4 nodos. Hoy en dia, un automóvil puede llegar a tener unas 70 ECU's (Unidades de Control Electronico) para varios subsistemas, como por ejemplo la ECU encargada del control del motor, del ABS, del sistema de sonido, elevalunas eléctricos… El bus CAN se implanta como canal de comunicación entre todos estos sistemas.

En automóviles por ejemplo, es posible que cables, contactos y las propias unidades de mando presenten alguna disfunción. Así que para el análisis de una avería, se debe tener presente que una unidad de mando averiada abonada al Can-Bus, como ya hemos visto, en ningún caso impide que el sistema trabaje con normalidad. Es decir: no será posible llevar a cabo las funciones que implican el uso de información que proporciona la unidad averiada, pero sí todas las demás. En este caso podemos detectar fallos en la red CAN mediante el sistema de diagnostico OBD (del ingles, On-Board Diagnostic). Este sistema permite controlar casi la totalidad del vehiculo, pues como hemos visto, toda la electrónica de los coches modernos se asienta sobre el bus CAN y esta interfaz de diagnostico y control.

Los fabricantes han conseguido muy rápidamente que también los fallos de otros sistemas se puedan leer a través del OBD. Esto ha llevado a que a través del interfaz del OBD a día de hoy prácticamente todo pueda ser leido, ajustado y reiniciado. En coches del consorcio VAG es posible a través de las funciones de confort: regular el cierre centralizado, se pueden ajustar las indicaciones del salpicadero, el aire acondicionado, el comportamiento de la transmisión automática, la regulación del sonido de la radio, y mucho más. Incluso la revisión de la emisión de gases o la regulación de los valores de CO2, pueden ser llevadas a cabo a través del interfaz del OBD.

En la industria, los protocolos de las capas superiores como CANopen y DeviceNet CAN logran la integración entre sistemas. El bus CAN se utiliza en una amplia variedad de industrias manufactureras, principalmente para el control de la máquina integrada, pero también para la automatización industrial, automatización de procesos, y la generación de energía. Un ejemplo es la oficina de ingeniería Suiza "Kyaser" lo introdujo a los fabricantes de maquinaria textil del país, que acabaron fundando el 'Grupo de usuarios textiles CAN'. Los dispositivos más frecuentes en este campo son sensores hidráulicos y neumáticos, actuadores, etc.

En aviación, la capa de aplicación CANaerospace se usa, principalmente, para el control de los sistemas de las aeronaves de pequeñas y medianas empresas. En satélites espaciales y otras aplicaciones del mismo campo, las redes CAN también aparecen, ya que por ejemplo, la ESA, Agencia Espacial Europea, utiliza CANopen como protocolo de capa de aplicación en las redes de sus satélites.

En domótica es cada vez más frecuente encontrar redes basadas en CAN. Muchas casas y oficinas que implementan sistemas domóticas controlan su calefacción-ventilación mediante redes basadas en CAN.

En campos como Agricultrua, Salud, Astronomia o equipamiento de Laboratorio también podemos encontrar redes CAN.

En todas las líneas de uso prima sobre otras soluciones la relación velocidad-coste del bus CAN, que supera con creces otras implementaciones como LIN. De todos modos, en muchas ocasiones, la implementación del bus CAN no excluye al uso del bus LIN, es mas, usualmente ambos buses se complementan, ya que dadas sus diferencias (sobre todo en velocidad, tamaño de datos y confiabilidad) hacen que cada una de las opciones sea mas apropiada para ciertos usos que la otra.

La presencia del bus CAN crece más cada dia, ya que puede adaptarse perfectamente a muchas otras situaciones, como por ejemplo en el Monorail de las Vegas, en los enormes camiones de uso minero-cantero de la marca Liebherr, o en máquinas de empaquetado industrial por Meurer.

BLOQUE 2: CAN EN EL DSPIC30F4013

1.- EMPEZANDO CON CAN EN EL dsPIC30F4013

1.1 Características básicas

CAN se implementa en nuestro microcontrolador como una interfaz serie, utilizada para la comunicación con otros módulos CAN u otros dispositivos del propio microcontrolador.

El módulo está formado por un 'motor de protocolo' (protocol engine) y por un control y almacenamiento de los mensajes. El motor de protocolo CAN maneja todas las funciones encargadas de recibir y transmitir mensajes a través del bus. El estado y los errores de la transmisión pueden consultarse leyendo en los registros apropiados. Además, como ya sabemos, cualquier mensaje detectado en el bus es sometido a un control de errores y después es filtrado para comprobar si debería ser recibido y almacenado en uno de los registros de recepción, o debería por lo contrario ser rechazado.

Básicamente las características técnicas del módulo son las siguientes:

- Implementa el protocolo 'CAN 2.0 A/B' tal y como lo define Bosch, es decir, soporta CAN 1.2, CAN 2.0A, CAN 2.0B Pasivo y CAN 2.0B Activo.
- Soporta tramas de datos estándar y extendidas.
- Máximo 8 bytes de longitud de datos.
- Velocidad de transferencia de datos de hasta 1 Mbit/segundo.
- Receptor de doble búfer
- 6 'filtros de aceptación completa' (full acceptance filter), tanto para mensajes con identificador estándar como extendido.
- 2 'máscaras de filtro de aceptación completa' (full acceptance filter masks)
- 3 bufers de transmisión con asignación de prioridades específicas y capacidad de aborto (cada búfer puede contener hasta 8 bytes de datos)
- Función de 'despertador' (wake-up) programable con filtros paso-bajo integrados.
- Modo Loopback programable con operación de 'auto-test' (self-test).
- Capacidad de señalización a través de interrupciones por parte de todos los receptores y transmisores de estados de error CAN.
- Reloj interno programable.
- 2 modos de baja potencia: Sleep (dormido) e Idle (parado).

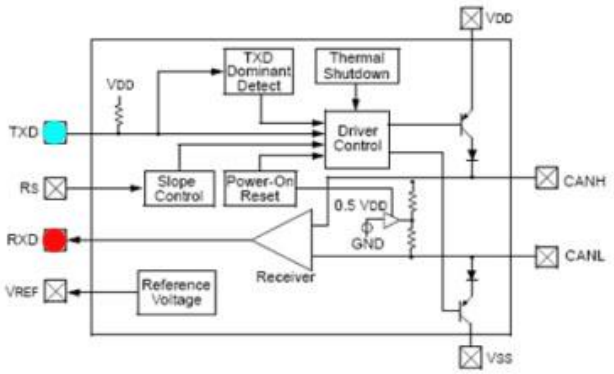
1.2. Transceptor (transceiver)

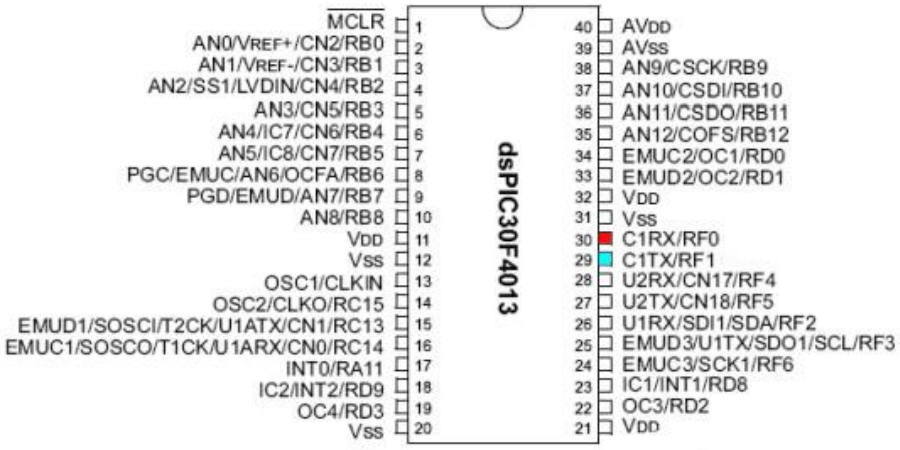
Como ya hemos visto anteriormente, dsPIC30F4013 implementa un controlador CAN integrado, pero el transceptor, es decir, el dispositivo que realiza funciones de envio y recepción dentro de un mismo módulo, no está dentro del propio microcontrolador.

Para poder utilizar el bus CAN en nuestro dsPIC deberemos conectar uno de estos transceptores al microcontrolador. El más sencillo dentro de la gama de Microchip es el MCP2551. Vamos a darle un pequeño vistazo.

Transceptor MCP2551

- Entrada de selección de modo
- Velocidad 1 Mb/s
- Cumple la norma ISO-11898 para la estandarización de la capa física
- Utilizable a 12V y 24V
- Gran inmunidad al ruido





La entrada del transceptor VREF proporciona el diferencial de tensión entre CANH y CANL.

El pin Rs permite elegir entre 3 modos de funcionamiento:

- High Speed mode: Se consigue conectando el pin Rs a VSS.
- Slope Control: Reduce considerablemente las emisiones electromagnéticas disminuyendo los tiempos de subida y de caída de CANH y CANL. Para activarlo, conectaremos una resistencia entre Rs y masa.

- Standby mode: Se consigue poniendo a nivel alto el pin Rs. En este modo, el transmisor está apagado, y el receptor funciona ralentizado, es decir, el pin del receptor RXD seguirá operativo, pero de forma más lenta.

TXD y RXD son los pines a conectar al microcontrolador

CANH y CANL son las salidas CAN del transceptor, es decir, las que adecuan la señal de entrada del bus al sistema.

Ahora que ya sabemos que necesitamos para utilizar el bus CAN con nuestro microcontrolador, vamos a centrarnos en como funciona el módulo de control que éste trae integrado.

2.- MÓDULO DE CONTROL CAN EN EL dsPIC30F4013

2.1 Modos de funcionamiento

El módulo puede trabajar en uno de entre varios modos de funcionamiento:

- Modo de inicialización (Initialization mode)
- Modo deshabilitado (Disable mode)
- Modo de funcionamiento normal (Normal operation mode)
- Modo de solo escucha (Listen only mode)
- Modo en bucle (Loopback mode)
- Modo de reconocimiento de errores (Error recognition mode)

Los modos se solicitan poniendo los bits del registro REQOP<2:0> (CiCTRL<10:8>), excepto el modo de reconocimiento de errores que se selecciona a través de los bits RXM<1:0> (CiRXnCON<6:5>, donde n=0 o n=1 representan un búfer de recepción en particular). La entrada en un modo se reconoce observando los bits del registro OPMODE<2:0> (CiCTRL<7:5>). El módulo no cambiará de modo, ni tampoco cambiará los bits OPMODE mientras el cambio de modo no sea correcto. Generalmente, para que sea correcto ha de hacerse durante el tiempo en que el bus está parado (Idle time), es decir, cuando se envían al menos 11 bits recesivos consecutivos.

Modo de inicialización (Initialization mode)

En este modo, el módulo no puede transmitir ni recibir datos. Al seleccionarlo, los contadores de error se ponen a 0 y las 'banderas de interrupción' (interrupt flags) permanecen inalterables. A la hora de programar, desde él tendremos acceso a ciertos registros de configuración a los cuales no se puede acceder desde ningún otro modo, con la opción de corregir daños que podamos ocasionar al protocolo CAN durante la programación.

Para poder acceder al *modo de configuración* (Configuration mode) no se puede estar en mitad de una transmisión; es por ello que ninguno de los registros que controlan la configuración del módulo puede ser modificado mientras el módulo esté conectado. Dicho modo de configuración protege los siguientes registros:

- Registros de control del módulo (Module control registers)
- Registros de tasa (velocidad de transmisión) en baudios (Baud rate registers) y registros de configuración de interrupciones (Interrupt configuration registers)
- Registros de temporización del bus (Bus timing registers)
- Registros identificadores de los filtros de aceptación (Identifier acceptance filter registers)
- Registros identificadores de las máscaras de aceptación (Identifier acceptance mask registers).

Modo deshabilitado (Disable mode)

Nos encontramos con que el módulo tampoco transmite ni recibe datos. Si hay actividad en el bus, el módulo pondrá a 1 los bits del registro WAKIF, mientras que las interrupciones sin ejecutar quedaran pendientes y los contadores de error conservaran su valor.
Si los bits REQOP<2:0> (CiCTRL<10:8>) = 001, el módulo entrará en el modo deshabilitado. Si en ese momento el módulo está activo, entonces esperará hasta que le lleguen 11 bits recesivos desde el bus, los cuales detectará e interpretará como a que el bus está sin utilizar (Idle bus); después aceptará el comando de módulo deshabilitado (module disable command). Cuando los bits OPMODE<2:0> (CiCTRL<7:5>) = 001, el módulo habrá entrado

satisfactoriamente al modo. Los pins de entrada/salida (I/O) volverán a su función normal cuando el módulo esté en el modo de módulo inutilizado (Module Disable mode).
El módulo puede ser programado para aplicar un filtro paso-bajo a la línea de entrada CiRX, mientras el módulo o la CPU estén en modo dormido (Sleep mode). El bit WAKFIL (CiCFG2<14>) es el que activa o desactiva este filtro.

NOTA: Si el módulo va a trabajar en un modo de funcionamiento particular y es solicitada una transmisión inmediatamente después de que el módulo CAN se haya colocado en ese modo de funcionamiento, esperará 11 bits recesivos consecutivos en el bus antes de empezar la transmisión. Si el usuario decide cambiar al modo deshabilitado (disable mode) dentro del periodo de esos 11 bits recesivos, la transmisión será cancelada y se colocará el bit TXABT y será borrado el bit TXREQ.

Modo de funcionamiento normal (Normal operation mode)

El modo de funcionamiento normal, se selecciona con los bits REQOP<2:0> = 000. Al entrar en este modo, el módulo transmitirá y recibirá mensajes del bus a través de los pins CxTX y CxRX.

Modo de solo escucha (Listen only mode)

Cuando activamos este modo, el módulo está pasivo. Los bufers de transmisión funcionan como puertos de entrada/salida (I/O), y los pins receptores por su parte, continúan como entradas. El receptor deja de enviar 'banderas de error' (error flags) y 'señales de reconocimiento' (acknowledge signals), y los contadores de error se desactivan. El modo de solo escucha puede ser utilizado para detectar la tasa (velocidad de transmisión) en baudios sobre el bus. Parece evidente que para poder utilizarlo, es necesario que haya al menos dos nodos remotos que se comuniquen el uno con el otro.

Modo de escucha de todos los mensajes (Listen all messages mode)

El módulo es fijado para ignorar todos los errores y poder recibir así cualquier mensaje que se ha enviado a través del bus. Para activarlo, debemos poner los bits de REQOP<2:0> = '111'. En este modo, los datos que están en el buffer de ensamblado de mensajes (message assembly buffer) hasta que ocurre un error, son copiados en el búfer de recepción y pueden ser leídos a
través de la interfaz de la CPU.

Modo loopback (Loopback mode)

Cuando el modo loopback se activa, la señal interna de transmisión y la señal interna de recepción se conectan en el límite del módulo, formando una especie de bucle cerrado. Los pins de transmisión y de recepción vuelven a sus funciones de puerto de entrada/salida I/O.

2.2 Recepción de mensajes

2.2.1 Bufers de recepción

El módulo del bus CAN se compone de 3 bufers de recepción. Dos de ellos son el RXB0 y RXB1 y reciben simultáneamente mensajes completos. Existe otro bufers llamado 'Búfer de ensamblado de mensajes' (Message Assembly Buffer o MAB) que se dedica continuamente a escuchar el bus en espera de mensajes entrantes.

Todos los mensajes son montados por el MAB y son transferidos a los bufers RXBn solo si se encuentran los filtros de criterios de aceptación. Cuando se recibe un mensaje, la bandera (flag) RXnIF (CiINTF <0> o CiINTF <1>) se pone a 1. Este bit solo puede ser activado por el modulo cuando se recibe un mensaje. La CPU pone el bit a cero cuando ha completado el procesado del mensaje en el búfer. Si el bit RXiIE (CiINTE <0> o CiINTE <1>) está a 1, se generará una interrupción cuando se reciba un mensaje. Los filtros RXF0 y RXF1 con la máscara RXM0 están asociados con RXB0. Los filtros RXF2, RXF3, RXF4 y RXF5 y la máscara RXM1 están asociados con RXB1.

2.2.2 Filtros de aceptación de mensajes

Los filtros y mascaras de aceptación de mensajes se usan para determinar si un mensaje del búfer de ensamblado de mensajes debe ser cargado en alguno de los bufers de recepción. Una vez un mensaje válido ha sido recibido por el MAB, los campos de identificación del mensaje son comparados con los valores de los filtros. Si hay una coincidencia, ese mensaje será cargado en el búfer de recepción apropiado. El filtro de aceptación busca en los mensajes entrantes el bit RXIDE (CiRXnSID<0>) para comparar los identificadores:

- Si el bit RXIDE esta desactivado, el mensaje es una trama estándar y solo se comparan los filtros con el bit EXIDE (CiRXFnSID<0>) a cero.
- Si el bit RXIDE está puesto a 1, el mensaje es una trama extendida, y solo son comparados los filtros con el EXIDE a 1. Configurando los bits RXM<1:0> a '01' ó a '10' se puede borrar el bit EXIDE.

2.2.3 Máscaras de filtro de aceptación de mensajes

Los bits de máscara determinan que bits se han de aplicar en el filtro. . Encontramos 2 máscaras de filtro de aceptación programables, una por cada búfer de recepción. Si un bit de máscara está a cero, entonces este bit será aceptado automáticamente independientemente del bit de filtro.

2.2.4 Sobrecarga de recepción:

Existen tres casos de sobrecarga de recepción:

- El MAB ha ensamblado un mensaje recibido válido,
- El mensaje es aceptado a través del filtro de aceptación,
- El búfer de recepción asociado con el filtro no ha sido designado como vacío por el mensaje previo.

El flag de error de sobrecarga, RXnOVR (CiINTF<15> o CiINTF<14>), y el bit ERRIF (CiINTF<5>) son puestos a 1 y el mensaje del MAB es descartado.

Si el bit DBEN está desactivado, RXB1 y RXB0 operan independientemente. Cuando esto ocurre, un mensaje dirigido a RXB0 no es desviado a RXB1 si el RXB0 contiene un mensaje no leído y el RX0OVR se pone a 1.

Si el bit DBEN esta activado, la sobrecarga del RXB0 se trata de forma diferente. Si se recibe un mensaje válido para RXB0 y el bit RXFUL = 0, el mensaje para RXB0 se carga en RXB1. No se genera un mensaje de error de sobrecarga para RXB0. Si se recibe un mensaje valido para RXB0 y el bit RXFUL = 1, tanto RXB0 como RXB1 están llenos, por lo tanto, el mensaje se perderá y se generará un mensaje de error de sobrecarga para RXB1.

2.2.5 Errores de recepción

El Modulo CAN puede detectar los siguientes errores de recepción:

- Error de Chequeo de Redundancia Cíclica (CRC)

- Error de Relleno de Bits (Bit Stuffing Error)

- Error de Mensaje Inválido Recibido

Estos errores de recepción no generan una interrupción. Sin embargo, si que se incrementa el contador de errores de recepción. Cuando el contador de errores de recepción llega al límite (96), advierte a la CPU generando una interrupción, todo esto se produce con el bit RXWAR (CiINTF<9>).

2.2.6 Interrupciones de recepción

Las Interrupciones de recepción se pueden dividir en 3 grandes grupos, cada uno de ellos incluye varias condiciones que generan interrupciones:

- Interrupción de recepción: Un mensaje se ha recibido correctamente y se ha cargado en uno de los bufers de recepción. Esta interrupción se activa inmediatamente después de recibir el campo Fin de Trama (EOF). El flag RXnIF indica que buffer de recepción ha causado la interrupción.

- Interrupción de despertador (Wake-up Interrupt): El módulo CAN ha despertado del modo deshabilitado (disable) o durmiendo (sleep)
- Interrupción de error de recepción: Se señala en el bit ERRIF. La fuente del error puede ser determinada comprobando los bits del registro Estatus de Interrupción CAN, CiINTF.
- Mensaje Inválido Recibido: Si ocurre algún tipo de error durante la recepción del ultimo mensaje, se indicará un error mediante el bit IVRIF.
- Sobrecarga de Receptor: El bit RXnOVR indica que ha ocurrido un error de sobrecarga.
- Advertencia del Receptor: El bit RXWAR indica que el contador de errores de recepción (RERRCNT<7:0>) ha alcanzado el límite de advertencia de 96.
- Error de Recepción Pasivo:El bit RXEP indica que el contador de errores de recepción ha excedido el límite de error pasivo de 127 y el módulo ha entrado en el estado error pasivo.

2.3. Transmisión de mensajes

El módulo CAN tiene 3 bufers de transmisión, con un tamaño de 14 bytes cada uno. Los ocho primeros bytes son los que puede ocupar como máximo el mensaje transmitido. El resto son los identificadores estándar, extendidos y otra información de control del mensaje.

2.3.1 Prioridad de transmisión del mensaje:

La prioridad de transmisión es una priorización dentro de cada nodo de los mensajes pendientes de transmitir. Hay 4 niveles de prioridad de transmisión:

-Si TXPRI<1:0> (CiTXnCON<1:0>, donde n = 0, 1 o 2 representa el búfer de transmisión en particular) para un búfer de mensaje en particular se establece a ‘11’, ese búfer tiene la máxima prioridad.

-Si TXPRI<1:0> para un búfer de mensaje en particular se establece a ‘10’ o a ‘01’, ese búfer tiene una prioridad intermedia.

-Si TXPRI<1:0> para un búfer de mensaje en particular se establece a ‘00’, ese búfer tiene la mínima prioridad.

2.3.2 Secuencia de transmisión:

Para iniciar la transmisión del mensaje, el bit TXREQ (CiTXnCON<3>) debe ponerse a 1. El módulo del bus CAN resuelve cualquier conflicto de tiempo entre el establecimiento del bit TXREQ y el Inicio de Trama (SOF), asegurando que si la prioridad ha sido cambiada, se ha resuelto correctamente antes de llegar al SOF. Cuando se pone a 1 TXREQ, los bits del flag TXABT (CiTXnCON<6>), TXLARB (CiTXnCON<5>) y TXERR (CiTXnCON<4>) son puestos a 0 automáticamente. Poner a 1 el bit TXREQ simplemente marca un búfer de mensaje como

puesto en cola para transmisión. Cuando el módulo detecta el bus disponible, empieza a transmitir el mensaje que tiene mayor prioridad.

- Si la transmisión se completa satisfactoriamente en el primer intento, el bit TXREQ se pone a 0 automáticamente y se genera una interrupción si TXIE había sido activado.
- Si la transmisión del mensaje falla, se activa uno de los avisos de condición de error y el bit TXREQ permanece a 1, indicando que el mensaje está aún pendiente de transmisión.
- Si el mensaje encuentra una condición de error durante el intento de transmisión, el bit TXERR se pone a 1 y puede causar una interrupción. Si el mensaje colisiona durante el intento de transmisión, se pone a 1 el bit TXLARB. No se genera interrupción para indicar la perdida.

2.3.3 Abortando la transmisión del mensaje

El sistema también puede cancelar en envío de datos de diferentes formas:

-Aborta el envío de un mensaje poniendo a 0 el bit TXREQ asociado con cada búfer de mensaje.

-Poniendo a 1 el bit ABAT(CiCTRL<12>) se solicita abortar todos los mensajes pendientes.

-Poniendo a 1 el bit TXABT y la bandera (flag) TXnIF no se activa automáticamente. Si el mensaje aún no ha iniciado la transmisión, o si el mensaje la ha iniciado pero es interrumpida por pérdida de arbitraje o por un error, se procesa el aborto.

2.3.4 Errores de transmisión

El modulo CAN detecta los siguientes errores de transmisión:

- Error de ACK

- Error de forma

- Error de bit

Estos errores de transmisión no generan necesariamente una interrupción, pero son indicados por el contador de errores de transmisión. Una vez que el contador de errores supera el valor de 96, los bits ERRIF (CiINTF<5>) y TXWAR (CiINTF<10>) son puestos a 1. Además, se genera una interrupción y se pone a 1 el bit TXWAR del registro de banderas de error.

2.3.5 Interrupciones de transmisión

Las Interrupciones de transmisión que destacan son las siguientes:

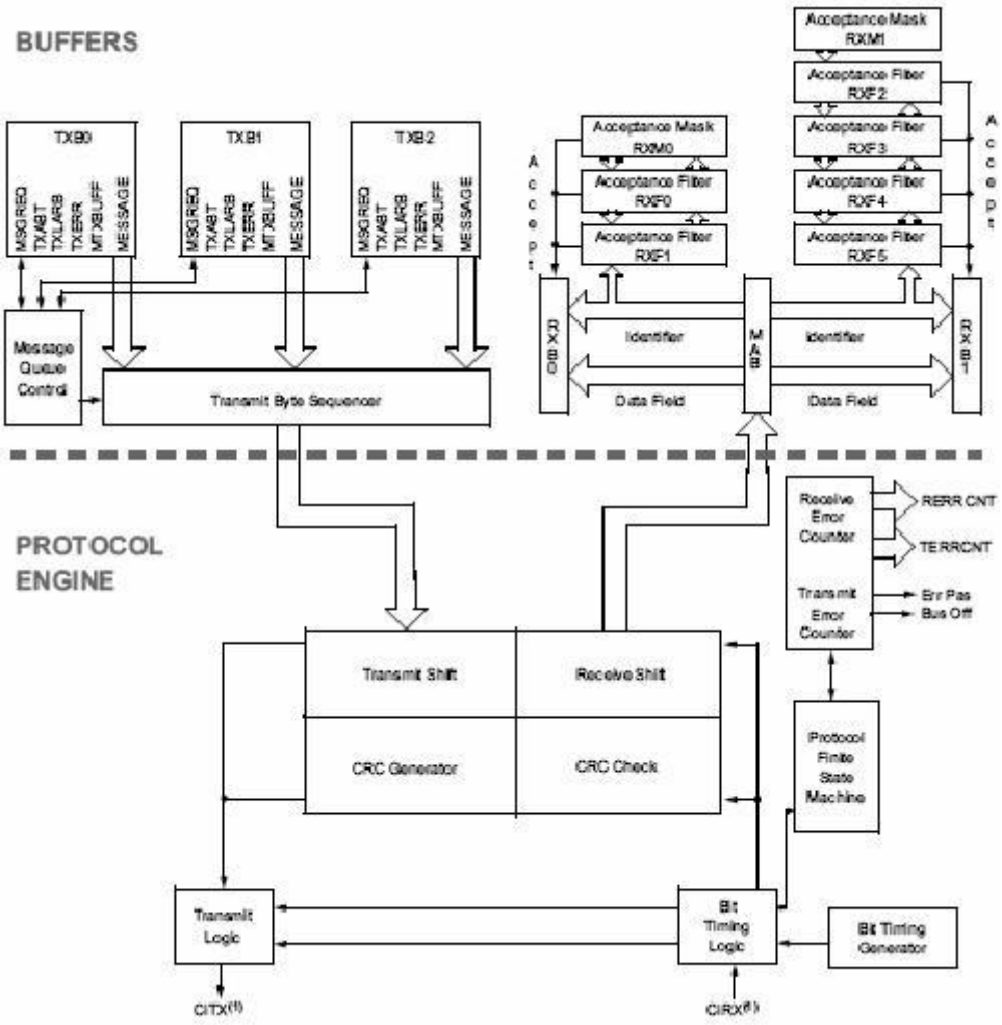
- Interrupciones de transmisión: Al menos uno de los tres bufers de transmisión está vacío y puede ser cargado una para programar transmisión de mensaje. Los flags TXnIF indican que búfer de transmisión está disponible y ha sido el causante de la interrupción.

- Interrupciones de errores de transmisión: Una interrupción error de transmisión se indica con el flag ERRIF. La fuente del error se puede determinar comprobando los flags de error en el registro de Estatus 50 de Interrupción del CAN, CiINTF. Los flags de este registro están relacionados con los errores de transmisión y de recepción.

- Interrupción de Advertencia del Transmisor: El bit TXWAR indica que el contador de errores de transmisión ha alcanzado el límite de advertencia de la CPU (96).

- Error Pasivo del Transmisor: El bit TXEP (CiINTF<12>) indica que el contador de errores de transmisión ha superado el límite de error pasivo de 127 y el modulo ha entrado en el estado de error pasivo.

- Bus Off: El bit TXBO (CiINTF<13>) indica que el contador de errores de transmisión ha superado 255 y el módulo ha entrado en el estado de bus off.



2.4 Temporización y sincronización de bits

2.4.1. Temporización de bits

Sabemos que las señales eléctricas del bus sufren las alteraciones propias de la distorsión
que se produce, por un lado por las características físicas de la línea, por los retardos de propagación, y los retardos producidos en los propios controladores, y como por las posibles fuentes externas de interferencia electromagnética.

Por otra parte, cada controlador CAN depende, normalmente, de un oscilador distinto, lo que provoca diferencias de frecuencia que pueden dar lugar a desfases en el muestreo de tramas de los distintos nodos. Por ello, los controladores siguen un proceso de muestreo y resincronización orientado a evitar los desajustes debidos a estas eventuales circunstancias eventuales. Y estos desajustes se manifiestan especialmente en el arbitraje de mensajes, cuando varios nodos transmiten de forma simultánea, y es casi imposible que esten completamente sincronizados; y la frecuencia del oscilador de ambos puede ser ligeramente distinta. Los receptores que se han sincronizado con el primer flanco de bajada detectado, Tienen que ir resincronizándose sucesivamente a los flancos producidos por el nodo que acabe transmitiendo. CAN utiliza señalización asíncrona y un tipo de codificación NRZ, en la que normalmente 0V representa un '0' lógico, y 5V representa un '1' lógico. Es necesaria, por tanto, una operación de muestreo por parte de los receptores, que se han de resincronizar periódicamente. Además, la tasa de transmisión (desde menos de 1 kbps hasta 1 Mbps) también es un factor determinante. Todos estos parámetros pueden programarse individualmente y ser ejecutados por el reloj lógico de CAN (Bit Timing Logic).

Para que el controlador, que es un sistema digital síncrono con una frecuencia de reloj generada por un oscilador (generalmente un oscilador de cuarzo), pueda muestrear la señal asíncrona recibida con seguridad y precisión ha de utilizar una frecuencia de muestreo superior a la de la señal transmitida. Y por otra parte, el controlador ha de utilizar una frecuencia de reloj que sea múltiplo de la 'frecuencia nominal de bit' (número de bis transmitidos en ausencia de resincronizaciones por un transmisor ideal). Esta frecuencia se obtiene como cociente de la frecuencia del oscilador. La velocidad de comunicación a la que queda configurado el controlador, para un oscilador de cierta frecuencia, queda definida por el valor del 'divisor de frecuencia' (BRP), también configurable, y por el número de 'TQ' por bit. TQ (quantum de tiempo) es una unidad de tiempo fija, derivada del periodo del oscilador.

Asi, debemos cambiar de escala para bajar la frecuencia del reloj, y lograremos adecuar el sistema a nuestras necesidades. Este valor para nuestro dsPIC30F4013, se define en su correspondiente manual como 'ajuste de preescalado' (prescaler setting) y se obtiene a partir de la siguiente ecuación:

$$TQ = \frac{2 \cdot (BRP<5:0>+1)}{FCAN}$$

donde FCAN es FCY (si el bit CANKS está puesto a 1) o 4·FCY (si CANKS está puesto a 0).

Además, FCAN no debe sobrepasar nunca los 30 MHz, es decir que si CANKS = 0, FCY no debe exceder de 7.5 MHz, ya que como hemos dicho FCAN=4·FCY en este caso.

Además, para confirmar la funcionalidad de la red, se incluye en una posición concreta del 'tiempo de bit' (del cual hablaremos más adelante), un 'punto de muestreo' (sample point en inglés). Este es el punto exacto en el que el nivel de bus es leído e interpretado como el valor de ese respectivo bit. En algunos microcontroladores, entre los que se incluye dsPIC30F4013, si la temporización de bits se retarda y tiene muchos TQ, es posible realizar también, un muestreo múltiple sobre el bus. En este caso, las muestras para realizar el cálculo pueden ser tomadas en el punto de muestreo o en dos posiciones anteriores a este con una distancia de TQ/2. Además, el módulo CAN permite al usuario elegir entre muestrear tres veces en el mismo punto o una vez en cada uno de los tres puntos, según pongamos a '0' o a '1' el bit
SAM (CiCFG2<6>).

Tenemos que en este modelo de muestreo el nivel determinado por el bus corresponde
al resultado por mayoría de tres valores. Para asegurar la validez del cálculo, este muestreo debe ocupar del 60 al 70 por ciento del tiempo de bit, dependiendo de los parámetros del sistema.

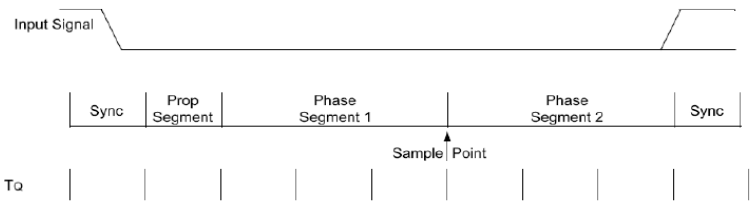
2.4.2 Resincronización de bits (Resynchronization)

Se basa en el acortamiento o alargamiento del tiempo de bit (o número de TQ por bit) para que el punto de muestreo se desplace con relación al último flanco recesivo-dominante detectado. Y esta decisión la toma el controlador tras detectar el flanco y evaluar el error.
En CAN se siguen dos métodos de sincronización del muestreo con la secuencia de bits de la trama:

Sincronización de trama ("Hard Synchronization"): Es la reinicialización de la lógica de muestreo de bit que ocurre ante el flanco de bajada (paso de recesivo a dominante) producido por un bit de inicio de trama. Resincronización o sincronización intra-trama ("Soft Synchronization" o "Resyncronization): Es un proceso de sincronización que ocurre en cada
flanco de bajada que se produce durante la transmisión de la trama Puede producir el alargamiento o el acortamiento de los segmentos de fase de los que hablamos en la siguiente sección.

2.4.3 Parámetros del tiempo de bit nominal

El tiempo de bit nominal se ideó como una división del tiempo en segmentos separados y no superpuestos. El microcontrolador dsPIC30F4013 define para el tiempo de bit nominal un mínimo de 8 TQ y un máximo de 25 TQ, mientras que define el mínimo tiempo de bit nominal como 1 µsegundo, correspondiente a una tasa máxima (velocidad de transmisión máxima) en bits de 1 MHz. Su esquema gráfico es el siguiente:



Y aquí podemos observar como el punto de muestreo del bit (sample point) se realiza en seguida del segmento Phase_Seg1 y antes de Phase_Seg2. Las funciones y los tiempos de cada uno de los segmentos son los siguientes:

- Segmento de sincronización (Sync_Seg): En este intervalo se espera que se cree cualquier cambio de recesivo a dominante en el bus. Su duración es de 1 TQ.

- Segmento de propagación (Prop_Seg): Encaminado a la compensación del atraso de tiempo de la red. Y este retardo es la suma del inducido por la línea y el encargado a la electrónica de la interfaz de los controladores. El segmento puede configurarse como de 1 a 8 TQ de
- duración poniendo a 1 los bits PRSEG<2:0> (CiCFG2<2:0>).

- Segmento de fase 1 (Phase_Seg1): Está oencaminado a la detección y reajuste de desfases entre nodos. Puede configurarse como de 1 a 8 TQ de duración poniendo a 1 los bits SEG1PH<2:0> (CiCFG2<5:3>), aunque puede prolonga se en operaciones de
- resincronización.

- Segmento de fase 2 (Phase_Seg2): Encargado de retardar la proxima transición de datos a transmitir. Su valor máximo puede ser igual al segmento de fase 1 más grande o al tiempo de procesado de la información (2 TQ). El segmento es programable desde 1TQ hasta 8 TQ de duración poniendo a 1 los bits SEG2PH<2:0> (CiCFG2<10:8>).

- Puede acortarse (nunca menos del tiempo de procesado de la información) en las operaciones de resincronización. Tenemos que un requisito imprescindible que deben cumplir los segmentos de fase para el ajuste de sus respectivas longitudes es el siguiente:

- Prop Seg + Phase 1 Seg >= Phase 2 Seg

Y terminando, en cuanto al cambio de longitudes por la resincronización, se define un parámetro adicional: el 'salto de resincronización' (Sjw o Resynchronization jump width) que establece el valor del salto máximo que se pueda llegar a efectuar como producto de la resincronización. En dsPIC30F4013 se especifica a través de los bits del registro SJW<1:0> (CiCFG1<7:6>), con valores de entre 1 y 4 TQ, pero no mayores que Phase_Seg2, ya que durante el proceso, el valor de este parámetro o bien será añadido al segmento de fase 1 o bien recortado del segmento de fase 2. Es decir, ha de cumplirse que Phase2 Seg > Synchronization Jump Width

3.- PROGRAMACIÓN

3.1 Programación

MPLAB es un editor IDE gratuito, destinado a productos de la marca Microchip. Este editor es modular, permite seleccionar los distintos microprocesadores soportados, además de permitir la grabación de estos circuitos integrados directamente al programador. En concreto permite trabajar con microcontroladores PIC y dsPIC desarrollados y fabricados por la empresa "Arizona Microchip Technology" (AMT), que son los que utilizaremos nosotros. MPLAB incorpora todas las utilidades necesarias para la realización de cualquier proyecto así como compiladores de alto nivel (nosotros usaremos el c30). Podremos ensamblador nuestro proyecto y simularlo en pantalla, pudiendo ejecutarlo posteriormente en modo paso a paso y ver como evolucionarían de forma real tanto sus registros internos, la memoria RAM y/o EEPROM de usuario como la memoria de programa, según se fueran ejecutando las instrucciones. Además el entorno que se utiliza es el mismo que si se estuviera utilizando un emulador.

Es un programa que corre bajo Windows y como tal, presenta las clásicas barras de programa, de menú, de herramientas de estado, etc. El ambiente MPLAB® posee editor de texto, compilador y simulación (no en tiempo real).

Programador Picstart Plus de Microchip

Una vez escrito y depurado el programa, se procede a la compilación. Para esto, desde el menú PROYECT se elige la opción BUILD ALL (construir todo) que, si no existen errores, devolverá un mensaje como BUILD SUCCESFULL. Los errores que muestra el compilador son del tipo sintácticos, es decir que si se llegara a tener un error, se espera que se ponga un bit en "0" y nunca pasa, se estará en un bucle infinito y el compilador compilará perfectamente porque no hay error de sintaxis.

También existen mensajes y advertencias; los mensajes pueden ser, por ejemplo, que se está trabajando en un banco de memoria que no es el bank 0, etc. Las advertencias tienen un poco más de peso, por ejemplo: el PIC seleccionado no es el mismo que esta definido en el programa, etc. En ambos casos, mensajes y advertencias, la compilación termina satisfactoriamente pero hay que tener en cuenta siempre lo que nos dicen estos para prevenir errores.

Terminada la compilación el MPLAB® nos genera un archivo de extensión *.hex* el cual es completamente entendible para el PIC. Es decir, solo resta grabarlo al PIC por medio de una interfaz como por ejemplo el programador Picstart Plus de microchip. Una vez completado esto, se alimenta al mismo y el programa ya se estará ejecutando.

2.5.1 Ejemplo de programación

En este ejemplo se utiliza la técnica de polling o espera activa para comprobar si un botón esta pulsado, y si es así, encender un LED. Esta técnica consiste en tener el dispositivo "preguntando" constantemente si está pulsado.

```
#include <p18cxxx.h>
#include "j1939.h"

/*Definir algunos valores arbitrarios que estén de acuerdo con los valores de los demás
nodos*/

#define OTHER_NODE 129
#define TURN_ON_LED 92
#define TURN_OFF_LED 94
void main( void )

unsigned char LastSwitch = 1;
unsigned char CurrentSwitch;
TRISBbits.TRISB4 = 1; // Switch pin
TRISD = 0; // LED pins
LATD = 0; // Turn off LED
J1939_Initialization( TRUE );

// Wait for address contention to time out

while (J1939_Flags.WaitingForAddressClaimContention){
J1939_Poll(5);
}
// Now we know our address should be good, so start checking for
// messages and switches.
while (1){
  CurrentSwitch = PORTBbits.RB4;
  if (LastSwitch != CurrentSwitch){
    Msg.DataPage = 0;
    Msg.Priority = J1939_CONTROL_PRIORITY;
    Msg.DestinationAddress        = OTHER_NODE;
    Msg.DataLength = 0;
    if (CurrentSwitch == 0)
      Msg.PDUFormat = TURN_ON_LED;
    else
      Msg.PDUFormat = TURN_OFF_LED;
    while (J1939_EnqueueMessage( &Msg ) != RC_SUCCESS){
      J1939_Poll(5);
      LastSwitch = CurrentSwitch;
    }
  }
  while (RXQueueCount > 0){
    J1939_DequeueMessage( &Msg );
    if (Msg.PDUFormat == TURN_ON_LED)
      LATDbits.LATD0 = 1;
    else if (Msg.PDUFormat == TURN_OFF_LED)
      LATDbits.LATD0 = 0;
  }
// Since we don't accept the Commanded Address message,
// the value passed here doesn't matter.
  J1939_Poll(20);
  }
}AN930
```