Introducción

- Para el uso de los STR es necesario un conocimiento del hardware:
  - El hardware influye en el comportamiento temporal del sistema. Debe ser determinista
  - Para obtener suficientes prestaciones en ocasiones es necesario programar a bajo nivel
  - A menudo son necesarios dispositivos específicos y hay que programarlos directamente
  - El mecanismo de interrupciones debe ser coherente con el manejo de prioridades
  - Los mecanismo de comunicación influyen en el comportamiento temporal. Es necesario que los buses / redes, y los protocolos utilizados sean deterministas

Entrada / Salida

- Suele ser necesario interactuar con dispositivos de E/S de propósito especial
- Existen dos clases generales de arquitecturas:
  - La memoria y los periféricos utilizan buses independientes a nivel lógico (normalmente con menos líneas para los periféricos)
  - La memoria y los periféricos comparten el mismo bus
- El interfaz con los periféricos suele ser un conjunto de registros
- Se usan instrucciones especiales para acceder a los periféricos:
  - Instrucciones especiales para acceder al bus de los periféricos
  - Para manejar tamaños de palabra diferentes a los del bus de la memoria
- Existen dos mecanismos para acceder y sincronizarse con los periféricos:
  - Control por supervisión de estado
  - Control por interrupción

Control por supervisión de estado

- Se deben realizar tests del periférico para conocer su estado, antes de realizar las acciones apropiadas
- Se usan tres tipos de instrucciones:
  - Operaciones de test para obtener el estado
  - Operaciones de control para preparar el periférico antes de las operaciones
  - Operaciones de entrada / salida
- Aunque han sido muy comunes, se están sustituyendo por periféricos con control por interrupción
- Las interrupciones presentan el inconveniente que introducen un comportamiento no determinista

Control por interrupción

- Existen tres variaciones en el control por interrupción
  - Controlada por programa
    - El periférico solicita la interrupción
    - Se activa el manejador que realiza la operación
    - Se restaura el estado del procesador
  - Iniciada por programa
    - Se utiliza un dispositivo de Acceso Directo a Memoria (DMA)
    - El DMA actúa como el procesador en la transferencia de información compartiendo el uso de la memoria
    - Al finalizar la operación se notifica con una interrupción
  - Controlada por canal programable
    - Es similar a la iniciada por programa por el DMA se sustituye por un dispositivo programable, eliminando al máximo la participación del procesador
    - Permite realizar operaciones de entrada / salida más complejas

Programación a bajo nivel

- Tradicionalmente se realizaba en ensamblador
- Los lenguajes de alto nivel modernos ofrecen mecanismos para programar a bajo nivel que permiten acceder directamente a los periféricos
- Las facilidades que deben ofrecer son:
  - Facilidades de modularidad y encapsulamiento:
    - Separar las secciones de software no portable
    - Acceder desde un interfaz compatible con el lenguaje
  - Modelos abstractos para el manejo de periféricos:
    - Representar, direccionar y manipular los registros del periférico (variable de programa, objeto o un canal de comunicación)
    - Una representación adecuada de las interrupciones

Representación de las interrupciones

- Formas de representación:
  - Procedimiento
  - Proceso esporádico
  - Evento asíncrono
  - Condición de sincronización por variable condición
  - Sincronización basada en paso de mensaje
- Cada lenguaje puede usar un modelo distinto. El más usual es el de procedimiento
- En Ada es un híbrido entre el modelo de procedimiento y el de sincronización por variable compartida. La interrupción se mapea sobre un procedimiento de un objeto protegido y los registros se ven como variables de programa
- EL KMOS adopta la sincronización basada en paso de mensaje

Facilidades del lenguaje Ada

- El manejo de un periférico se encapsula en una unidad protegida. El manejador de una interrupción se trata con un procedimiento sin parámetros de la unidad protegida
- Para manejar los registros se utilizan las cláusulas de representación
  - Representación de enumeración
  - Representación de registros
  - Definición de atributos
- Las cláusulas se utilizan para hacer coincidir la representación de los tipos que hace el lenguaje con las características del periférico

Cláusulas de representación en ADA

```
type Mode_select is new integer range 0..5;
type Format_select is (Latch, MSB_only, LSB_only, LSB_and_MSB);
type Counter_select is new integer range 0..2;
type Timer_control is record
    BCD : Boolean;
    Mode: Mode_select;
    Format: Format_select;
    Counter: Counter_select ;
end record;

for Format_select use (Latch => 0, MSB_only => 1,
               LSB_only >= 2, LSB_and_MSB => 3);

for Timer_control use record
    BCD     at 0 range 0..0;
    Mode    at 0 range 1..3;
    Format  at 0 range 4..5;
    Counter at 0 range 6..7;
end record;

for Timer_control'Size use 8;
for Timer_control'Alignement use 1;
for Timer_control'Bit_Order use Low_Order_First;

Timer_control_Reg: Timer_control;
for Timer_control_Reg use System.Storage_Element.To_Address(16#43#);

Timer_control_Reg := (BCD => False, Mode =>3,
                      Format => Latch, Counter => 0);
```

Ensamblador en Ada

- Ada permite el uso de código ensamblador integrado en los programas
- Los módulos que contengan código ensamblador no pueden tener sentencias de Ada
- El subprograma en ensamblador dispondrá de una parte visible en formato Ada de definición de subprograma

```
Procedure In_Op; pragma Inline(In_op);

procedure In_Op is
     use System.Machine_code;
begin
     My_Machine_Format'(Code => In_instruction,
                        Reg => 1, Port => 1);
     My_Machine_Format'(Code => SAVE, Reg => S'Address);
end;
```


Facilidades en C / C++

- Al no soportar la multitarea, no existe un mecanismo para comunicar los manejadores de interrupción con las tareas
- Se suele usar una librería o SO que lo soporte
- Diferencias entre un manejador y un procedimiento
  - No debe alterar ningún registro de la CPU
  - Debe restaurar el registro de flags
- La construcción del manejador se puede hacer:
  - Con un servicio del sistema que soporta la multitarea
  - Con código en ensamblador que llama a la función de C

Facilidades en C / C++

- Se pueden definir estructuras a nivel de bit

```
typedef struct {
            unsigned char bcd     : 1;
            unsigned char mode    : 3;
            unsigned char format  : 2;
            unsigned char counter : 2;
} Timer_control;

enum { Latch = 0, MSB_Only = 1, LSB_Only = 2, LSB_and_MSB = 3 };

Timer_control *register, shadow;

shadow.bcd = 0;
shadow.mode = 3;          /* Onda cuadrada */
shadow.format = LSB_and_MSB;
shadow.counter = 0;

*register = shadow;
```

- En algunos compiladores se pueden incluir código ensamblador

```
#define INT_DOS asm(“ int $0x21”);
#define disable() asm(“ cli”);
```


Planificación y drivers

- Se debe incluir en la planificación el tratamiento de los dispositivos de E/S
- Consideraremos solo el control por interrupción y supervisión de estado (no DMA ni canal programable)
- Si una interrupción activa un proceso esporádico puede producir una inversión de prioridad
- Las interrupciones hardware suelen tener mayor prioridad que los procesos. Se modelan con un proceso extra de mayor prioridad
- En el control por supervisión de estado se utiliza un proceso normal. El problema es como espera el proceso a que el periférico cambie de estado
  - Espera ocupada. Aceptable en retardos pequeños
  - Suspender el proceso. Tiene impactos negativos en la planificación
  - Aplazar la acción hasta el siguiente periodo. Posible si $S \leq P - D$

Sincronización por interrupción

- La forma de tratar las interrupciones puede variar mucho entre distintos sistemas
- Aunque pueden haber varias entradas de interrupción, lo normal es que sean menos que los periféricos
- Se necesitan mecanismos para compartir interrupciones:
  - Polling. Tras la interrupción se busca el periférico que la ha producido
  - Interrupciones vectorizadas. El periférico se identifica en el bus de datos y la CPU selecciona el manejador
  - Controladores de interrupción. Disponen de varias entradas de interrupción y producen interrupciones vectorizadas en la CPU

Polling

- Es la solución más sencilla para que varios periféricos compartan una línea de interrupción
- La CPU debe ver la función OR de las interrupciones de todos los periféricos
- Todos los periféricos activan el mismo manejador, y este debe verificar el estado de todos los periféricos
- El periférico debe indicar en alguno de sus registros que está solicitando interrupción
- Una vez encontrado el periférico, el manejador la debe tratar para que retire la interrupción
- Si hay otra interrupción se volverá a activar el manejador, aunque se podrían tratar varios periféricos en la misma activación
- El orden en que se revisan los periféricos es importante
- Se usa en un solo periférico si produce varios tipos de interrupción

Interrupciones vectorizadas

- Es un mecanismo hardware para evitar buscar el periférico
- Se basa en la capacidad de activar distintos manejadores desde una solo línea de interrupción
- El procesador, al recibir la petición de interrupción, activa otra línea de reconocimiento
- El periférico que pide la interrupción, al ver esta línea pone en el bus de datos un número de manejador
- La CPU activa el manejador asociado a ese número
- El mecanismo de reconocimiento debe prever que varios periféricos pueden solicitar interrupción a la vez
- Son mas complejos de utilizar. En cada periférico hay que programar el vector asociado a su manejador
- Un mismo periférico podría manejar varios vectores

Controladores de interrupción

- Su función es aumentar el número de líneas de interrupción y proporcionar un mecanismo para distinguir el periférico solicitante, normalmente con interrupciones vectorizadas
- Tienes varias entradas de interrupción, y se conecta a la CPU en una sola línea de interrupción y, como el resto de periféricos, al bus de datos y direcciones
- Tiene capacidad para enmascarar individualmente cada interrupción y establecer prioridades entre interrupciones
- Tras procesar cada interrupción, la CPU debe indicarlo al controlador para poder procesar las interrupciones de menor nivel
- La arquitectura PC utiliza dos 8259A, accesibles en los puertos 0x20 (master) y 0xA0 (slave)

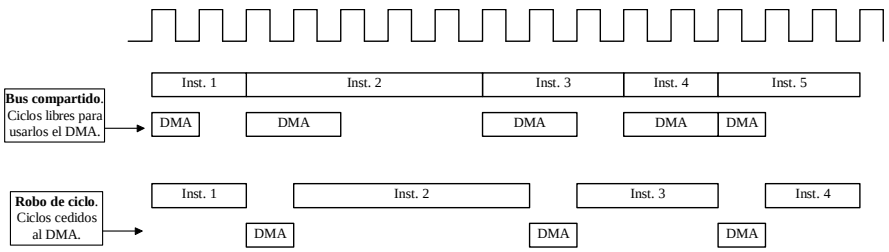
Transferencia de datos con DMA

- Se distingue por la transferencia rápida de bloques entre la memoria y los periféricos
- Se usan posiciones de memoria consecutivas. El dispositivo puede acceder de forma autónoma en los ciclos de reloj que no la usa la CPU
- La CPU y el DMA deben coordinar el uso de la memoria
  - La CPU indica la posición inicial y final de la memoria a utilizar
  - Se establece un protocolo para indicar el inicio y fin de la transferencia
  - Se debe establecer un protocolo entre el DMA y el periférico para armonizar sus velocidades

Sincronización en el uso del BUS

- Durante las transferencias por DMA, el bus de direcciones, de datos y la linea R/W debe estar aislada del procesador para que el DMA pueda usar la memoria
- Métodos para compartir el bus
  - Parada del procesador. La CPU se para mientras el DMA usa la memoria. Es el acceso más rápido
  - Robo de ciclo. Cada vez que finaliza una instrucción cede los buses al DMA durante uno o varios ciclos
  - Bus compartido. El DMA usa la memoria en los ciclos que la CPU no usa el bus
  - Por demanda. El DMA solicita el bus cuando tiene datos para transferir
  - Data a dato. Cuando el DMA solicita el bus solo transfiere un dato

Bus compartido y robo de ciclo



Retardos introducidos por el DMA

- El retardo depende del método utilizado para compartir el BUS
- En los casos de parada del bus o robo de ciclo se relentiza la CPU
- Hay que determinar el número de ciclos que precisa el DMA, pero este no depende solo del tamaño del bloque pues el DMA se debe sincronizar con el periférico
- En los otros tipos de sincronización puede ocurrir que, una vez el DMA toma el bus, use más ciclos de los que deja libres la CPU
- En general el retardo dificulta el test de planificabilidad, de ahí que sea poco utilizado en los STR estricto

Servicios de tiempo

- La gestión de tiempo debe ofrecer los siguientes servicios:
  - Actualizar un calendario
  - Activación de tareas en un cierto instante
  - Retardar la ejecución de una tarea
  - Gestionar los time-outs de las situaciones de espera
  - Enviar mensajes o señales a una hora determinada
  - Activación de funciones de forma asíncrona
  - Obtener medidas de tiempos
- Para realizar estas funciones se precisa un apoyo hardware. Los dispositivos usados se llaman timers
- La forma de usarlos es:
  - Se programan para que generen interrupciones periódicas con periodo igual a la mínima resolución temporal que debe tener el sistema
  - En cada interrupción el manejador incremente la hora actual y actualiza un calendario. También maneja una lista de acciones a realizar ordenadas por tiempo
  - En un instante se puede conocer a la fracción de tick accediendo a los registros del periférico, lo que puede dar mayor resolución en las medidas de tiempo

Timers

- Son circuitos programables que pueden realizar las siguientes funciones:
  - Generar un pulso de una determinada duración
  - Generar una onda cuadrada
  - Provocar una interrupción al cabo de un tiempo
  - Provocar interrupciones periódicas
- Los timers funcionan con un contador interno que se incrementa / decrementa por pulsos de un reloj a frecuencia constante
- Al pasar por el valor cero genera el pulso o la interrupción
- Variando la cuenta inicial se consiguen diferentes tiempos
- La hora se obtiene contando los ticks desde una hora conocida
- Se puede obtener mas precisión en la hora leyendo el valor del contador (fracción de tick)
- La arquitectura PC usa el timer 8254-2 con un reloj de 1.193.180 Hz accesible en el puerto 0x40

Circuitos de Watch-dog

- Son similares a los timers, pero la acción que realizan es provocar un reset del sistema
- Se utilizan como elementos de seguridad contra errores software en sistemas empotrados
- En funcionamiento normal, alguna tarea debe estar reprogramando su instante de actuación (reset del watch-dog)
- El software se debe diseñar para que cualquier fallo no recuperable acabe deteniendo la reprogramación del watch-dog (no es trivial en un sistema concurrente)
- Una posibilidad es que el reset del watch-dog lo haga una tarea (monitor) que vigila al resto de tareas
- El los STR estricto no siempre es válido reiniciar el sistema pues se podrían perder plazo. En estos casos se provoca un cambio de modo o la activación de un sistema auxiliar

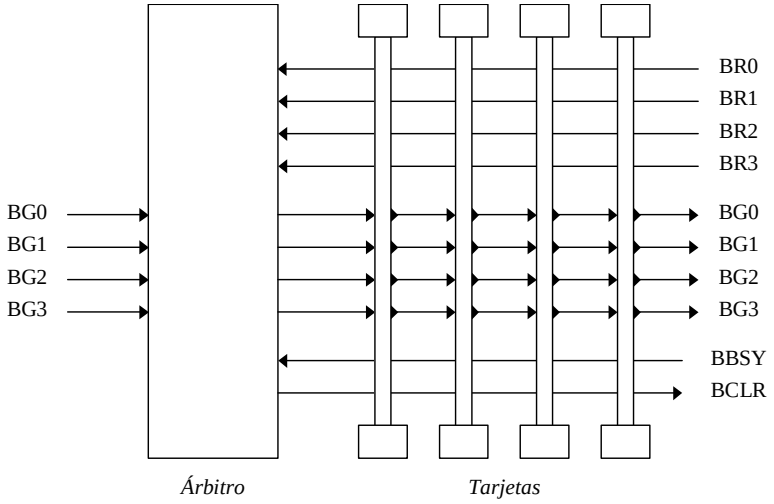
Buses

- Permite conectar diferentes elementos de forma que puedan transferirse información (procesadores, memorias, dispositivos E/S)
- Se usan en sistemas complejos con un diseño modular
- Al ser un elemento compartido, su uso tendrá efectos en el comportamiento temporal
- Los tiempos de utilización deben estar acotados. Los factores a tener en cuenta son:
  - Disponer de una información temporal precisa
  - La respuesta debe ser rápida y predecible para los eventos urgentes
  - Debe tener un grado alto de planificabilidad
  - Debe ser estable bajo sobrecargas transitorias

Bus VME

- Es un bus estándar muy difundido en los STR diseñado para el microprocesador MC68000 de 16 bits
- Existen una amplía gama de componentes para este bus
- El arbitraje es la técnica usada para poner orden en la utilización del bus. Consiste en:
  - El árbitros del bus. Recibe las peticiones y gestiona su concesión
  - Varios solicitadores. Realizan las solicitudes y el reconocimiento de su concesión
- Se prevén varios niveles de prioridad en la solicitud del bus y el árbitro se encargar de atender siempre la más prioritaria

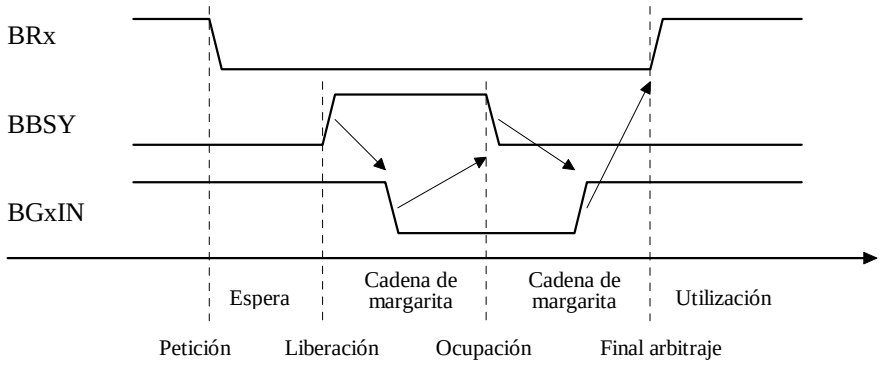
Líneas de Control en el bus VME



BUS VME. Arbitro

- Para solicitar el bus existen 4 líneas BR0-BR3 (activas bajo) asociadas a 4 niveles de prioridad (BR3 la mayor)
- El árbitro lee estas líneas junto con la señal BBSY (activo bajo) que indica que se está usando el bus
- Cuando el bus está libre (BBSY alto) activa una de las señales BG0IN-BG3IN (activas bajo) para conceder el uso a la petición más prioritaria
- Cada módulo lee las líneas de concesión del bus y las procesa en una cadena de margarita (daisy chain). Si es el módulo que realizó la solicitud activará la línea BBSY, si no pasará la concesión al siguiente módulo activando la línea BG0OUT-BG3OUT correspondiente
- Cuando el árbitro detecta la señal BBSY retira la concesión y cuando esto le llegue al solicitador, éste retirará su solicitud y podrá usar el bus. A partir de entonces el árbitro queda listo para una nueva solicitud
- Cuando el módulo termina de usar el bus libera la señal BBSY y el árbitro podrá procesar otras solicitudes
- Con la señal BCLR el árbitro puede indicar a un módulo que adelante la liberación del BUS si aparece una solicitud más prioritaria de la que está en curso

Arbitraje del bus VME



Redes. Capas inferiores

- Al igual que los buses, las redes usadas para interconectar STR deben cumplir ciertas propiedades que garanticen los límites temporales
- Los factores que se debe tener en cuenta son:
  - El ancho de banda
  - Los niveles de prioridad
  - El tiempo máximo de latencia
  - El mecanismo para la recuperación de errores
- No todas las redes son adecuadas para los STR (ethernet)
- Hay que prestar especial atención a los protocolos utilizados
- Los protocolos se suelen estructurar en capas (estándar ISO)
- Normalmente el soporte de TR se da en los niveles inferiores

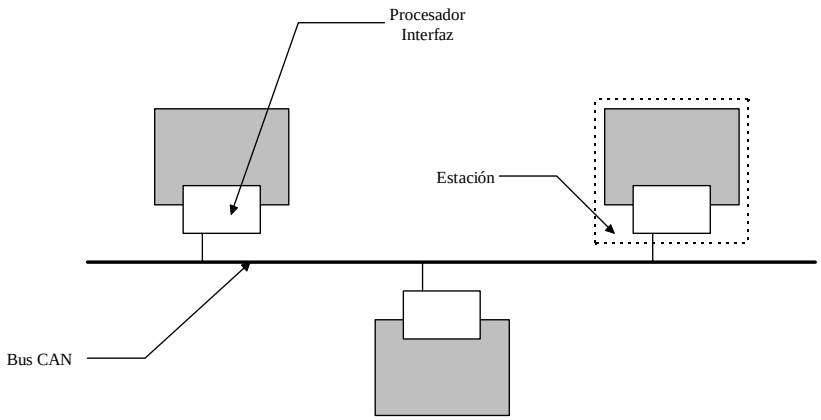
Modelo de capas OSI

Nivel	**Descripción**
Físico	Comprende los circuitos y el hardware que forman la red. Transmite secuencias de datos binarios por medio de señales analógicas, utilizando modulación en frecuencia o en amplitud de señales eléctricas (en circuitos de cable), señales luminosas (sobre fibra óptica) o señales electromagnéticas (en circuitos de radio o microondas).
Enlace	Es responsable de la transmisión libre de errores entre ordenadores que están conectados directamente. En una WAN las conexiones son entre pares PSEs (packets-switches exchanges). En una LAN la conexión es entre pares de hosts.
Red	Transfiere paketes de datos entre ordenadores en una red específica. En una WAN o red internet esto supone la generación de una ruta entre PSEs o ruters. En una LAN sencilla no es necesario el enrutamiento.
Transporte	Este es el nivel más bajo en el que se manejan mensajes (en vez de paquetes). Los mensajes se direccionan a puertos de comunicaciones. Los protocolos en este nivel pueden ser orientados a conexión o sin conexión.
Sesión	En este nivel se establece la comunicación entre procesos y se realiza la recuperación de errores. No es necesario para la comunicación sin conexión.
Presentación	A este nivel el protocolo transmite datos en la representación de red que es independiente de la representación utilizada en los ordenadores individuales, las cuales pueden ser distintas. La encriptación se realiza en este nivel en caso de ser necesaria.
Aplicación	Los protocolos son diseñados para responder a los requerimientos de un aplicación específica, a menudo definiendo el interfaz para un servicio.

Bus CAN

- Está diseñado para enviar mensajes de control cortos en tiempo real, en un área pequeña a velocidades de hasta 1 Mbit
- Es un bus de uno a todos (broadcast)
- Una fuente de datos es un mensaje de entre 1 y 8 bytes. Puede ser transmitida de forma periódica, esporádica o bajo demanda
- Comparada con el modelo de referencia ISO, la estructura de CAN es de tres niveles
- Según las prestaciones del controlador, podemos distinguir dos tipos:
  - FullCAN. El controlador almacena todos los mensajes. La CPU le indica los mensajes que desea leer y éste le avisa cuando recibe alguno de ellos. Se precisa poco trabajo de la CPU
  - BasicCan. Los mensajes se almacenan en la memoria de la CPU, y la CPU debe realizar todo el trabajo para obtener los mensajes deseados

Arquitectura CAN



Niveles OSI en bus CAN

Nivel	**Descripción**
Físico	En la mayoría de los casos, el nivel físico o medio de transmisión es un bus diferencia a 2 hilos, utilizado como par trenzado.
Enlace	Viene fijado por la especificación del protocolo CAN y está implementado en gran cantidad de circuitos integrados de distintos fabricantes.
Red Transporte Sesión Presentación	Niveles vacíos
Aplicación	Se implementa según las necesidades del usuario.

Bus CAN

- La versión 1.0 a la fuente de datos se le asigna un identificador de 11 bits (2032 valores, pues CAN prohíbe los 7 bits de mas peso a 1)
- La versión 2.0 usa 29 bits para los identificadores
- El identificador filtra los mensajes en la recepción y les asigna una prioridad
- Cuando hay colisión con dos mensajes, en el bus prevalece el identificador con mayor prioridad
- El bus CAN funciona como una puerta AND y cada estación puede también leer el bus para detectar colisiones. El bit ‘0’ es dominante y el bit ’1’ recesivo
- Cada estación espera a que el bus este libre para transmitir, y se transmite el mensaje más prioritario de la cola de salida (primero el bit más significativo)
- Si una estación transmite un bit recesivo pero recibe uno dominante, detecta una colisión y deja de transmitir, esperando a que el bus quede libre de nuevo
- Los identificadores deben ser únicos, siendo el 0 el mas prioritario
- Se puede calcular fácilmente el tiempo más largo desde que el mensaje de mayor prioridad es encolado hasta que es recibido (130 mseg.)
- Para los mensajes menos prioritarios también se puede obtener un valor máximo teniendo en cuenta el resto de mensajes más prioritarios que se pueden transmitir

Arbitraje en el Bus CAN