2012

Instrumentación Electrónica Avanzada

Instrumentación Inteligente

Este libro explica las aplicaciones de la incorporación de tratamiento digital en sistemas de instrumentación electrónica.

José Bosch, Manuel Carmona
Departament d'Electrònica, Universitat de Barcelona
13/07/2012

Autors:
Manuel Carmona.
José Bosch
José María Gómez
Manel López

Índice

1. SISTEMAS DIGITALES DE ADQUSICIÓN DE DATOS

1.1 INTRODUCCIÓN

La tecnología actual de semiconductores permiten, por un lado, tener mayor precisión en las medidas y, por otro lado, incluir algún tipo de "inteligencia" a los transductores. Ejemplos de estos segundos son los microcontroladores (MCUs o μCs), 'Digital Signal Processors' (DSPs), ASICs y FPGAs, aunque los primeros están más destinados a tareas de instrumentación y control. Algunas tecnologías son capaces de integrar todo (sensor + inteligencia) en un único encapsulado (integración monolítica), generando así los llamados sensores inteligentes (o 'smart sensors').

Todo ello permite el desarrollo de Sistemas de Adquisición de Datos (DAS) con muy buenas prestaciones y de forma rápida. Se usan ampliamente en tareas como la medición de parámetros y el control de sistemas.

Hoy en día, el uso de sistemas digitales para analizar/procesar las señales está muy extendido. Esto es debido a las múltiples ventajas que presentan: bajo coste, mayor precisión, facilidad de implementación, etc. El uso de microcontroladores (μC) o microprocesadores (μP) es común en muchas industrias para el control de procesos. Ejemplos son la industria alimenticia, del papel, textil, químicas, etc.

1.2 BENEFICIOS DE LA UTILIZACIÓN DE SISTEMAS DIGITALES EN LA INSTRUMENTACIÓN ELECTRÓNICA

1.2.1 Inmunidad

Una de las ventajas es la insensibilidad al ruido proporcionada, en general, por los sistemas digitales.

Por la misma razón anterior, otra ventaja consiste en la insensibilidad a las variaciones o pérdidas de amplitud (una vez convertidas las señales a digitales).

1.2.2 Aislamiento galvánico

Con técnicas analógicas, este tipo de aislamiento se realiza mediante transformadores. Con técnicas digitales, podemos utilizar optoacopladores. Estos componentes presentan diversas ventajas. Una de ellas es que son más rápidos que los transformadores. Además, los circuitos acoplados pueden tener referencias de tensión diferentes.

1.2.3 Organización y manejo de secuencias de medida

La capacidad de sincronización y control de operaciones de un sistema digital permite poder programar de forma muy versátil cualquier secuencia de medida.

1.2.4 Almacenamiento de información

Los distintos tipos de memoria existentes en sistemas digitales, permite el almacenamiento de información diversa en diferentes tipos de continentes: constantes, tablas, programas, etc.

Un ejemplo práctico consiste en la linealización de un transductor no lineal, cuya característica podría almacenarse en memoria para su posterior uso tras las medidas. En el caso de un termopar, analógicamente supondría incluir, entre otros componentes, varios amplificadores operacionales.

1.2.5 Procesado de las señales medidas

Una de las mayores virtudes de los sistemas digitales es su capacidad de procesado de la información. Como ejemplos de procesado de las señales, se puede mencionar: combinar las medidas de varios parámetros diferentes, extraer medidas de un conjunto, calcular correlaciones entre señales.

1.2.6 Calibración automática

Una utilidad bastante extendida en sistemas digitales consiste en la realización de autocalibración de sensores de forma relativamente sencilla.

1.2.7 Presentaciones inteligentes

La capacidad en cuanto a la presentación de los resultados obtenidos por el sistema digital es muy importante, pudiéndose adaptar a diferentes tipos de pantallas y organizar la presentación de datos de forma muy versátil.

1.2.8 Flexibilidad en el diseño y manipulación del sistema de medida

Los microcontroladores combinan una unidad de procesado, varios tipos y tamaños de memoria, un reloj oscilador, distintas opciones de I/O, y todo en un mismo chip. Esto proporciona flexibilidad para la confección de numerosos sistemas embebidos, así como

DAS. Esto hace disminuir el coste de otros sistemas que requerirían otro tipo de componentes para construir la misma solución.

1.2.9 Facilidad en la transmisión de señales y en el control remoto

Los microcontroladores suelen incorporar módulos para el control de las comunicaciones con dispositivos externos. Esto permite una mayor facilidad en el tratamiento de las comunicaciones, sin necesidad de usar otros elementos externos. Típicamente, suelen incluir interfaces para comunicaciones SPI e I^2C.

1.3 ELEMENTOS DIGITALES EN LOS SISTEMAS DE ADQUISICIÓN DE DATOS

1.3.1 Introducción

Un DAS (Sistema de adquisición de datos) es un sistema que permite la medida y/o control de una serie de magnitudes físicas ("mundo real") mediante dispositivos digitales. Constituyen la interfaz entre el mundo real (analógico) y el artificial (digital).

Un sistema DAS incluye, habitualmente, transductores (sensores y actuadores) para la medida (o aplicación) de variables físicas (generalmente no electrónicas, como por ejemplo temperatura, velocidad de un fluído, etc), acondicionadores de la señal y un entorno digital. Esta fase de interfície constituye una parte importante de los sistemas de medida.

Nosotros consideraremos aquellos sistemas DAS en los que hay, al menos, un μC (microcontrolador).

1.3.2 Elementos habituales en un DAS

Los elementos de los que suele disponer un DAS son:

- Entradas y salidas digitales.
- Entradas y salidas analógicas.
- Temporizadores.
- Contadores: Usados en conversores A/D y en múltiples aplicaciones.
- Lógica de control (μC, etc.).
- Elementos de transmisión de la información: UARTs.

- Lógica de tratamiento de las magnitudes medidas: Se refiere al software que controlará el sistema.

1.3.3 Parámetros para la definición de la estructura de un DAS

Para definir la estructura del DAS tendremos que tener en cuenta parámetros como los siguientes:

- Número de magnitudes físicas a medir y/o controlar: Determinará el número de entradas y salidas, analógicas y/o digitales.
- Velocidad de cambio de estas magnitudes: Determinará un mínimo de la velocidad de operación de los componentes.
- Precisión de las medidas: Puede definir el número de bits necesarios, así como los sistemas de acondicionamiento, etc.
- Finalidad del sistema: Puede definir el tipo de componentes necesarios.

1.3.4 Sistema DAS de un sólo canal

Es el caso de DAS más sencillo.

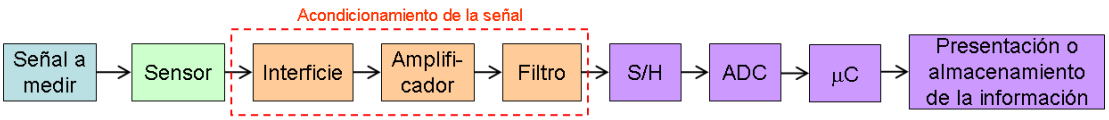


Figura 1.1. Estructura de un DAS de un único canal.

Existen dos posibilidades:

- En bucle abierto (Figura 1.1): La finalidad del sistema consiste en medir una magnitud para presentarla, de alguna forma, al usuario, o bien para procesarla posteriormente.
  - Señal a medir: Magnitud física a medir (P, T, posición, tensión).
  - Sensor: Dispositivo que transforma la señal a medir en señal eléctrica (si la señal a media no es ya eléctrica). Sensores de presión, termómetros, etc.
  - Elementos de acondicionamiento de la señal: Esta etapa es analógica, que puede estar compuesta de elementos como potenciómetros, puentes de Wheatstone, adaptadores de impedancia, amplificadores de instrumentación, etc.
    - Interfície: Adecúa la señal en sí a lo que "podemos" adquirir. Ej: una señal en corriente podemos desear pasarla a tensión. Ej: la

salida del sensor puede ser de alta impedancia y tenemos que capturarla sin afectar al sensor.

- Amplificador: Adecúa los niveles de la señal a valores más "manejables".
- Filtro: Para eliminar rangos de frecuencias, ruido, etc. de la señal.

- S/H: Para poder "leer" el valor de la señal en un instante de tiempo. El S/H mantiene a la salida el valor tomado en la entrada en momento específicos, controlados por un reloj proveniente de la lógica de control (μC).
- ADC: Paso para obtener la señal digital.
- μC (o μP): Permite tratar la señal obtenida para los fines deseados. La lógica de control se ocupa de la secuenciación y de controlar el modo de trabajo.
- Presentación/almacenamiento de la información: Definición del destino final de la información obtenida.

- En bucle cerrado (Figura 1.2): La magnitud medida (y su posible procesamiento) se utiliza en un proceso de control.

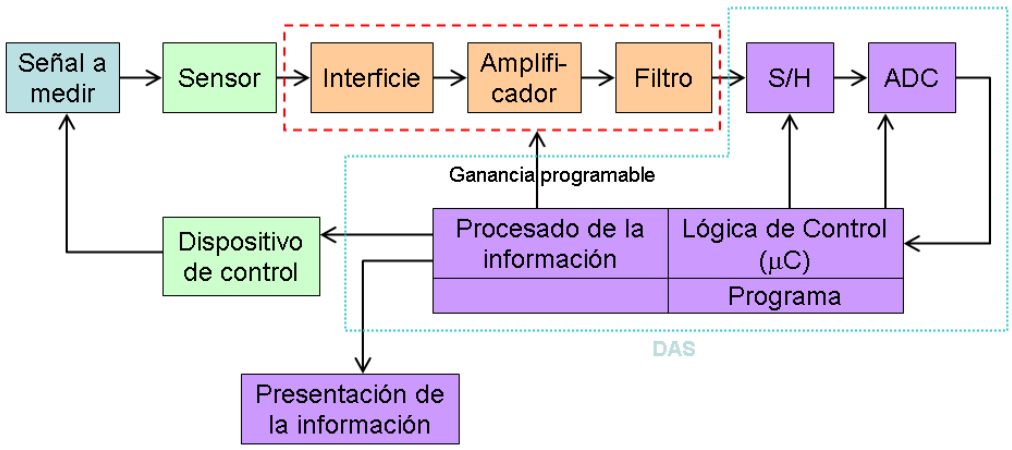


Figura 1.2. Estructura de un DAS de un solo canal en bucle cerrado.

La información extraída tras el procesado de la información puede utilizarse con dos fines: influenciar sobre el sistema a medir y/o modificar la etapa de acondicionamiento de la señal (por ejemplo, variando la amplificación).

Entre la etapa de procesado y el dispositivo de control, típicamente hay un registro, un DAC y otra posible etapa de acondicionamiento de la señal. El registro es una especie de equivalente del S/H en la etapa de adquisición, pero en digital ya que mantiene el valor deseado en la entrada del DAC hasta la siguiente actualización del valor.

Ejemplo: Control de la temperatura de un horno (Figura 1.3).

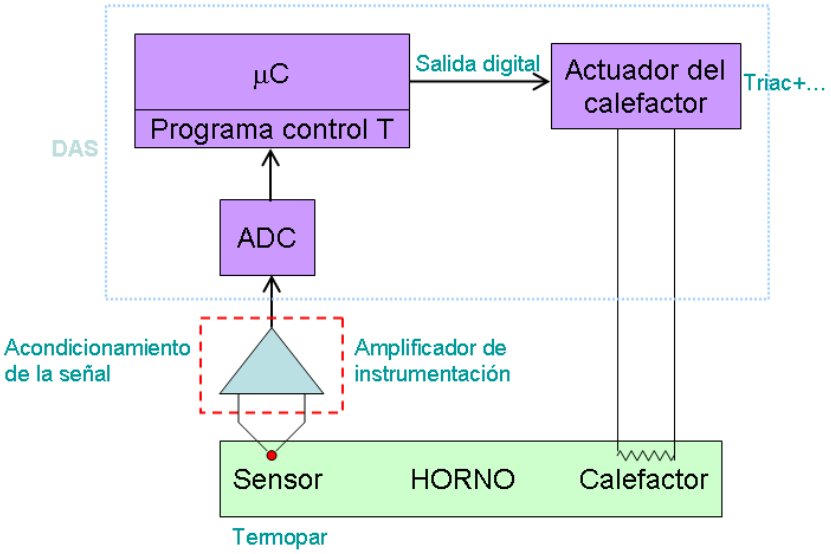


Figura 1.3. Esquema del control de temperatura de un horno.

Existen múltiples alternativas para el sistema de control. Dependerá del software del sistema. Esto da mucha flexibilidad a estos sistemas para implementar diferentes métodos de control. Ejemplos son: ON/OFF, PID, etc. El sistema PID tiene en cuenta la dinámica del sistema y puede ajustarse sus parámetros de forma que el control tenga las características deseadas (por ejemplo en rapidez).

1.3.5 Sistemas DAS multicanal

En muchas aplicaciones se han de medir varias magnitudes simultáneamente, así como controlar varios actuadores. Nosotros veremos casos en los que el control lo realiza un µP o un µC.

Veamos por separado los casos con entradas digitales y analógicas.

1.3.5.1 Entradas digitales de un DAS multicanal

El componente más sencillo que se puede considerar es el Buffer triestado (Figura 1.4).

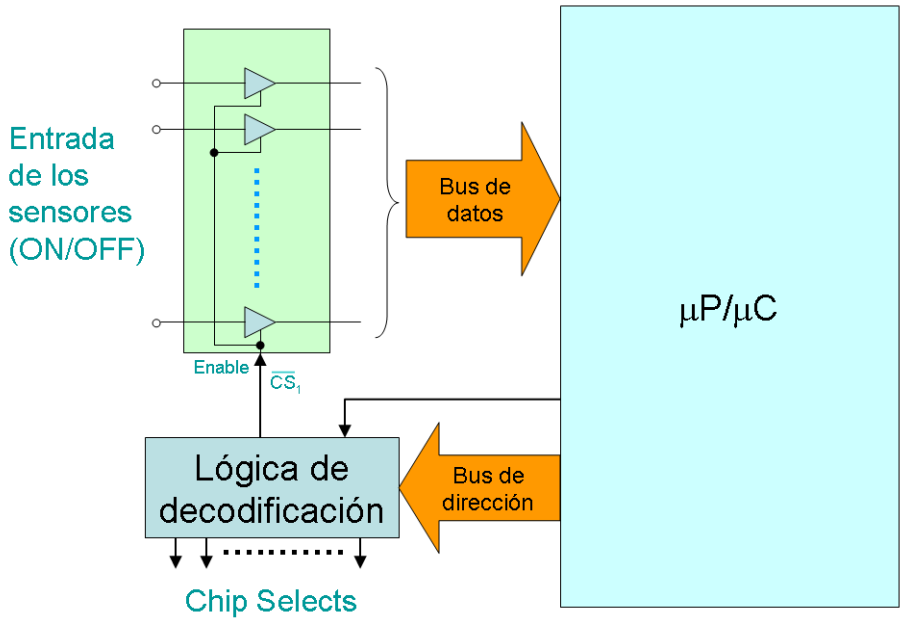


Figura 1.4. DAS multicanal con buffers triestado.

El buffer triestado permite conectar las diferentes salidas a una misma línea (bus de datos) sin que se interfieran entre ellas. En cada momento sólo habrá un buffer activado, y el resto en estado de alta impedancia.

Una modificación a esta configuración sería substituir el Buffer por un Registro (también con salida triestado). Esto permitiría mantener los datos, aunque sólo permanezcan en la entrada durante un tiempo limitado. (Esto es como un S/H digital).

Un punto que nos puede interesar en multitud de ocasiones es el de realizar un aislamiento galvánico entre las señales de entrada y el DAS. Esto lo conseguiremos mediante optoacopladores (Figura 1.5).

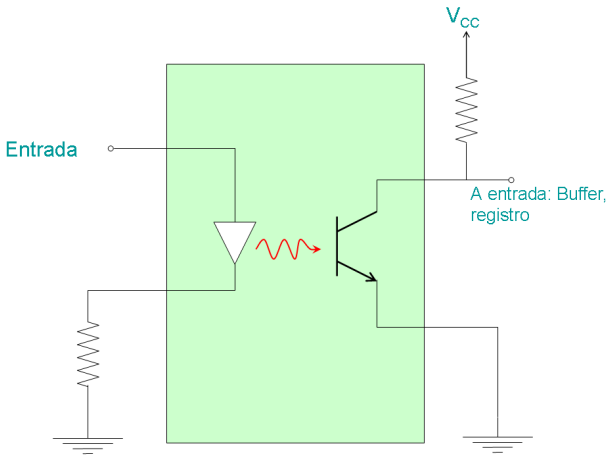


Figura 1.5. Uso de optoacoplador para aislar las entradas del DAS.

*******(Ventajas de los optoacopladores frente a los transformadores).

1.3.5.2 Salidas digitales de un DAS multicanal

En general, la etapa de salida digital consiste también en buffers o en registros.

Se pueden utilizar también elementos de salida de potencia, como tiristores o triacs, y etapas de aislamiento entre el DAS y las salidas.

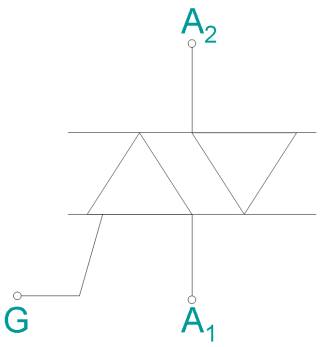


Figura 1.6. Triac para etapas de salida de potencia.

1.3.5.3 Entradas analógicas de un DAS multicanal

Para realizar este tipo de medidas, utilizaremos convertidores Analógico-Digitales (ADCs).

Cuando tenemos varios canales a medir, las posibles soluciones son:

- Multiplexar en el tiempo (mediante multiplexores analógicos):

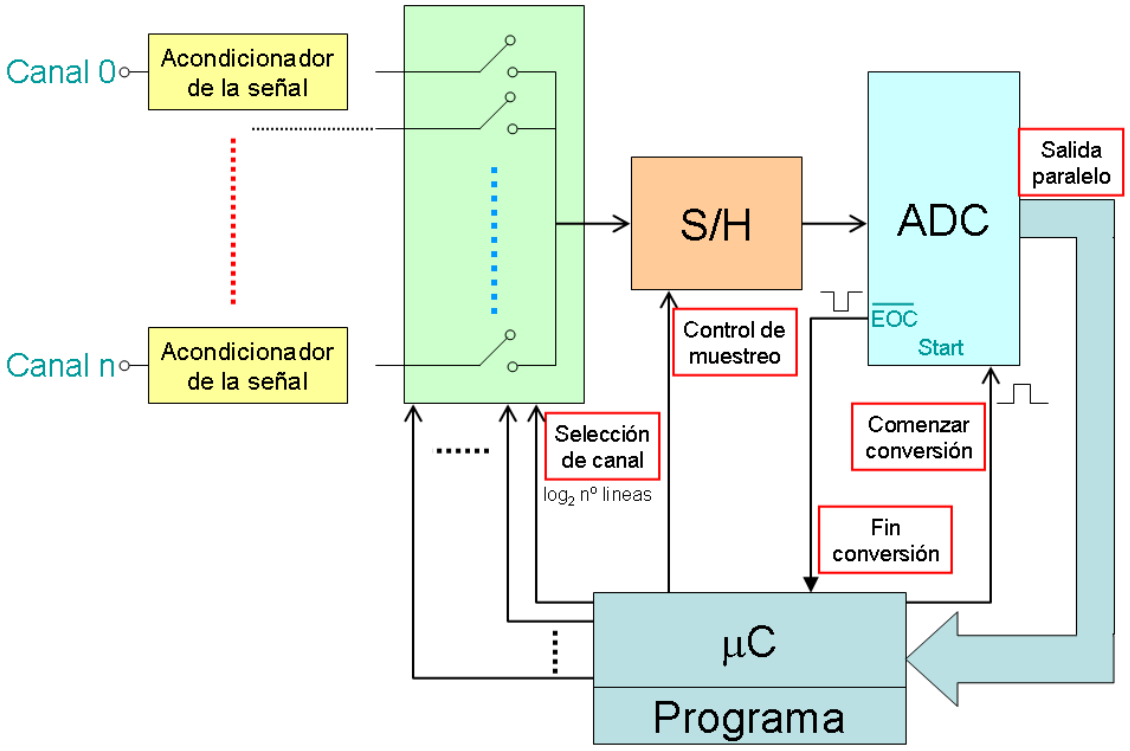


Figura 1.7. Multiplexación en el tiempo de las entradas analógicas.

Para organizar y controlar las operaciones de los componentes del sistema, el µP/µC ha de generar y/o muestrear las siguientes señales:

- Generar las señales que seleccionen el canal a medir.
- Generar la señal de comienzo de conversión del ADC.
- Muestrear (o por interrupciones) la señal de la final de conversión (EOC) que genera el convertidor ADC para indicar que la medida está disponible.
- Leer el valor de la medida entregado por el ADC.

Aquí, se cumple:

$$f_m = \frac{f_{ADC}}{N},$$

donde *N* es el número de canales.

El tipo de ADC escogido (como doble rampa (integrador), rampa (seguidor), aproximaciones sucesivas, flash) dependerá de los requerimientos del sistema y el coste del mismo.

- El esquema anterior tiene varios inconvenientes. Primero, la medida de cada canal corresponde a tiempos diferentes. Segundo, la velocidad de adquisición está muy reducida puesto que se realiza la operación de S/H cada vez después de multiplexar cada canal. Para mejorar esta situación podemos utilizar el siguiente esquema:

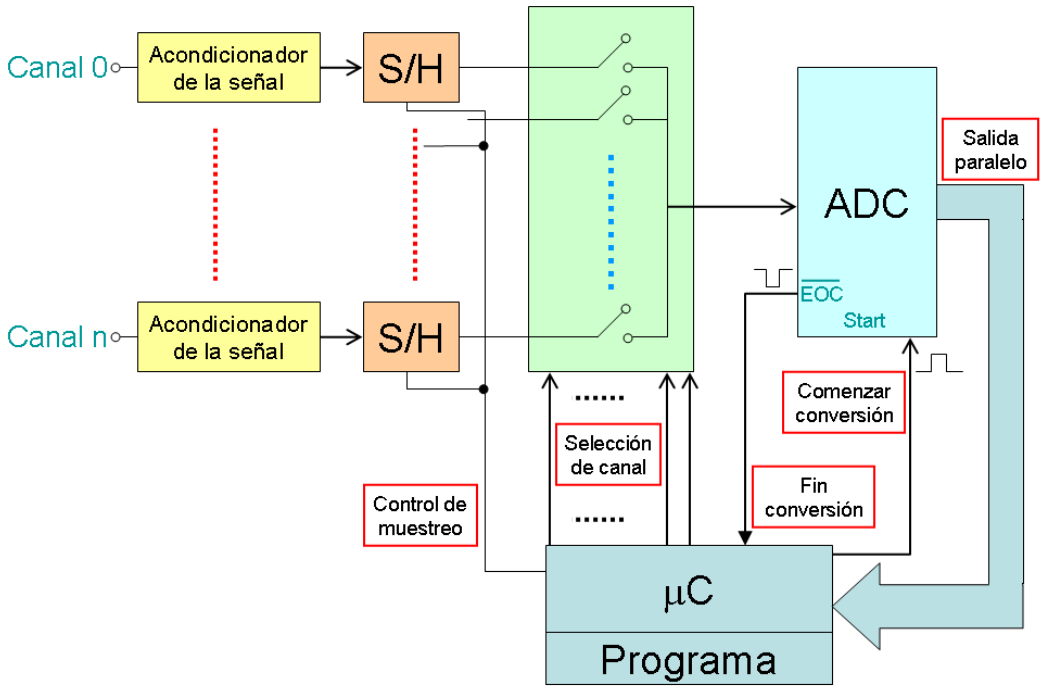


Figura 1.8. Multiplexación usando un S/H por cada canal analógico de entrada.

Aquí no hay que esperar a que se realice el muestreo para cada medida, después de escoger el canal a medir mediante el multiplexor analógico.

El mayor inconveniente de esta arquitectura consiste en un aumento del coste del equipo.

➡ Si las señales a medir varían muy rápidamente, los dos sistemas anteriores no serán los más convenientes. En estos casos utilizaremos, además, un ADC para cada canal. Y si queremos mucha velocidad, estos serán de tipo Flash. (no de aproximaciones sucesivas o de rampa como los anteriores).

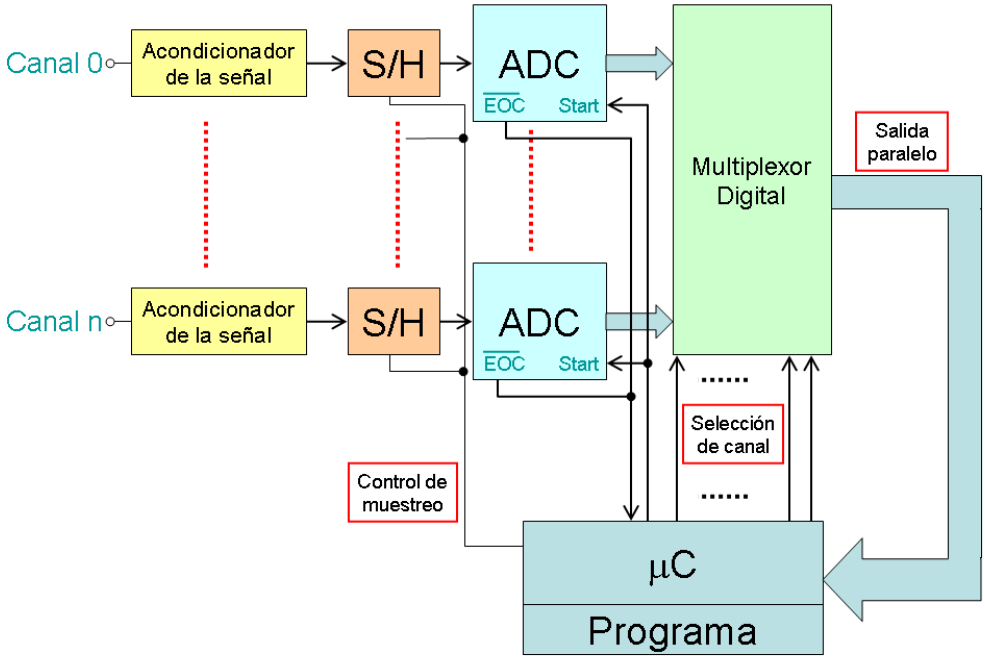


Figura 1.9. Multiplexación usando un S/H y un ADC por cada canal analógico de entrada.

Mediante este sistema aumentamos la velocidad de muestreo en un factor del orden de N.

Este sistema sigue teniendo el inconveniente de tener mayores costes, ya que hemos aumentado el número de ADCs.

➡ Una variante del sistema anterior puede conseguirse si los ADCs tienen salida triestado. De esta forma podemos conectar todas las salidas de los ADCs.

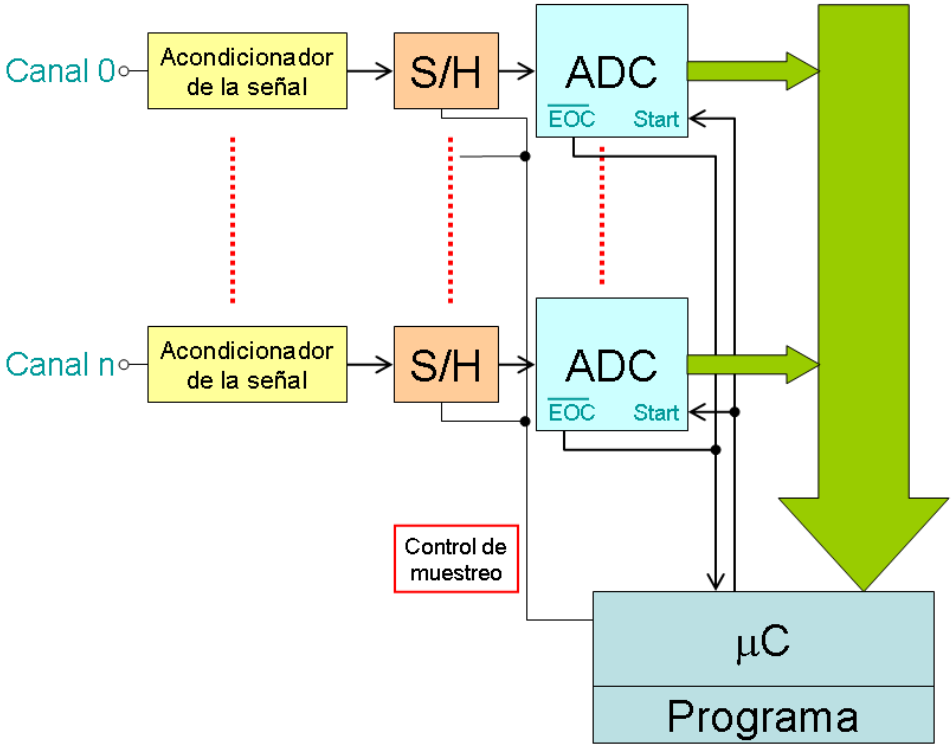


Figura 1.10. Variante para ADCs con salidas triestado.

Si tenemos una señal que varía muy rápidamente, podemos utilizar este sistema para muestrear la señal con varios canales, cada uno desfasado $2\pi/(n+1)$. Un ejemplo para un sistema de tres canales se muestra en la Figura 1.11.

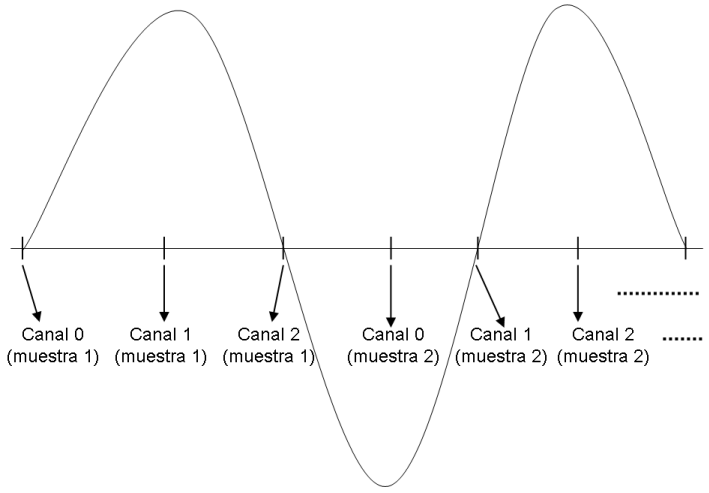


Figura 1.11. Adquisición rápida de una señal usando múltiples canales.

1.3.5.4 Salidas analógicas de un DAS multicanal

Las señales digitales necesitarán obligatoriamente de un DAC para obtener salidas analógicas. En el caso de tener una salida multiplexada, se ha de tener en cuenta que el valor no se mantendrá en la salida.

1.3.5.5 Entradas y salidas analógicas de un DAS multicanal

Algunos de los elementos que podemos tener tanto en entradas como en salidas analógicas son los siguientes:

a) Puertas de paso
b) Multiplexores analógicos.
c) Tener en cuenta la resistencia interna (varía con la tensión de alimentación).

Un ejemplo mostrado en la Figura 1.12, consistente en un amplificador de ganancia programable.

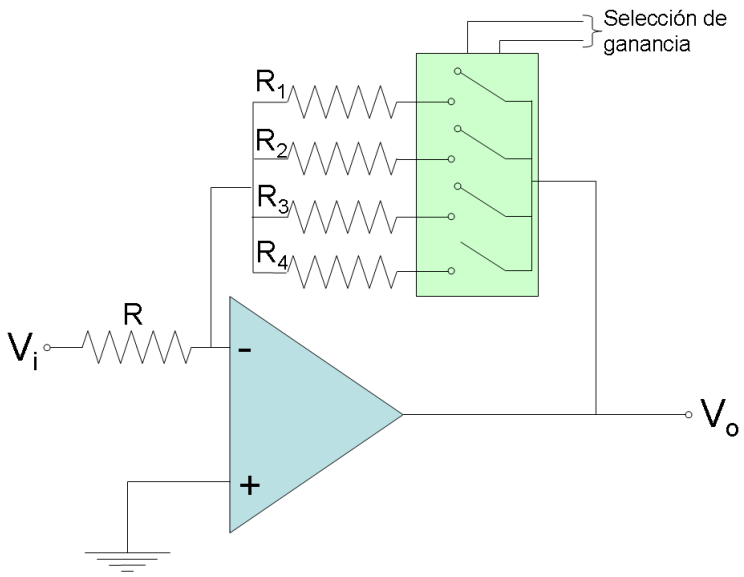


Figura 1.12. Amplificador de ganancia programable.

Estos elementos de E/S analógicos también sirven para E/S digitales.

1.3.6 Contadores como sensores

1.3.6.1 Medida de la frecuencia

Esquema:

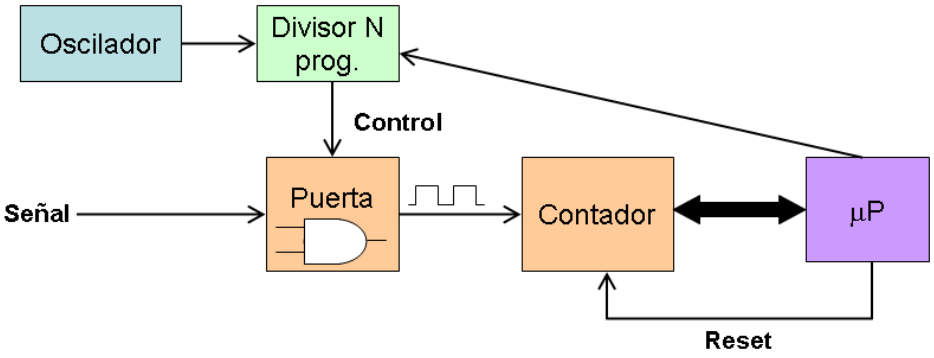


Figura 1.13. Contador como sensor de frecuencia.

El divisor se utiliza para poder medir distintos rangos de frecuencias.

El funcionamiento consiste en dejar pasar la señal durante uno de los semiperíodos del oscilador y contar el número de pulso de la señal durante ese tiempo. Al conocer la frecuencia del oscilador y la división realizada, podemos obtener la frecuencia de la señal con la siguiente relación:

$$f_s = (nº\ pulsos) \cdot 2 \cdot f_{oscilador}/N$$

Ejemplo de la secuencia de señales:

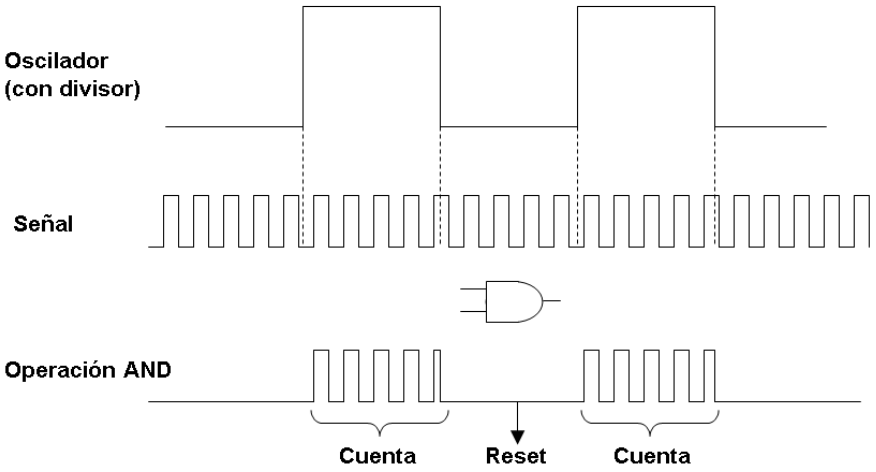


Figura 1.14. Secuencia de señales en la medida de la frecuencia con un contador.

1.3.6.2 Medida del período

Se realiza una operación similar, pero intercambiando el papel del oscilador y de la señal:

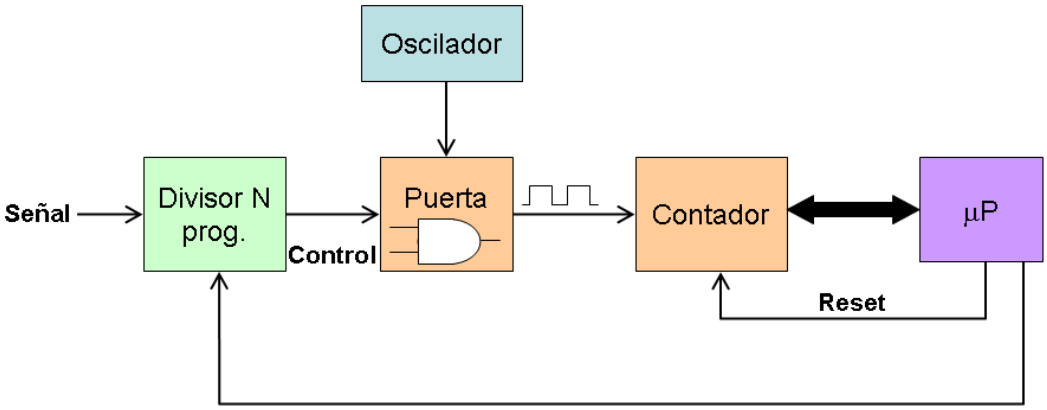


Figura 1.15. Contador como sensor de período.

Aquí contamos el número de pulsos del oscilador que pasan por cada semiperíodo de la señal. Por tanto, la medida del período vendrá dada por:

$$T_s =\cdot 2 \cdot T_{oscilador} \cdot (nº\ pulsos)/N$$

Un ejemplo de la secuencia de señales sería el siguiente:

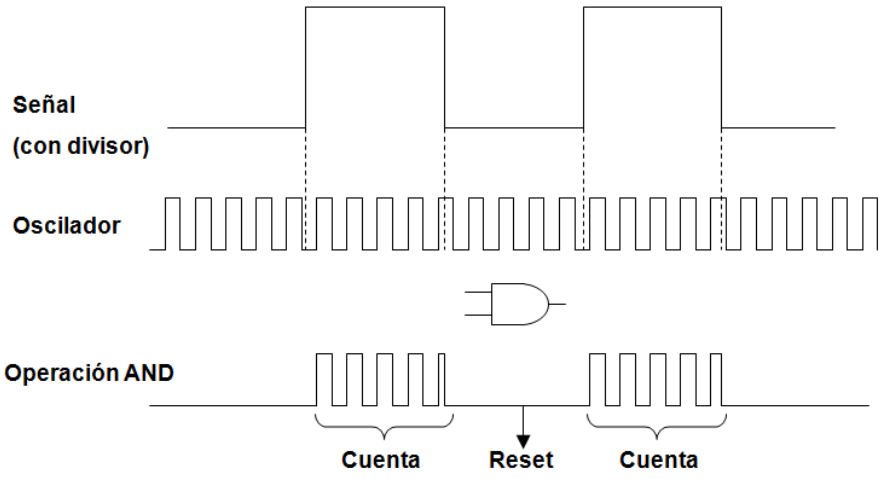


Figura 1.16. Secuencia de señales en la medida del período con un contador.

1.3.7 Contadores como actuadores

Los contadores también pueden tener una función de actuadores, como:

a) Bases de tiempos.
b) Actuadores propiamente (generadores de señales periódicas).
c) Generar frecuencias (exterior o interrupciones)
d) Retardar una señal.

1.4 DISEÑO DE UN DAS CON μP/μC

En primer lugar, el diseñador ha de delimitar todas las funciones y características del sistema a desarrollar, con el fin de seleccionar los módulos que mejor se ajusten a las necesidades de la aplicación.

1.4.1 Selección del μP/μC

Hay que tener en cuenta que el sistema incluye Hardware y Software, y la frontera entre las funciones de estos dos elementos no es rígida; es decir que hay funciones que se pueden implementar tanto por hardware como por software.

En general las funciones implementadas por hardware se realizan a mayor velocidad, mientras que las implementadas por software tienen una mayor adaptabilidad.

Los aspectos a tener en cuenta en la selección del μP/μC son:

- Memoria no-volátil (ROM, PROM, EPROM, EEPROM, FLASH).

  Función: Almacenar el programa a ejecutar por el DAS, así como tablas, ctes, etc.

- Memoria volátil.

  Función: Almacenar datos y, en ocasiones, programas.

- Puertos de E/S. (Número y programabilidad).

  Función: Entrada y salida de señales digitales.

- Convertidores A/D y D/A.

  Función: La propia...

  Factores a tener en cuenta:

  - Número de canales.
  - Resolución en bits.
  - Amplificación (fija, programabla, automática).
  - Frecuencia de muestreo (Nyquist: $f_e > 2 \cdot n \cdot f_{max}$).
  - Referencias internas.
  - Sensibilidad a la temperatura.

- Temporizadores / Contadores.

  Funciones:

- Generar señales periódicas.
- Generar interrupciones periódicas o no periódicas.
- Contar eventos externos.
- PWM.

- Circuitos de interfície.

  Función: Comunicarse con otros elementos del sistema o externos.

  Sobre todo son muy interesantes las interfícies serie: SPI, SCI, I^2C, CAN, etc.

  También los de presentación ('displays').

- Buses para el sistema.

  Función: Permiten conectar memorias, interfícies, etc.

- Controladores de interrupciones y/o DMA (Direct Memory Access).

  Función: Darán una "medida" de la capacidad de trabajo del sistema en tiempo real y de adquisición y almacenaje a gran velocidad.

  Permiten modos de trabajo en bajo consumo.

- El SET de instrucciones. (así como la existencia de compiladores de lenguajes de alto nivel).

  Posibilidad de manejo de bits independientes.

  Capacidad de cálculo.

  etc.

- La existencia de herramientas de desarrollo. (Emuladores, programadores, evaluadores, ensambladores, compiladores).

1.4.2 Ejemplos de microcontroladores

1.4.2.1 ADuC812

Sistema de adquisición de Analog Devices de 12 bits (reales) con ADC multicanal (8 canales de entrada), DAC dual (un único conversor que alterna con dos canales, usando S&H) y un microcontrolador programable de 8 bits (compatible con las instrucciones 8051).

Posee memorias FLASH y RAM.

El microcontrolador proporciona funciones como 'Watchdog Timer'. Tiene 32 líneas de I/O programables, interfaz serie I^2C y SPI, así como puertos UART.

P* son puertos de I/O, aunque.también se usan para algunas funciones secundarias.

Los prefijos D ó A se refieren a la parte digital o a la analógica respectivamente.

T? son las entradas para los contadores (aumenta el contador en las transiciones de 1 a 0 de estas entradas). (T2EX controla algunas funciones del contador 2).

INT* son entradas para interrupciones.

EA: External Access enable.

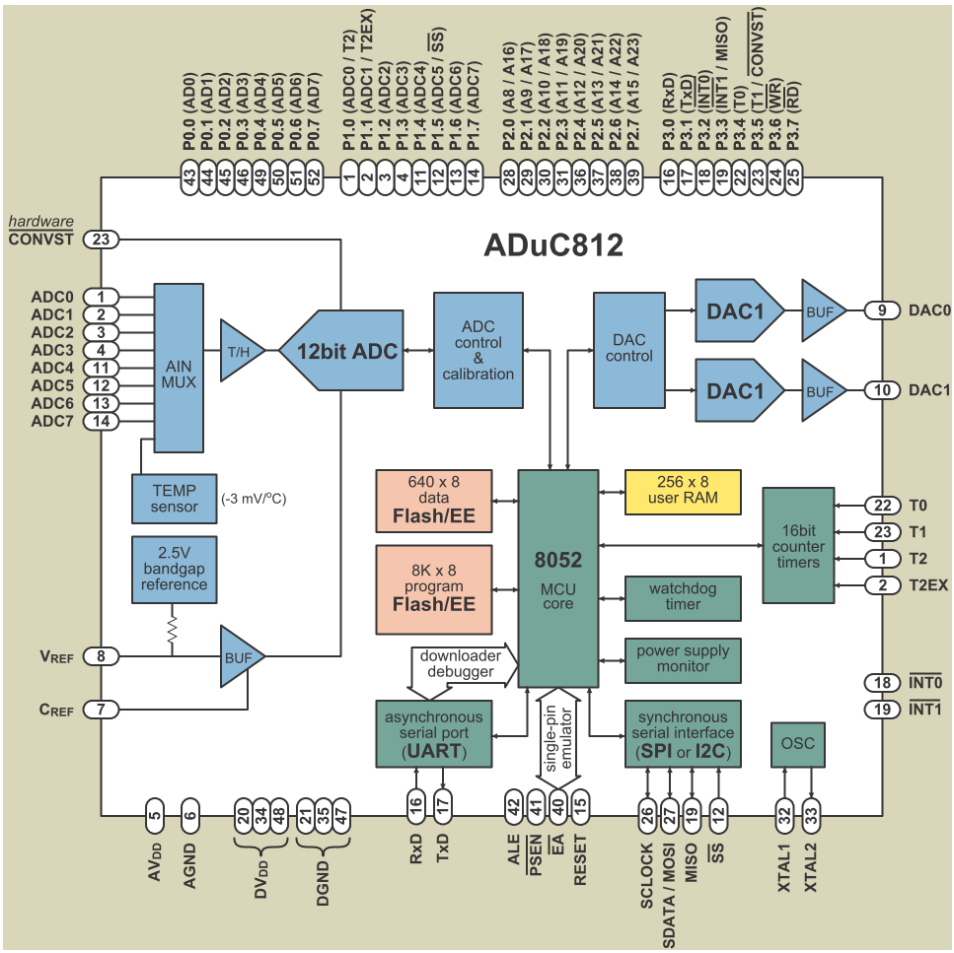


Figura 1.17. Esquema funcional del ADuC812. (Figura obtenida de documentación proviniente de Analog Devices, Inc. (www.analog.com)).

En cuanto a la memoria, dispone de direcciones separadas para el programa y los datos.

Programa: Si EA está a 0, el dispositivo ejecutará el programa localizado en un espacio de memoria externo. Si está a 1, ejecutará el existente en la memoria FLASH interna de 8 Kbytes.

Datos: Hay memoria interna y externa. La interna consta de 4 bloques distintos: partes alta y baja de RAM (cada una de 128 bytes), 128 bytes para los registros SFRs ('Special Function Register', que proporciona una interfaz entre la CPU y los diferentes elementos periféricos en el mismo chip), y 640 bytes de espacio libre para el usuario (memoria FLASH). Estos datos se acceden indirectamente a través de un grupo de registros de control mapeados en los registros SFRs. La memoria externa puede extenderse hasta 16 Mbytes.

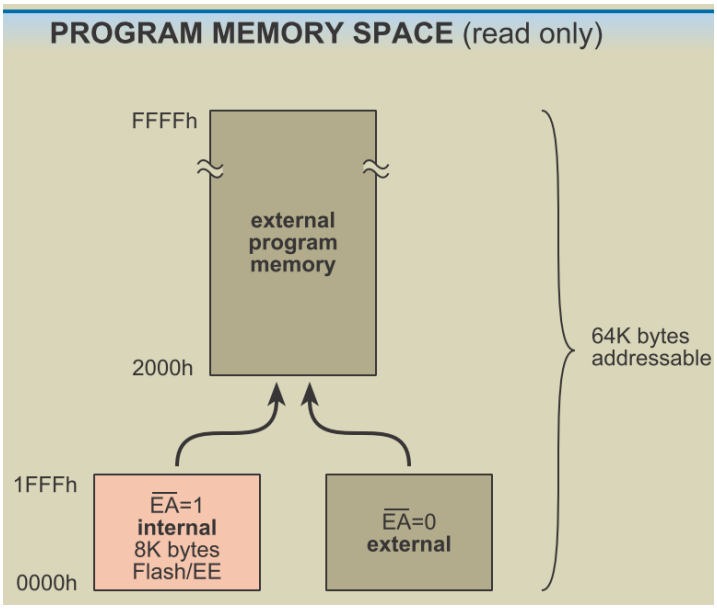


Figura 1.18. Memoria para programas. (Figura obtenida de documentación proviniente de Analog Devices, Inc.(www.analog.com)).

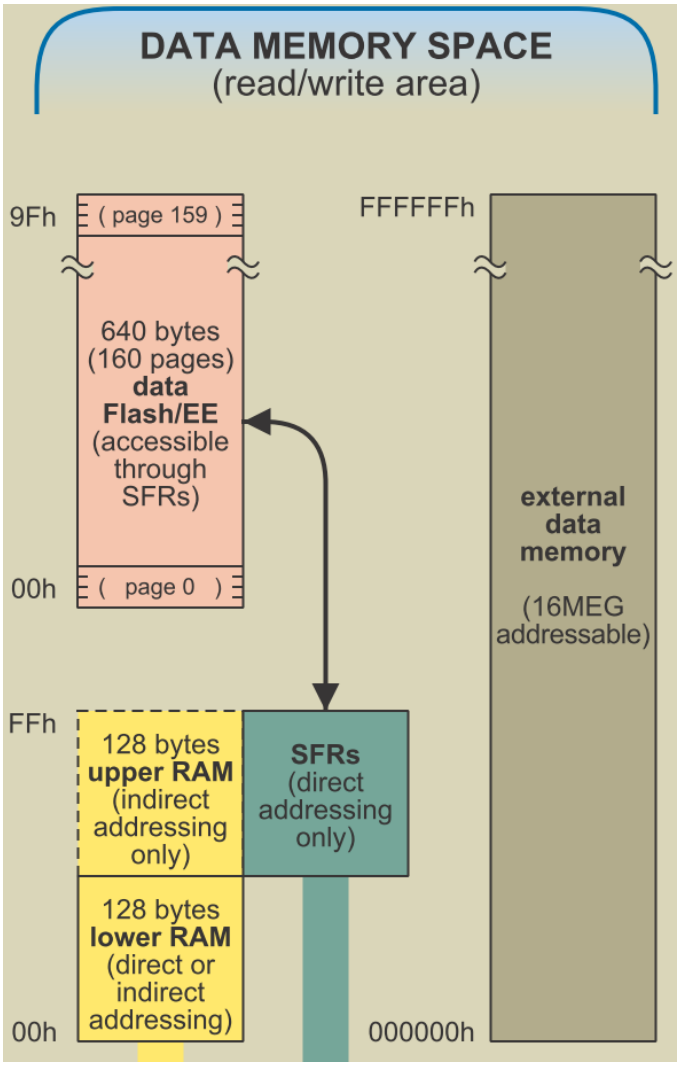


Figura 1.19. Memoria para datos. (Figura obtenida de documentación proviniente de Analog Devices, Inc.(www.analog.com)).

1.4.2.2 ADuC824

A diferencia del ADuC812, este componente posee dos conversores ADCs (sigma-delta) de alta resolución (uno de 24 bits con ganancia programable y otro de 16 bits). Está especialmente indicado para la medida de pequeñas señales, aunque de baja frecuencia.

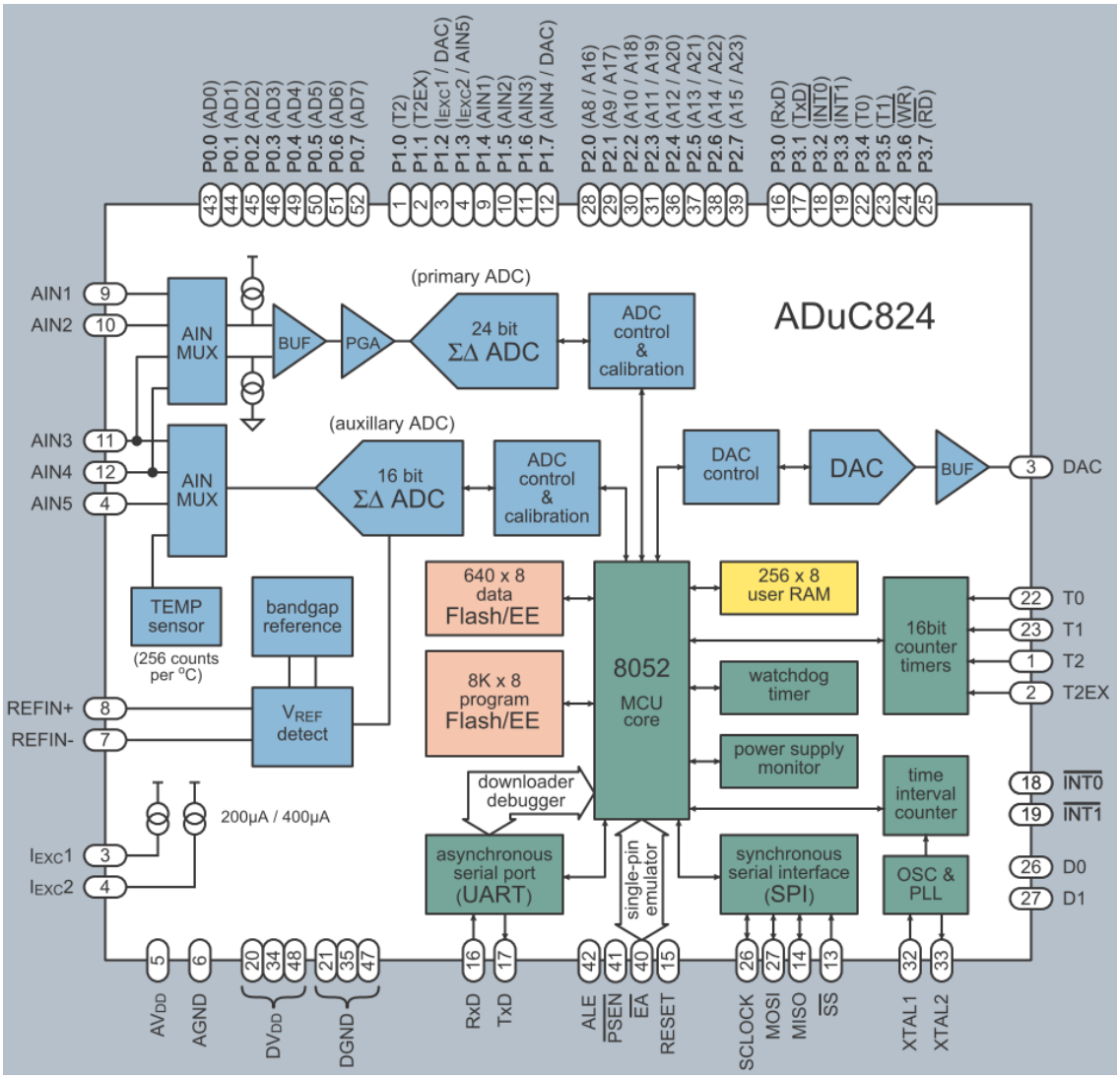


Figura 1.20. Esquema funcional del ADuC824. (Figura obtenida de documentación proviniente de Analog Devices, Inc. (www.analog.com)).

1.4.2.3 MSP430

Es una familia de procesadores de 16-bit de Texas Instruments (TI) de muy bajo consumo y de señales mixtas. Un oscilador controlado digitalmente (DCO) permite activar el dispositivo desde modos 'sleep' de forma rápida. Está especialmente indicado para aplicaciones portátiles que necesiten un bajo consumo. El procesador es de tipo RISC.

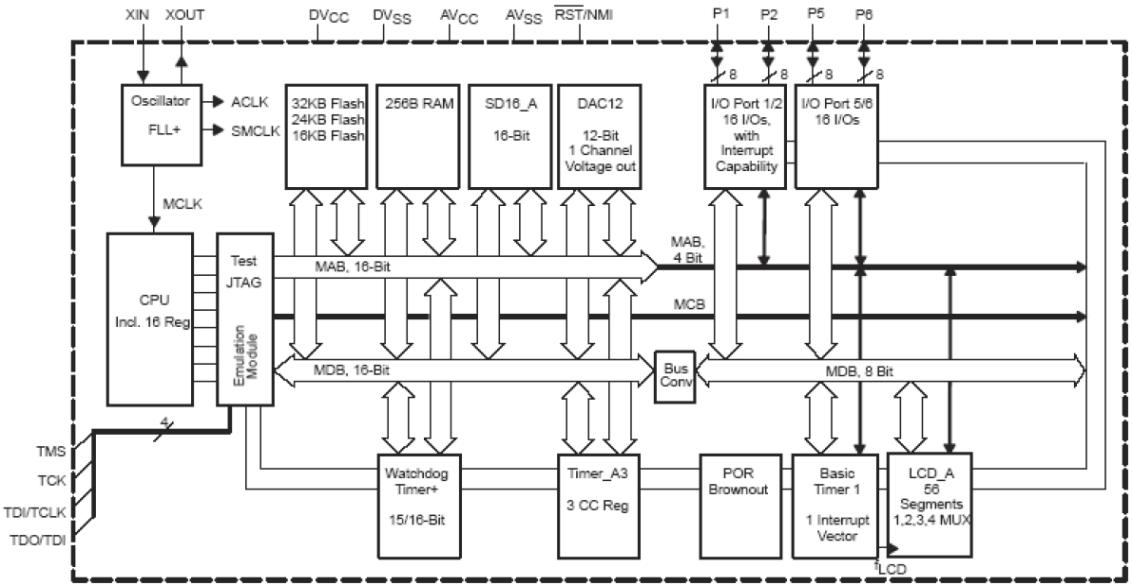


Figura 1.21. Esquema funcional del MSP430. (Figura obtenida de Texas Instruments, a la que se le aplica todas las garantías, condiciones, limitaciones y notas originales de Texas Instruments).

Módulo de conversión sigma-delta del MSP430F42x0:

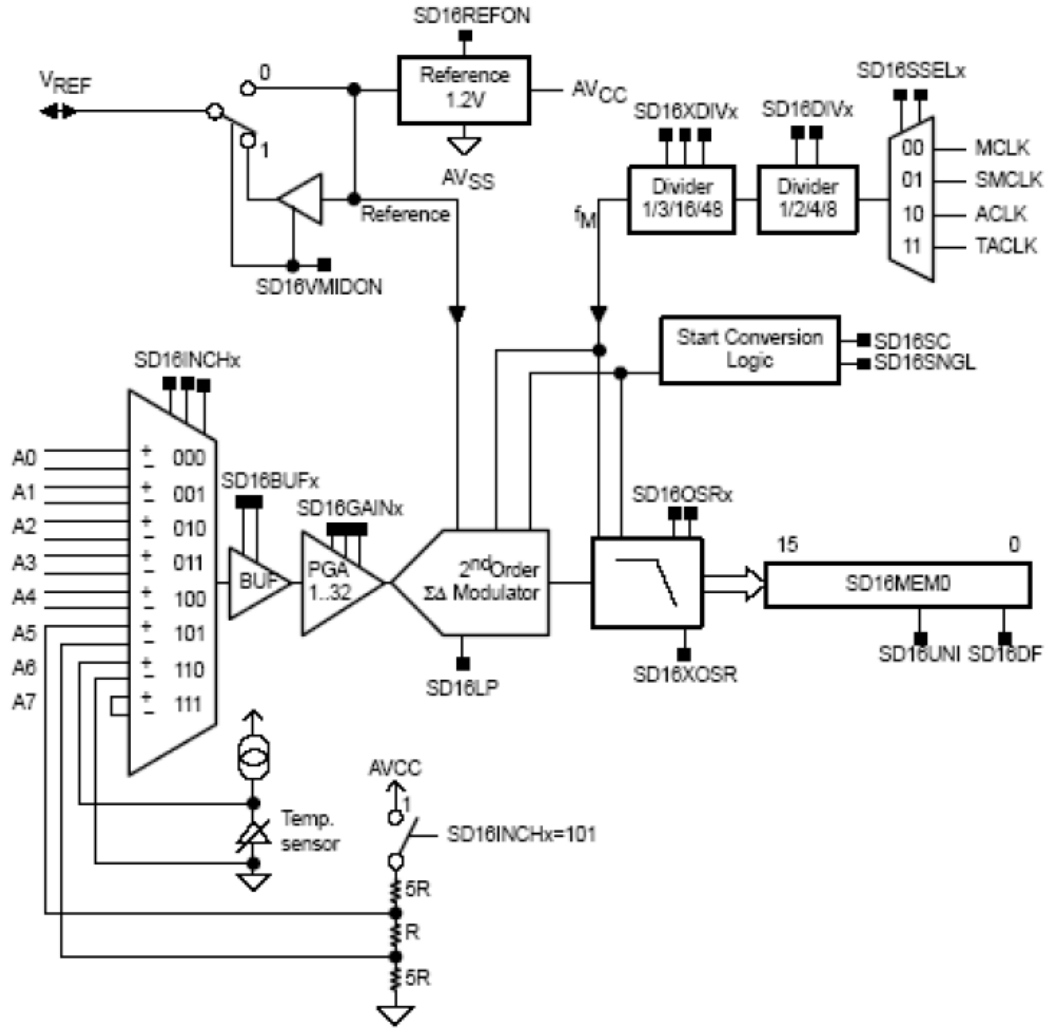


Figura 1.22. Módulo de conversión sigma-delta del MSP430. (Figura obtenida de Texas Instruments, a la que se le aplica todas las garantías, condiciones, limitaciones y notas originales de Texas Instruments)

Esta serie de la familia utiliza conversores sigma-delta.

1.4.2.4 Z8 Encore!

(De Zilog)

Familia de microprocesadores basados en una CPU eZ8 de 8 bits. Posee un carácter general y se puede aplicar en el campo de sensores, domótica, etc.

La resolución es de 10 bits.

Tiene protocolos de comunicación serie I^2C y SPI.

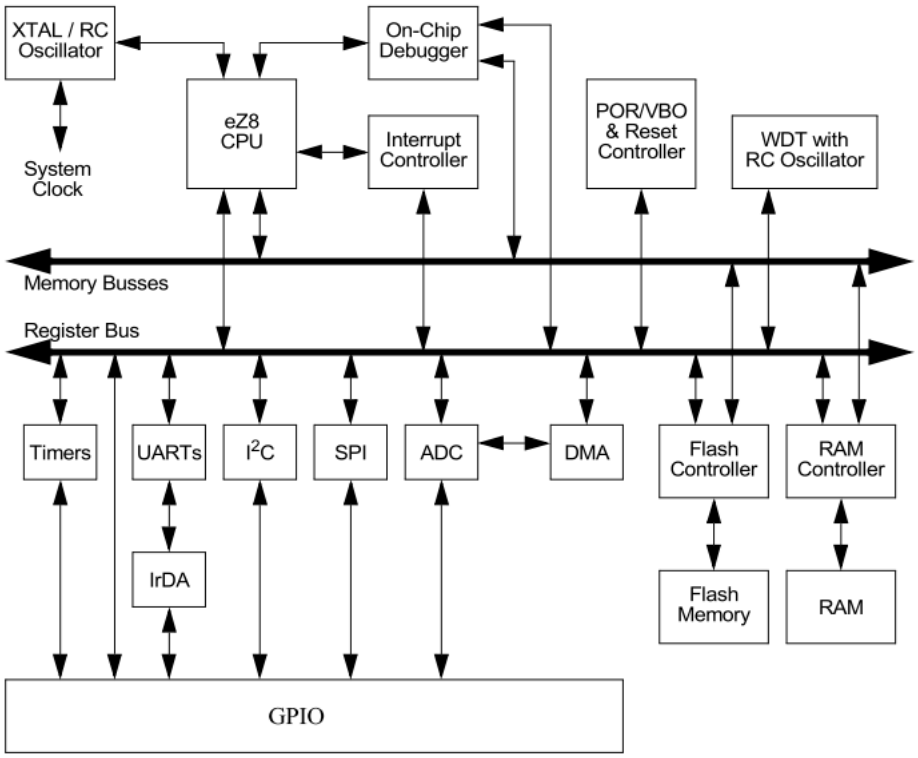


Figura 1.23. Diagrama de bloques del Z8 Encore!. (Reproduced by permission. Copyright 2003 Zilog, Inc. (www.zilog.com)).

2. COMUNICACIONES EN INSTRUMENTOS DIGITALES

2.1 CLASIFICACION DE BUSES DE TRANSMISIÓN DE INFORMACIÓN EN INSTRUMENTOS DIGITALES

Por tipo de transmisión de datos:

- Paralelo (velocidad): Corta distancia.
- Serie (economía): Larga distancia.

Por distancia:

- Dentro del instrumento (entre dispositivos):
  - Paralelos:
    - Propios de los procesadores: De direcciones, datos, control. Entre los distintos componentes del procesador.
    - Buses de sistema: ISA, PCI, VME, …
  - Serie: SCI, SPI, I^2C, Microwire.
- Entre instrumentos próximos:
  - Paralelo (del orden de algunos centímetros a muy pocos metros (2 ó 3m): IEEE488 (de 8 bits a 1Mbyte/s), CompactPCI, VXI.
  - Serie (de centímetros a cientos de metros, según bus y red): CAN, Profibus, I^2C, Red local.
- Instrumentos lejanos: Redes WAN ('Wide Area Networks').

Nosotros veremos las comunicaciones serie dentro del instrumento y para instrumentos próximos.

Protocolo:

- Nivel de bit.
- Nivel de palabra.
- Nivel de mensaje.

2.2 TERMINOLOGÍA DE COMUNICACIONES

Transmisor: Dispositivo que envía información al bus de datos.

Receptor: Dispositivo que recoge información del bus.

Maestro ('Master`): Dispositivo que inicía una transferencia, controla la señal de reloj y finaliza la transferencia.

Esclavo ('Slave'): Dispositivo direccionado por el maestro.

Multi-Maestro: Sistema donde más de un maestro puede intentar controlar el bus a la vez, sin corromper el mensaje.

Arbitraje: Procedimiento para permitir únicamente un Maestro en el bus y, a la vez, que el mensaje no sea corrompido.

2.3 BUS SPI ('SERIAL PERIPHERAL INTERFACE')

2.3.1 Introducción

Es un bus de comunicaciones entre dispositivos sencillo (la transmisión es sencilla y es fácil de implementar).

Pensado para dispositivos relativamente lentos y a los que se accede de forma interrumpida.

En comparación con otros buses del mismo estilo (como I^2C), este bus está más pensado para aquellas aplicaciones más relacionadas con flujos de datos contínuos ('data streams') y no para las comunicaciones de acceso 'puntual' a dispositivos esclavos. Como ejemplos están las comunicaciones entre microprocesadores (o en DSPs ('Digital Signal Processors'), o en microcontroladores (μC)) o la transferencia de datos desde ADCs. Aparecen comúnmente en dispositivos de mano o portátiles.

Debido a su falta de direccionamiento de dispositivos, este bus es más eficiente cuando sólo existen dos dispositivos: un maestro y un esclavo. Si existen más dispositivos, esta misma característica obliga a un mayor equipamiento hardware de los dispositivos.

Otra de sus características es la alta velocidad. Puede funcionar entorno a las decenas de de MHz.

No existe una especificación oficial. Eso hace que siempre tengamos que recurrir a las hojas de características de los componentes que se vayan a usar.

2.3.2 Características generales

Es un bus serie síncrono, donde un dispositivo maestro puede comenzar una comunicación con otros dispositivos esclavos.

Aparece generalmente en procesadores.

Utiliza una relación de comunicación Maestro-Esclavo.

La comunicación es 'Full-Duplex'. Es decir, puede transmitirse información en ambos sentidos simultáneamente. El Maestro y el Esclavo han de juzgar si los bytes recibidos son significativos o no.

En microcontroladores PICmicro se implementa por un módulo llamado SSP ('Synchronous Serial Port') o el MSSP ('Master SSP').

2.3.3 Conceptos del bus SPI

Este bus serie posee 4 líneas diferentes (cada una de ellas unidireccional):

- MISO ('Master In Slave Out'): Datos del Esclavo al Maestro.
- MOSI ('Master Out Slave In'): Datos del Maestro al Esclavo.
- SCLK ('Serial Clock'): Generada por el Maestro. Común a todos los dispositivos.
- SS ('Slave Select') (o CS ('Chip Select')): Controlada por el Maestro, tendrá que haber tantas como esclavos.

Las comunicaciones son síncronas. El reloj es común para todos los dispositivos. Esta señal la genera el Maestro. El reloj determina cuándo los datos pueden cambiar y cuándo son válidos para ser leídos. El hecho de tener una señal de reloj permite que esta señal pueda cambiar sin que haya interrupción de datos. Por tanto, no tiene que haber un control muy estricto de la señal del reloj (podría ser generada por un oscilador RC poco preciso).

Es un protocolo Maestro-Esclavo. Sólo el Maestro puede controlar la señal del reloj. Sin manipular esta señal, no podrá haber intercambio de información. Todos los esclavos se controlan mediante esta señal de reloj. En los PICmicro, los registros del SSP controlan cómo responden los dispositivos a esta señal de reloj.

También es un protocolo 'Data Exchange'. Ningún dispositivo puede ser únicamente transmisor o únicamente receptor. Así, cada dispositivo posee una línea de entrada y otra de salida. Cuando se transmite datos, antes de volver a transmitir se ha de leer la entrada. Si no se hace así, estos datos se perderán. (es decir, no hay 'Acknowledge'). Esta lectura se ha de hacer, incluso aunque no se prevea necesitar ese dato. Cuando existen más de dos dispositivos, se utiliza SS para seleccionar el dispositivo al que se accede. Esta señal indica al Esclavo que el Maestro quiere iniciar un intercambio de

datos (normalmente, se indica poniendo a línea a 0). Habrá un SS por cada dispositivo Esclavo.

La comunicación es 'Full-Duplex' ya que se puede transmitir información en ambos sentidos simultáneamente una vez se ha abierto el proceso de comunicación.

Sólo hay un protocolo a nivel de bit (mediante SCK).

Ejemplo de sistema con tres esclavos:

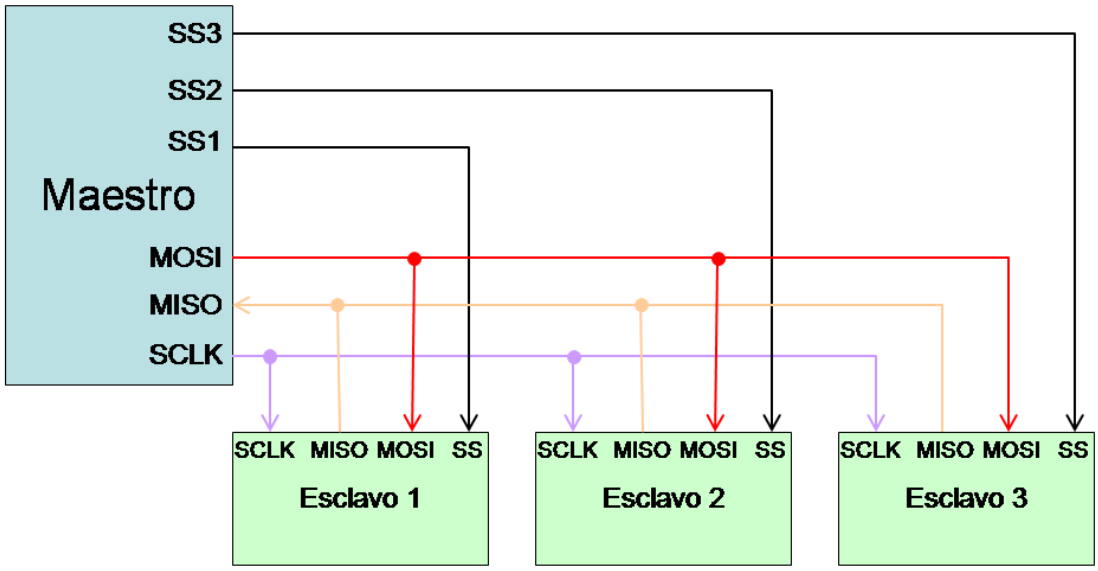


Figura 2.1. Conexión de instrumentos en el bus SPI.

El estándar, no especifica una forma concreta de implementación con un sistema multi-Maestro.

Tampoco tiene un sistema de 'Acknowledge' ni tampoco de control de flujo.

Aunque no están especificadas las transiciones donde realizar el “latching” de los datos, en la práctica existen cuatro tipos (o modos), controlados por CPOL ('Clock POLarity': indica si el estado “idle” se considera el nivel alto o el bajo) y CPHA ('Clock PHAse': O flanco de subida o bajada). (la razón es que estos dos factores pueden controlarse en los microprocesadores de Motorola). Si CPHA=0, CPOL=0 significa “latching” en flanco de subida, mientras que CPOL=1 en el de bajada. Si CPHA=1, el significado es justo el opuesto. (estas variables de control están ubicadas en un registro).

Los datos se ponen a la salida (cambian) al entrar en el flanco de subida o bajada (el que corresponda), y se leen al entrar en el flanco opuesto del anterior.

Transfiere datos siempre en bloques de 8 bits. Se utiliza un registro de desplazamiento para ir poniendo o adquiriendo los datos del bus. Al adquirir un byte, éste se copia a un registro de buffer, del cual se lee. Si se ha de enviar datos, estos se escriben primero en el buffer.

2.3.4 Ejemplos

2.3.4.1 ADC con bus SPI (LTC1290)

Veamos primero el diagrama de bloques del LTC1290 ('12-bit 8-channel DAQ'):

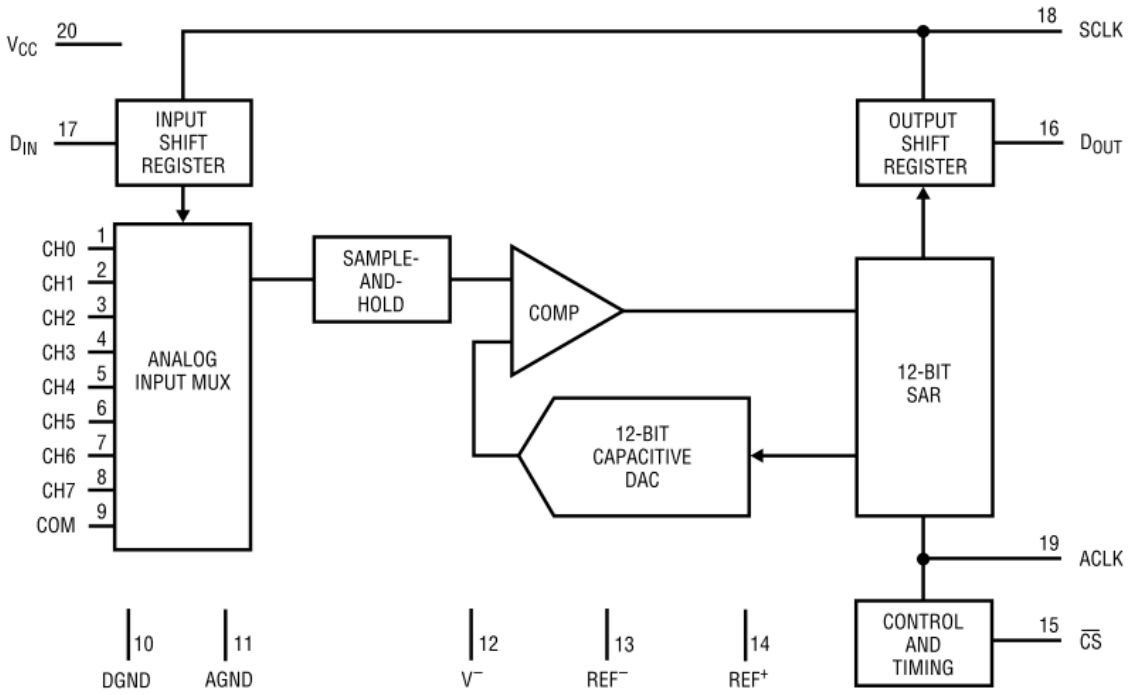


Figura 2.2. Diagrama de bloques del ADC LTC1290. (Figura obtenida de Linear Technology (www.linear.com)).

El LTC1290 se comunica por un bus SPI. Transmite en el 'falling edge', mientras que recibe en el 'rising edge'.

Tras colocar SS bajo (inicio de comunicación), primero lee 8 bits de DIN ('Digital Data Input') que configura el LTC1290 para la conversión (unipolar/bipolar,). Simultáneamente, el resultado de la conversión A/D anterior se pone en DOUT. Después, comienza la conversión. En este momento, SS debería ponerse a nivel alto, y esperarse un cierto tiempo t_{conv} para asegurar que finalice la conversión. Tras este tiempo, el resultado de la conversión estará disponible en el siguiente paso de reloj. Y así sucesivamente:

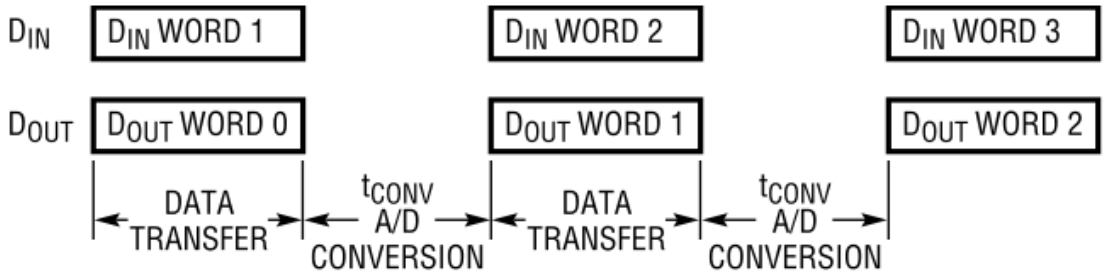


Figura 2.3. Diagrama de bloques del ADC LTC1290. (Figura obtenida de Linear Technology (www.linear.com)).

Los 8 bits de DIN son:

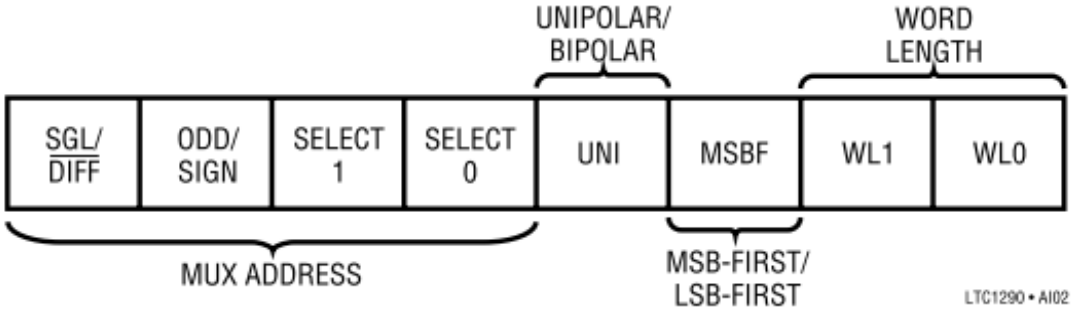


Figura 2.4. Primer byte de configuración. (Figura obtenida de Linear Technology (www.linear.com)).

Con ODD/SIGN y los SELECT se selecciona el canal de entrada a medir y la polarización (si se ha escogido diferencial). UNI se refiere al tipo de conversión por aproximaciones sucesivas (SAR) (sólo positivos o negativo/positivo (con bit de signo)). La longitud del dato de salida puede ser 8, 12 o 16 bits. (estos bits también se usan para hacer un 'power down' (01).

La secuencia de operación para un caso concreto de configuración de la conversión se muestra en la siguiente figura (DIN: 01010110):

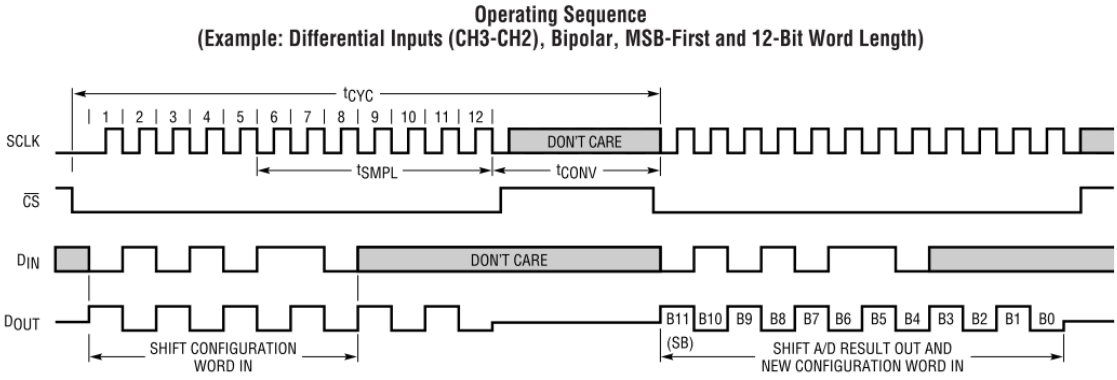


Figura 2.5. Secuencia para una conversión de entrada diferencial, bipolar, MSB-primero y 12-bits. (Figura obtenida de Linear Technology (www.linear.com)).

Fijarse que, mientras se entra DIN, en DOUT se envía el dato adquirido previamente (en este ejemplo no se conoce porque no se ha incluído).

Fijarse también que los bits en DIN se ponen en el 'falling edge' para poder capturarlos en el 'rising edge'.

La implementación de esta aplicación se muestra en la siguiente figura:

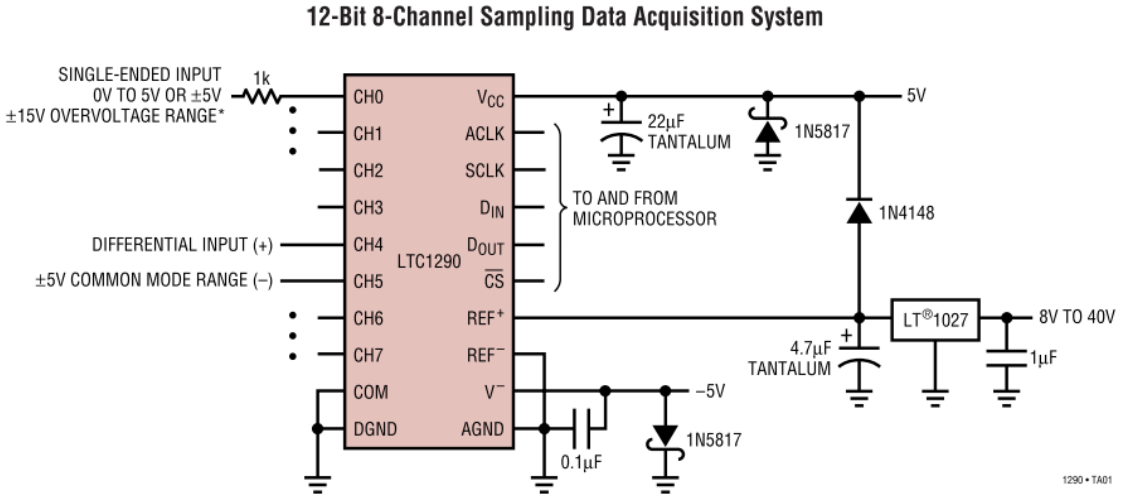


Figura 2.6. Implementación del conversor. (Figura obtenida de Linear Technology (www.linear.com)).

El LT1027 es una referencia de tensión de 5V. El 1N5817 es un diodo Schottky.

2.3.4.2 Memoria con bus SPI

Veamos el diagrama de bloques de la memoria EEPROM SPI X5163:

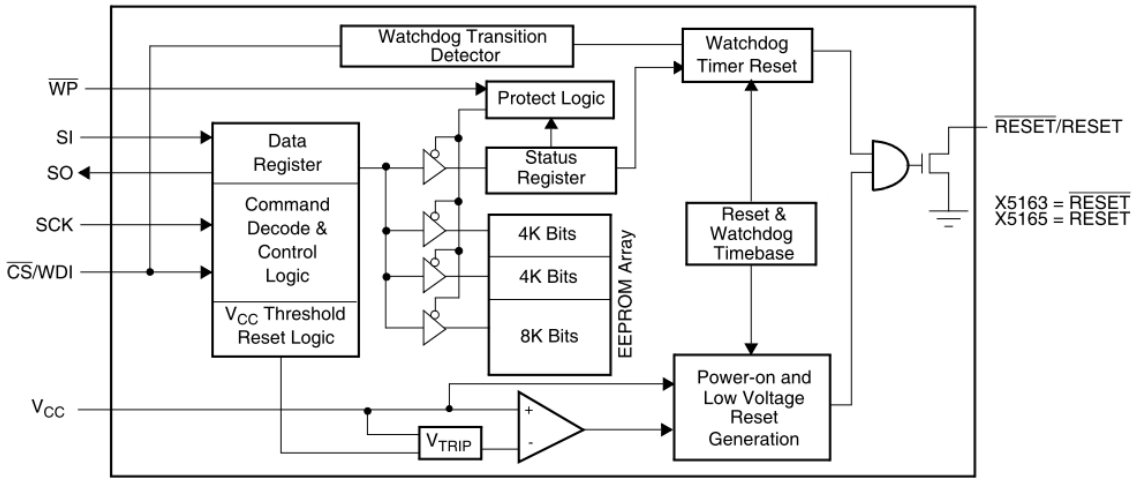


Figura 2.7. Diagrama de bloques de la memoria EEPROM SPI X5163. (Copyright Intersil Americas Inc. 2005, 2006. All Rights Reserved (www.intersil.com)).

(WP: 'Write Protect').

Puede verse que es una memoria de 16kbits. Posee un registro de estado. El Watchdog Timer Reset proporciona un mecanismo de protección independiente para

microcontroladores. Si el microcontrolador no reinicía el temporizador dentro de un cierto intervalo de tiempo (elegible) (lo hace conmutando la entrada CS/WDI), el dispositivo activa la señal RESET/RESET ('Reset Output').

Como indica su nombre, posee una interficie de comunicación SPI. Lee datos en el flanco de subida, mientras que transmite datos en el flanco de bajada. MSB se transfiere primero.

Contiene un registro de 8-bit de instrucciones.

Para poder escribir, primero se debe activar el 'Write Enable Latch'. Esto lo realiza la instrucción WREN (00000110), mientras que la instrucción WRDI (00000100) resetea el latch. Sin embargo, este latch se resetea automáticamente después de cada 'power-up' y después de cada escritura exitosa. La siguiente imagen muestra cómo realizar la instrucción WREN:

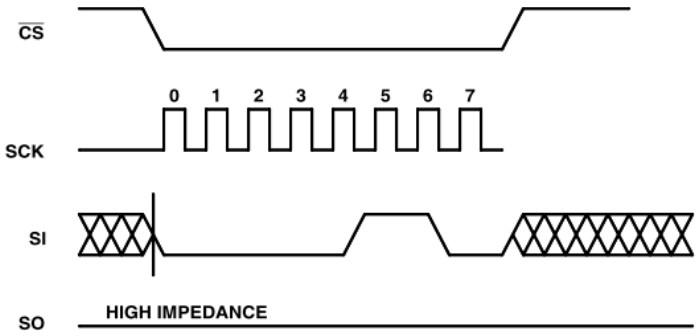


Figura 2.8. Secuencia del 'Write Enable' (WEN). (Copyright Intersil Americas Inc. 2005, 2006. All Rights Reserved (www.intersil.com)).

Posee un registro de estado, con el siguiente formato:

7	6	5	4	3	2	1	0
WPEN	FLB	WD1	WD0	BL1	BL0	WEL	WIP

Figura 2.9. Formato del registro de estado de la memoria. (Copyright Intersil Americas Inc. 2005, 2006. All Rights Reserved (www.intersil.com)).

(WIP: Write-in-progress (se lee con RDSR), WEL: Write enable latch, BL0, BL1: Block lock bits (permite proteger porciones (bloques) de la memoria), WD0, WD1: Watchdog timers bits (seleccionan el Watchdog time out period), FLAG: usado por el sistema, WPEN: Enable Write Protection).

Secuencia para leer la memoria:

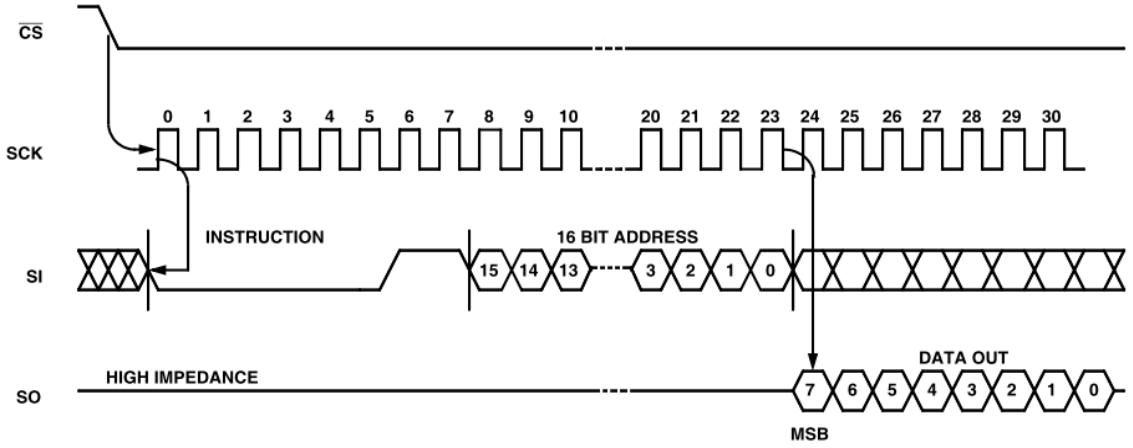


Figura 2.10. Secuencia para la lectura de la memoria. (Copyright Intersil Americas Inc. 2005, 2006. All Rights Reserved (www.intersil.com)).

Tras poner CS a nivel bajo, se envía la instrucción de lectura (8-bit, 00000011), seguida de la dirección a leer (16-bit) y automáticamente este registro se 'shiftea' en la línea de salida (SO). Si queremos seguir leyendo datos secuenciales al inicial, sólo hemos de mantener CS bajo y seguir enviando pulsos de reloj. Para finalizar, hacemos CS alto.

Secuencia para escribir en la memoria:

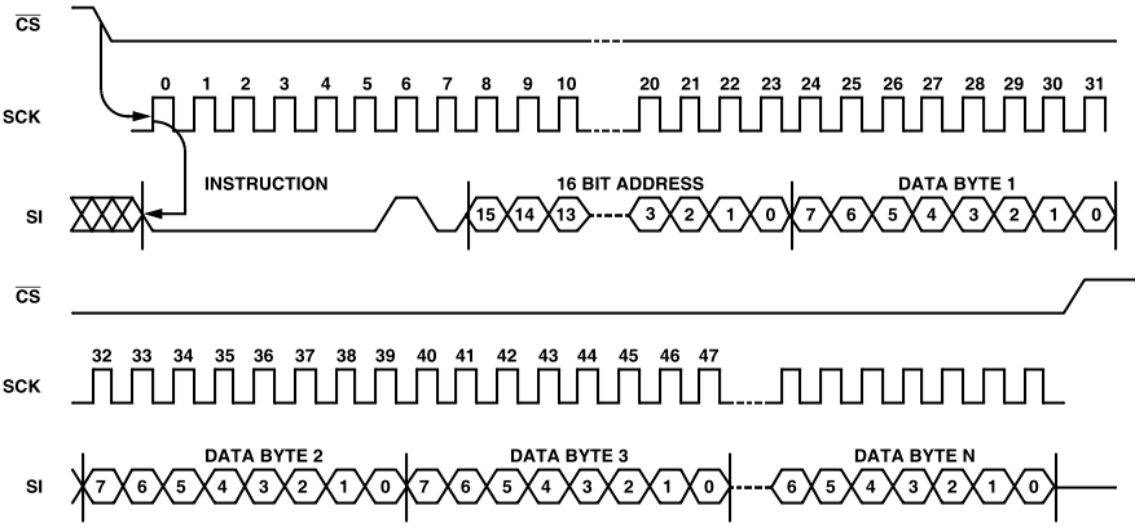


Figura 2.11. Secuencia de escritura de la memoria. (Copyright Intersil Americas Inc. 2005, 2006. All Rights Reserved (www.intersil.com)).

Antes de iniciar la escritura, el primer paso ha de ser activar WEL ('Write Enable Latch'), mediante la instrucción WREN tal y como se ha visto antes. Es decir, primero se pone CS a nivel bajo, seguido de la instrucción y finalmente se pone CS a nivel alto (si no se hace, se ignorará la instrucción de escritura).

Para proceder a la escritura, se sigue la secuencia de la figura: CS se pone a nivel bajo, se envía la instrucción de escritura (00000010), seguido de la dirección de escritura (16-bit) y a continuación los datos (8-bit). Si no se pone CS a nivel alto, se seguirá escribiendo en la siguiente posición de memoria secuencialmente. Se acaba cuando se pone CS a nivel alto (justo después del último bit del último byte).

Set de instrucciones:

INSTRUCTION NAME	INSTRUCTION FORMAT*	OPERATION
WREN	0000 0110	Set the Write Enable Latch (Enable Write Operations)
SFLB	0000 0000	Set Flag Bit
WRDI/RFLB	0000 0100	Reset the Write Enable Latch/Reset Flag Bit
RDSR	0000 0101	Read Status Register
WRSR	0000 0001	Write Status Register (Watchdog,BlockLock,WPEN & Flag Bits)
READ	0000 0011	Read Data from Memory Array Beginning at Selected Address
WRITE	0000 0010	Write Data to Memory Array Beginning at Selected Address

NOTE: *Instructions are shown MSB in leftmost position. Instructions are transferred MSB first.

Figura 2.12. Set de instrucciones de la memoria. (Copyright Intersil Americas Inc. 2005, 2006. All Rights Reserved (www.intersil.com)).

2.3.4.3 Memoria EEPROM con bus Microwire

(Microwire es un sistema SPI, pero más reducido (subconjunto))

La conexión más fácil entre una EEPROM y otros dispositivos (Microwire) es a través de 4-cables:

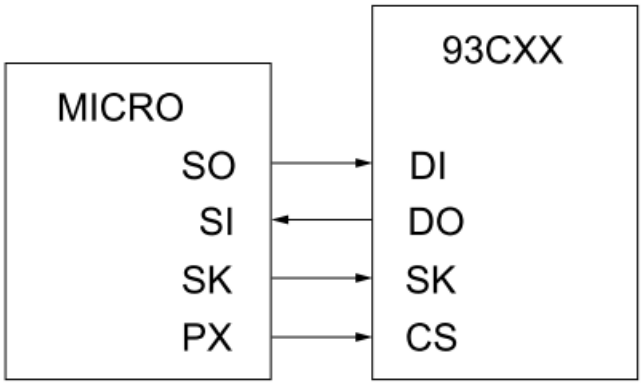


Figura 2.13. Conexión de 4 cables. (Copyright Intersil Americas Inc. 2005, 2006. All Rights Reserved (www.intersil.com)).

Sin embargo, a veces es necesario una línea bidireccional, que puede realizarse con 3 cables de la siguiente forma:

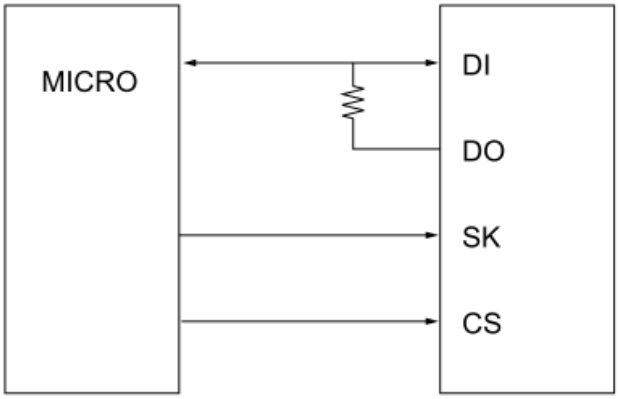


Figura 2.14. Conexión de 3 cables. (Copyright Intersil Americas Inc. 2005, 2006. All Rights Reserved (www.intersil.com)).

En este caso, el conjunto de instrucciones son:

Instruction	SB	Op Code	Address	Data	Comments
READ	1	10	A7/A5–A0		Reads data stored in memory.
EWEN	1	00	11XXXX		Write enable must precede all programming modes.
ERASE	1	11	A5–A0		Erase register A5A4A3A2A1A0.
WRITE	1	01	A5–A0	D15–D0	Writes register.
ERAL	1	00	10XXXX		Erase all registers.
WRAL	1	00	01XXXX	D15–D0	Writes all registers.
EWDS	1	00	00XXXX		Disables all programming instructions.

Figura 2.15. Set de instrucciones de la memoria en bus Microwire de 3 cables. (Copyright Intersil Americas Inc. 2005, 2006. All Rights Reserved (www.intersil.com)).

Secuencias de lectura y escritura:

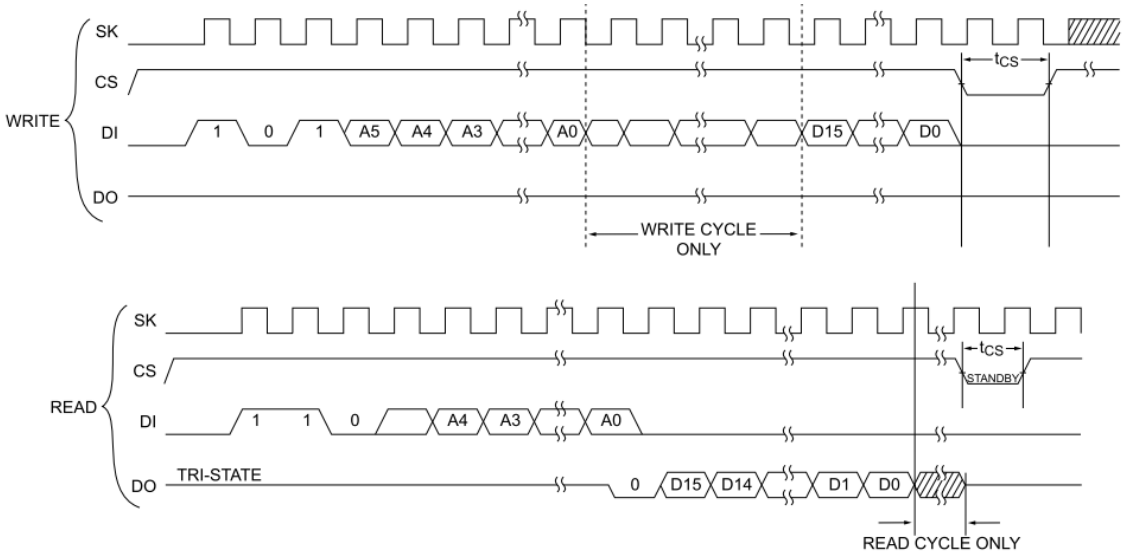


Figura 2.16. Secuencias de lectura y escritura. (Copyright Intersil Americas Inc. 2005, 2006. All Rights Reserved (www.intersil.com)).

Tanto la lectura como la escritura, la secuencia es poner CS alto, un bit de inicio (1) y dos bits de indicación de la operación, la dirección y los datos. Se acaba al conmutar CS a 0.

2.4 BUS I²C ('INTER-INTEGRATED CIRCUIT')

Al igual que el bus SPI, I^2C es un bus de dispositivos. Y, además, está pensado también para dispositivos relativamente lentos y a los que se accede de forma interrumpida.

Fue desarrollado por Philips Semiconductors (actualmente NXP) para controlar los circuitos integrados mediante un bus simple (bidirecional de dos hilos). Está orientado a transferencias de 8 bits. Se pueden conseguir hasta 100kbit/s en el modo estándar, aunque también puede llegar a 400 kbit/s en el modo rápido, o incluso mayores en otros modos (1 Mbit/s o 3.4Mbit/s (modo de alta velocidad)).

2.4.1 Concepto del bus I²C

Bus serie síncrono, con un sistema de comunicación Maestro-Esclavo.

Físicamente, el bus consiste en 2 hilos (3 con el de masa (referencia)) que transportan información entre los dispositivos conectados al bus:

- SDA ('Serial DAta Line'): Datos serie. Half duplex (bidireccional, pero no simultáneo).
- SCL ('Serial CLock Line'): Reloj serie.

(En general, estos sistemas suelen compartir la misma masa, con lo que la tercera línea no es necesario tenerla en cuenta).

Cada dispositivo se reconoce mediante una dirección única (propia) y puede operar como receptor y/o transmisor, dependiendo de la función del dispositivo.

El maestro inicía la transmisión y genera la señal de reloj. En este momento, los dispositivos direccionados son considerados como esclavos.

En sistemas multimaestros, varios maestros pueden intentar controlar el bus simultáneamente. Sólo uno lo conseguirá, pero los mensajes no son destruídos. Existe un proceso de arbitraje para conseguir el control del bus.

Orientado a 8 bits (1 bytes).

Es bidireccional.

2.4.2 Características generales

SDA y SCL son líneas bidireccionales, conectadas a una fuente positiva mediante resistencias de 'pull-up' (o de polarización):

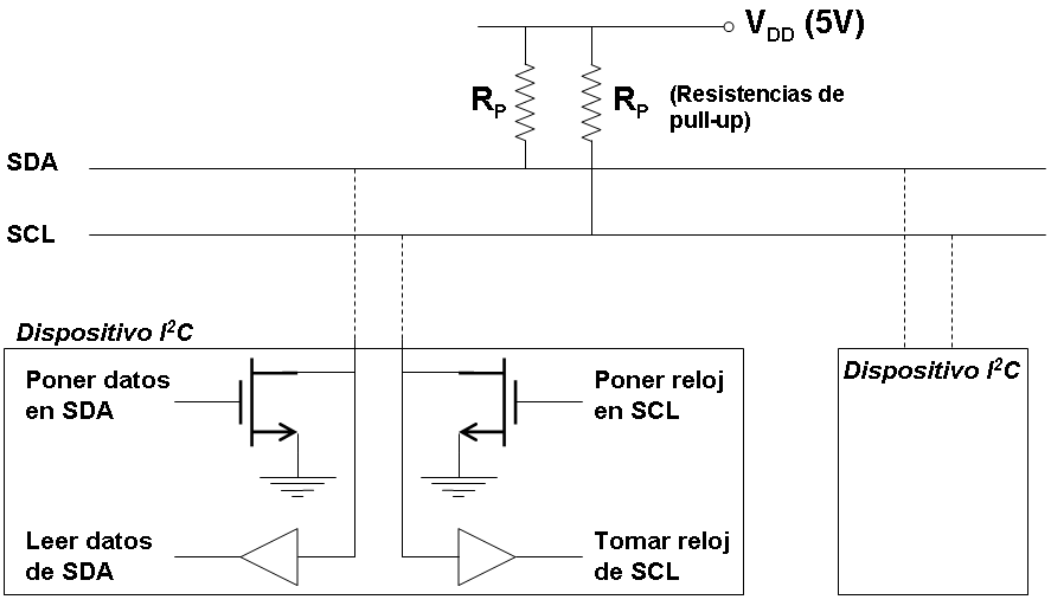


Figura 2.17. Conexión de dispositivos al bus I^2C.

(los dispositivos muestrean sus propias señales enviadas a las líneas) → permite Arbitrar y Sincronizar.

Cuando el bus está libre, las dos líneas están a nivel Alto.

Las salidas de los drivers de los dispositivos son open-drain (o colector abierto en TTL) para realizar una función Wired-AND (la salida es '1' si todas las salidas de los dispositivos es '1'). Tienen la ventaja de ser fáciles de implementar y baratos y, a la vez, evitan cualquier posibilidad de cortocircuitos.

Según la velocidad de transmisión, tiene varios modos de operación:

- Estándar: 100 kbits/s.
- Rápido: 400 kbits/s.
- Alta velocidad: 3.4 Mbits/s.

El número de dispositivos conectados al bus sólo depende de la capacidad límite: 400 pF.

Los dispositivos pueden ser tanto transmisores como receptores. Además, pueden ser también Maestros o Esclavos.

Los niveles de tensión correspondientes al '0' y '1' lógicos no están fijados por el estándar. Dependerán del valor de V_{DD}. Los niveles de referencia de entrada sí se definen, como $0.3 \cdot V_{DD}$ y $0.7 \cdot V_{DD}$.

2.4.3 Transferencias de bit

Se necesita un pulso de reloj por cada bit a transferir.

El dato en SDA ha de permanecer estable durante el período Alto del reloj (SCL). SDA sólo puede cambiar de estado durante la parta baja de SCL:

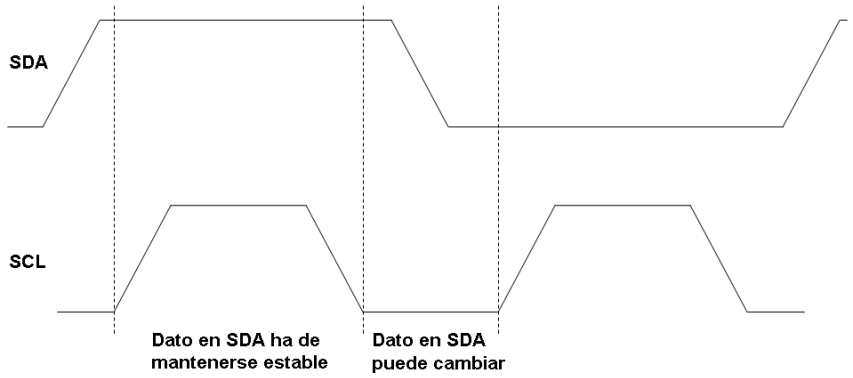


Figura 2.18. Trasferencia de un bit en SPI.

Condiciones de START y STOP: Ambas ocurren con la SCL a nivel Alto:

- START: Se indica con la transición de SDA de nivel Alto a nivel Bajo:

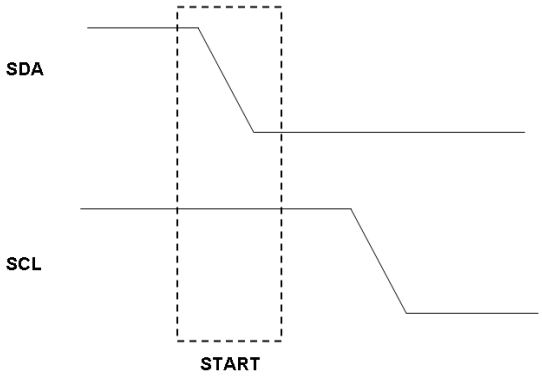


Figura 2.19. Condición START.

- STOP: Se indica con la transición de SDA de nivel Bajo a nivel Alto:

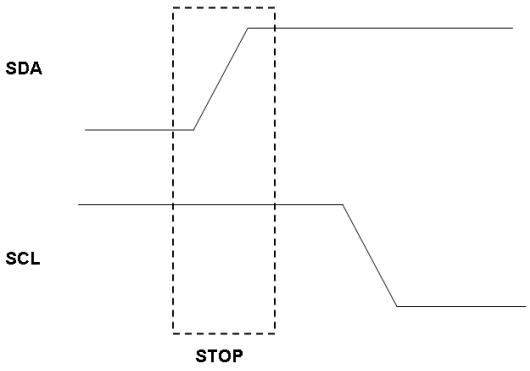


Figura 2.20. Condición de STOP.

Después de haber tenido lugar una condición START, el bus se considerará OCUPADO ('BUSY'). Mientras que después de un cierto tiempo de haber tenido lugar una condición STOP, el bus se considerará LIBRE ('FREE').

Por hardware es fácil de detectar estas condiciones. Si el dispositivo no posee este hardware, se ha de hacer por software, habiendo que muestrear al menos dos veces por cada período del reloj.

2.4.4 Transferencia de datos

Los datos se transfieren después de una condición START.

Los datos se transmiten siempre en formato de byte (8 bits), con el MSB ('Most Significant Bit') primero. No hay límite en el número de bytes transmitidos en cada transferencia.

Bit de 'Acknowledge': Es obligatorio. Se emite después de transferirse un dato (8 bits) y sirve para que el Esclavo indique al Maestro que ha recibido de forma correcta el último byte. El Maestro genera un pulso de reloj adicional. Durante ese pulso, el Transmisor ha de poner la línea SDA a Alta y el receptor ha de bajarla y mantenerla estable durante la parte alta de este ciclo de reloj.

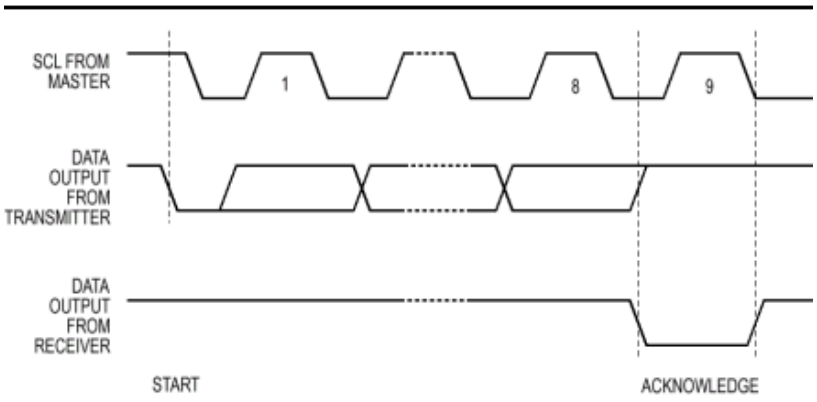


Figura 2.21. 'Acknowledge' en el bus I^2C.

Tras el 'Acknowledge', si el receptor recibe un dato y necesita cierto tiempo antes de aceptar un nuevo dato, este mismo tiene la opción de mantener la señal SCL a nivel Bajo para que el Maestro sepa que todavía no está listo. Cuando pueda recibir el próximo dato, liberará la señal SCL para que la controle el maestro.

Si no se produce el Acknowledge, el Maestro emitirá una condición de STOP, o un START repetido para volver a transferir el dato.

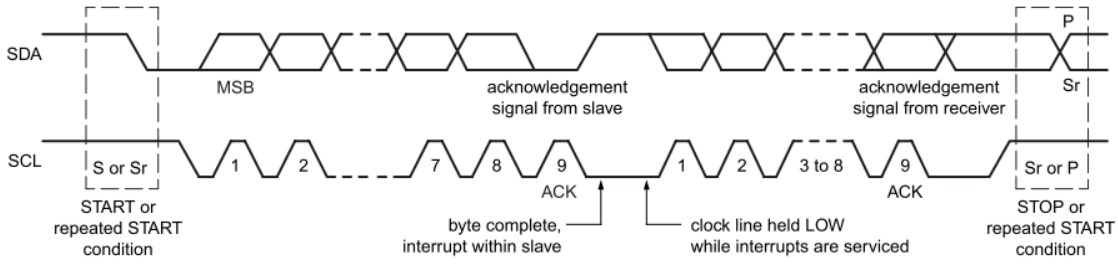


Figura 2.22. Respuesta del sistema a un 'not Acknowledge'.

Si el Maestro es receptor, ha de indicar el final de la transferencia al Esclavo-Transmisor. Esto lo realiza no dando el 'Acknowledge' en el último dato.

2.4.5 Arbitraje y generación de reloj

En sistemas multi-maestro, es necesario un sistema para decidir quién toma control del bus en cada momento. Este sistema se realiza mediante el arbitraje y la sincronización del reloj. En sistemas con un sólo Maestro, esto no es necesario.

2.4.5.1 Sincronización

Todos los maestros han de tener su propio reloj.

La sincronización del reloj en la línea SCL viene definida por la estructura Wire-AND de conexión:

- Estado Bajo del SCL lo definirá el maestro que tenga el t_L más largo. (ya que basta que uno lo tenga en estado Bajo, para que la línea siga en estado Bajo).
- Estado Alto de SCL lo definirá el maestro que tenga el t_H más corto. (por la misma razón que antes).

Con SCL a nivel Alto, cuando uno de los Maestros lo ponga a nivel bajo, todos los Maestros empezarán a 'contar' su nivel bajo. (es decir, que se pondrán a nivel bajo y empezará a contar su t_L particular cada uno).

Con SCL a nivel bajo, los Maestros que lo hayan puesto a nivel Alto, permanecerán así en estado de espera. Cuando todos se hayan puesto a nivel Alto, los relojes empezarán a 'contar' su nivel alto.

De esta forma, SCL tendrá un t_L igual al del reloj con t_L más largo, mientras que tendrá un t_H igual al del reloj con t_H más corto.

(Cada maestro dispone de dos contadores internos. Uno cuenta el tiempo en estado bajo de SCL y el otro el tiempo en estado alto).

Esquema de la sincronización del reloj durante el procedimiento de arbitraje:

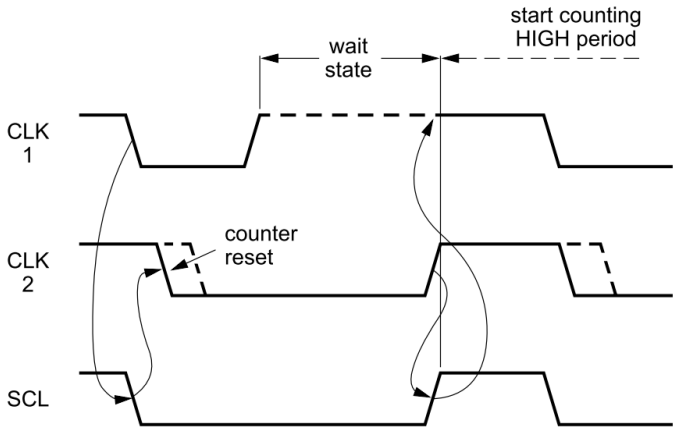


Figura 2.23. Esquema de sincronización durante el arbitraje.

2.4.5.2 Arbitraje

Los maestros sólo pueden iniciar una transferencia si el bus está libre. Pero dos o más Maestros pueden generar una condición START (durante el tiempo mínimo de 'hold'). El procedimiento de arbitraje se encarga de decidir cuál de los Maestros realizará la operación.

El arbitraje tiene lugar mediante la línea SDA, mientras SCL está en Alto.

Si un maestro transmite un '1', mientras otro envía un '0', el primero pierde el arbitraje y coloca su salida en 'OFF' (la línea estará a 0 ya que es tipo Wire-AND). Esto puede hacerse porque cada Maestro, además de sacar su dato, muestrea la línea SDA; si no coinciden, hace la operación anterior. El reloj también será una combinación Wire-AND de los relojes de ambos Maestros (sincronización).

El maestro que pierde el arbitraje, puede seguir enviando ciclos de reloj hasta finalizar el byte y esperar a que el bus esté inactivo para intentar de nuevo su operación. Sin embargo, debe cambiar a su modo Esclavo, puesto que el Maestro 'ganador' puede intentar direccionarlo.

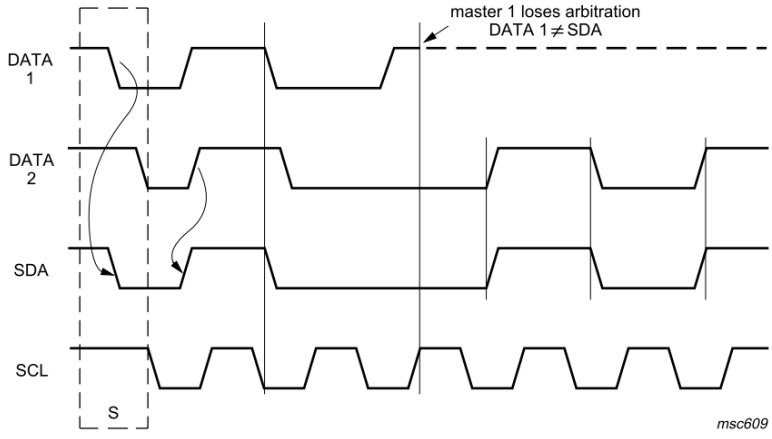


Figura 2.24. Ejemplo de arbitraje del bus I^2C.

Este arbitraje, puesto que se hace utilizando la dirección e información a transferir, implica que:

- No se pierde información ni tiempo.
- No hay un maestro central ni sistema de prioridades.

(La señal de reloj del bus, controlada por el Maestro, sólo puede ser modificada en dos casos: cuando un esclavo es lento y mantiene el reloj a nivel bajo, y durante el arbitraje, en el que dos Maestros puede estar controlando el reloj simultáneamente).

2.4.6 Formato de las tramas (dirección de 7 bits)

Las transferencias siempre empiezan con la siguiente secuencia:

- Generación de la condición START.
- El Maestro envía la dirección del Esclavo (de 7 bits de longitud).
- El octavo bit indica la dirección de la transferencia (R/W). ('0' es una transmisión, '1' es una petición de datos).
- Un bit de Acknowledge. (Esclavo).
- La transferencia de datos, al final, acabará en una condición de STOP. El Maestro siempre tiene la opción de volver a transferir datos, con lo que, en lugar de una condición de STOP, puede realizar una condición repetida de START (Sr), direccionando a otro Esclavo.

Esquema general de una secuencia de transferencia de datos:

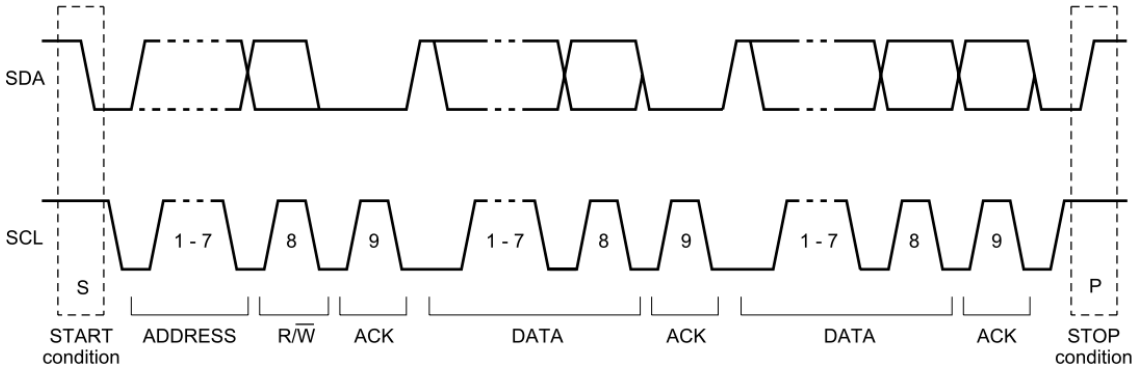


Figura 2.25. Ejemplo de secuencia de transferencia de datos.

Las secuencias siguientes dependerá del caso:

- Maestro-Transmisor y Esclavo-Receptor: (el bit de (R/W estará a 0)

- Envía dato de 8 bits.
- Genera un paso de reloj de 'Acknowledge' y el receptor ha de confirmar en SDA. Repite esta secuencia tantas veces como datos se tengan que transferir.
- El Maestro genera una condición de STOP.

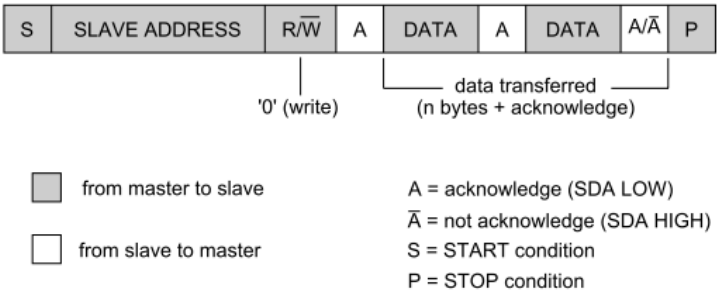


Figura 2.26. Secuencia para Maestro-Transmisor y Esclavo-Receptor.

- Maestro-Receptor y Esclavo-Transmisor: (el bit de (R/W estará a 1)
  - En el momento del primer 'Acknowledge' (generado por el esclavo), el Maestro-Transmisor pasa a ser Maestro-Receptor y el Esclavo-Receptor pasa a Esclavo-Transmisor. A partir de aquí, los 'Acknowledge' los genera el Maestro (receptor). Además, generará la condición de STOP, enviando un not-acknowledge previamente.

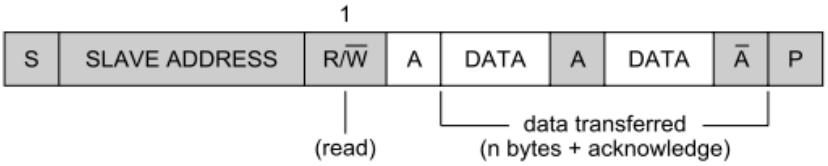


Figura 2.27. Secuencia para Maestro-Receptor y Esclavo-Transmisor.

- Combinado:
  - Cuando hay cambios de sentido durante la transferencia, se repiten: la condición de START y la dirección del Esclavo, pero invirtiendo el estado de R/W. (No hay STOP entre medio).

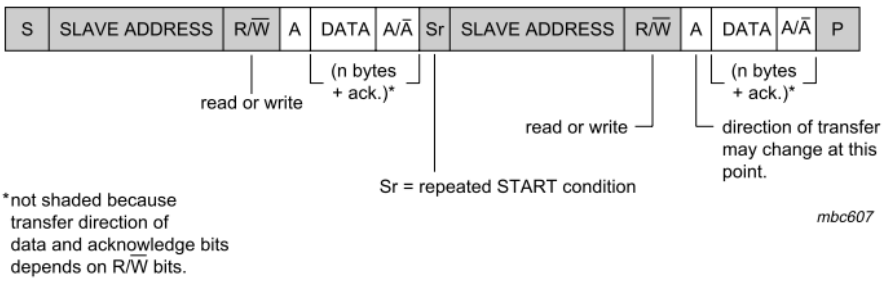


Figura 2.28. Secuencia para un modo combinado de los dos anteriores.

El caso combinado puede ser útil en casos como el de la comunicación con memorias, donde en primer lugar se ha de enviar un byte de configuración de la memoria, y posteriormente se puede leer.

Resumen de las secuencias:

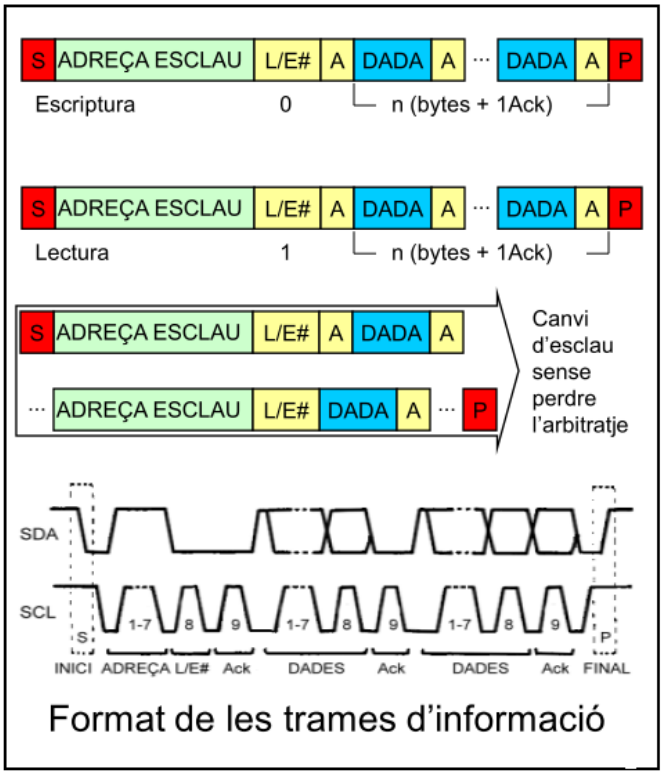


Figura 2.29. Resumen de las secuencias del bus I^2C.

2.4.7 Definición de los bits del primer byte tras START

Consta de 7 bits de dirección y 1 bit de sentido de R/W:

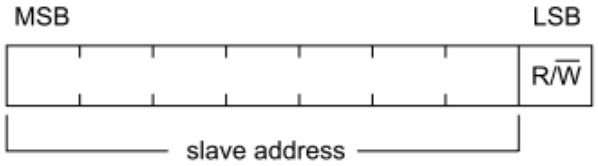


Figura 2.30. Esquema del byte tras START.

Cuando se envía una dirección, todos los dispositivos del sistema comparan esos 7 bits con su dirección. Si alguno coincide, se conectará como Esclavo-Receptor o Esclavo-Transmisor, según esté R/W.

La dirección de un esclavo puede tener dos partes: Fija y Programable (mediante Hardware).

La asignación de direcciones a los dispositivos la realiza el "I^2C-bus Committe".

Hay dos grupos de cuatro direcciones que están reservados para propósitos específicos:

Slave address	R/$\overline{W}$ bit	Description
0000 000	0	general call address[1]
0000 000	1	START byte[2]
0000 001	X	CBUS address[3]
0000 010	X	reserved for different bus format[4]
0000 011	X	reserved for future purposes
0000 1XX	X	Hs-mode master code
1111 1XX	X	reserved for future purposes
1111 0XX	X	10-bit slave addressing

Figura 2.31. Direcciones reservadas del bus I^2C.

Dirección de llamada general (0000 000): en principio es para llamar a todos los dispositivos del bus.

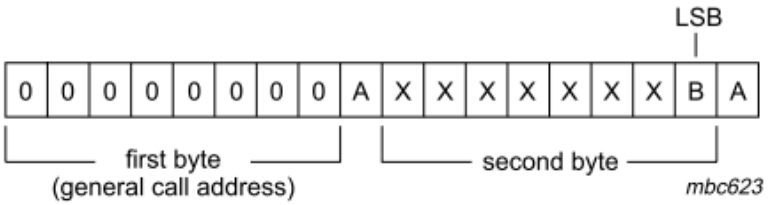


Figura 2.32. Esquema de llamada general en el bus I^2C.

Esta llamada general se utilizada para diversos objetivos: resetear los dispositivos, modificar su dirección (la parte programable), identificación, etc.

El START byte se usa para los casos en que un microcontrolador no dispone de una interfície I^2C. Para no tener que muestrear el bus muchas veces, en estos casos se utiliza un procedimiento de START más largo. La secuencia es: Condición START, START byte, Acknowledge, condición START repetida (Sr).

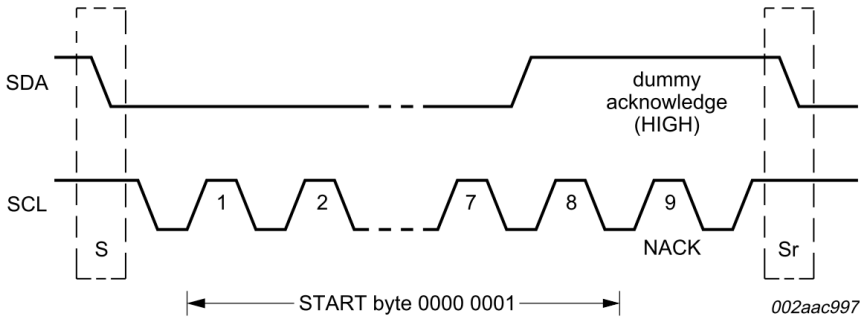


Figura 2.33. Secuencia de START byte.

Después de la condición de START, el microcontrolador sólo ha de detectar uno de los 0 del START byte, y por tanto puede muestrear a menor velocidad. Cuando lo detecte, el microcontrolador puede conmutar a un estado de mayor velocidad para encontrar Sr, que servirá de sincronización.

2.4.8 Extensiones de las especificaciones del bus I²C

Se han realizado dos extensiones:

- Modo rápido: 400 kbits/s.
- Modo rápido Plus: 1 Mbits/s.
- Modo de alta velocidad: 3.4 Mbits/s.
- 10 bits de direcciones: permite direccionar hasta 1024 dispositivos. El formato es el siguiente:

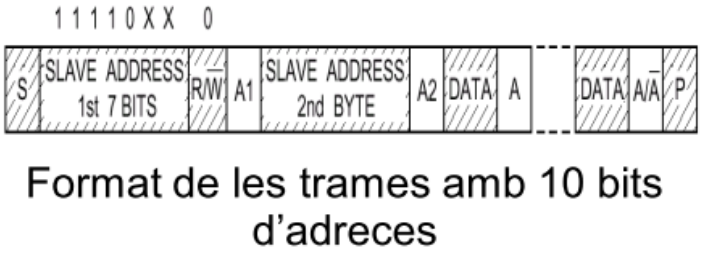


Figura 2.34. Formato del direccionamiento de 10 bits.

Para indicar que se va a indicar una dirección de 10 bits, los primeros bits tienen la codificación 11110, seguido de otros dos bits (XX). La dirección de 10 bits viene dada por XX más los 8 bits del segundo byte.

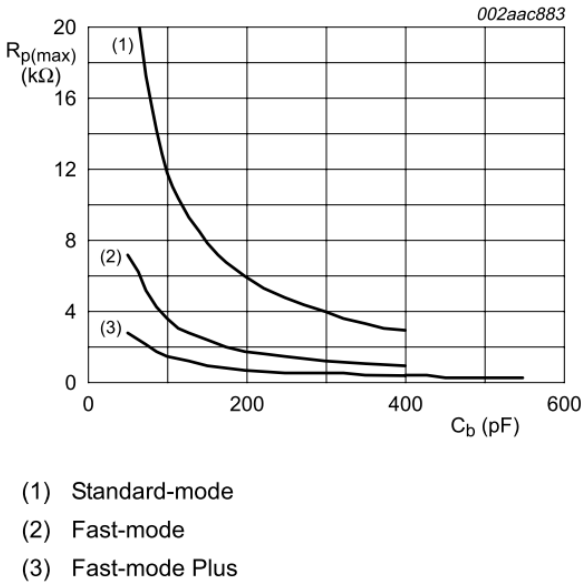
De esta forma, en un mismo bus se pueden mezclar dispositivos con 7 y 10 bits de dirección.

2.4.9 Características eléctricas de los dispositivos para el bus I²C

Como hemos visto, los valores considerados Alto y Bajo dependen de V_{DD}:

- $V_{ILmax} = 0.3 \cdot V_{DD}$.
- $V_{IHmax} = 0.7 \cdot V_{DD}$.

El estándar especifica un máximo de la capacidad del bus (400 pF) (cables, conexiones, pines). Debido a los requerimientos de los tiempos de subida de las señales, el valor máximo de R_P está limitado. Su valor más restrictivo corresponde al caso en que la capacidad del bus es máxima, siendo así un valor de unos 3 kΩ. A menor capacidad, mayor puede ser este valor de resistencia:



(1) Standard-mode
(2) Fast-mode
(3) Fast-mode Plus

Figura 2.35. R_{Pmax} en función de la capacidad del bus (C_b).

$$R_{Pmax} = \frac{t_r}{0.8473 \cdot C_b}$$

El valor máximo de R_P también viene limitado por el valor máximo de la corriente de entrada de cada dispositivo cuando está en nivel alto (10 μA). Esta condición, junto al límite del margen de ruido de $0.2 \cdot V_{DD}$ en el estado alto, determina este máximo:

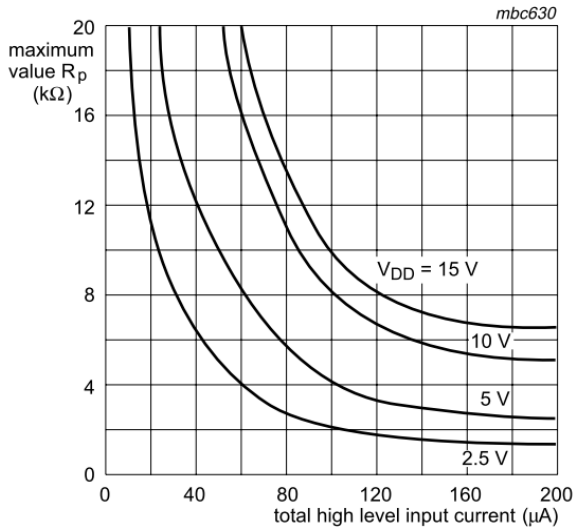
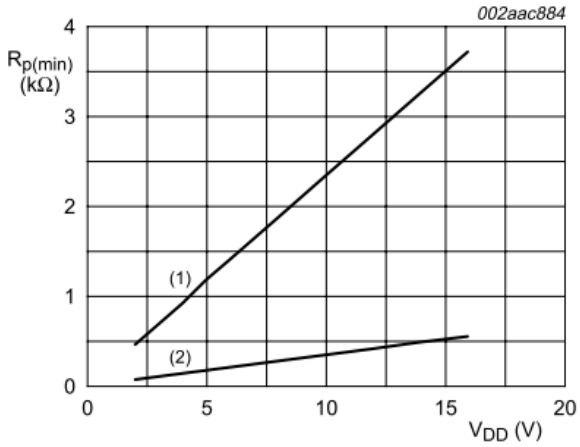


Figura 2.36. R_{Pmax} en función de la corriente de entrada total a los dispositivos en nivel alto.

El máximo valor de V_{DD} limita el valor mínimo de R_P, debido al valor mínimo de corriente de 'sink' (3mA en el modo estándar):



(1) Fast-mode and Standard-mode
(2) Fast-mode Plus

Figura 2.37. R_{Pmin} en función de V_{DD} debido a la mínima corriente de 'sink'.

$$R_{Pmin} = \frac{V_{DD} - V_{OLmax}}{I_{OLmin}}$$

Se pueden colocar resistencias en serie R_S (por ejemplo de unos 300 Ω) para proteger los dispositivos contra picos de alta tensión en las líneas SCL y SDA.

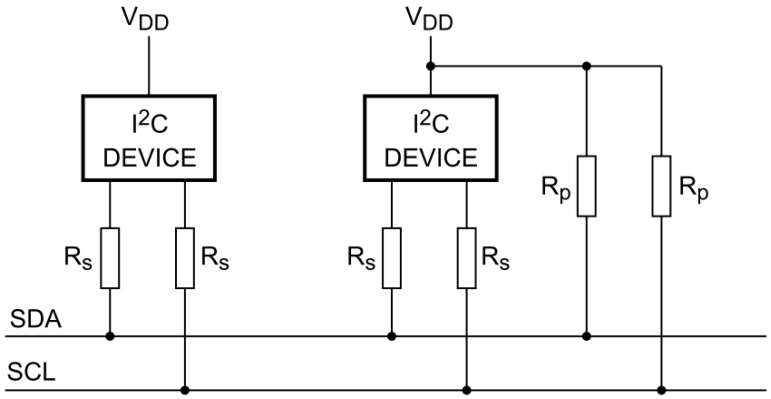


Figura 2.38. Disposición de R_S en el bus I^2C.

Esta resistencia, sin embargo, tendrá que tenerse en cuenta para el cálculo de la máxima capacidad permitida.

El valor máximo está limitado por la restricción en el margen de ruido en el estado bajo de $0.1 \cdot V_{DD}$. (esta resistencia limitará el tiempo de bajada de la salida).

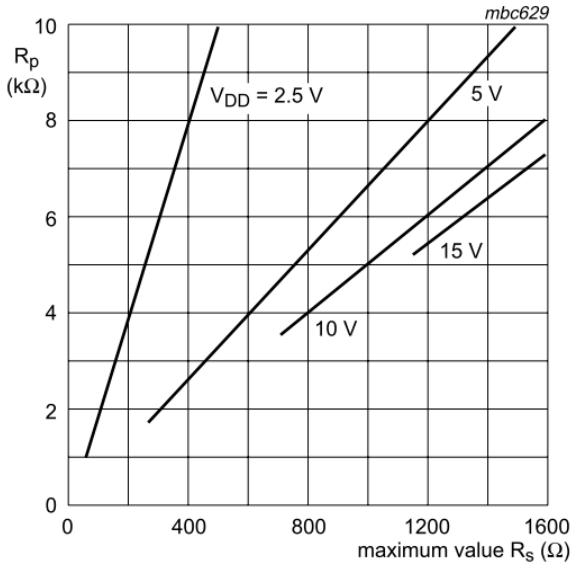


Figura 2.39. R_{Smax} en función de R_p.

(Capacidad máxima de cada "pin" de I/O: 10 pF).

2.4.10 Ejemplos

2.4.10.1 Memoria con bus I^2C

Diagrama de bloques de la memoria EEPROM de 16kbits con bus I^2C:

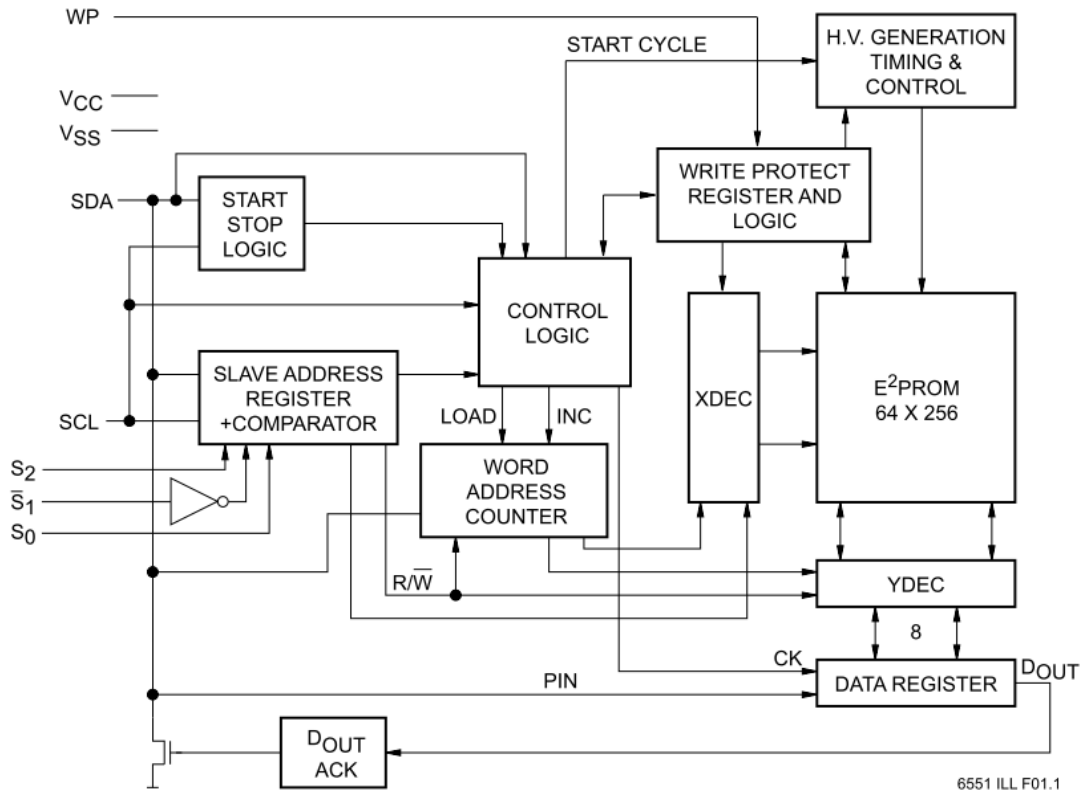


Figura 2.40. Diagrama de bloques de una memoria EEPROM de 16kbits con bus I^2C. (Copyright Intersil Americas Inc. 2005, 2006. All Rights Reserved (www.intersil.com)).

Las entradas de selección S_0, S_1 y S_2 permiten concurrir 8 de estos dispositivos en el mismo bus. La primera operación que realiza es comparar la dirección de esclavo enviada, con la propia del dispositivo. Si coinciden, seguidamente envía un acknowledge.

El registro de datos contendrá tanto los datos de entrada, como los pedidos de la memoria. XDEC e YDEC direccionan la memoria para activar la posición deseada. El dato se transmite a través de YDEC.

Posee un contador de memoria, de forma que mantiene la dirección de la última posición de memoria leída +1, o la última posición escrita.

Dirección del Esclavo:

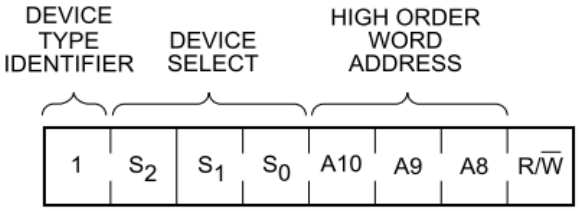


Figura 2.41. Dirección de la memoria como esclavo. (Copyright Intersil Americas Inc. 2005, 2006. All Rights Reserved (www.intersil.com)).

Los bits A_{10}, A_9 y A_8 son una extensión de la dirección de memoria (que se añaden a los 8 bits de la dirección del array) para tener acceso a todo el array de 2048x8 posiciones (la memoria se divide en 8 bloques de 2048 cada uno).

Escritura de un byte:

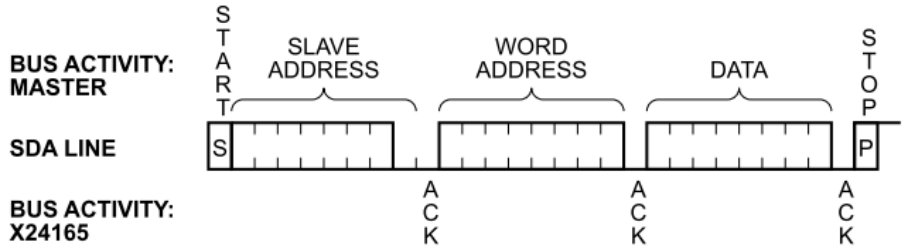


Figura 2.42. Trama para la escritura de un byte en la memoria. (Copyright Intersil Americas Inc. 2005, 2006. All Rights Reserved (www.intersil.com)).

La memoria necesita de una segunda palabra, enviada después del primer acknowledge, para determinar la posición de memoria. Vuelve a responder con un acknowledge y esperar la recepción del dato a almacenar. Si no hubo ningún problema, responde con un acknowledge y el Maestro envía una condición de STOP. En este momento, la memoria inicía las operaciones internas para almacenar el dato. Durante esta operación, la entrada estará inhabilitada y no responderá a ninguna petición del Maestro.

Escritura de una página (32 bytes):

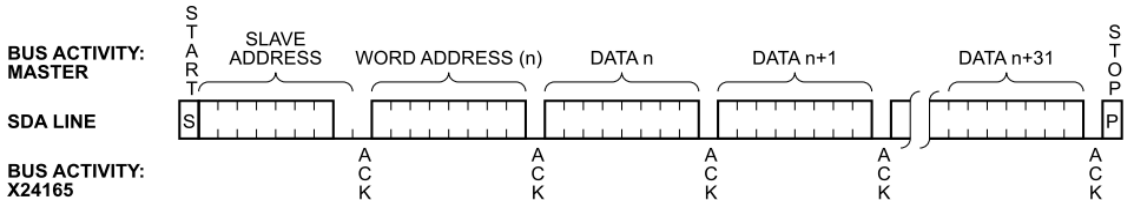


Figura 2.43. Trama para la escritura de una página. (Copyright Intersil Americas Inc. 2005, 2006. All Rights Reserved (www.intersil.com)).

La operación es similar a la de un sólo byte, pero ahora, en lugar de acabar la operación en el primer byte, el maestro sigue enviando bytes, y la memoria realiza un acknowledge para cada byte. Las direcciones son secuenciales respecto la memoria indicada por el Maestro. Al igual que antes, la entrada estará inhabilitada durante la operación interna de escritura, tras la condición de STOP.

Lectura de la posición actual del contador:

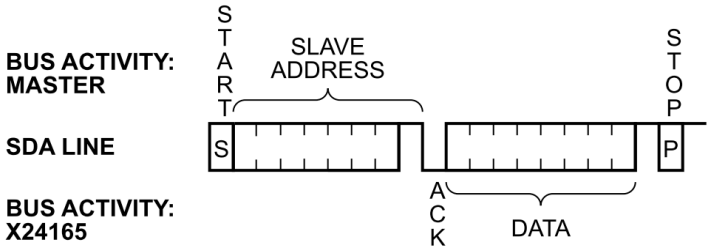


Figura 2.44. Lectura en la posición actual del contador. (Copyright Intersil Americas Inc. 2005, 2006. All Rights Reserved (www.intersil.com)).

La memoria, bajo una petición de lectura (con R/W alto), responde con un acknowledge y devuelve el dato de la memoria en la posición indicada por el contador de memoria. Para finalizar la lecctura, el maestro debe realizar un 'not-acknowledge' y condición de STOP. Al realizar esta lectura, el contador aumenta en una posición.

Lectura de una dirección aleatoria:

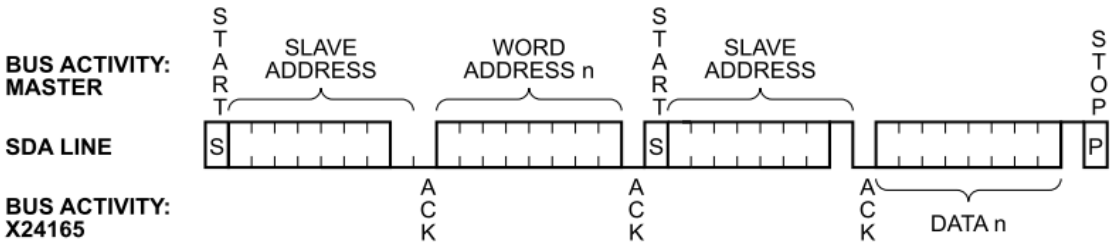


Figura 2.45. Lectura de una dirección aleatoria. (Copyright Intersil Americas Inc. 2005, 2006. All Rights Reserved (www.intersil.com)).

Para este fin, primero se ha de hacer una escritura 'dummy', en el que se indica la dirección que queremos empezar a leer. Después del segundo acknowledge, el Maestro emitirá inmediatamente otra condición de START. Esta vez, se realiza la medida de lectura como se ha especificado para un byte.

Lectura de direcciones consecutivas (secuenciales):

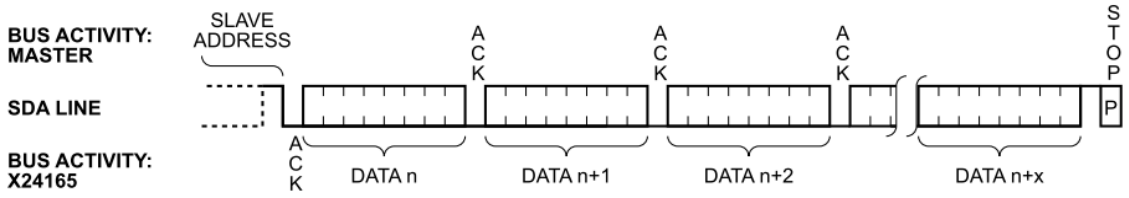


Figura 2.46. Lectura en direcciones consecutivas. (Copyright Intersil Americas Inc. 2005, 2006. All Rights Reserved (www.intersil.com)).

La lectura secuencial se puede hacer tanto partiendo de la dirección actual de la memoria, como empezando por una dirección aleatoria. Para ello se sigue los mismos procedimientos anteriores, sólo que en lugar de responder con una condición de STOP tras la lectura del dato, el Maestro responderá al acknowledge indicando que necesita más datos. La memoria irá leyendo secuencialmente la memoria tras cada acknowledge.

2.4.10.2 Adaptador de puerto paralelo a bus I²C

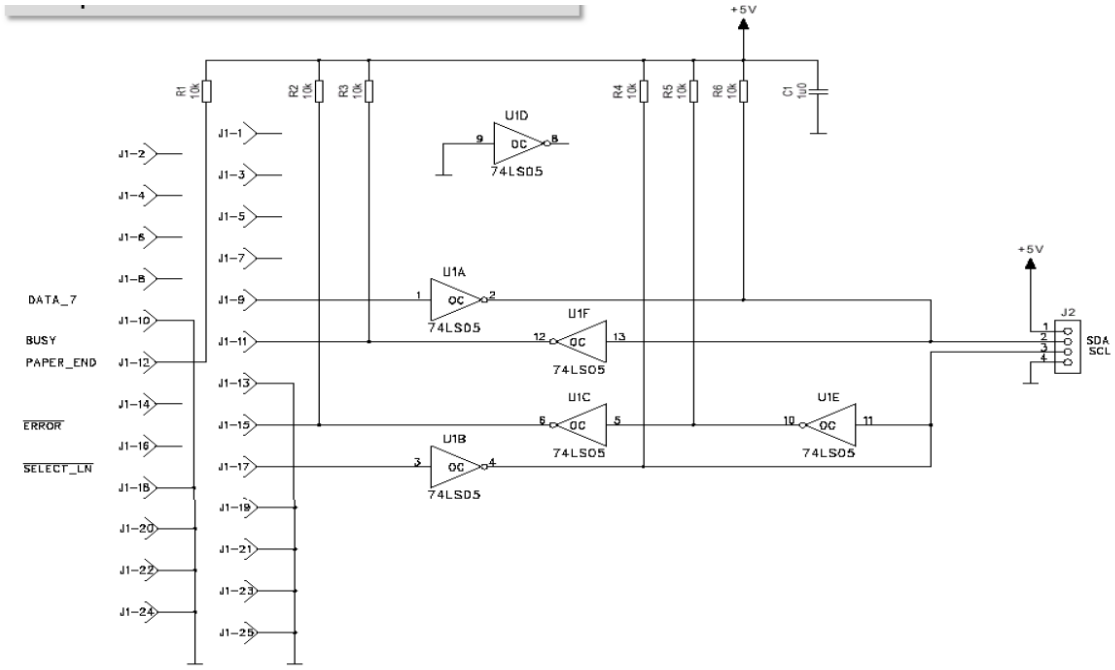


Figura 2.47. Adaptador de puerto paralelo a bus I²C. (Figura obtenida de http://users.tkk.fi/iisakkil/stuff.html).

De esta forma tan sencilla podemos tener una interfaz I²C a partir del puerto paralelo.

Los 74LS05 son inversores, con salida 'open drain', con lógica positiva ('1' es una tensión mayor a la correspondiente al '0'). Estos componentes tienen varias funciones:

- Proporciona aislamiento entre el puerto paralelo y el resto del circuito.

- Convierte las salidas TTL del puerto paralelo a drenador abierto, que es necesario para I^2C.
- Drenador abierto también es necesario para conectar la señal SDA bidireccional a una entrada y una salida dedicadas unidireccionales del puerto paralelo. Lo mismo ocurre con SCL si quiere implementarse 'clock stretching' (control del reloj por parte del esclavo).

Por tanto, SDA se forma mediante una salida (por ejemplo D7, pin 9) y una entrada (por ejemplo Busy, pin 11).

El reloj se genera utilizando también una entrada (nError, pin 15) y una salida (nSelect, pin 17).

Como las salidas son de drenador abierto, necesitan de una resistencia de pull-up.

(Para cumplir con el estándar, se ha de poder leer también la línea. Por ello, se utiliza también un inversor de vuelta.)

2.4.10.3 Adaptador de puerto paralelo a bus I^2C optoaislado

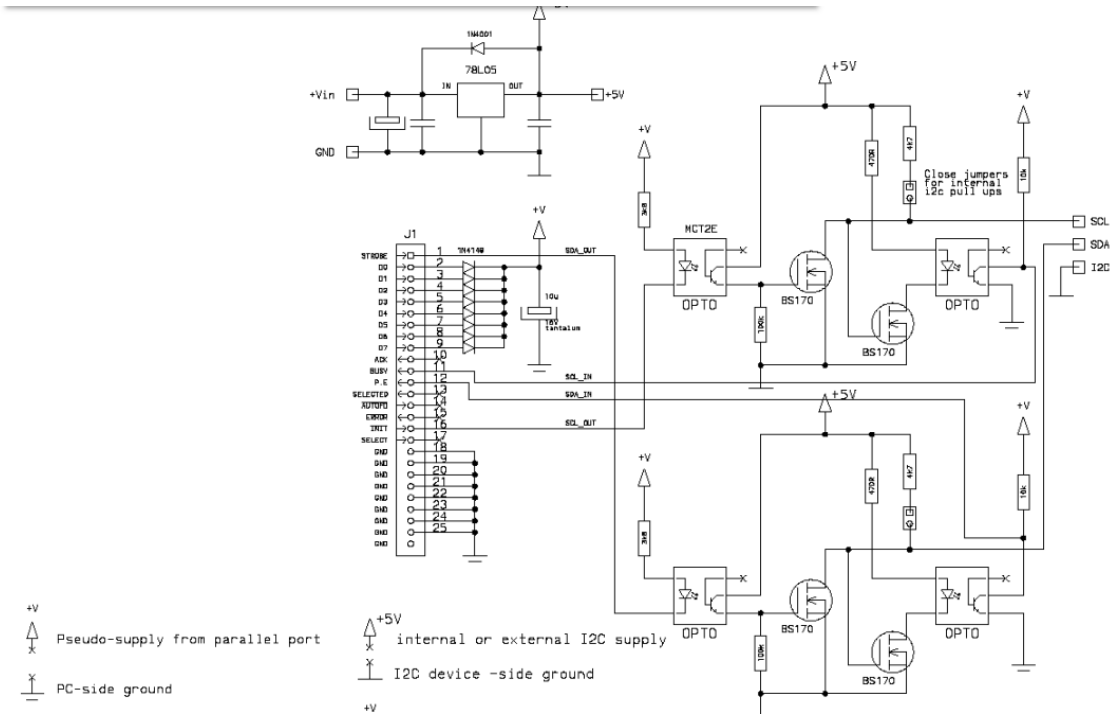


Figura 2.48. Adaptador de puerto paralelo a bus I2C optoaislado. (Figura obtenida de http://users.tkk.fi/iisakkil/stuff.html).

2.4.11 Conexión de dispositivos con diferentes niveles lógicos

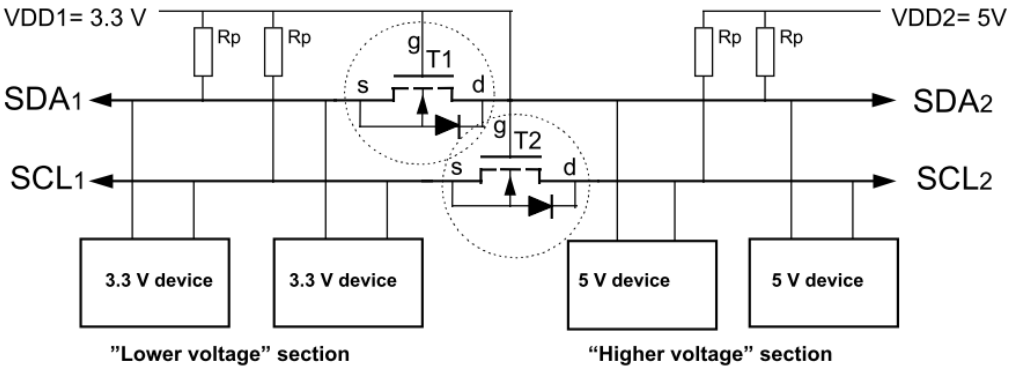


Figura 2.49. Conexión de dos sistemas I^2C con diferentes niveles lógicos.

Una tensión de alimentación de 3.3 V es relativamente común hoy día, al igual que los 5V. Esta representación muestra una posibilidad para adaptar dos redes I^2C alimentadas a dos tensiones diferentes de una forma sencilla. Como las señales SCL y SDA son bidireccionales y no hay un indicador de dirección, no es una opción utilizar 'level-shifters' convencionales.

La solución consiste en interponer N-MOSFETs de enriquecimiento. Para ver su funcionamiento hemos de comprobar todas las posibles combinaciones de tensiones:

- Si los dos líneas están libres (a nivel alto), el MOSFET no conducirá, ya que tenemos V_{GS} = 0 V. Por tanto, los dos sistemas están aislados.
- Si unos de los dispositivos en la línea de 3.3V pone la línea a nivel bajo, el transistores conducirá y, por tanto, la línea del sistema de 5V también se pondrá a nivel bajo.
- Si uno de los dispositivos del sistema de 5 V pone la línea a nivel bajo, en primer lugar conducirá el diodo (poniendo la línea del sistema de 3.3V a un nivel suficientemente bajo como para tener $V_{GS}>V_T$) y seguidamente también se pondrá en conducción el transistor.

Este sistema de conexión tiene también otras ventajas:

- El sistema de tensión menor puede apagarse, dejando al otro sistema totalmente operativo, ya que el transistor siempre estará en corte.
- Proporciona una buena protección al sistema de menor tensión frente a picos de tensión producidos en el sistema de mayor tensión, ya que si la línea aumenta de tensión, también lo hará la fuente hasta que, se cerrará el transistor.

Otro sistema con mejores propiedades viene dado en la siguiente figura:

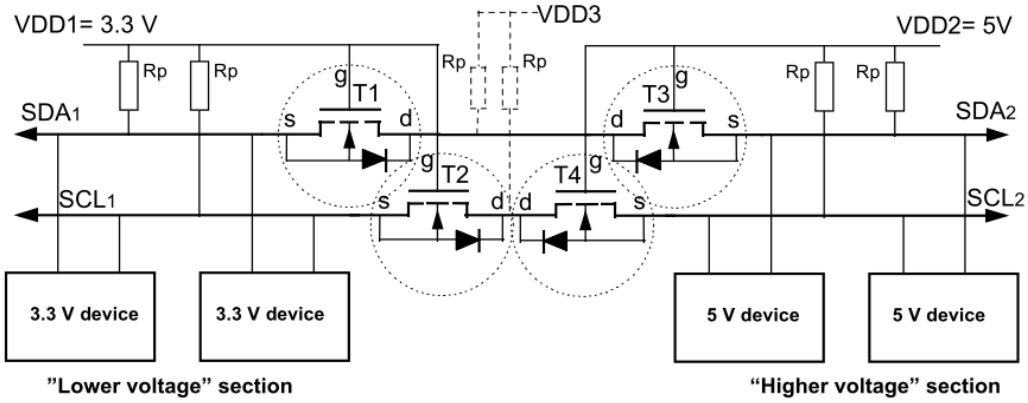


Figura 2.50. Sistema alternativo de conexión de dos sistemas I^2C con diferentes niveles lógicos.

Aquí, el aislamiento se produce en ambos sentidos. Permite un mejor aislamiento entre los sistemas a diferentes tensión (de forma que, al desconectar cualquiera de ellos, no afectará a ninguno de los otros conectados). En el drenador común puede conectarse otras sistemas a distintas tensiones.

2.5 BUS CAN ('CONTROLLER AREA NETWORK')

2.5.1 Introducción

El bus CAN (siglas de 'Controller Area Network') es un protocolo de comunicaciones serie síncrono que soporta control distribuido en tiempo real, con un alto nivel de seguridad. Fue desarrollado por Bosch para aplicarlo en la industria del automóvil. Este sistema permite reducir el cableado necesario debido a la implementación de la multiplexación.

Ejemplos de utilización: (Bosch) en el automóvil → multiplexación de cableado, control del motor, sistemas antideslizantes, sensores, etc.

Puede trabajar hasta a 1Mbit/s.

2.5.2 Concepto del bus CAN

2.5.2.1 Capas

El estándar del bus CAN viene definido por el estándar ISO 11898 (alta velocidad) e ISO 11519-2 (baja velocidad). Nosotros nos centraremos en el primero.

Se divide en tres capas:

- Objeto.
- Transferencia.

- Física.

Las dos primeras comprende todas las funciones y servicios de la capa de enlace de datos del modelo OSI:

- Data Link Layer:
  - LLC ('Logical Link Control') subcapa.
  - MAC ('Medium Access Control') subcapa.

Los objetivos de la capa Objeto son:

- Determinar qué mensajes se han de enviar.
- Decidir qué mensajes recibidos por la capa de transferencia se van a utilizar.
- Proveer una interficie con el hardware relacionado con la capa de aplicación.

La primera y segunda opción constituyen el filtrado de mensajes.

Los objetivos de la capa de Transferencia son:

- Misión: Protocolo de transferencia.
- Controlar las tramas.
- Arbitraje
- Detectar errores.
- Señalar errores.
- Decidir si el bus está libre para enviar o cuando comienza una recepción.
- Temporización.

La capa física define cómo los datos son realmente transmitidos. Sus objetivos son:

- Trasferencia real de bits entre distintos nodos, con unas determinadas características eléctricas.

Existe mucha libertad en cuanto a la capa física aunque, por supuesto, en un caso determinado real todos los nodos han de tener la misma capa física

2.5.2.2 Propiedades

- Priorización de mensajes.
- Garantía de tiempos de latencia.
- Flexibilidad de configuración.
- Recepción multicast con sincronización.

- 'Wide data consistency' del sistema.
- Multimaestro.
- Detección de error y señalización.
- Retransmisión automática de mensajes corruptos una vez el bus se pone inactivo.
- Distinción entre errores temporales y fallos permanentes de nodos. Apagado autónomo de nodos defectuosos.

2.5.2.3 Mensajes

La información se envía mediante mensajes de formato fijo, de longitud variable pero limitada.

Cuando el bus está libre, cualquier dispositivo conectado al bus puede comenzar a transmitir un nuevo mensaje.

2.5.2.4 Información del routeado

En este sistema, los nodos no utilizan información sobre la configuración del sistema (por ejemplo, de las direcciones de las estaciones). Las consecuencias de este hecho son:

- Mayor flexibilidad del sistema: Se pueden añadir o quitar nodos al sistema sin necesidad de ningún cambio de hardware o software en los otros nodos.
- Ruteado de los mensajes: Cada mensaje se etiqueta mediante un Identificador, el cual no indica el destino del mensaje, sino que describe el significado de los datos. Así, cada nodo de la red es libre de decidir mediante filtrado de mensajes si le interesan o no esos datos.
- 'Multicast': Debido al filtrado de mensajes, cualquier número de nodos puede leer un mismo mensaje.
- Consistencia de los datos: El sistema garantiza que los mensajes puedan ser aceptados por cualquier número de nodos. Por tanto, la consistencia se asegura por la multidifusión ('Multicast') y por el manejo de errores realizado por distintos nodos.

2.5.2.5 Velocidad

Puede variar de un sistema a otro, pero para un sistema determinado, la velocidad de transferencia es fija. Como máximo será de 1Mbit/s.

2.5.2.6 Prioridad

El Identificador define un sistema de prioridades durante el acceso al bus.

2.5.2.7 Petición remota de datos

Mediante el envío de un 'Remote Frame', un nodo puede pedir datos a otro nodo. Este le enviará un 'Dataframe' con los datos pedidos.

El Identificador del 'Dataframe' y el del 'Remote Frame' han de ser iguales.

2.5.2.8 Multimaestro

Cuando el bus está libre, cualquier nodo puede comenzar a transmitir un mensaje. El nodo con el mensaje a transmitir de mayor prioridad ganará el acceso al bus. Nosotros seremos capaces de fijar la prioridad según el tipo de mensaje.

2.5.2.9 Arbitraje

El mecanismo de arbitraje garantiza que no hay pérdidas de información ni de tiempo.

Si se envían simultáneamente un “Dataframe” y un “Remoteframe” con el mismo Identificador, gana el arbitraje el “Dataframe”.

Durante el período de arbitraje, cada transmisor compara el nivel del bit transmitido con el que monitoriza en el bus. Si los niveles son iguales, el nodo continúa transmitiendo. Cuando se envía un nivel “Recesivo” y se monitoriza un nivel “Dominante”, el nodo deja de transmitir; pierde el arbitraje.

(En cierta forma ,es igual que el arbitraje en I^2C).

2.5.2.10 Seguridad

En cada nodo se implementan una serie de medidas (potentes) para detectar, señalar y auto-detectar errores, para conseguir una gran seguridad de los datos transmitidos.

Para detectar errores:

- Monitorización (los transmisores van comparando los niveles del bus con los emitidos).
- Chequeo cíclico de redundancia (CRC: 'Cyclic Redundancy Check').
- 'Bit Stuffing': Insertar bits específicos en posiciones determinadas).
- 'Message Frame Check'.

2.5.2.11 Señalización de errores y tiempo de recuperación

Los mensajes corrompidos son señalizados por cualquier nodo que detecte el error. Dichos mensajes son detenidos y se retransmitirán automáticamente.

El tiempo desde que se detecta un error y en que comienza el nuevo mensaje es como máximo de 29 intervalos de bit.

2.5.2.12 Confinamiento de fallos

Los nodos CAN han de poder distinguir entre errores espúreos y permanentes. Los nodos defectuosos se apagan.

2.5.2.13 Conexiones

El bus serie de conexión utilizado por el CAN puede tener un número de nodos que, teóricamente, no tiene límite. En la práctica, el límite viene dado por los tiempos de retardo y la carga eléctrica de la línea.

2.5.2.14 Canal simple

El bus consiste en un único canal simple que transporta bits.

El método que se utiliza para implementar este canal no queda especificado en la norma:

- 'Single Ended' (+masa).
- Dos hilos diferenciales.
- Fibra óptica.

2.5.2.15 Valores en el bus

El bus puede tomar uno de dos valores lógicos posibles:

- Dominante.
- Recesivo.

Si se envían simultáneamente al bus un bit Dominante y otro Recesivo, prevalecerá el Dominante.

Ejemplo: En una implementación Wire-AND, 0 sería dominante, y 1 sería recesivo.

El estándar no define ningún nivel de tensión para estos valores lógicos.

2.5.2.16 Reconocimiento (‘Acknowledge’)

Todos los receptores comprueban la consistencia del mensaje recibido, y reconocerán (acknowledge) un mensaje consistente y señalarán uno inconsistente.

2.5.2.17 Modo ‘Sleep’/’Wake-up’

Con el fin de reducir el consumo del sistema, un dispositivo CAN puede configurarse en modo 'Sleep', sin actividad interna y con los drivers de bus desconectados. Esta situación finaliza cuando hay actividad en el bus, o cuando así lo decida el mismo dispositivo. Para poder despertar a otros nodos, se puede utilizar como 'Wake-up' el mensaje con identificador de menor prioridad:

rrr rrrd rrrr

(donde r = recesivo y d = dominante)

2.5.3 Transferencia de mensajes

En este bus, hay cuatro tipos de tramas:

- Trama de datos ('Data Frame'): Transporta datos desde un transmisor a los receptores.
- Trama remota ('Remote Frame'): Es transmitida por un nodo para pedir la transmisión, por parte de otro nodo, de un 'Data Frame' con el mismo identificador.
- Trama de error ('Error Frame'): Lo transmite cualquier nodo que detecte un error en el bus.
- Trama de saturación ('Overload Frame'): Utilizado para conseguir un retardo extra entre un 'Data Frame' o 'Remote Frame' y el siguiente.

Las tramas de datos y remotas se separan de la precedente mediante un 'Interframe Spacing' (espaciado entre tramas).

Un nodo sólo está permitido a enviar una trama si el bus está inactivo ('idle').

2.5.3.1 Trama de datos

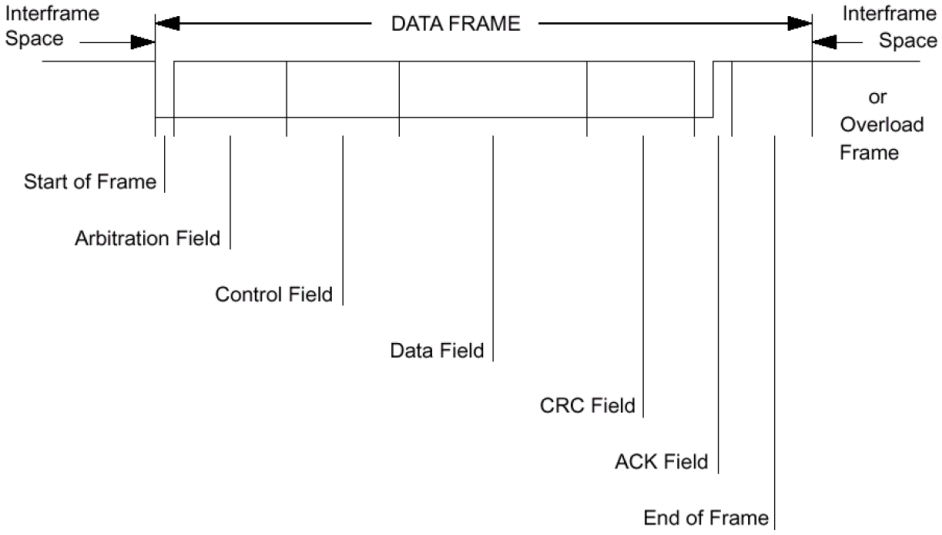


Figura 2.51. Esquema de la trama de datos en el bus CAN.

Se compone de siete campos:

- Comienzo de la trama ('Start of Frame'): Marca el principio de las tramas de datos y remotas. Es un único bit dominante. Sirve para sincronizar todas las estaciones cuando una comienza a transmitir.
- Campo de arbitraje ('Arbitration Field'): Se compone de dos partes:
  - Identificador: Longitud de 11 bits. Primero se transmite el más significativo. Los 7 más significativos no pueden ser todos recesivos.
  - Bit RTR ('Remote Transmission Request'): Distingue si la trama es de datos o remota:
    - Dominante: de datos.
    - Recesivo: remoto.

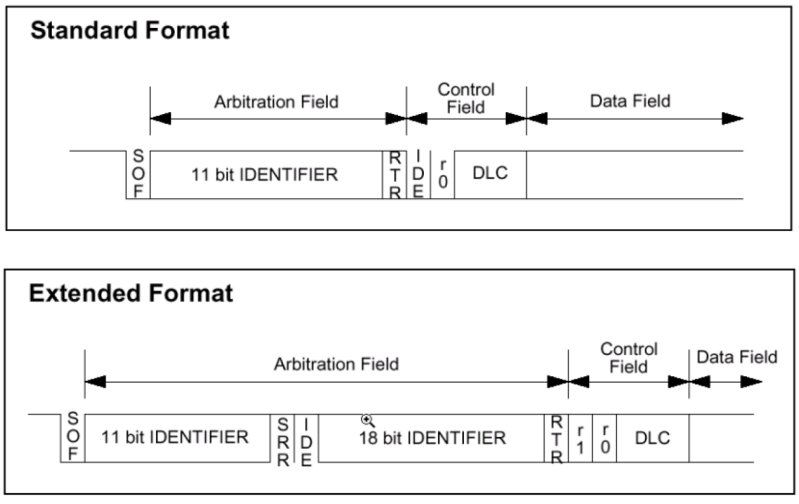


Figura 2.52. Campos de las tramas en formato entrada y extendido.

- Campo de control ('Control Field'): Se compone de 6 bits:
  - 4 bits en los que se indica el código de la longitud del datos ('Data Length Code').
  - 2 bits están reservados: Estos han de ser dominantes.

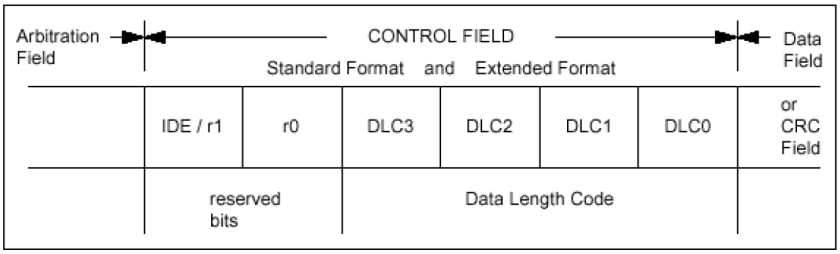


Figura 2.53. Campo de control.

Number of Data Bytes	Data Length Code			
	DLC3	DLC2	DLC1	DLC0
0	d	d	d	d
1	d	d	d	r
2	d	d	r	d
3	d	d	r	r
4	d	r	d	d
5	d	r	d	r
6	d	r	r	d
7	d	r	r	r
8	r	d	d	d

Figura 2.54. Código DLC.

- Campo de datos ('Data Field'): Datos a transmitir. Puede ir de 0 a 8 bytes. Para cada byte, el MSB se envía primero.
- Campo CRC ('CRC Field'): Contiene la secuencia de CRC (de 15 bits), seguido de un delimitador de CRC (que consiste en un único bit recesivo). (para la detección de errores).

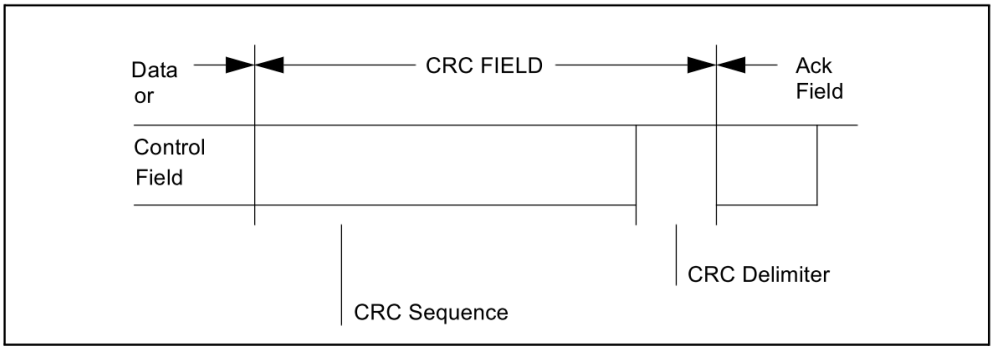


Figura 2.55. Campo CRC.

- Campo de ACK ('ACK Field'): Tiene dos bits de longitud. Contienen el 'ACK Slot' y el 'ACK Delimiter'. El transmisor pone dos bits recesivos. Si el receptor ha recibido el mensaje correctamente (es decir, coinciden la secuencia de CRC), pondrá un bit dominante en el 'ACK Slot'.

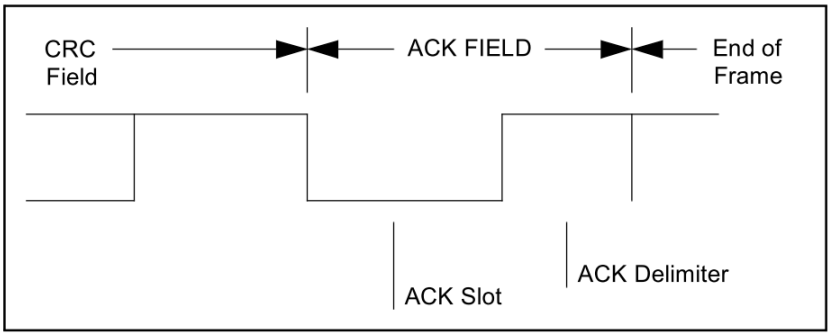


Figura 2.56. Campos ACK.

- Fin de trama ('End of Frame'): Consiste en 7 bits recesivos consecutivos.

2.5.3.2 Trama remota (‘Remote Frame’)

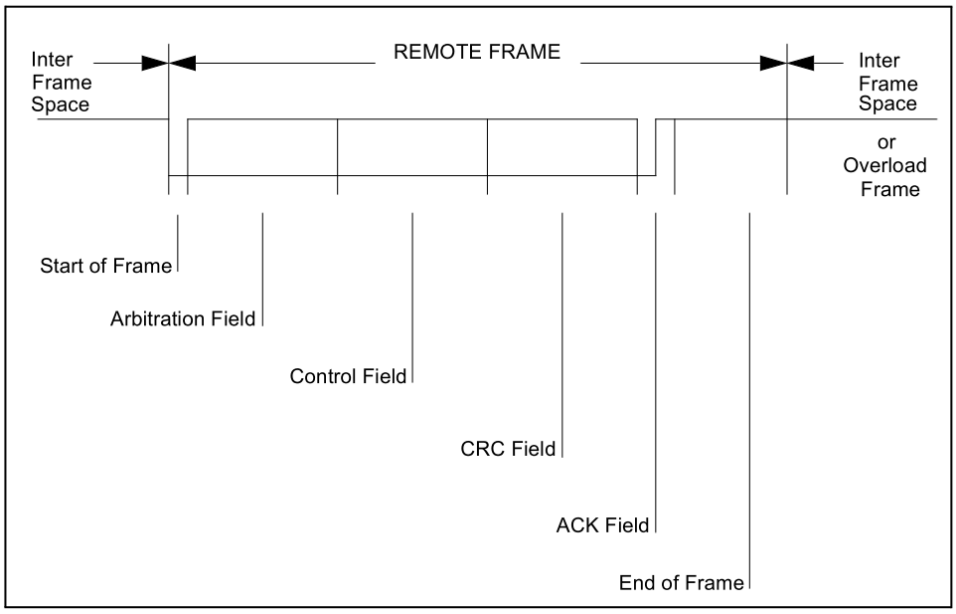


Figura 2.57. Trama remota.

La utiliza una estación que, actuando como receptora, quiere que otro nodo inicíe la transmisión de estos datos.

Es igual que la trama de datos, pero con dos diferencias:

- El bit RTR es recesivo. Es el bit que indica que es una trama remota.
- Sólo tiene 6 campos. No existe el campo de datos.

2.5.3.3 Trama de error

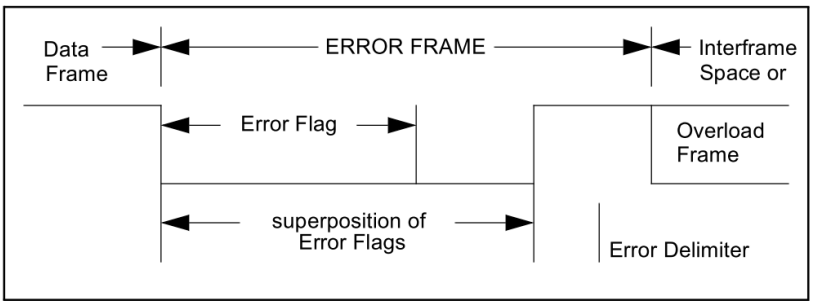


Figura 2.58. Trama de error.

Contiene dos campos:

- Indicadores de error ('Error Flags'): Es una superposición de los indicadores de error de las diferentes estaciones. Tiene una longitud entre 6 y 12 bits.
- Delimitador de error ('Error Delimiter'): Consiste en 8 bits recesivos.

Error Flags: Hay dos tipos:

- Activo: Seis bits dominantes consecutivos.
- Pasivo: Seis bits recesivos consecutivos (a no ser que que alguno de ellos esté sobreimpuesto por un bit dominante de otro dispositivo).

Los 'Errors Flags' violan la ley de 'bit stuffing'. Eso provoca que los demás dispositivos detecten el error y comienzan la emisión de 'Errors Flags'. Por tanto, se observará una superposición de todas las 'Errors Flags'.

2.5.3.4 Trama de saturación (‘Overload Frame’)

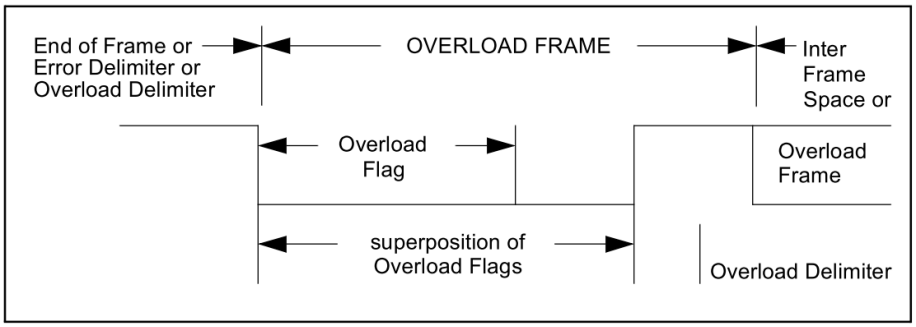


Figura 2.59. Trama de saturación.

Hay dos tipos de condiciones de saturación:

- Cuando un receptor necesita un retardo para la siguiente trama de datos o remota. Sólo puede empezar en el primer pulso de una 'Intermission' esperada.

- Cuando se detecta un bit dominante durante una 'Intermission'. Empieza un pulso después de detectar este bit dominante.

Como máximo pueden generarse dos tramas de saturación para retrasar el siguiente dato o trama remota.

Contiene dos campos de un byte cada uno de longitud:

- Indicador de saturación ('Overload Flag'): Consiste en una superposición de indicadores de saturación (que son de 6 bits) de los diferentes dispositivos.
- Delimitador de saturación ('Delimiter Flag'): Consiste en 8 bits recesivos.

2.5.3.5 *Espaciado entre tramas ('Interframe Spacing')*

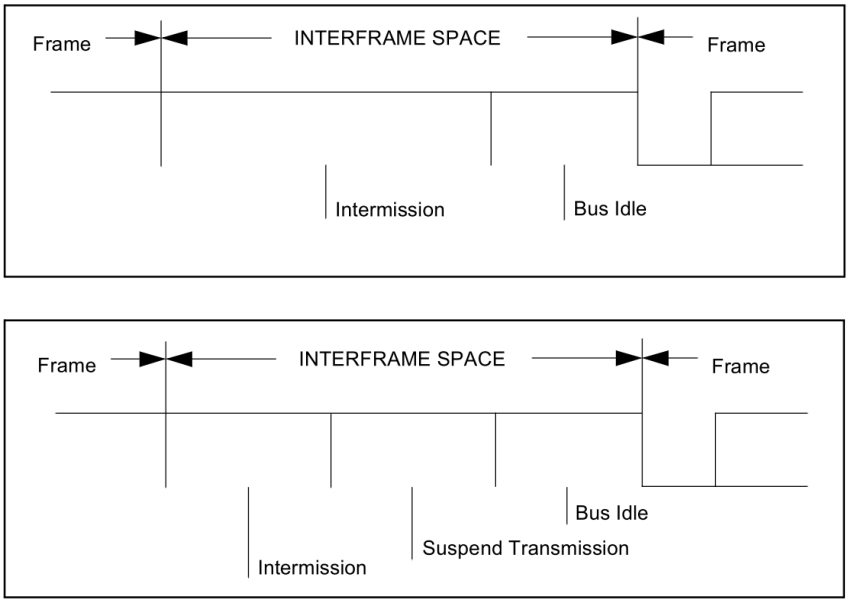


Figura 2.60. Espaciado entre tramas.

Las tramas de datos y remotas han de separarse de las tramas precedentes por un campo de bits llamado 'Interframe Spacing'. (Las de errores y saturación no lo necesitan).

Se compone de dos campos de bits:

- 'Intermission': Consiste en tres bits recesivos. Durante 'Intermission' sólo se permiten tramas de saturación.
- 'Bus Idle': Permite reconocer cuando el bus está libre. La aparición de un bit dominante en el bus se considera como un comienzo de trama.

Para estaciones 'error passive' que han sido transmisores del mensaje previo, existe otro campo llamado 'Suspend Transmission'. Se utiliza en el caso que, antes de enviar un nuevo mensaje, otra estación envía antes otro mensaje, pasando esta estación a ser receptora. (segunda figura).

2.5.3.6 Codificación del flujo de bits

Los segmentos 'Start of Frame', 'Arbitration Field', 'Control Field', 'Data Field' y 'CRC Sequence' están codificados mediante el método de 'bit stuffing'. Cuando un trasmisor detecta 5 bits consecutivos de valor idéntico, inserta automáticamente un bit complementario.

Tanto los campos 'CRC Delimiter', 'ACK Field' y 'End of Frame', como las tramas de error y de saturación son de forma fija (y por tanto no son 'stuffed').

Por lo demás se utiliza codificación NRZ ('Non-Return-to-Zero'). Por tanto, durante todo el 'bit time', el nivel del bit generado se mantendrá igual.

2.5.4 Ejemplos

2.5.4.1 Controlador del bus CAN (SJA1000)

Diagrama de bloques:

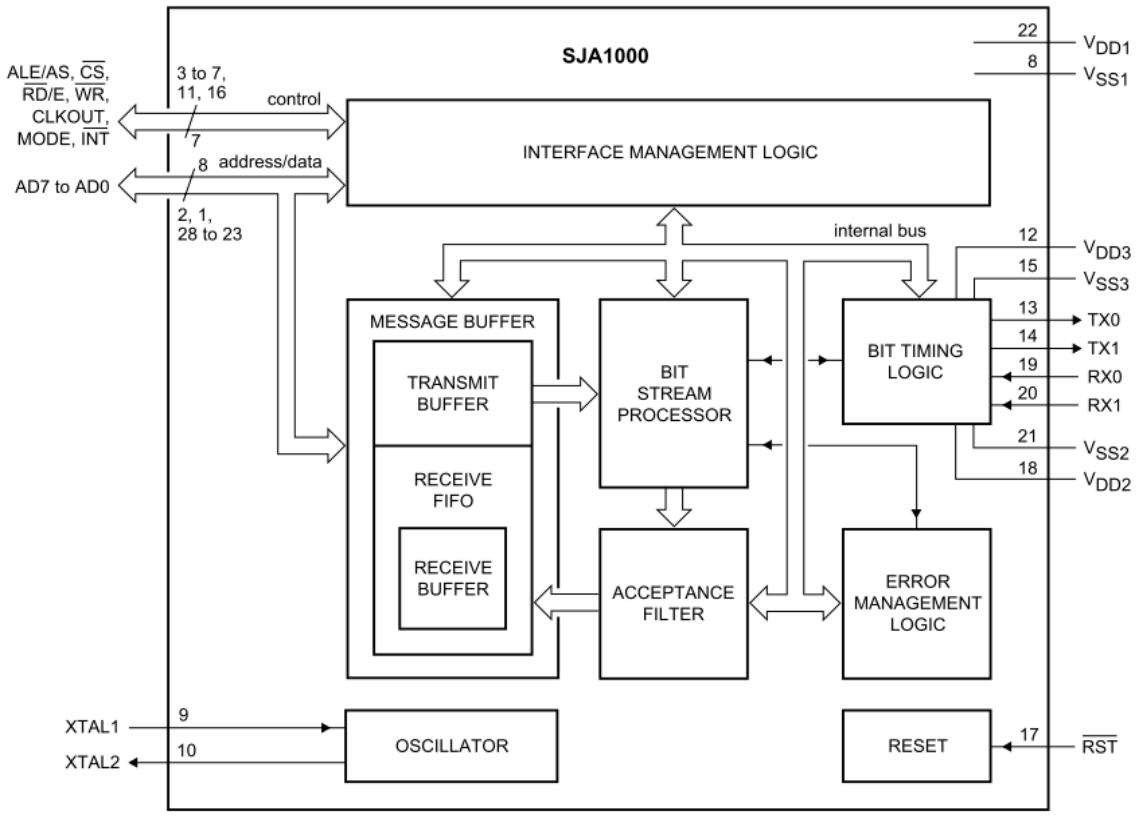


Figura 2.61. Diagrama de bloques del controlador de bus CAN SJA1000. (Figura obtenida de Philips (www.philips.com)).

La 'interface management logic' interpreta comandos de la CPU, controla el direccionamiento de los registros, proporciona interrupciones, información de estado, etc.

El buffer de transmisión almacena un mensaje completo para ser transmitido por el bus, a través de TX. Para ello, la CPU guarda este mensaje, mientras que es el BSP (Bit Stream Processor) el que lo lee para transmitirlo. El BSP es un secuenciador que controla el flujo de datos entre el buffer de transmisión y el bus.

El buffer de recepción es donde se almacenan los mensajes recibidos y aceptados. Posee un cierto tamaño para que la CPU puede procesar un mensaje, mientras puede estar recibiendo otros.

El BTL ('Bit Timing Logic') monitoriza el bus y controla la temporización del bus.

El buffer de recepción es una parte de una memoria FIFO en el que puede almacenarse 64 bytes.

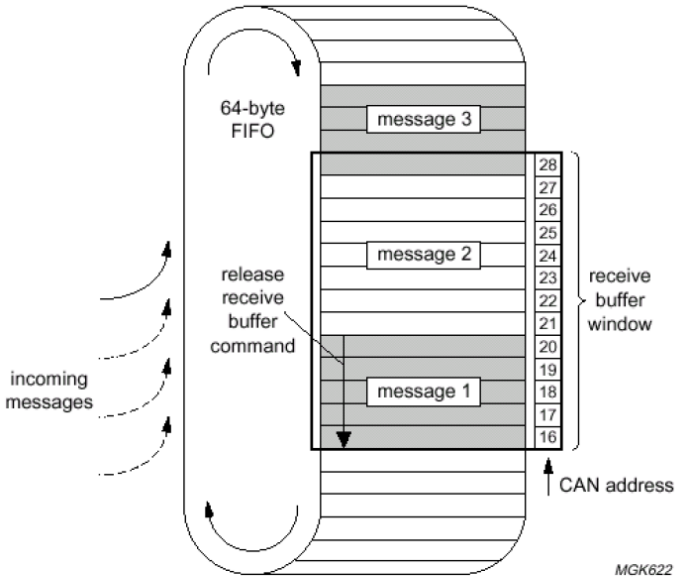


Figura 2.62. Memoria FIFO de recepción de mensajes. (Figura obtenida de Philips (www.philips.com)).

Para que un dispositivo acepte un mensaje, existe un proceso de filtrado. En una trama estándar, se usa todo el identificador y los dos primeros bytes de datos. También pueden aceptarse mensajes que no contienen datos (al estar el bit RTR activo), o que sólo tengan un dato (el código de longitud). En este filtrado, todos los bits han de coincidir para aceptar el mensaje.

El filtro está definido en los 'Acceptance Code Registers' (contiene realmente los patrones de bits que pueden ser aceptados) y los 'Acceptance Mask Registers' (define algunas posiciones de bits que son 'don't care').

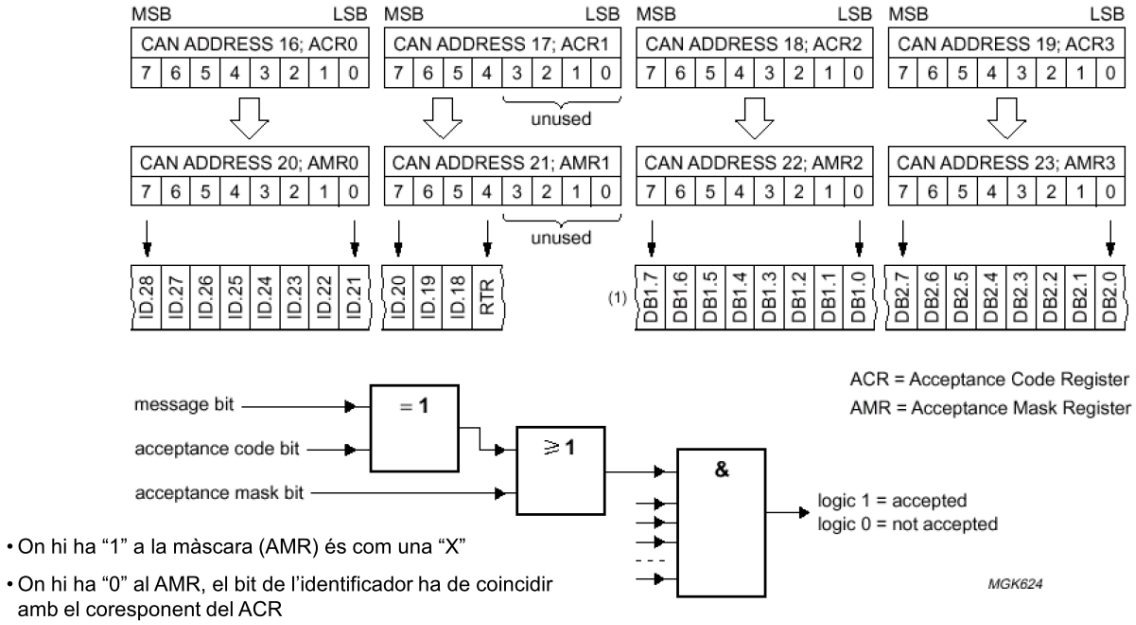


Figura 2.63. Filtro de aceptación de mensaje. (Figura obtenida de Philips (www.philips.com)).

2.5.4.2 Transceptor por bus CAN (PCA82C250/51)

El componente PCA82C250/251 proporciona la interfaz entre un controlador del bus CAN y la implementación física del bus.

También puede realizar con aislamiento galvánico, mediante optoacopladores.

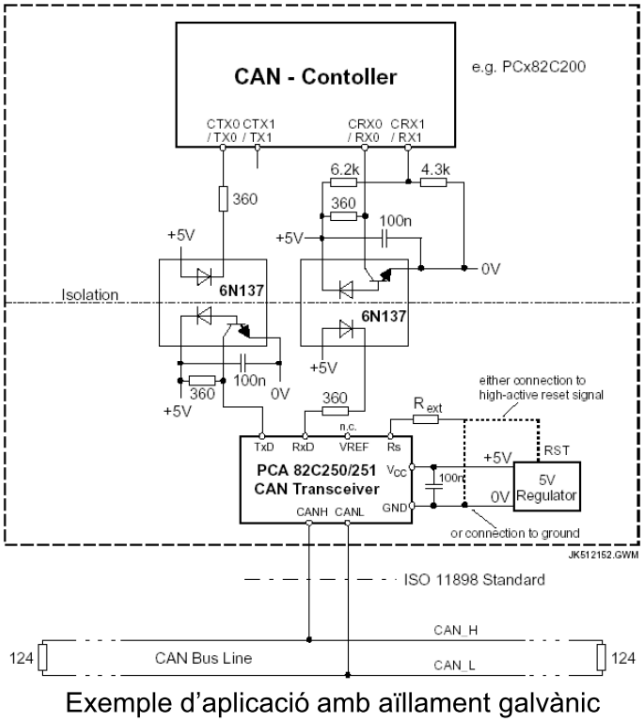


Figura 2.64. Transceptor con aislamiento galvánico. (Figura obtenida de NXP/Philips (www.nxp.com)).

Nivel de tensiones para estos drivers:

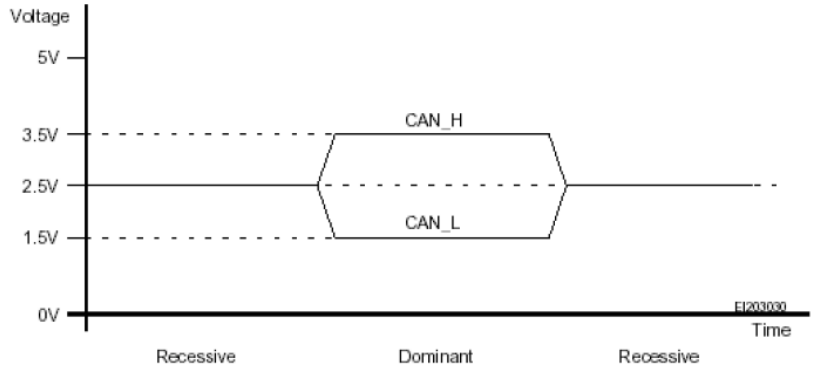


Figura 2.65. Niveles de tensión en el bus para estos transceptores.

Las velocidades máximas de transmisión dependerán de la longitud del bus:

Bit Rate (kbit/s)	Bus Length (m)
1000	30
500	100
250	250
125	500
62.5	1000

Figura 2.66. Velocidades de transmisión.

Cable Type	L_{max} (k_{sm} = 0,2)[1]			L_{max} (k_{sm} = 0,1)[2]		
	n = 32	n = 64	n = 100	n = 32	n = 64	n = 100
DeviceNet™ (thin cable) and/or ISO 11898 cable	200 m	170 m	150 m	230 m	200 m	170 m
DeviceNet™ (thick cable)	800 m	690 m	600 m	940 m	810 m	700 m
0.5 mm^2 (or AWG 20)	360 m	310 m	270 m	420 m	360 m	320 m
0.75 mm^2 (or AWG 18)	550 m	470 m	410 m	640 m	550 m	480 m

Figura 2.67. Máxima longitudes del cable en función del tipo y del número de nodos.

2.5.4.3 Conexión entre SJA1000 y PCA82C250/51

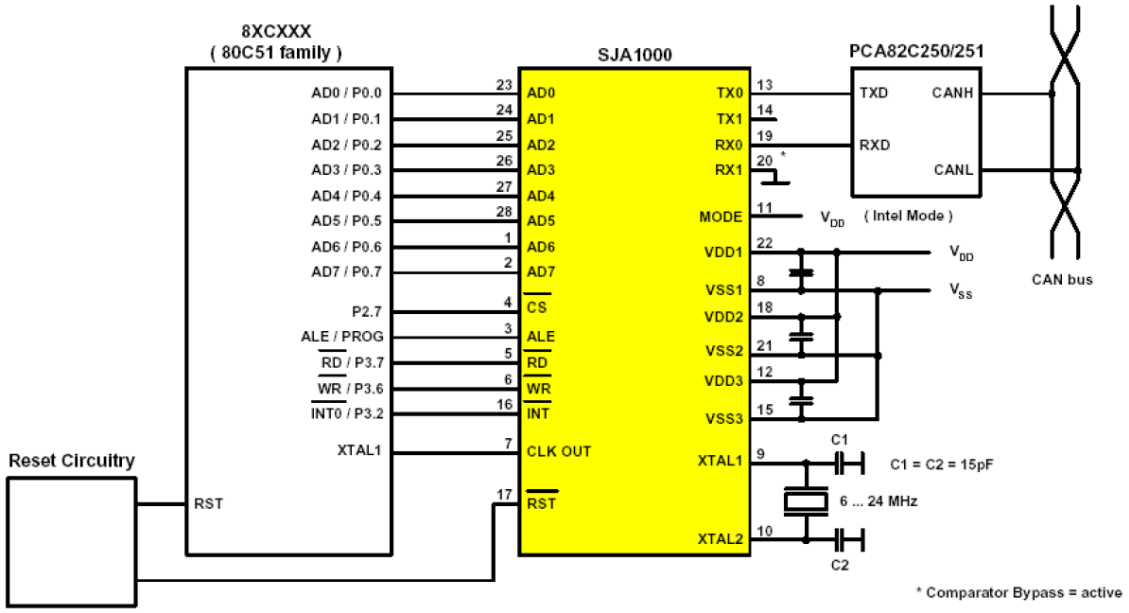


Figura 2.68. Conexión entre SJA1000 y PCA82C250/51. (Figura obtenida de NXP/Philips (www.nxp.com)).

El 80C51 es un microcontrolador.

El controlador CAN funciona como un generador de reloj, mientras que la señal de reset es externa.

3. INSTRUMENTACIÓN INTELIGENTE

3.1 INTRODUCCIÓN

Existen enormes dificultades a la hora de integrar diferentes tipos de sensores y actuadores (transductores) en una misma red. Las causas son varias:

- Existen muchos tipos diferentes de transductores y con distintos tipos de señales:
  - Bucle de corriente 4-20 mA.
  - Niveles de tensión: 0-5V, 0-12V, …
  - Salida en frecuencia.
  - Etc.

POSSIBLES SORTIDES DELS SENSORS			
TIPUS	Paràmetre Variable	Comentaris	Exemples
Passiu	Resistència		Pressió, Galgues
Passiu	Capacitat		Humitat
Passiu	Inductancia		LVDT
Actiu	Tensió	Molts Rangs (μV-V)	Termoparell
Actiu	Intensitat	Molts rangs (el més típic 4-20mA)	AD590 (temperatura)
Actiu	Freqüència		
Actiu	Càrrega	Necessiten un amplificador de càrrega	Piezoelèctrics
Digital	On/Off		Termòstat
Digital	Polsos	Rang molt ampli	Humitat, acceleròmetres.

Figura 3.1. Diferentes tipos de salidas de sensores.

- Existen muchas redes de control distribuido (CAN, LonWorks, etc.). (típicamente, el μP se encarga del protocolo de la red, además de realizar las medidas y actuar (filtrar, etc.)).

Diferentes buses de campo (aquellos usados en procesos industriales para facilitar la instalación y control de máquinas) que se utilizan son:

BUSOS DE CAMP UTILITZATS A LA INDÚSTRIA					
BUS	BITBUS	SDS	PROFIBUS	LonWorks	DeviceNet
Promotor	Intel	Honeywell	DIN Aleman	Echelon	Allen-Bradley
Velocitat	375Kbits/s	500kbits/s	500kbits/s	125Mb/s	1Mbit/s
Nº Nodes màx.: - amb repetidors -sense repetidors	 250 32		 127 32	milers	
Màx. distància: - amb repetidors - sense repetidors	 13.2km 1.2km	0.5km	 0.8km 0.2km	0.5km	0.5km
Medi físic	Parell trenat	Parell trenat	Parell trenat (RS485)		
Arbitratge	Mestre/ Esclau	Tipus CAN	Pas per Token		Tipus CAN
Principals aventatges	Molt utilitzat	Molt efectiu p-p Basat CAN	Determinista Potent Missatges		Sistema obert Basat en CAN
Aplicacions	E/S intel·ligent Control distribuït	Adquisició Remota de dades	Inter PLC-PC Automatització Industrial	Automatització Indus. Edificis Intel·ligents	Automat. Industrial Control distribuït

Figura 3.2. Propiedades de buses de campo utilizados en la industria.

Buses en la automatización de edificios:

BUSOS UTILILITZATS A L'AUTOMATITZACIÓ D'EDIFICIS	
Protocol	Promotor
BACnet	Building Automation Industry
LonTalk	Echelon Cosp.
IBIbus	Intelligent Building Institute
Batibus	Merlin Gerin (France)
EIBus	Alemanya
CAB	Canada

Figura 3.3. Buses utilizados en la automatización de edificios.

Buses en la automatización de casas (para domótica):

BUSOS UTILILITZATS A L'AUTOMATITZACIÓ DE CASES	
Protocol	Promotor
X-10	X-10 Corp.
Smart House	Smart House L.P.
CEBus	EIA
LonTalk	Echelon Corp.
TTP	Universitat de Viena
HS (Home Systems)	Europa

Figura 3.4. Buses utilizados en domótica.

Buses utilizados en automoción:

BUSOS UTILILITZATS AL AUTOMÒVIL	
Protocol	Promotor
J-1850	Society of Automotive Enginners (SAE)
J-1939 (CAN)	SAE
J-1567 (C2D)	SAE (Chrysler)
J2058 CSC	Chrysler
J2106	SAE (GM)
CAN	Robert Bosch GmbH
VAN	ISO (Renault)
A-Bus	Volkswagen AG
D2B	Philips
MI-Bus	Motorola
TTP	Universitat de Viena, Àustria

Figura 3.5. Buses utilizados en el automóvil.

El fabricante de transductores puede actuar de tres formas:

- Fabricar transductores que proporcionen señales de los tipos estándar o más habituales.
- Desarrollar interfícies propias entre los transductores y el µP que soporta el protocolo de red.
- Desarrollar transductores que se conecten directamente a la red. Para ello es necesario:
  - Conocer los detalles de los protocolos.
  - Integrar Transductores + Acondicionamiento de la señal + NCAP ('Network-Capable Application Processor').
  - Hacer todo lo anterior para cada red que el fabricante quiera soportar.

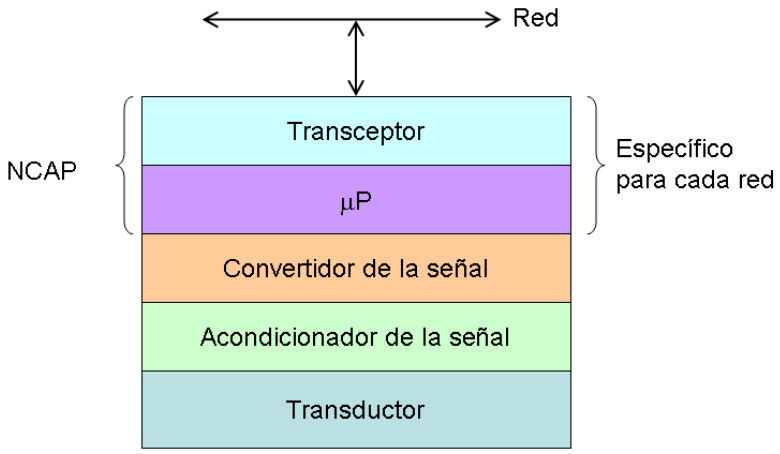


Figura 3.6. .

Para simplificar esta tarea, se define el estándar IEEE 1451. Este estándar establece una serie de señales y un protocolo entre los elementos transductores (STIM, 'Smart Transducer Interface Module') y los NCAP.

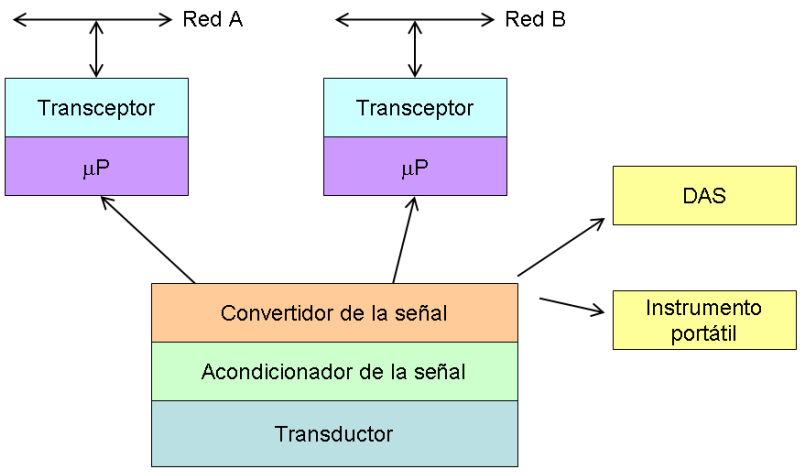


Figura 3.7. .

3.2 ESTÁNDAR IEEE 1451

IEEE 1451 consiste en una serie de estándares para sensores inteligentes ('Smart Sensors'), que pretende integrar diferentes tipos de sensores de forma fácil y barata.

3.2.1 Estandarización

IEEE ha desarrollado varios estándares entorno al 1451. Estos son:

- IEEE 1451.0 (2007): Trata de definir aspectos comunes entre los distintos estándares. Concretamente, define:
  - La estructura de las hojas de datos electrónica de los transductores (TEDS, 'Transducer Electronic Data Sheet').
  - La interfície entre el estándar 1451.1 y el resto de estándares 1451.
  - Protocolos de intercambio de mensajes.
  - Comandos de los transductores.
- IEEE 1451.1 (1999): Especifica el modelo de información del NCAP. Es decir, la obtención y distribución de información sobre la red convencional IP.
- IEEE 1451.2 (1997): Especifica el protocolo de comunicación entre transductores y µPs y los formatos de las TEDS.
- IEEE 1451.3 (2003): Especifica los formatos de la comunicación digital y el formato de las TEDS para sistemas distribuidos 'Multidrop'.
- IEEE 1451.4 (2004): Especifica los protocolos de comunicación y formato de las TEDS en modo Mixto (Analógico y Digital).

- IEEE 1451.5 (2007): Especifica los protocolos de comunicación 'Wireless' y los formatos de las TEDS.

Algunas definiciones utilizadas en este estándar:

- STIM ('Smart Transducer Interface Module'): Un módulo que contiene : el transductor, los TEDS, la lógica para implementar la interfície del transductor, y cualquier conversión o acondicionamiento de la señal que sea necesaria. En operación, todos estos módulos han de ir unidos.
- TEDS ('Transducer Electronic Data Sheet'): Hoja de características electrónica del transductor que puede ser leída y que describe al transductor correspondiente.
- NCAP ('Network Capable Application Processor'): Dispositivo entre el STIM y la red. Realiza las tareas de: comunicaciones en la red, comunicaciones con el STIM y funciones de conversión de datos.
- Transductor (XDCR).

Norma	Interficie HW	TEDS	Distancia	Nivel Medida	Signal converter
1451.1	N/A	Soporta	N/A	Soporta	N/A
1451.2	10-wire digital	Si	Corta (P2P)	Raw digital engineering units	Si
1451.3	4-wire digital	Si	Media (Multidrop)	Raw digital engineering units	Si
1451.4	2 a 4 wire A/D	Si	Media (P2P)	Analog	Si
1451.5					

3.2.2 Aspectos generales

El núcleo del sistema se basa en asociar físicamente las TEDS a los transductores.

Estos TEDS tienen una estructura que soporta una amplia gama de transductores. Son utilizadas para almacenar parámetros que definen todas las características necesarias de los transductores, para que los controle el NCAP, incluso la autoidentificación.

Permite, además, implementar sistemas 'Plug & Play' (y Redes 'Ad Hoc').

Una interficie digital (IEEE 1451-2) permite al NCAP:

- Acceder a las TEDS.
- Leer datos digitalizados de los sensores.
- Escribir en los actuadores.
- Disparar sensores y actuadores.

Y a los transductores les permite:

- Indicar cuándo se pueden leer datos de los sensores o cuándo se ha realizado un actuación.

➡ Indicar al NCAP cuándo se ha producido una excepción.

Un sistema basado en este estándar podría representarse como en la siguiente figura:

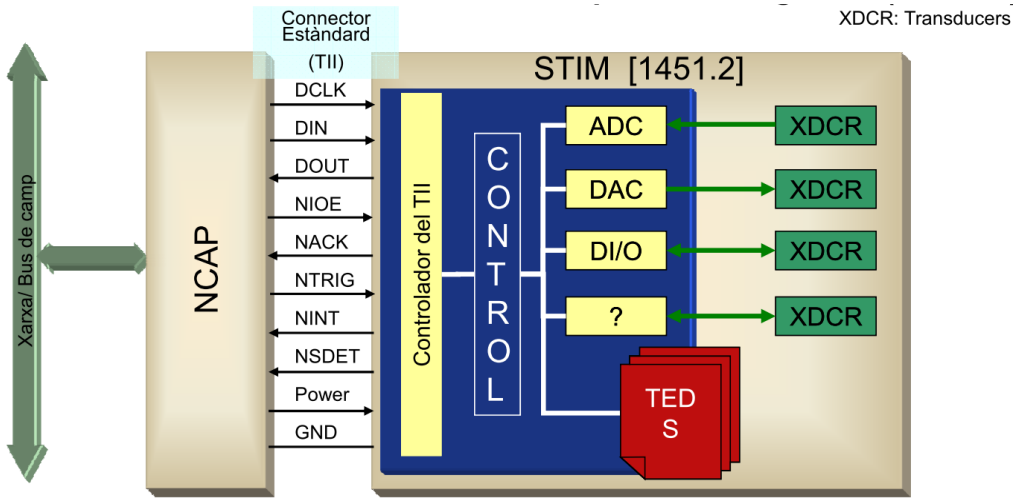


Figura 3.8. Sistema basado en el IEEE 1451.

El TII ('Transducer Independent Interface') está formado por las líneas físicas que conectan el NCAP con el STIM. El significado de cada línea puede verse en la siguiente tabla:

Senyal	Lògica	Control	Funció
DIN	Nivell	NCAP	Línia de dades del NCAP al STIM.
DOUT	Nivell	STIM	Línia de dades del STIM al NCAP.
DCLK	Flanc pujada	NCAP	Línia de rellotge. El flanc de pujada registra les dades a DIN i els de baixada les de DOUT.
NIOE	Nivell (baix)	NCAP	Activa el STIM. Límits de les trames de dades.
NTRIG	Flanc baixada	NCAP	Línia de (NCAP és el mestre).
NACK	Flanc baixada	STIM	2 funcions: - Reconeixement del *Trigger*. - Reconeixement de dada rebuda.
NINT	Falling Edge	STIM	La fa servir el STIM per demanar atenció al NCAP
NSDET	Nivell (baix)	STIM	La fa servir el NCAP per a detectar que s'ha connectat el STIM.
POWER	N/A	NCAP	Alimentació del STIM, 5V (màxim 75mA)
GND	N/A	NCAP	GND.

(NTRIG: línea de 'trigger').

El protocolo general de transferencia entre el STIM y el NCAP puede verse en el siguiente esquema:

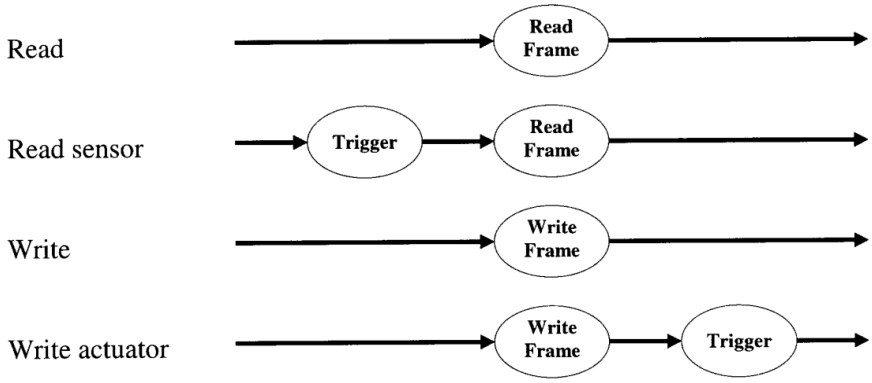


Figura 3.9. Protocolo general de transferencia de datos.

Los datos se transfieren en modo 'bit-serial' del NCAP al STIM via DIN, o a la inversa via DOUT.

Un ejemplo de lectura de una trama de datos (un par de datos en este caso) se muestra en la siguiente figura:

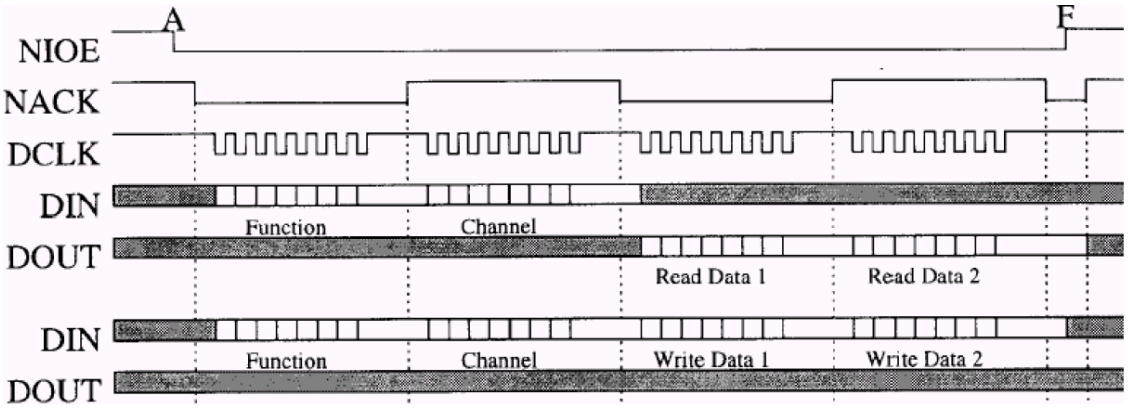


Figura 3.10. Ejemplo de lectura de una trama de datos.

(El estado inicial para triggering, lectura o escritura consiste en tener las líneas NTRIG, NACK y NIOE negadas).

La transferencia viene controlada por DCLK. En el primer flanco de bajada, el transmisor poner el dato, y el subsiguiente flanco de subida, el receptor lo captura ('latch').

Aunque podría hacerse una comunicación full-duplex (usando DIN y DOUT), aquí sólo se hace half-duplex.

Primero el NCAP fuerza NIOE a nivel bajo, y espera a obtener un 'Acknowledge' del STIM (si éste fuerza NACK a nivel bajo). Entonces NCAP envía, a través de DIN, la dirección funcional (siguiendo el formato de la escritura de un byte) y después la dirección de canal. Por cada byte, el STIM ha de realizar un 'Acknowledge' modificando el estado de NACK. Seguidamente, el STIM coloca los datos (de 0 a n bytes) (en este caso 2 bytes) en DOUT (siguiente el formato de la lectura de un byte). Al acabar, el NCAP fuerza NIOE a nivel alto. El STIM realiza un último cambio de NACK.

3.2.3 TEDS

Los TEDS pueden contener diferentes bloques. En total pueden haber hasta 8, aunque sólo dos de ellos son obligatorios. Los ochos TEDS posibles se muestran en la siguiente tabla:

TYPES	Reader	Mand./Opt.
Meta TEDS	**Màquina**	**Obligatòria**
Channel TEDS (1 per canal)	**Màquina**	**Obligatòria**
Calibration TEDS (1 per canal corregit)	**Màquina**	**Opcional**
Generic-Extension TEDS	**Màquina**	**Opcional**
Meta-ID TEDS	**Usuari**	**Opcional**
Channel-ID TEDS (1 per canal)	**Usuari**	**Opcional**
Calibration-ID TEDS (1 per canal corregit)	**Usuari**	**Opcional**
End User Application-Specific TEDS	**Usuari**	**Opcional**

Figura 3.11. Posibles TEDS.

Veamos con algo de detalle los tres primeros:

- Meta-TEDS (1 por STIM): Contiene parámetros globales del STIM (que son comunes a los transductores del STIM), permitiendo el acceso a los canales. Son constantes de sólo lectura. Es un bloque obligatorio. Contiene:
  - Parámetros sobre las temporizaciones del hardware de interfície: por ejemplo la velocidad máxima de transferencia (ya que el estándar especifica únicamente una velocidad de transferencia mínima, pero no máxima).
  - Parámetros de identificación: fabricante, nº de dirección, fecha, etc...
  - Información sobre los grupos de canales: número de canales por agrupación, número de agrupaciones,...
- TEDS de los canales: Hay TEDS de canal para cada canal implementado. Es un bloque obligatorio. Son constantes y de sólo lectura. Los campos que contiene cada uno son:
  - Información sobre el transductor: tipo de canal que posee, clave de auto-test, etc...
  - Información sobre la conversión de datos: frecuencia de accesso, tiempos de setup de lectura y escritura, etc...
- TEDS de calibración: Proporciona toda la información necesaria para el software de corrección. Utiliza una función de corrección multinomial (contiene todas las combinaciones del producto de las variables (datos obtenidos de diversos canales)). Las constantes de estos polinomios se almacenan aquí.

3.2.4 STIM ('Smart Transducer Interface Module')

Un STIM es un módulo que contiene los TEDS, la lógica necesaria para implementar la interfície del transductor, el transductor(s) y, si es necesario, conversión de señal y acondicionamiento de la señal. Estos módulos no pueden ser físicamente separados cuando se encuentra operando.

Un único STIM puede medir y/o control diversos fenómenos físicos (o variables físicas). Cada una de estas variables estará asociada a una canal del transductor del STIM. En el estándar se definen 6 tipos de canales:

- Sensor: Ha de devolver un valor digital representando la magnitud medida.
- Actuador: La acción será triggerizada.
- 'Buffered sensor': La salida sólo tiene un simple nivel de 'buffering' de datos; será triggerizado.
- 'Data sequence sensor': Adquiere datos contínuamente, con periodo controlado por el STIM; la obtención de datos es triggerizada, ofreciendo los últimos datos obtenidos justo antes del trigger.
- 'Buffered data sequence sensor'.
- 'Event sequence sensor': Produce una señal cuando ocurre algún evento; la señal es el mismo acknowledge para eventos de otros transductores.
- Transductor general: Para futuras ampliaciones de otros tipos de transductores.

Cada STIM ha de ser capaz de realizar una serie de tareas obligatorias (como direccionamiento, triggerización, interrupciones, etc.), y otras opcionales. Lo mismo ocurre para cada canal del STIM. (como canalizar los TEDS, estado, control, etc.).

La dirección especifica si los datos se van a leer o escribir (dirección funcional), a qué función corresponde y a qué canal del STIM (dirección de canal: cada transductor tiene asignado un número de canal; máximo de 255 por cada STIM; canal 0 se refiere a todos los canales). Consta de dos bytes y su estructura es la siguiente:

Functional address Most significant byte								Channel address Least significant byte							
r/w	Function code							Channel number							
msb							lsb	msb							lsb

Figura 3.12. Bytes de la dirección.

El STIM ha de ser capaz de comunicarse con el NCAP, que es el que realizará la conexión con el bus físico. Para ello, el estándar también impone ciertas especificaciones. Los datos se transmiten siempre en bytes. Cada byte de datos se transmitirán en una forma 'bit-serial', con el bit más significativo primero. El NCAP avisará cuando el bus está activo, mientras que el STIM hará un 'Acknowledge' para indicar que está listo. La velocidad de transmisión la determinará el NCAP. Triggering ha de estar soportado, y permite al NCAP enviar una orden al STIM para una determinada acción, y también para que el STIM indique que la acción ha acabado.

Su funcionamiento es el siguiente:

- Al arrancar, los TEDS de cada STIM se leen.
- El NCAP utiliza la información de cada TEDS para realizar una auto-configuración.
- Para realizar la lectura de un sensor, el NCAP dispara ('trigger') el transductor, espera a un 'Acknowledge', y lee los datos convertidos. Entonces, el NCAP utiliza la información de los TEDS para convertir la lectura a unidades estándar de ingeniería.
- Para activar un actuador, el NCAP convierte un valor de unidades estándar a valores que entienda el actuador (mediante las TEDS) y lo escribe a través de la interfície hardware. Se envía un 'trigger' al actuador y se espera un 'Acknowledge' del STIM.

3.2.5 NCAP

Definido por el 1451.1.

En la siguiente figura puede verse el modo de transductor inteligente en red. Se indica la parte correspondiente al NCAP (limitado por la línea gruesa discontinua de la derecha):

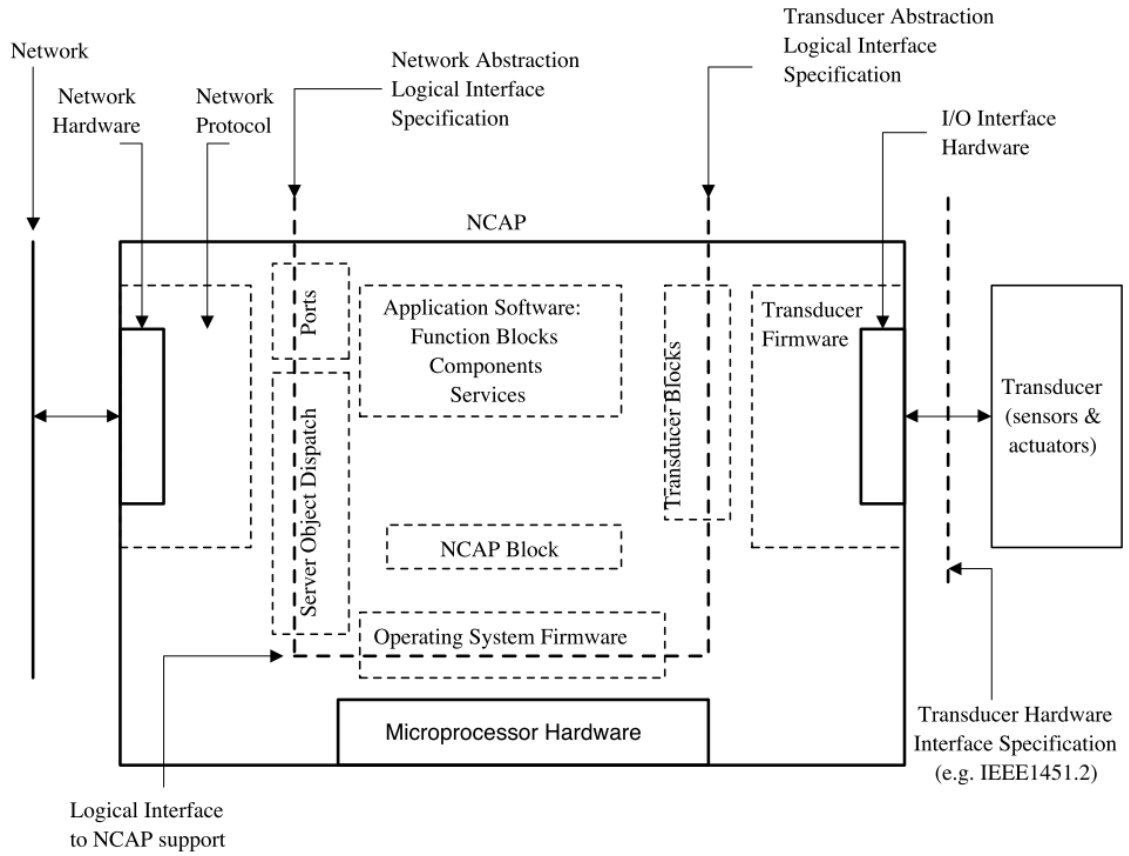


Figura 3.13. Modo de transductor inteligente en red.

(Las líneas sólidas indican los componentes, mientras que las líneas discontinuas indica las funciones lógicas).

El NCAP consiste el el microprocesador y la circuitería asociada así como el hardware implementando físicamente la red a la que se une, así como las entradas y salidas a los transductores.

El sistema operativo proporciona una interfície para las aplicaciones (indicado por 'Logical Interface to NCAP support).

La 'Network Abstraction Logical Interface Specifications' proporciona una abstracción para esconder los detalles de comunicación específicos de una red determinada, a través de una serie de métodos de comunicación.

La 'Transducer Abstraction Logical Interface Specification' realiza la misma abstracción, pero para transductores.

Las aplicaciones se modelización como 'Function Blocks', junto con Componentes y Servicios.

3.2.6 Protocolo de lectura de un sensor

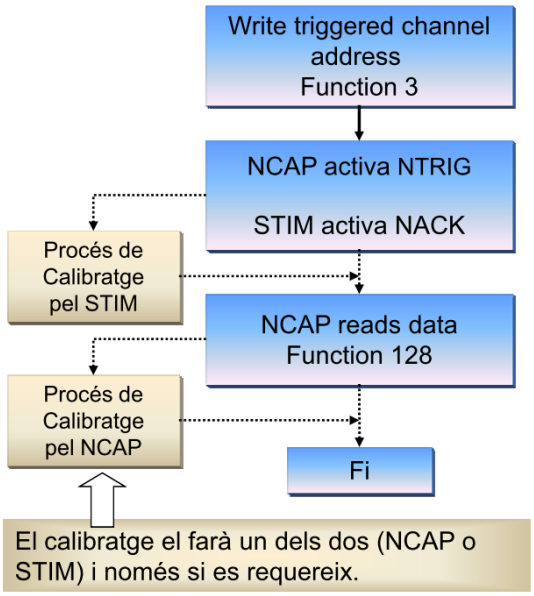


Figura 3.14. Protocolo de lectura de un sensor.

- El NCAP envía una funció al STIM con la dirección del canal a medir (write triggered channel address = 3).
- El NCAP envía el trigger al STIM (activa NTRIG). Entonces comienza la medida. El STIM ha de reconocer que ha recibido el trigger (activa NACK).
- El NCAP espera el tiempo necesario para que el STIM acbe la medida (eso lo sabe leyendo la TEDS correspondiente). Después envía una 'data channel read function' (128) al STIM para leer los datos leídos.
- Si es necesario, el NCAP hará el proceso de calibración (esto también lo ha podido hacer previamente el STIM).

3.2.7 Protocolo de control de un actuador

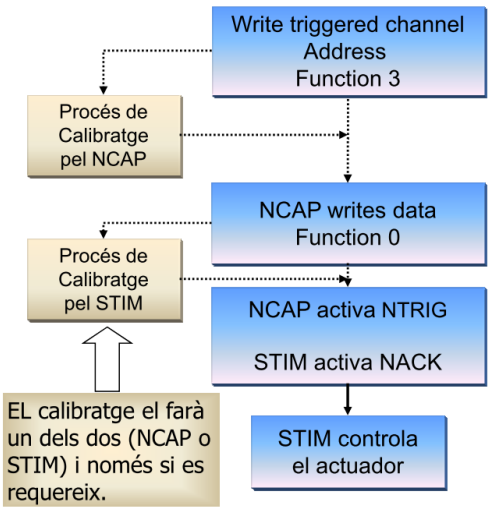


Figura 3.15. Protocolo de control de un actuador.

- El NCAP envía una función al STIM con la dirección del canal sobre el que se actuará (write triggered channel address = 3).
- El NCAP envía un 'data channel write function' (0) con el dato a escribir en el actuador cuando se le envíe el 'trigger'.
- El NCAP envía el 'trigger' al STIM (activa NTRIG). Entonces comienza el control del actuador. El STIM ha de reconcer que ha recibido el 'trigger' (activa NACK).
- Si es necesario, el NCAP hará el proceso de calibración (esto también lo ha podido hacer previamente el STIM).

3.2.8 IEEE 1451.4

En el estándar IEEE1451.4 se establece un mecanismo para incluir Auto-Identificación en los transductores analógicos convencionales.

Se define el concepto de transductor Mixed-mode, con dos interfícies:

- Una analógica, que proporcional la señal proporcional a la magnitud a medir (tensión, temperatura, presión,...) en la forma tradicional.
- Se incluye una memoria en el transductor (con las TEDS) para describir e identificar al transductor. La interfície proporciona esta información.

Se definen dos tipos de interfícies mixed-mode:

- IEEE1451.4 class 1: Las dos interfícies comparten la conexión eléctrica (sólo se necesitan dos cables).

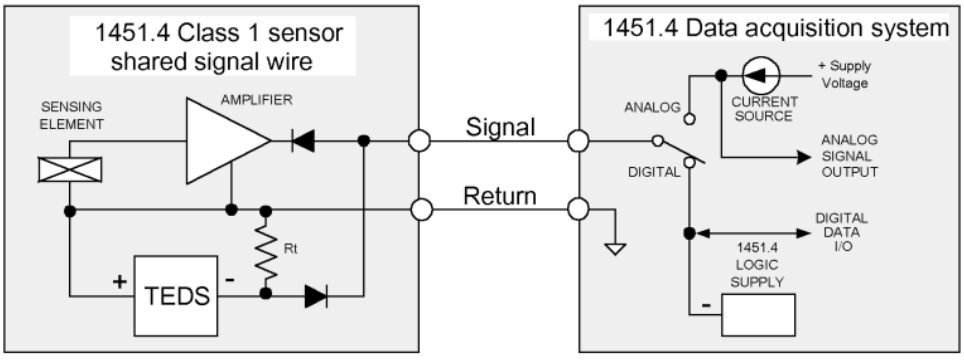


Figura 3.16. Interfície de clase 1.

(según el tipo de salida del sensor, la interconexión puede cambiar)

- IEEE1451.4 class 2: La interfície de la TEDS es independiente de la señal del transductor analógico. De hecho, no hay ningún cambio en la interficie analógica original, sólo se añade una línea para la interfície digital. Como están separadas, ambas pueden accederse simultáneamente.

En la siguiente figura puede verse un ejemplo de una interfície de la IEEE1451.4 class 2 para un sensor con una salida en puente de Wheatstone:

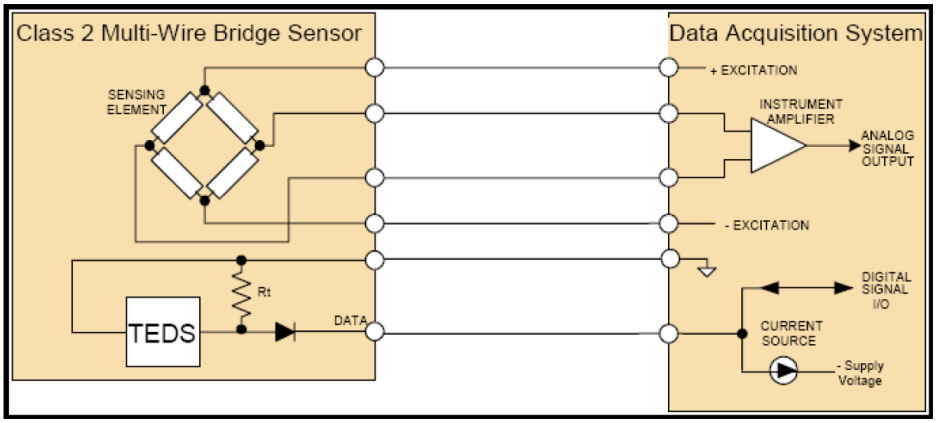


Figura 3.17. Ejemplo de la interficie IEEE1451.4 de clase 2.

4. EJEMPLOS DE PROGRAMACIÓN C BASADOS EN LOS μC Z8 ENCORE

Ejemplos de programación en C (también pueden encontrarse en ensamblador) para el μC Z8 Encore, se pueden encontrar en la página web de Zilog, tanto en notas de aplicación como directamente el código C en archivos comprimidos (http://www.zilog.com/index.php?option=com_doc&Itemid=99). Como ejemplos se pueden mencionar:

- Sensor de presión → AN0195.
- Programación software de protocolo de comunicación (UART) → AN0147.