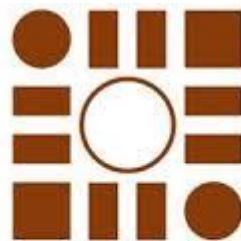




Universidad de Alcalá

Estación meteorológica basada
en LPC1768.



ESCUELA POLITECNICA
SUPERIOR

Autor: Palomo Alonso

Alberto

Asignatura: Sistemas electrónicos digitales avanzados.

ESCUELA POLITÉCNICA SUPERIOR - UAH.

Índice de contenido:

Abstract.	5
Capítulo 1: Introducción y objetivos.	6
Introducción.	6
Objetivos a cumplir.	7
Posibles mejoras al sistema en un futuro.	10
Capítulo 2: Componentes del sistema.	11
Mini-DK2	11
Anemómetro.	12
LDR.	12
Sensor UV.	13
Servo motor.	13
Amplificador + altavoz.	14
Micrófono.	14
Sensor de temperatura.	15
Sensor de presión.	15
TFT.	16
Capítulo 3: Estructura del sistema.	17
0. Esquema general.	17
1. PWM1 y control de servo.	18
2. WEB.	18
3. Usuario.	18
4. Medio exterior.	19
5. Timer controlador de medidas.	19
6. Grupo de sensores 1.	19
7. Grupo de sensores 2.	19
8. WDT.	19
9. RTC.	19
10. UART.	20
11. ADC.	20
12. DMA + DAC.	20
Capítulo 4: Implementación del sistema.	21
Carpeta MiniDK-2.	21
Carpeta README.	21
Carpeta Startup.	21
Carpeta Principal.	21
Carpeta Menu.	22
Carpeta Setup.	24

Carpeta I2C.	24
Carpeta Anemómetro.	25
Carpeta ADC.	25
Carpeta OneWire.	28
Carpeta PWM.	29
Carpeta DAC.	30
Carpeta Database.	30
Carpeta WDT.	31
Carpeta UART.	31
Carpeta TCP/IP.	32
Carpeta RTC.	34
Carpeta Timers.	34
Carpeta SDCARD.	35
Carpeta CMSIS.	35
Carpeta GLCD.	35
Capítulo 5: Pruebas, ejecutabilidad y conclusiones.	36
Pruebas.	36
Ejecutabilidad.	37
Conclusiones.	38
Capítulo 6: Manual de usuario.	39
TFT.	39
WEB.	42
UART.	44
Referencias	45
Anexos.	46
Anexo I - Esquemático.	46
Anexo II - Código fuente.	47
Readme.md	47
Archivos extensión startup.	48
Archivos extensión C.	55
Archivos extensión H.	497
Archivos extensión CGI.	629
Anexo III -Hojas de datos.	633

Alberto Palomo Alonso.

Abstract.

Microcontrollers, also known as its acronym MCUs (Micro Controller Unit), are embedded systems which, nowadays, its computational power can overcome microprocessors designed in early 90s as P54C (an old intel processor family) which CPU clock were around 100 MHz and were expensive. Only 20 years later, MCUs overcame, with a remarkable lower cost, its computational power.

In this document, I will talk about an LPC1768 based embedded system which holds a Cortex M3 processor. This MCU has a 100 MHz CPU clock speed and has a Thumb/Thumb-2 subset architecture developed by ARM. In this case, the project to document has been developed in a Mini-DK2 card manufactured by NXP, which contains the following features:

- 100M Ethernet network interface.
 - TFT color LCD with SPI interface.
 - Two USB interfaces.
 - Some sets of buttons, one of which can produce interrupts.
 - Serial ISP download JTAG (the code can also be flashed via USB).
 - SD card with SPI interface.
 - Generic input and output ports (GPIO).
 - A 12 bit ADC and 10 bit DAC.
 - One memory protection unit divided by regions.
 - One real time clock.
 - Interrupt controller.
 - A quadrature encoder interface.
 - System of pulse width modulator (PWM) and other for motor control.
 - An I2C bus and 3 SPI buses.
 - Four timer modules.
 - Four universal asynchronous receiver transmitter buses.
(Reference from: www.hotmcu.com/lpc1768minidk2-development-board-p-55.html)
-
- NXP sponsors the user a manual (UM10360) which can be obtained by typing the following link: <https://www.nxp.com/docs/en/user-guide/UM10360.pdf>
 - The IDE is MDK (Keil uVision4) for the Cortex M3 processor, it can be bought in ARM website.

The project to build in this embedded system consists in a weather station with the capacity to measure some weather conditions such as temperature or air pressure. It will be able to communicate with a computer via UART and it will contain a website reachable via Ethernet interface.

Capítulo 1: Introducción y objetivos.

Introducción.

Una estación meteorológica basada en LPC1768 puede llegar a necesitar un sistema de control medianamente complejo. Pasar de instrucciones máquina que utiliza el procesador de ARM (Cortex M3) a controlar sensores de manera simple y eficaz necesita varias capas de interfaces, esto es, la acción que realizan los sistemas operativos en cualquier tipo de máquina que necesite interacción sencilla con el usuario. Es por ello que definiré más adelante una estructura del sistema operativo que actúa como interfaz, tanto para los sensores como para el usuario, cuyo código estará contenido en memoria flash.

Para empezar, hablaré un poco sobre las características que tiene la placa que vamos a utilizar.

La placa **Mini-DK2** contiene en su interior un procesador de ARM llamado Cortex M3 que funciona a 100MHz de velocidad de reloj. Además, tiene un oscilador de cristal para los periféricos denominado XTAL de 12MHz. A continuación, en la '*Tabla 1.- Características de la placa Mini-DK2*' resumo la gran mayoría de sus características.

<i>Característica.</i>	<i>Número de elementos.</i>	<i>Notas.</i>
100M interfaz Ethernet	1	
Interfaz TFT de colores (LCD) con interfaz SPI.	1	
Puertos de propósito general (GPIO).	4 x 32	
Interfaz USB.	2	
Bus I2C	1	
Bus SPI	3	
Interfaz UART.	4	
Reloj en tiempo real.	2	
Oscilador de cristal.	1	12MHz
Módulos temporizadores y de captura.	4	
Módulo PWM.	1	
Módulo PWM especializado para motores.	1	
Acceso directo a memoria (DMA)	1	
Unidad de protección de memoria.	1	8 regiones
Conversor de analógico a digital.	1	12 bits
Conversor de digital a analógico.	1	10 bits
Memoria RAM.	1	64kB
Memoria flash.	1	512kB
Interfaz de encoder de cuadratura	1	
Slot de SD con interfaz SPI	1	
Interrupciones externas.	4	
Leds	N/A	
Interfaz de carga de código vía JTAG por ISP.	1	
Controlador de interrupciones anidadas (NVIC).	1	
Entrada de interrupción no enmascarable (NMI).	1	

Tabla 1.- Características de la placa Mini-DK2.

Para abordar la práctica hablaré de la estructura definida de ese 'sistema operativo' en el **Capítulo 3: Estructura del sistema** y más adelante explicaré cada parte específicamente. Cabe destacar que el código empleado utiliza una estructura jerárquica para controlar el sistema y utiliza variables como señales de comunicación entre diferentes zonas de código.

Más adelante explicaré qué sensores tiene el proyecto, qué miden cada uno de ellos y cómo funcionan a rasgos generales. Todo ello explicado en el **Capítulo 2: Componentes del sistema**.

Objetivos a cumplir.

Los objetivos establecidos para el cumplimiento de la práctica se pueden resumir en la siguiente tabla (Tabla 2.- Objetivos a cumplir):

<i>Objetivo general.</i>	<i>Objetivos específicos.</i>
<i>Crear un menú táctil con el TFT de la tarjeta.</i>	Crear un statechart. Manejar las librerías para la generación de caracteres y zonas en pantalla. Vincular todos los datos y señales de forma coherente. Permitir al usuario modificar variables del sistema. Permitir al usuario interactuar con el sistema. Calibrar la pantalla de manera estática o dinámica. Manejar el brillo de manera manual o automática con el módulo PWM.
<i>Controlar un servo motor para la visualización de temperatura.</i>	Ser capaz de alternar dos modos PWM para controlar el servo y el brillo a la vez. Ser capaz de dar valores coherentes a las variables PWM. Interconectar el módulo PWM con los datos.
<i>Crear un servidor web accesible vía Ethernet</i>	Manejar las librerías TCP-IP. Configurar variables para realizar una conexión TCP-IP. Escribir con lenguaje de marcado una página web. Crear funciones que interactúen con la página web mediante callbacks. Mantener una conexión TCP. Proporcionar datos a la página web. Recibir datos de solicitudes del usuario mediante la web.
<i>Medir la velocidad del viento.</i>	Realizar un análisis físico de cómo realizar las medidas. Realizar un análisis a nivel de electrónica de las medidas. Configurar un módulo temporizador para medir frecuencia de pulsos. Traducir los pulsos a medidas. Conectar los datos recibidos sin que haya interferencias con todo el sistema anterior y posterior, incluyendo gestión de recursos de la tarjeta.

	Examinar si las medidas son precisas.
Tener un reloj en tiempo real.*	Configurar el RTC de la tarjeta.
	Exportar los datos de manera coherente.
	Ser capaz de configurar la fecha, día y hora de manera manual.
	Mostrar por el TFT la hora actual.
	Conectar los datos recibidos sin que haya interferencias con todo el sistema anterior y posterior, incluyendo gestión de recursos de la tarjeta.
Tener un sistema de control de bloqueo del sistema.	Configurar un Watchdog timer.
	Seleccionar una zona del código donde se ejecute con frecuencia, para cualquier estado de funcionamiento, la alimentación del Watchdog sin interferir con el sistema anterior y posterior.
	Elegir un valor adecuado para las variables del Watchdog timer.
	Examinar que este temporizador no salte de manera espuria.
Medir temperatura.	Comprender y manejar el protocolo OneWire.
	Comprender documentación en mandarín o en su defecto comprender documentación mal planificada en inglés*.
	Configurar un pin para un timer en modo captura.
	Gestionar los recursos para no quedarse sin timers ni interferir con el sistema anterior y posterior.
	Estructurar y traducir los datos recibidos adecuadamente.
Reproducir audio.	Configurar adecuadamente el DAC.
	Exportar vía DMA los datos solicitados.
	Gestionar la memoria de manera eficaz para tener 3 segundos de audio almacenados en memoria.
	Gestionar los recursos para no quedarse sin timers ni interferir con el sistema anterior y posterior.
	Configurar adecuadamente otro timer para reproducir el audio guardado en memoria.
	Hacer sonar el audio cada vez que una medida supera un umbral.
	Establecer dicho umbral para todas las medidas.
	Implementar un botón de reproducción del audio cada vez que el usuario lo desee.
	Seleccionar las variables adecuadas, así como la frecuencia de exportación del audio.
Grabar audio.	Configurar un ADC para tener dos modos de funcionamiento.
	Configurar un pin de la placa para leer audio.
	Seleccionar adecuadamente las variables, así como la frecuencia de muestreo del audio.
	Ser capaz de alternar ambos modos de funcionamiento del ADC sin interferir con

	las demás medidas y ser capaz de bloquearlas mientras se produce la grabación del audio.
	Exportar los datos de manera adecuada al sistema.
	Entrar en el modo de grabación cuando el usuario lo desee.
<i>Medir el brillo.</i>	Traducir de manera adecuada resistencia a voltaje y de voltaje a código del ADC.
	Escoger una resistencia de pull-up adecuada para el LDR seleccionado.
	Configurar el ADC en modo ráfaga sin que interfiera con la grabación de audio.
	Leer el código del ADC para traducirlo a nivel de brillo.
	Controlar automáticamente el brillo del LCD en función del dato.
	Examinar la validez de las medidas.
	Gestionar la frecuencia de muestreo.
<i>Medir el índice UV.</i>	Traducir de manera adecuada el código recibido al índice.
	Usar el modo del ADC ya configurado para esta medida sin interferir con el sistema anterior y posterior.
	Examinar la validez de las medidas.
	Exportar los datos al sistema.
<i>Medir presión atmosférica.</i>	Crear un controlador que sea capaz de leer registros de un sensor vía I2C.
	Examinar la validez de las medidas.
<i>Medir posición GPS¹.*</i>	Conseguir leer un sensor GPS vía UART.
<i>Conexión WiFi.*</i>	Conseguir conectar un módulo WIFI vía UART a la placa.
	Ser capaz de modificar variables.
	Comprender los protocolos utilizados para conectar un dispositivo WIFI con el módulo.
	Buscar una interfaz adecuada.
<i>Conexión UART con un PC.</i>	Conectar el módulo UART del lpc1768 con el controlador de un ordenador convencional.
	Modificar variables mediante la conexión serie asíncrona.
	Conseguir exportar datos vía conexión seria asíncrona.

Tabla 2.- Objetivos a cumplir.

Como podemos observar, esta lista de objetivos a cumplir de manera obligatoria para el funcionamiento del sistema, contiene todo lo que debemos de realizar a grandes rasgos y será utilizada como guía para los siguientes capítulos.

¹ Los objetivos marcados con * representan objetivos opcionales pero que incluyen en su implementación modelos marcados como obligatorios.

Posibles mejoras al sistema en un futuro.

Pese a que los objetivos generales son muy completos, los objetivosopcionales son sólo suplementarios y añaden funciones extras a esta estación meteorológica para hacer pequeñas mejoras del sistema y que sea aún más completo. No tabularé los objetivos opcionales, que no prometo para nada cumplir, pero que puede que unas versiones futuras del proyecto aparezcan, son los que enumero a continuación y evalúo su complejidad:

- Añadir memoria extra con la interfaz SPI en la tarjeta SD.
(Complejidad media)
- Añadir funcionalidad a la página web para modificar más variables del sistema.
(Complejidad baja)
- Hacer un seguimiento histórico de los datos.
(Complejidad baja)
- Exportar los datos a la comunidad.
(Complejidad media)
- Mejorar la interfaz TFT.
(Complejidad baja)
- Comunicarse con más estaciones del mismo tipo a gran distancia.
(Complejidad alta)
- Realizar los objetivos marcados como * en el apartado anterior.
(Complejidad media)

Capítulo 2: Componentes del sistema.

Mini-DK2

El componente fundamental de sistema es la tarjeta antes mencionada en el *Capítulo 1: Introducción y objetivos a cumplir*. Será la unidad de control y todos los sensores y aparatos electrónicos del sistema se conectan a este componente. Su función es ejecutar el sistema operativo antes mencionado en el Capítulo 1 y proporcionar control sobre todo el sistema. En la *Figura 1* podemos ver el esquema de pines de la tarjeta con la que vamos a trabajar. En la *Figura 2* podemos ver una imagen de la tarjeta.

Si desea ver las características generales de este componente puede recurrir a la Tabla 1 o si desea más información, puede recurrir a su fabricante; NXP.

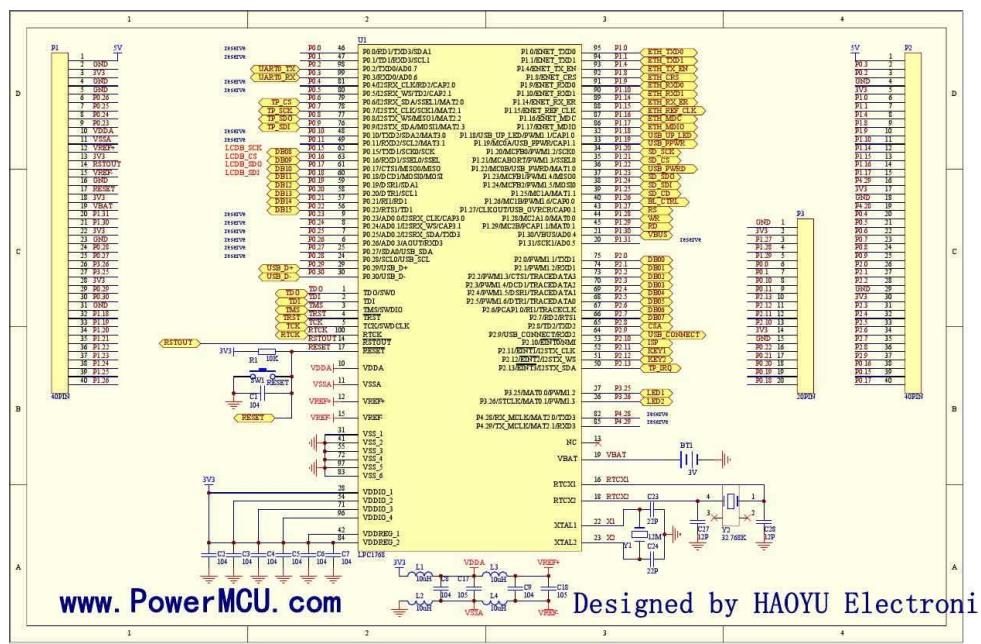


Figura 1.- Esquema general de la tarjeta Mini-DK2.

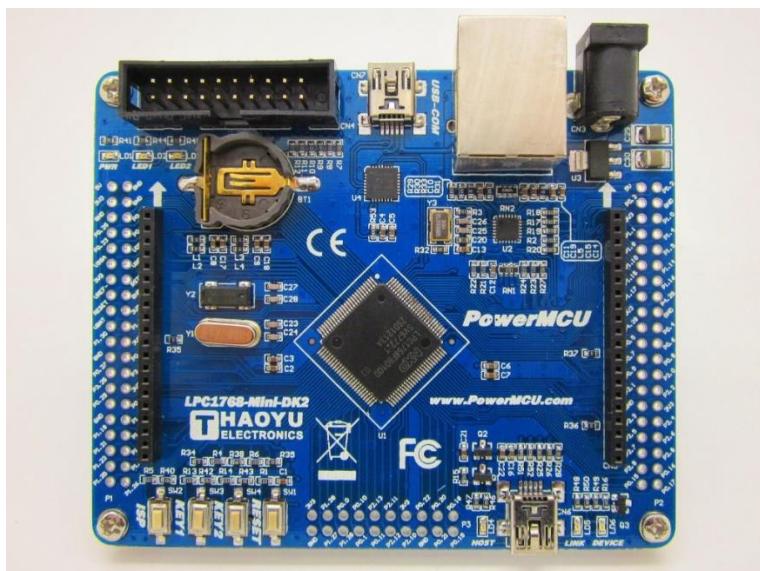


Figura 2.- Imagen de la tarjeta Mini-DK2.

Anemómetro.

Este componente nos permitirá medir la velocidad del viento. Se trata de un eje rotatorio conectado a tres cuencas de baja densidad que forman un ángulo de 120° entre ellas y permiten rotar sobre el eje si existe viento.

Tiene dos pines de salida que son cortocircuitados por zonas cada 90° , por lo que cada vuelta se cortocircuita dos veces.

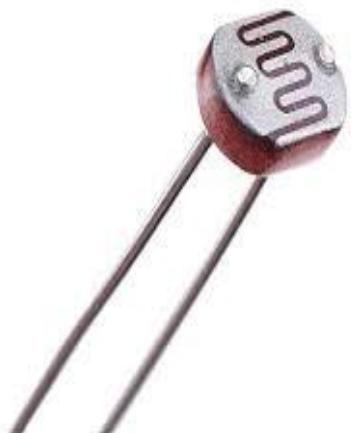
Su funcionamiento consiste en generar cortocircuitos entre los dos pines de salida a medida que el eje va rotando, por lo que podemos generar pulsos cuadrados a medida que el eje rota si lo conectamos adecuadamente.

En la siguiente figura, *Figura 3* podemos ver una imagen de este anemómetro del cuál no he encontrado documentación.



Figura 3.- Anemómetro.

LDR.



Este componente nos permitirá medir la cantidad de brillo que hay en la zona de manera que este varía la resistencia entre sus dos patillas en función de la luz que recibe. Esto sigue una relación lineal con una tolerancia entre los LUX en el ambiente y la resistencia entre ambas patillas. Conectándolo adecuadamente podemos traducir mediante un divisor resistivo y escogiendo un valor de pull-up adecuado esta resistencia nos dará los LUX que mida.

En la *Figura 4* incluyo una imagen del LDR y en la *Figura 5* una gráfica en la mejor calidad que he encontrado que representa la relación resistencia-LUX del componente.

Figura 4.- LDR.

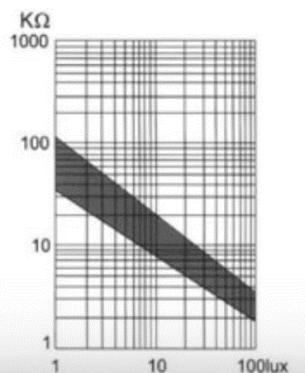


Figura 5.- Gráfica R-LUX

Sensor UV.

Este sensor nos permitirá medir el índice UV que reciba, este sensor traduce el índice UV que recibe a voltaje directamente sin necesidad de aplicar un circuito conversor resistivo como en los dos casos anteriores, en cambio, viene en forma de tarjeta pequeña con 3 pines de interés:

- Alimentación (+3.3V)
- Masa. (-0.0V)
- Salida de voltaje. [0, 3.3V]

Este sensor mide la índice UV y saca un índice del 0 al 10 en forma de la siguiente ecuación:

$$V_o = \frac{V_{cc} \cdot IndiceUV}{IndiceUVmax}$$

En la Figura 6 se muestra el sensor UV a utilizar (VMA30A)

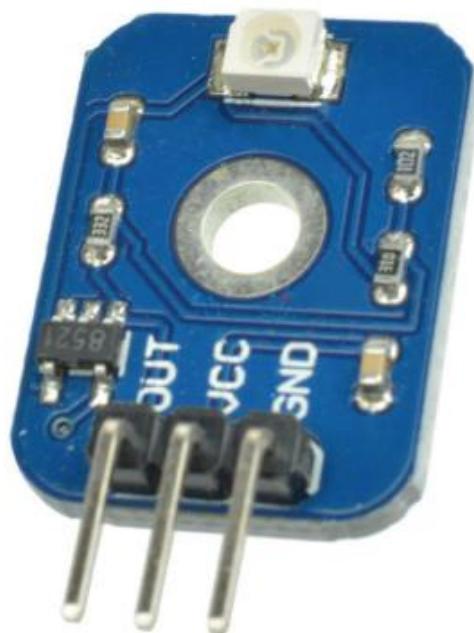


Figura 6.- Sensor UV30A.



Servo motor.

Este componente nos permitirá representar la temperatura que hay entre dos valores límite a elegir. El mínimo significará izquierda, mientras que el máximo de esa temperatura será a la derecha. Este servo controla el ángulo de posición en función de una señal modulada por pulso (PWM) y, en función de su ciclo de trabajo o tiempo a nivel alto, tendrá un ángulo u otro. En la Figura 7 aparece una imagen del servo motor a utilizar (SG90) y en la Figura 8 una gráfica del fabricante que sólo va a servir para identificar un problema que surge a raíz de utilizar la tarjeta Mini-DK2 con este servo.

Figura 7.- SG90, servo.

Como podemos observar en la Figura 8 la señal PWM generada ha de tener un valor a nivel alto de +5.0V. Es por ello que esta gráfica sólo nos va a dar una referencia del periodo de la señal a generar, pero no del tiempo en alta, dado que estos servos se guían por la potencia recibida que depende de la amplitud del pulso y el voltaje. Habrá que hacer una corrección en potencia del pulso.

PWM=Orange (↑↑)
Vcc=Red (+)
Ground=Brown (-)

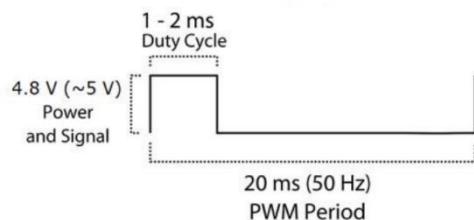


Figura 8.- Pulso a recibir.

Amplificador + altavoz.

Se utilizará un altavoz de 8Ω conectado a un amplificador de señal adaptado de 4 a 8Ω (PAM8320A) que contiene el siguiente patillaje:

- Vdc - +5.0V.
 - GND - -0.0V.
 - Shutdown - N/C.
 - Audioin+ - Audio.
 - Audioin- - GND.

Esta configuración permite pasar de una señal eléctrica de audio a audio en sí. En la Figura 9 muestro una imagen del amplificador soldado al altavoz de 8Ω.

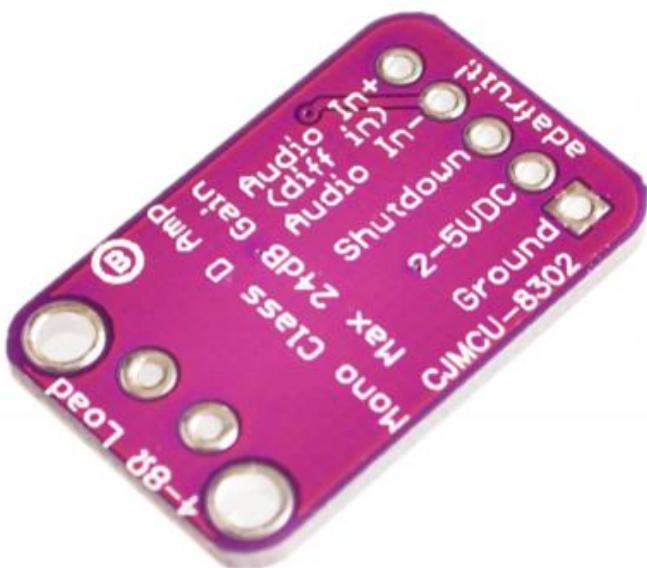


Figura 9.- PAM8320A + Altavoz

Micrófono.

Este micrófono (MAX9814) es un micrófono de alta calidad y bajo coste que nos permitirá hacer vibrar la señal continua con un offset de +1.25V (amplitud máxima de +2Vpp) en función del audio recibido. Directamente podemos convertir la señal por el pin de salida con la siguiente configuración del patillaje:

- AR - N/C
 - Gain - N/C
 - Out - Salida.
 - Vdd - +3.3V
 - GND - -0.0V

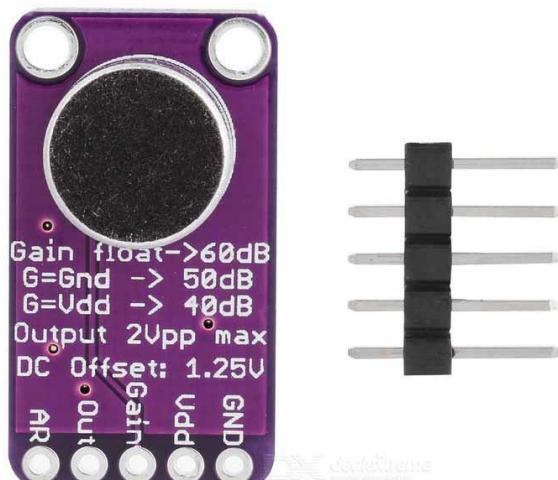


Figura 10.- MAX9814, micrófono.

Se podría configurar para que obtuviese ganancia personalizada, pero no conectar da una ganancia fija también que no tendremos en cuenta dado que nuestro objetivo no es mejorar la calidad de audio, sino simplemente tener una señal de audio. Lo mismo pasa con el pin AR, que fija el factor de compresión del audio; no conectarlo implica que tenga un valor fijo, dado que ambas entradas al aire no forman efecto de alta impedancia.

En la figura 10 tenemos una imagen de dicho micrófono (MAX9814) con parte de los valores más significativos de la datasheet impresa en la tarjeta.

Sensor de temperatura.

El sensor que medirá la humedad y la temperatura en interiores a utilizar será el DHT22. Este sensor pese a tener 4 patillas, una no hay que conectarla y la otra utiliza el protocolo OneWire. Las otras dos son de diferencia de potencial para alimentación. El conexionado es el siguiente, siendo ordenados los pines de izquierda a derecha:

- Pin 1 - +5V Vcc.
- Pin 2 - OneWire.
- Pin 3 - NC
- Pin 4 - 0V GND.

Cabe destacar que este sensor requiere de una resistencia de pull-up de $4k1\Omega$.



Figura 11.- Sensor DHT22.



Figura 12.- Módulo BMP180.

Sensor de presión.

El sensor a utilizar es un módulo que incorpora el sensor BMP180, que funciona por protocolo I2C.

Cabe destacar que el módulo contiene un regulador y las resistencias necesarias de pull-up para el funcionamiento de los pines SDA y SCL del protocolo. El conexionado del módulo es el siguiente:

- GND - -0.0V
- Vcc - +5.0V
- SDA - Datos I2C.
- SCL - Reloj I2C.

Este sensor utiliza un driver software que puede ser extraído del fabricante, pese a eso, se ha desarrollado un driver específico para estación meteorológica utilizando los mismos algoritmos que utiliza el driver del fabricante para leer los registros por I2C del sensor.

TFT.

Se utilizará un display táctil (TFT), para poder mostrar y recibir por pantalla datos relativos a la interfaz del usuario y variables ajustables del sistema. El panel se llama HY32-B y utiliza 8 bits para comunicarse. El conexionado de los 10 bits, los 8 anteriormente mencionados y dos de control se muestra a continuación:



Figura 13.- Panel TFT-LCD.

Puerto / pin utilizado.	Función que desempeña.
P0.6	Touchpannel - CS
P0.7	Touchpannel - SCK
P0.8	Touchpannel - SDO
P0.9	Touchpannel - SDI
P0.15	LCD-SCK
P0.16	LCD-CS
P0.17	LCD-SDO
P0.18	LCD-SDI
P1.26	Control del brillo por PWM.
P2.13	Interrupción del touchpannel.

Tabla 3.- Conexionado del TFT.

Capítulo 3: Estructura del sistema.

Durante este capítulo nos dedicaremos a analizar cómo he estructurado de manera general el sistema y cómo pretendo hacer funcionar y gestionar los recursos de la placa para que no interfieran entre ellos.

Cabe mencionar que este es un análisis que hay que hacer desde un principio y en mi caso no lo he podido hacer debido a que he tenido que añadir las cosas por partes separadas sin conocer el funcionamiento de las posteriores. Por ello, he tenido que aplicar la estrategia de: *la utilización de los menores recursos posibles para que funcione y ya añadiré más recursos si me sobran y si quiero mejorar la calidad*, cosa que si pudiese haber evitado lo hubiese hecho.

0. Esquema general.

Mi esquema se basa en ramificar jerárquicamente en torno a la interfaz de usuario, es decir, el menú, los demás módulos de control del sistema según la Figura 1.

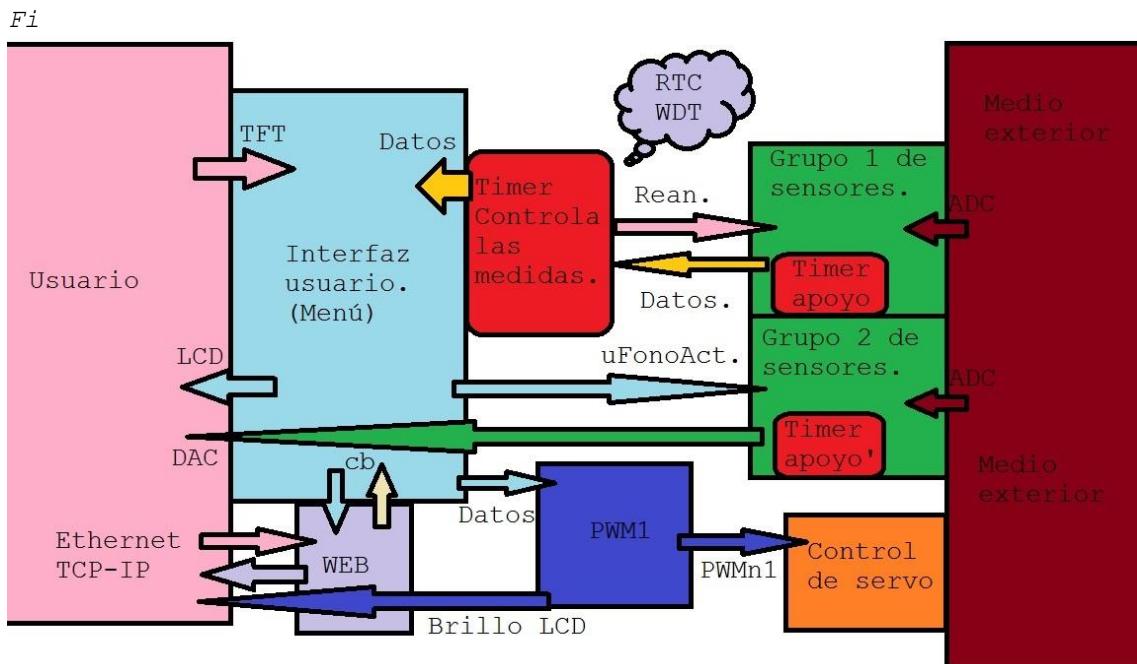


Figura 1.- Esquema de primera versión.

Cabe destacar que el esquema de la Figura 1 es la primera versión. Es decir, sin que exista conexión UART posible ni WiFi. Añadiré una versión así más adelante que únicamente añade esos dos módulos.

El esquema en general consta de módulos que se comunican todos con la interfaz de usuario indirecta o directamente. Por eso que es un requema ramificado a partir de la interfaz de usuario (que actúa como sistema de control). También llama la atención un timer que controla las medidas. Esto lo explicaré en la sección de este mismo capítulo *Timer controlador de medidas*.

A grandes rasgos, existe un timer que controla las medidas del grupo 1, es por eso que he separado en dos grupos los sensores: los controlados por timer y los controlados por la interfaz de usuario; grupo 2. Evidentemente, los del grupo 2 son los correspondientes a grabar y

reproducir audio y los demás corresponden al grupo 1. El tiempo de muestreo es, por tanto, marcado dentro del timer que controla esos sensores para el grupo 1 y constante para el grupo 2 de sensores. Todos ellos leen del medio exterior los datos con el único ADC de la placa que consta de varios canales multiplexados.

El usuario interactúa con el interfaz de usuario mediante el TFT y el LCD, además es posible comunicarse mediante UART y mediante Ethernet (TCP-IP).

1. PWM1 y control de servo.

El módulo PWM se utiliza para controlar un servomotor que se encargará de monitorizar las medidas de temperatura y presión del sistema. Los datos son almacenados en memoria y el sistema se encarga de, cada vez que se leen los sensores, actualizar el módulo PWM para enviar un pulso de manera adecuada para que el servo se posicione en función de unos valores máximos y mínimos que pueden ser modificados por el usuario en todas sus interfaces.

La estación meteorológica utiliza el servo expuesto en el capítulo 2 y lo actualiza el manejador de interrupción del Timer0 del LPC1768.

2. WEB.

La interfaz web utiliza una pila de protocolos y librerías TCP-IP para proporcionar los recursos que permiten montar un servidor WEB integrado en el LPC1768. Mediante un compilador de CGI compila un archivo con extensión de formato CGI que incorpora un lenguaje de marcado HTML, para configurar la interfaz de usuario WEB. Más adelante se explica el funcionamiento.

3. Usuario.

El usuario puede interactuar con las diferentes interfaces que se presentan para leer los datos de la estación meteorológica. Además, este usuario puede modificar mediante todas sus interfaces los valores modificables del sistema.

- Interfaz WEB: Interfaz mediante un servidor WEB sobre un cable ethernet, se puede acceder mediante un navegador y el usuario puede modificar las variables del sistema excepto la hora actual.
- Interfaz UART: Interfaz mediante el protocolo de comunicación UART sobre un cable USB, se puede acceder mediante una aplicación que monitorice la conexión serie asíncrona como Termite (por ejemplo).
- Interfaz TFT-LCD: Interfaz mediante el protocolo de comunicación SPI sobre los pines de la placa MiniDK-2. El usuario lee por la pantalla del LCD la interfaz creada en el programa mediante la iluminación del array de pixeles. Además, el usuario puede modificar las variables del sistema desde la pantalla.

Las variables que el usuario puede modificar son: máximos y mínimos valores de alarma y presión. Si se superan, suena una alarma. La hora puede ser modificada, también el umbral de brillo, que es un valor expresando en segundos de el tiempo que permanece encendida la pantalla en el modo ULP². Otra variable a modificar es la selección del servo, es decir, qué variable (si presión o temperatura) es representada.

² Ultra Low Power Mode. Ver manual de usuario.

4. Medio exterior.

El medio exterior se utiliza para obtener los dartos, los sensores utilizan el medio exterior para obtener valores que la estación meteorológica lee mediante los diferentes protocolos.

5. Timer controlador de medidas.

Se utilizan varios módulos temporizadores para apoyar a los sensores. A continuación, se exponen las funcionalidades de cada uno de los timers si se utilizasen:

- Timer 0: Controla las medidas, marca el tiempo de muestreo de cada uno de los sensores y se ocupa de actualizar el módulo PWM cada vez que interrumpe, en caso de haber cambiado un valor máximo o mínimo.
- Timer 1: Este Timer cumple **tres funciones**: la primera es utilizar el modo capture para medir los pulsos del **anemómetro**; la segunda es utilizar un MR (Match Register) para desactivar el **DAC** una vez haya acabado de reproducir audio; y la tercera es utilizada en caso de **grabar audio**, tener un MR que sirva para activar las cuentas.
- Timer 2: No usado.
- Timer 3: Sirve de apoyo al protocolo OneWire (actúa de contador).

6. Grupo de sensores 1.

Definimos como el primer grupo de sensores al grupo perteneciente al sensor BMP180 y DHT22, debido a que al tener incorporadas esperas activas es recomendable medirlo cuanto menos. Cabe recalcar que estos sensores miden variables que no cambian con gran velocidad, por lo que pueden ser medidas cada 5 segundos incluso más si se desease ajustar. Estas variables son la temperatura, la presión y la humedad. En caso de querer medirlas más rápido, cambiar el símbolo CsCAP definido en el archivo Systemsymbols.h y poner un valor de cuentas adecuado para la función que se deseé desempeñar.

7. Grupo de sensores 2.

Definimos como segundo grupo de sensores al grupo perteneciente a los sensores LDR y UVA. Debido a que necesitan un tiempo de muestreo más regular, la luz y el índice UV pueden cambiar rápidamente, por lo que es aconsejable medirlo con más frecuencia. Si se desease modificar el valor del tiempo de muestreo, referirse a CsADC definido en el archivo Systemsymbols.h y poner un valor de cuentas adecuado para la función que se deseé desempeñar.

8. WDT.

El WatchDogTimer es un contador regresivo que si se deja llegar a 0 provoca un reinicio del sistema. Así, si el sistema es bloqueado, el WDT lo desbloquea reiniciando el sistema. Cabe destacar que la hora debe de ser ajustada de nuevo y todas las variables que el usuario haya modificado.

9. RTC.

El RTC es un reloj en tiempo real que sirve para tener un reloj en la estación meteorológica. Es un contador que se utiliza para mostrar por la página WEB y por pantalla la hora actual de las medidas.

10. UART.

Es un módulo que se encarga de transmitir y recibir información por el protocolo de comunicación seria asíncrona universal. Sirve como interfaz para que el usuario pueda recibir información de la estación y mandar comandos a la misma.

11. ADC.

El módulo ADC, es un módulo integrado en la placa que tiene la capacidad de codificar señales analógicas y convertirlas a un formato digital, esto permite leer señales analógicas, en nuestro caso, el sensor LDR y el UVA.

12. DMA + DAC.

Estos módulos actúan unidos. El DMA se encarga de liberar carga computacional al procesador, transfiriendo el valor de los datos del audio al siguiente módulo, el DAC, mediante una señal de inicio de conversión. Mientras que el DAC es un módulo que genera una señal analógica en función de un valor digital recibido. El funcionamiento reside en activar el DMA cuando se deseé producir audio y desactivarlo cuando este termine, produciendo las transferencias de datos con el DMA y no con el procesador.

Se recuerda que el Timer encargado de desactivar la transferencia es el Timer 1.

Capítulo 4: Implementación del sistema.

En este capítulo veremos cómo se implementan tanto en software como en hardware los sistemas y módulos anteriormente mencionados en el *Capítulo 3: Estructura del sistema*. Intentando resolver tantos objetivos como nos sea posible a lo largo del capítulo. Además, se explica el fundamento de la solución adoptada.

En el Anexo I se incluye un esquemático de todo el **conexionado hardware del sistema**, cumpliendo las características del *Capítulo 2: Componentes del sistema*.

En el Anexo II se incluye el **código fuente**. Para leer las descripciones de las funciones y variables, leer en el código comentado de las secciones.

Carpeta MiniDK-2.

En esta carpeta se incluyen todas las carpetas del proyecto.

Carpeta README.

Contiene información del proyecto.

Carpeta Startup.

Contiene los ficheros de inicialización del sistema, así como librerías internas que utiliza el LPC1768 para iniciar. Incluyen las librerías:

- Startup LPC17XX.s: Inicializa el sistema. La única variable modificada en este archivo es Stack_Size y se le ha otorgado un valor de 0x200.
- Core cm3.c: Inicializa el core.
- System LPC17XX.c: Ajusta variables y define macros internos sobre todo de los relojes de la placa MiniDK-2.

Carpeta Principal.

Contiene el programa principal en su interior, dentro del archivo main.c. En el programa principal se crean variables globales y punteros a dichas variables globales, para utilizarlas de manera más sencilla.

Dentro del programa principal sólo es necesario llamar a la función de configuración del sistema. Luego es necesario crear un bucle infinito para llamar a la función mainLoop() que está definida en Statechart.c y a la función mantenerTCP() que se encuentra en el archivo HTTP_SOURCE.c. La función mainLoop() es la máquina de estados que rige la interfaz de usuario por el LCD y mantenerTCP() mantiene la conexión TCP mediante la librería de TCP haciendo un llamado a la función main_TcpNet().

Cabe destacar que no todas las variables importadas se encuentran en este archivo, dado que sólo residen las de mayor importancia como pueden ser los datos almacenados y señalizadores del sistema.

Carpeta Menu.

En esta carpeta se encuentra la librería TouchPanel.c y el archivo responsable de generar la máquina de estados, Satatechart.c. En la Figura 1. Podemos observar la máquina de estados implementada en el archivo. La función `_mainLoop()` se encarga de generar la máquina de estados, utilizando la variable global `ESATADO` como una variable de 8 bits que almacena el valor del estado que se quiere asignar. En este archivo se declara la variable `MODIFICABLES`, que contiene todos los valores modificables por el usuario en cualquiera de sus interfaces mencionados anteriormente.

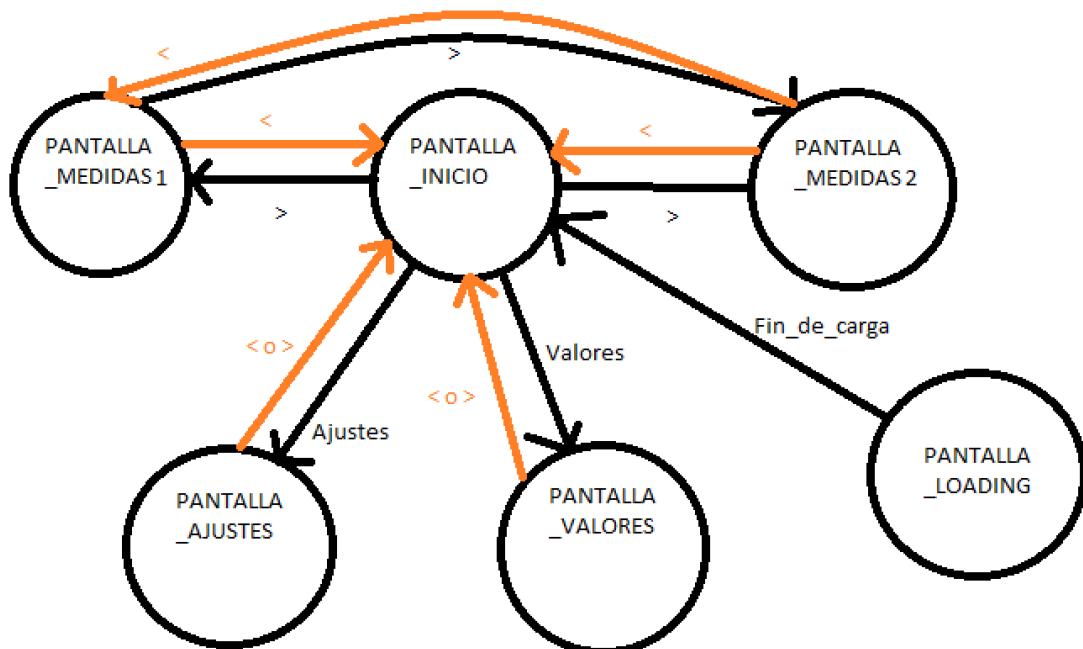


Figura 1.- Modelo de pantallas.

Además, se definen variables como `Modo_brillo`, `Modo_energetico` que guardan el estado del brillo, es decir, cualquiera de sus 4 niveles de brillo y el modo de ahorro de energía que se encuentra el sistema. La variable `pressedTouchPannel` refleja si se ha tocado la pantalla, para que se ilumine cuando esta, en su modo de ultra ahorro de energía; `ULPM`, se haya apagado.

Las variables internas `_brilloFade` y `_brilloAuto`, son variables que guardan si el sistema debe de apagar la pantalla o no (`_brilloFade`) y si deben de leer el LDR para proporcionar un control automático de brillo (`_brilloAuto`). El sistema cada iteración de la función principal, comprueba si `_brilloAuto` con ayuda del SysTickTimer cada 100ms si el estado es de brillo automático y actualiza el módulo PWM para proporcionar el pulso adecuado para generar un nivel de brillo que oscila entre el 20% y el 80% de luminosidad. Esto lo hace refiriéndose a la tabla en el archivo `LUT.c`, contretamente en el array `Brillo2ciclo_LDR`, que traduce el brillo al porcentaje del ciclo de trabajo del módulo PWM1.

Se definen variables de tipo `zona` que son variables que necesita la librería para generar la interfaz. Se le pasa como parámetro el `exe X e Y` y la longitud en `X` y en `Y` que ocupa el recuadro generado. Además, dichas variables admiten color y texto tanto como de la línea como del texto.

En el archivo se encuentran las siguientes funciones:

- _pintaX_(): pinta la pantalla X.
Dentro de estas funciones puede haber más estados que representen un tipo de color, por ejemplo, existen unas barras de colores que son interfaces gráficas para que el usuario sepa la temperatura que hace en función de los valores máximos y mínimos de la misma, cambiando el color en función del porcentaje.
Dentro de estas funciones se utilizan actualizadores, estos actualizadores sirven para actualizar el valor de los datos, es decir, mostrarlos por pantalla una vez sean medidos. Cuando un módulo termine de realizar una medida, este módulo debe de hacer un toogle al actualizador correspondiente para indicarle a la máquina de estados que se ha realizado un cambio en el valor de la medida y así poder mostrarla por pantalla.
- _configuraLCD_(): configura el LCD.
- squareButton(): Dibuja un botón cuadrado en función de la zona, los colores y un texto.
- squareBox(): Dibuja una caja en una zona de un color.
- checkTouchPanel(): Comprueba si se ha tocado un botón, además si se ha tocado cualquier zona de la pantalla, resetea el contador que apaga la pantalla en ULPM.
- ZoneNewPressed(): Comprueba si se ha tocado cierta zona de la pantalla, sirve para saber qué botón se ha pulsado. Si se toca la pantalla cuando esta ha sido apagada (esto sucede dentro del manejador del SysTickTimer) esta se recupera con un ciclo del 60% de PWM, es decir al 60% del brillo. Además, recupera las variables anteriormente mencionadas y resetea el contadorLuz, que es el contador encargado de controlar que no se supere el tiempo de apagar pantalla (este tiempo es modificable).

Dentro de cada pantalla, podemos visualizar o modificar variables.

- En la pantalla de medidas 1: Podemos ver los siguientes valores.
 - Velocidad del viento en m/s.
 - Humedad del aire en % relativo.
 - Claridad recibida en LUX.
 - Índice UV en UVs.
 - La altura respecto al nivel del mar en metros.
- En la pantalla de medidas 2:
 - Temperatura en °C.
 - Presión en mBar.
 - Gráfico de barras horizontales de colores para los valores máximos y mínimos de cada una.
- En la pantalla de ajustes: Podemos visualizar la IP del servidor WEB y además modificar las horas, minutos y segundos del reloj, además del día.
- En la pantalla de valores: Podemos modificar los valores de presión y temperatura máximos y mínimos que generan alarma, además de poder modificar si el servo representa la temperatura o la presión. El botón tendrá letras verdes para la presión y letras rojas para la temperatura. En dicho botón se representan los rangos máximos y mínimos de temperatura y presión a representar.

- En la pantalla de inicio: Podemos, además de acceder a otras pantallas, modificar el brillo, siendo 1 el mínimo y 4 el máximo. Además, está la selección de A que es brillo automático, lo que automáticamente le hace cambiar a LPM (Low power mode). También podemos grabar y reproducir audio con los botones de load y play, que lanzan las señales de grabar y reproducir audio. Desde esta pantalla podemos modificar el modo de ahorro de energía.

Importante:

Dentro de la librería TouchPanel.c, podemos encontrar una matriz denominada como matrix. Esta matriz contiene unos valores de calibración para calibrar la pantalla, cada una tiene sus valores y deberían de ser ajustados por el usuario. En caso de querer calibrar manualmente la pantalla cada vez que se inicia el sistema, descimentar la línea 68 del archivo configura.c, donde dice:
//TouchPanel_Calibrate();.

Carpeta Setup.

En esta carpeta se encuentra el archivo de configuración configura.c. Se encarga de mandar a pintar las diversas pantallas de carga, de iniciar las variables y de mandar a configurar todos los módulos. Además, inicia el módulo PWM con el 90% de ciclo para el servo y 50% de brillo. Inicia las variables a cero y las modificables con el valor de las macros.

Temperatura: [-10, 50] °C.

Presión: [500, 1500]mBar.

Tiempo de brillo: [10]seg.

Variable medida: Temperatura.

Carpeta I2C.

En esta carpeta hay una librería de I2C (I2Clib.c) que contiene todo lo necesario para realizar las llamadas a las funciones de comunicación de I2C. (Mandar un byte, recibir un byte, recibir una dirección). Luego está el archivo que se encarga de utilizar la librería de I2C para leer el sensor BMP180.

En el archivo I2C.c podemos encontrar funciones que se encargan de usar la librería para leer los registros de acuerdo con la datasheet del fabricante (ver la hoja de datos de BMP180). Utiliza la lectura de ciertos registros del sensor para calibrarlo y luego llama a medirBMP(), que utiliza un algoritmo que leyendo ciertos registros y trabajando con ellos podemos obtener la temperatura y la presión. Cuando se desee medir, solo hace falta llamar a medirBMP() para que automáticamente guarde en la variable DATOS la presión y temperatura. Además, se calcula la altura sobre el nivel del mar con una fórmula proporcionada por el fabricante.

Nota: La función procesarDato() admite parámetros que en versiones anteriores servían para elegir presión o temperatura, en la versión final se incluyen la altura, temperatura y presión. Se ha decidido tomar

la temperatura con este sensor, dado que es de mayor calidad que el DHT22.

Carpeta Anemómetro.

En esta carpeta se encuentra Anemometro.c que incluye dos funciones, una de configuración y otra de medición. El anemómetro funciona de la siguiente manera: necesita una resistencia de pull up (vale la configuración estándar de la placa), dado que cada cuarto de vuelta del ciclo de giro cortocircuita ambas patillas, por lo que la conexión debe de ser input-masa, la resistencia de pull up debe de estar en el lado de input. Cada vuelta genera 2 pulsos, dado que cortocircuita en dos cuartos de ciclo, por lo que el anemómetro genera 2 pulsos por vuelta.

Dicho esto, podemos utilizar el módulo CAPTURE del Timer 1 para calcular la diferencia de tiempos entre el inicio o fin de ambos flancos (subida o bajada a elegir). Podemos calcular la velocidad que recorre el viento de la siguiente manera:

$$Distancia_{recorrida} = Perímetro_{anemómetro} = \pi \cdot D_{anemómetro}$$

$$Tiempo_{medidoCapture} = \frac{Perímetro_{anemómetro}}{Pulsos_{vuelta} \cdot Velocidad_{viento}}$$

Podemos esperar a que se realicen dos pulsos y luego medir como si se hubiese realizado un pulso para reducir carga computacional:

$$Velocidad_{viento} = \frac{Perímetro_{anemómetro}}{Diferencia_{pulsos}} = \pi \cdot \frac{D}{T_{reloj} \cdot (CAPTURE_x - CAPTURE_{x-1})}$$

Medidas realizadas estiman el diámetro del anemómetro en **14 centímetros**.

Carpeta ADC.

En esta carpeta están los archivos necesarios para hacer funcionar todo el módulo del ADC y leer los sensores LDR y UVA30A. Además, se encuentra la lectura del micrófono. Los archivos inherentes son Ldr.c, uFono.c y UVA30A.c.

En el archivo LDR.c contiene toda la configuración del ADC, que se ejecuta en modo BURST. Dicha configuración realiza conversiones a todos los canales activados, que en este caso es el del LDR y el del UVA, podrían haberse metido más sensores sin ningún problema y sin aumentar la carga computacional, solo que se utilizan más pines del ADC en este modo BURST. El valor de CLKDIV es el máximo que se puede poner (0xFF), por dejar tiempo al ADC a que convierta con tranquilidad, podría reducirse dicho valor hasta cierto punto.

La interrupción que hace leer los registros del ADC es la penúltima, dado que hay un 'lag' de una conversión desde que se muestrea hasta que se produce la llamada al Handler del ADC. Cuando el Handler entra en acción, el último canal ya ha sido convertido. El penúltimo canal es del del LDR (1), por lo que la interrupción la provoca este fin de conversión.

En el archivo UVA30A.c sólo se configura el ADC para permitir un canal más en modo BURST, si no se ha configurado el LDR, el configurador del

UVA30A llama al del LDR primero, dado que hay que configurar todo el ADC primero para configurar que el UVA entre en modo BURST. El canal asociado al UVA es el 2.

Una vez teniendo las medidas en modo BURST, cuando se deseé grabar el micrófono, se lanzará una señal que lo indique. Esta señal es la función lanzaUFONO(), que configura todo el ADC y el Timer 1 para que se obtengan muestras cada MATCH0, pare la conversión de audio y vuelva a medir en modo BURST. Cabe destacar que las medidas se bloquean mientras el modo audio está activado. Este bloqueo lo representa la variable YaPuedesMedir, que además se utiliza en el Handler del ADC para saber en qué modo (si BURST o audio) se encuentra el sistema. Una vez alcanzado el número de muestras del audio, se lanza la desactivación del ADC en modo audio y se recupera el contexto del ADC para el modo BURST, además de activar los actualizadores y de marcar que ya se puede medir en modo BURST.

La configuración del micrófono configura el ADC para que cada 8000kHz se obtenga una muestra de audio y se guarde en la variable AUDIO, que ocupa un total de 8 bits por muestra a 16000 bytes. 16kB de audio.

Dentro del manejador de la interrupción del ADC podemos encontrar dos modos:

- YaPuedesMedir = 0: Mete las muestras recibidas por el canal 0 al array definido para el AUDIO, cuando se alcanza el número de muestras MUESTRAS_AUDIO, se reconfigura el ADC para medir en modo BURST.
- YaPuedesMedir = 1: Lee los canales 1 y 2 del ADC por cada interrupción.
- Para obtener el brillo: Se lee el canal 1 del ADC, se calcula el porcentaje relativo de voltios del canal referido a 3.3V; luego se calcula su resistencia y se pasa dicho valor a una look-up-table para traducir dicho valor a LUX.

$$R_{LDR} = R_{pull} \cdot \frac{\frac{ADC_1}{0xFFFF}}{1 - \frac{ADC_1}{0xFFFF}}$$

- Para obtener el índice UV: Es una función lineal, por lo que se obtiene el porcentaje relativo respecto a 3.3V del canal 2 de entrada y se multiplica por el máximo.

$$Indice_{UV} = Indice_{maximo} \cdot 3.3V \cdot \frac{ADC_2}{0xFFFF}$$

La resistencia de pull-up escogida son 70kOhm, debido a que linealiza mucho su respuesta. Es muy importante escoger una resistencia de pull-up adecuada, debido a que una pequeña variación en el error podría producir una gran variación de la resistencia. A continuación, gráficas que muestran dicho comportamiento.

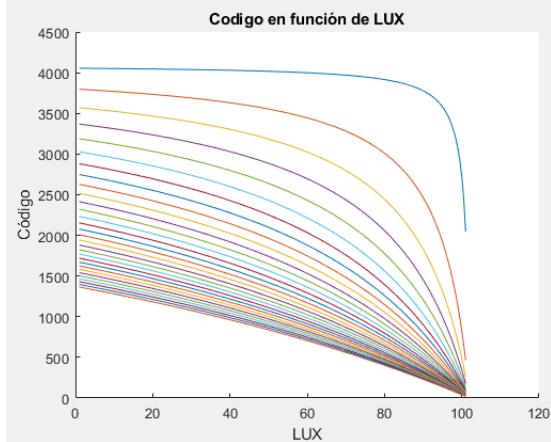


Figura 2.- Código en función de LUX.

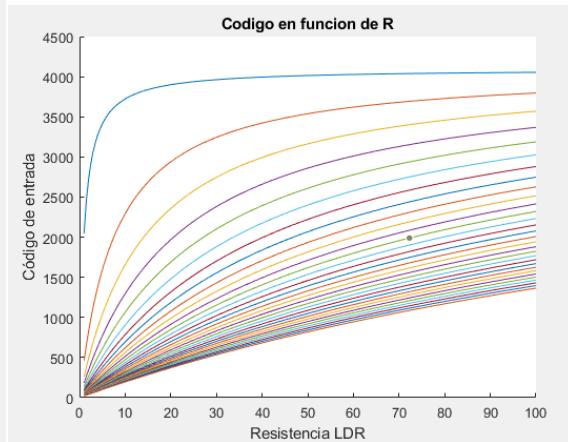


Figura 3.- 2 En función de los ohm.

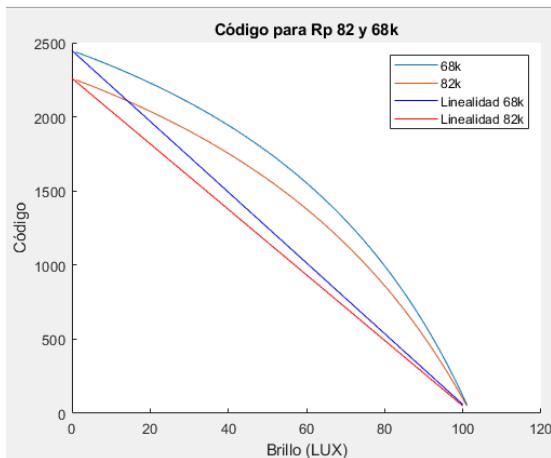


Figura 4.- Linealidad a 68 y 82k.

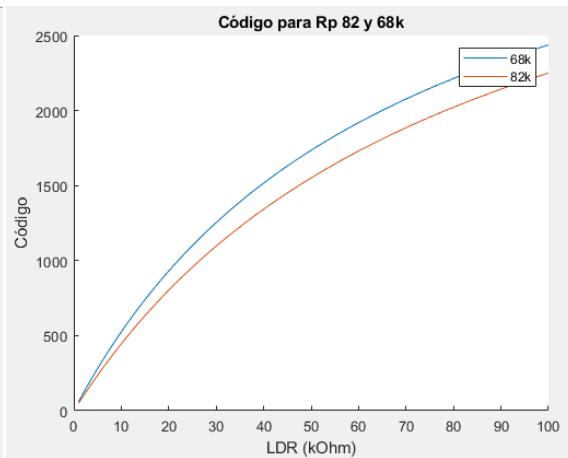


Figura 5.- 4 En función de los ohm.

Se han escogido valores de 68kohm debido a que son valores estándar y tienen una buena respuesta en la linealidad sin sacrificar mucho porcentaje del código que se ha recibido. Perdemos la mitad del rango, que podría arreglarse seleccionando una fuente de voltaje adecuada, pero ganamos linealidad y por ende, más margen de error.

Carpeta OneWire.

Pese a que el código de este apartado es muy claro y este sí está bien comentado, explico por encima las funciones que realiza.

En la datasheet del DTH22 especifican un protocolo de lectura de monofilo, lo que requiere que exista una configuración dinámica del pin de datos, para mandar una señal de inicio de conversión y configurar dicho GPIO como entrada. He utilizado GPIO como forma de lectura utilizando esperas activas del orden de los microsegundos ayudándome de el contador del Timer 3, dado que este no lo toca ningún otro módulo y estaba sin usar. Dado que realizando pruebas este sensor tiene una cantidad de ruido importante, utilizar el modo capture era realmente un desafío. Por lo que he optado utilizar esperas activas, que no dañan a penas la ejecutabilidad del programa dado que la relación ejecución/uso (0.076% del tiempo) es muy baja; se llama cada 5 segundos a la función en el Handler del Timer 0 y ocupa 3.8ms de media en realizarse. Algo parecido pasa con I2C que utiliza librerías de espera activa.

En este archivo se ejecuta el protocolo y se almacena en una variable local a `mideTemperatura()`, que es la señal de inicio de medida, que se llama Rx. Se divide en 4 etapas, marcadas en el código en el marcador `@state`:

- Inicio de medida: Se configura el pin como salida de datos y se crea un pull down, luego se configura como entrada.
- Señal de respuesta: Para esta señal, se espera recibir 50us de nivel bajo más otros 50 de nivel alto, con un margen de 10us. Esto se ejecuta sobre la función `compruebaRespuesta()`. No se tiene en cuenta el tiempo que tarda el sensor en volver a poner el tiempo en alto, por lo que hay que esperar a que lo haga, con un valor de timeout de 45us.
- Lectura de datos: Se leen los datos en función de la duración del flanko de subida, si se excede el tiempo de timeout, se aborta la medición.
- Comprobación de checksum: Se lee el último byte y se compara con el algoritmo de checksum del fabricante. Si coincide, se insertan los nuevos datos, si no coincide se aborta la medida y se espera a la siguiente llamada.

Existe la función `reinicia cuenta`, que devuelve el valor del contador TC, que está reducido por un valor de 25, lo que provoca cuentas cada 1 us, y acto seguido lo reinicia. Por eso podemos obtener el valor del tiempo fácilmente.

Cabe destacar que si no se resuelven los márgenes, se sale de la función con un código de salida de 1 en la función `compruebaRespuesta()` y un código de 0 para la función `LeerByte()`. Los valores de timeout de datos son de 100us, si no se ha recibido un cambio en ese periodo, se sale de la medición y queda abortada.

Carpeta PWM.

En esta carpeta se encuentra todo lo referente al módulo PWM, residente en el archivo PWM.c. Existen dos funciones: `_configuraPWM()` y `modificaPulso()`. El código está comentado y explica todos los input y output de las funciones.

- La función de configuración lo que hace es recibir una frecuencia para el pulso de PWM y los puertos y pines que se quieren activar. En nuestro caso sólo los pines 2_1 y 6_1, uno para el servomotor y otro para el brillo del LCD.
 - La función modifica pulso, lo que hace es modificar el ciclo del pulso PWM, para cambiar su frecuencia, hay que reconfigurarla. Las entradas son:
 - PWMn: El pin PWM seleccionado.
 - Modo: Indica si el siguiente argumento es para el servo o para el LCD.
 - Ciclo: Si es modo ciclo, se utiliza este ciclo de trabajo del módulo PWM.
 - Grados: Si es modo servo, se utiliza este argumento como los grados a los que debe inclinarse la aguja del servo.
- PARA EL MODO SERVO:
- Mínimo: Valor mínimo del pulso a nivel alto en segundos.
 - Máximo: Valor máximo del pulso a nivel alto en segundos.

IMPORTANTE: Cabe destacar que el servomotor utiliza ciclos de 5V, cuando la placa LPC1768 utiliza ciclos de 3.3V. Esto es un problema dado que el servo parece utilizar la potencia del pulso y no precisamente su duración, es por ello por lo que no funciona de 0 a 180° y funciona de 45° a 135°. Esto puede corregirse modificando los valores de duración mínimo y máximo del pulso.

Experimentalmente se ha decidido que los valores por los que hay que multiplicar dichas constantes sean las siguientes:

- KMX: Constante del máximo = 1.3
- KMN: Constante del mínimo = 0.6

Lo que implica proporcionar más potencia al servo con la señal de 3.3V, dado que el mínimo disminuye y el máximo aumenta.

Las fórmulas utilizadas son, en función del ciclo o los grados, las siguientes:

$$MR_x = MR_0 \cdot \frac{Ciclo}{100}$$

$$MR_x = \left(Maximo + (Maximo - Minimo) \cdot \frac{Grados}{180} \right) \cdot \frac{1}{T_{reloj}} - 1$$

Carpeta DAC.

En esta carpeta se encuentra todo lo referente a la señalización y configuración del DAC. Para configurarlo, se utiliza el prototipo de función que se ha utilizado hasta ahora: `_configuraDAC_()`, que incluye la puesta a nivel alto de los LED de la placa, cuando están encendidos significa que está disponible la escritura de audio. Lo mismo para la lectura de audio. Esta función únicamente configura esos pines dado que el DAC se utiliza con el DMA y por ende se configura junto a él.

Las funciones `activarDac()` y `desactivarDac()` realizan la función de activación o desactivación del DAC.

Al activar el DAC, se activa el canal correspondiente del DMA, es decir, se manda una señal de inicio. Además, se configura un Timer para llamar a `desactivarDac()` pasados los 2 segundos. Se apaga el LED de la placa que señala escritura de audio.

Para desactivar el DAC, se señala al sistema con el actualizador que se ha acabado el tiempo del DAC y desactiva el canal del DMA, es decir, se manda una señal al DMA de fin. Además, se enciende el led correspondiente a escritura de audio y se escribe un valor de 0 en el DAC, dado que no hay señal de salida una vez finalizada la conversión.

Carpeta Database.

En esta carpeta se encuentran todos los archivos referentes a look-up-tables (LUT.c) y transferencia de memoria (DMA.c). En el archivo LUT.c podemos ver que reside la función `goto_LUT` que admite los siguientes parámetros:

- Variable: Variable de entrada.
- LUTn: El tipo de tabla que vamos a usar.
- Ret_x: Son un grupo de parámetros por los que hay que señalar una dirección en la que se va a guardar el resultado. Los tipos admisibles son de enteros sin signo de 8 a 64 bits y flotantes.

Las tablas que existen son Brillo_LDR, que traduce el valor en ohmios medido por el ADC a brillo, y Brillo2Ciclo_LDR, que traduce la variable brillo o claridad expresada en LUX a un ciclo de trabajo para controlar el servomotor. Sólo el LDR utiliza estas tablas.

En el archivo de DMA.c existen tres funciones y las tres son de configuración.

La primera función es la función `_configuraDMA_()`, que genera un tono de 32 muestras y activa la salida analógica (P0.26). Luego hace una llamada a la función `_configuraTono_()`, que deja configurado el audio pregrabado, que es un tono de 400Hz. Este es activado si se supera el valor mínimo de la alarma. Si se superase el valor máximo, se llamaría a la función `_configuraAudio_()`, esta función configura el DMA para reproducir el audio grabado. En ambos casos las muestras son de 8 bits y se crea una estructura (LLI0) que contiene las direcciones de origen y destino de las transferencias, así como que el destino se incremente y el número de muestras a transferir. Además, en ambos se configura que la transferencia es de memoria a periférico y que se transfiera al DAC. Dentro del DAC, hay que configurar que las transferencias son realizadas vía DMA.

La frecuencia a la que se transfieren las muestras son las siguientes:

- Para el tono:

$$T_{dma} = \frac{Num_{muestras}}{400Hz}$$

- Para el audio:

$$T_{dma} = \frac{Duracion_{audio}}{F_s} = 4kHz$$

Las funciones `_configuraTono()` y `_configuraAudio()` son llamadas si se exceden los valores límite de alarma, para que el DMA lance la transferencia ya sea por audio o por alarma pregrabada. Tras configurar el DMA así, se lanza un Timer (Timer 1), que desactiva el canal pasados 2 segundos.

Carpeta WDT.

Esta carpeta contiene los archivos referentes al WatchDogTimer, en WDT.c. Existen dos funciones:

- `_configuraWDT()`: Se encarga de configurar el WDT, la configuración seleccionada es de un timeout de 10 segundos utilizando el reloj de Fclk/4. La acción a realizar tras vencer el temporizador es de reiniciar el sistema, por lo que es muy importante estar alimentando el WTD continuamente.
- `alimentaWDT()`: Es la función que reinicia el contador del WDT. Se escribe en su registro de alimentación el código 1 y acto seguido el código 2 para reiniciarlo y evitar el reinicio del sistema. El código 1 se define como 0xAA y el código 2 se define como 0x55.

Esto se hace cada ciclo de `_mainLoop()`. Si no se entra a dicha función en 10 segundos, este temporizador reinicia el sistema. Esto evita bloqueos indeseados.

Carpeta UART.

Esta carpeta utiliza una librería proporcionada por la signatura SEDA de la UAH. Y el archivo UART0.c.

Dentro del archivo UART0.c podemos observar dos funciones, una de configuración y una de procesado del comando.

Respecto a la configuración, la función hace referencia a la librería para configurar, con 9600 baudios, el UART0 que es la conexión por USB (controlador de UART de la placa). En la librería se tiene una función para el cálculo de los registros de configuración de los baudios con un método iterativo. También se encuentra el manejador de la interrupción de UART0, que se activa cada byte recibido y por cada vez que el buffer de transmisión esté lleno.

Cuando se envía una cadena, hay que usar la función de la librería `tx_cadena_UART0`.

Para procesar el comando recibido, se guarda cada byte en un buffer (bufferx) y al recibir el carácter 13 (\r) se considera como acabado y se llama a la función de procesar comando: procesarComando(char * string).

Esta función tiene una máquina de estados implementada como se muestra en la figura 6. Donde se espera a recibir un tipo de comando para ejecutar una determinada acción. Los comandos pueden observarse en el Capítulo 6: Manual de usuario, en el apartado de UART.

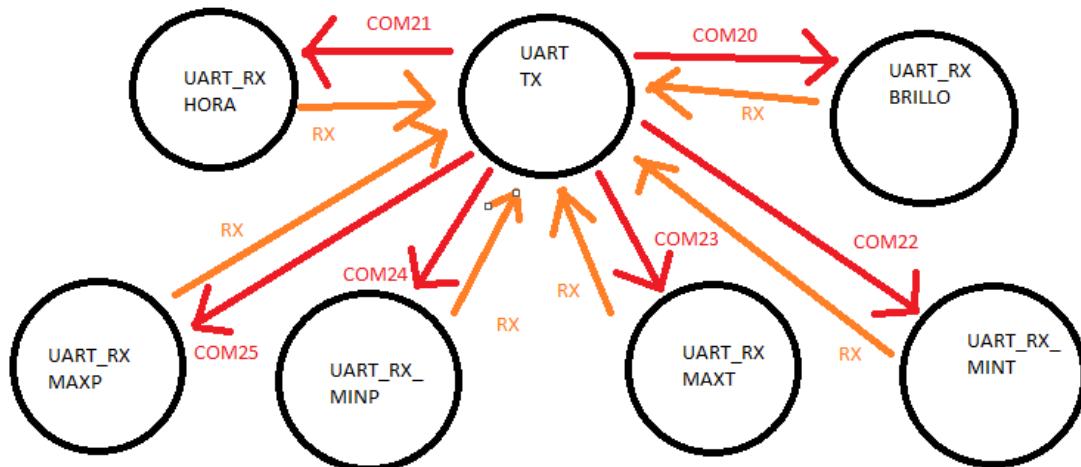


Figura 6.- Diagrama de estados de UART0.

Una vez obtenido el comando de modificación necesario, se procede a introducir el valor de la variable a modificar, el nombre del estado hace alusión a la variable a modificar. (Ver manual de usuario para más información sobre los comandos y el formato de salida).

La cadena a transmitir se guarda en un buffer y luego se transmite (UART0_BUFFER_TX[]) con la función de la librería (tx_cadena_UART0).

Carpeta TCP/IP.

En esta carpeta se encuentran los archivos referentes a la pila de protocolos TCP/IP que forman en servidor web. Se incluyen dos librerías: TCP_CM3.lib (la del fabricante) y EMAC_LPC17XX_LAN8720.c (la que se encarga del servicio sobre ethernet). Además, existe el archivo WEB.imp que guarda la orden de compilación:

```
index.cgi to WEB.C nopr root (.../TCPIP)
```

Esta orden manda compilar el archivo CGI a otro llamado WEB.C que contiene la página web, convirtiendo el archivo de marcado en un objeto de C.

Luego se encuentran los siguientes archivos:

- `HTTP_SOURCE.c`: Contiene la configuración de la página web (`_configuraWEB_()`) que llama a la función `init_TcpNet()` de la librería, que inicia la conexión TCP. Además, encapsula en `_mantenerTCP_()` a la función de la librería `main_TcpNet()` que es la que hay que llamar con cierta frecuencia (se encuentra en el programa `main.c`).
- `NET_Config.c`: Contiene la configuración de la conexión TCP, lo más relativo de este fichero se encuentra definido en el archivo `miGlobal.h`, que define la IP del servidor, el Gateway y la máscara subred. En este archivo se define con `DEFAULT` la ip 192.168.1.120 con un Gateway de 192.168.1.20 y una máscara subred de 255.255.255.0. Además, el usuario y la contraseña se definen como `user` y `Alver` respectivamente.
- `Index.cgi`: contiene el lenguaje de marcado en formato CGI y html de la página web a compilar. Esta pasa a convertirse en `WEB.C`.
- `HTTP_CGI.c`: Se encarga de la parte automática de la página web, mediante llamadas a callbacks y al método GET.
- **Callbacks**: se encuentra en la función `cgi_func`, función llamada cuando una línea de `index.cgi` empieza por la letra c. Se utiliza un espacio y una letra para determinar la acción de dicha callback:
 - t para temperatura.
 - v para velocidad del viento.
 - p para presión.
 - h para humedad.
 - i para índice UV.
 - b para brillo.
 - a para altitud.
 - X no usado.
 - Y no usado.
 - A año.
 - M mes.
 - D día.
 - H horas.
 - T minutos.
 - S segundos.Así podemos sacar por este callback y la función `sprintf` la variable en cuestión y poder representarla en el formato de la página WEB.
- **GET**: Se utiliza para enviar comandos a la estación, se define una tabla en la que se pueden escribir valores para enviar a la estación, todos los modificables comentados con anterioridad exceptuando la hora, que es modificable por pantalla y por UART. Primero se lee de la cadena si coincide la variable con las siguientes cadenas:
 - `Tmin=` para temperatura mínima.
 - `Tmax=` para temperatura máxima.
 - `Pmin=` para presión mínima.
 - `Pmax=` para presión máxima.
 - `Vart=` para seleccionar temperatura.
 - `Varp=` para seleccionar presión.Si coincide con dicha string, se pasa a procesar el número que viene a continuación y se guarda en la variable en el campo de la estructura correspondiente `MODIFICABLES.[campo]`.

Carpeta RTC.

Dentro de esta carpeta, se encuentra el archivo RTC.c, lo que contiene la configuración y el manejador de la interrupción del RTC. Se configura con `_configuraRTC_()` y se incicia el reloj a principios de año de 2020. Se configura para interrumpir cada segundo para actualizar una variable llamada `Clock`, que contiene una string con el valor de la fecha actual, además tiene un contador de segundos.

Carpeta Timers.

En esta carpeta se encuentra todo lo referido a los Timers, es la carpeta más importante dado que es donde se hacen todas las llamadas a las funciones de las demás carpetas. Dentro de esta carpeta se encuentran las configuraciones y los manejadores de interrupción de los temporizadores y del SysTickTimer.

- `_configuraSysTick_()`: Es la función que configura el SysTick para interrumpir cada 100ms.
- `SysTick_Handler()`: En esta función se hace una llamada a una función necesaria para la librería TCP/IP: `timer_Tick()`. Además, incrementa un contador (`contadorLUZ`) cada 100ms. Si este alcanza el tiempo límite sin tocar la pantalla (recuerdo que tocar la pantalla reinicia dicho contador) y este se encuentra en ULPM, esta desactiva el brillo automático y pasa a mandar una señal PWM de brillo del 1%. Por lo que se apaga, dejando un pequeño margen de luz del 1%.
- `_configuraTimer0_()`: Configura el Timer0 para interrumpir cada 0.5 segundos. En esta configuración se activan todos los timer a usar, aunque no el manejador de sus interrupciones.
- `Timer0_IRQHandler()`: En este manejador, tenemos tres funciones: medir el grupo 1 de sensores cada 0.5 segundos, medir el grupo 2 de sensores cada 5 segundos y actualizar el servo cada 0.5 segundos. Este temporizador controla el tiempo de muestreo.
- `Timer1_IRQHandler()`: Si interrumpe el MR1, se considera interrupción por fin de salida del DAC y se desactiva el mismo. Si interrumpe el modo capture del (CAP1.0), se considera que ha llegado un pulso del anemómetro y se llama a la función que lo mide. (Hay que tener en cuenta que quien activa el anemómetro es el Timer0 y se desactiva solo tras obtener una medida, si este no manda un pulso, la velocidad saldrá como 0 m/s indicando que no existe viento en ese momento).

Carpeta SDCARD.

Esta carpeta contiene los ficheros diskio.c, ff.c, SPI_MSD_Driver.c en caso de que en un futuro se quiera utilizar la tarjeta SD en la estación meteorológica. En esta versión no se incluye la tarjeta SD.

Carpeta CMSIS.

En esta carpeta se encuentra el software del fabricante CMSIS para utilizar en las librerías, varias de ellas utilizan los archivos de esta carpeta. Entre ellos se encuentran:

- lpc17xx_ssp.c
- lpc17xx_pinsel.c
- lpc17xx_gpio.c
- lpc17xx_clkpwr.c
- lpc17xx_libcfg_default.c

Carpeta GLCD.

En esta carpeta se encuentra la librería que se encarga de comunicar los pines de la placa con el TFT, utilizando funciones para mandar datos por dichos pines. (Ver esquema del Anexo I).

Capítulo 5: Pruebas, ejecutabilidad y conclusiones.

En este capítulo se explican los bugs que puede llegar a tener la estación meteorológica en esta versión con pruebas que se han realizado. Luego un estudio de ejecutabilidad y una recapitulación de objetivos logrados.

Pruebas.

Pruebas realizadas con la estación concluyen que puede haber los siguientes bugs:

- Aparición de números en las medidas no deseables: Cuando se realiza una medida de una variable que por un momento se ha excedido de un valor alto, puede que, si al no cambiar de pantalla el valor se reduce lo suficiente como para pasar de 3 a 2 dígitos, por ejemplo, el último dígito no sea borrado de la pantalla y parezca que sigue ahí. Esto es resuelto cambiando de pantalla. Puede observarse reduciendo valores de alarma de presión cómo la letra acaba por duplicado. El error viene de las funciones de la librería TouchPanel.c.
- Caída de la conexión TCP: Depende del tiempo de ejecución y de lo que se haya realizado hasta el momento, puede haber una caída de la conexión TCP y que la página WEB se caiga, esto se resuelve reiniciando el sistema. La media medida en la que se produce este fallo oscila entre los 9 o 10 minutos de ejecución del programa, habiendo veces que no sucede y habiendo veces que al minuto 2 está caída.
- Bloqueo del sistema por caída de la conexión TCP: Depende de si se ha caído la conexión TCP o no, puede bloquearse el sistema por un error que denomina la librería como ERROR_FREE_MEMORY, que es un error de liberación de memoria. Inmediatamente este error reinicia el sistema debido al WatchDogTimer. Si no se cae la conexión TCP, este error no sucede. La media medida de este error oscila las 2:30 horas de ejecución, aunque si no se produce la caída no se forma, y aun produciéndose puede tardar bastante más en caerse. Su record medido es de 6 horas después de caerse la conexión TCP a los 8 minutos de ejecución. Este error es resuelto automáticamente con el WatchDogTimer.

Las medidas de la estación son bastante precisas para la calidad y el precio de los sensores utilizados. Aunque las pruebas realizadas no han sido posibles con el sensor UVA30A, se ha probado a simular su comportamiento y el comportamiento es correcto.

Ejecutabilidad.

Con el simulador de Keil uVision4 podemos obtener la diferencia de tiempos entre dos breakpoints y medir las diferentes tareas que componen el sistema, en la Tabla 1 se muestran las tareas, el tiempo máximo que hay para realizarlas, el tiempo mínimo que tardan en ser requeridas y el tiempo máximo que tardan en ejecutarse.

Tarea	Deadline (D)	T.mín (T)	T.ejec (R)	Unidades
RTC	1000	1000	0.41	ms
Timer0	500	500	1.21+0.69	ms
Timer0*	5000	5000	10.32	ms
SysTick	100	100	0.92	ms
UART	1.04	1.04	0.12	ms
ADC (microfono)	0.125	0.125	0.10	ms
WDT	/	10000	108.729	ms
Statechart.				

Tabla 1.- Datos de ejecutabilidad del sistema.

El RTC al interrumpir cada segundo y actualizar el reloj, su D = T es de ese segundo.

El Timer0 interrumpe cada 0.5s, siendo cada 5s el mayor tiempo de ejecución que tiene, incluyendo el protocolo OneWire e I2C que son los que incluyen esperas activas, es por eso por lo que incluyo dos Timer0, el largo cada 5s y el corto cada 500ms.

El SysTick tiene 100ms para mantener abierta la pila de protocolos TCP.

El Timer1 no tiene tiempo mínimo de ejecución, dado que lee los pulsos del anemómetro y cuando necesite y se cierra, esto está incluido en el tiempo de ejecución del Timer0 cada 500ms.

UART tiene que mandar 9600 bits por segundo y lanza 11 bits, 8 de datos y 1 de start y stop y otro de paridad.

WDT reinicia al sistema cada 10s, por lo que hay que el tiempo máximo es de 10s.

El ADC tiene una frecuencia de muestreo de 8000kHz, por lo que su tiempo de muestreo es el máximo que tenemos para ejecutar las funciones dentro del manejador.

Nota: Los valores 0.001 son valores que no se han conseguido medir dado que el simulador los ha considerado despreciables, dado que rondan el orden de las 3 o 4 líneas de código.

Nada es bloqueante excepto el ADC, pero como se ha comentado con anterioridad, cuando se mide con el ADC la voz, se bloquean las medidas, por lo que el Timer0 no ejecuta código. La suma de la duración de todas las tareas no es mayor que los 100ms del SysTick. Luego está la tarea del Statechart, pero al no estar en una interrupción su prioridad es mínima, por lo que la máxima interrupción de Statechart sería de 12ms aproximadamente en caso de que todo interrumpiese a la vez. Cosa que el usuario no notaría para nada.

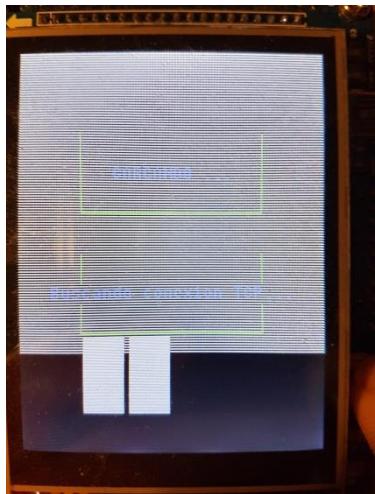
Conclusiones.

La práctica en su totalidad está completa, aunque no se han añadido muchos de los objetivos opcionales (sólo el RTC y el DMA y alguna mejora de la interfaz de pantalla de usuario), por lo general funciona muy bien en cuanto a lectura de sensores y funcionamiento de pantallas refiere. La UART también funciona muy bien, aunque personalmente me gustaría mejorar los bugs que tiene la pila de protocolos TCP/IP que bloquean el sistema tras 4 o 5 horas de ejecución, de hecho, si comento la línea de configuración de TCP/IP, el sistema no se bloquea nunca al no ser que se mande el comando KILL (ver manual de usuario).

La parte más costosa ha sido el ADC y el DMA junto con la página WEB, dado que he tenido muchos problemas al leer el audio y reproducirlo. La poca documentación sobre DMA que disponía me ha hecho tener que referirme a ejemplos colgados en internet y el largo tiempo que me ha llevado modificar pequeñas variables de la página web para que funcione ha sido determinante para centrarme en la calidad de esta. Además, no disponía ninguna experiencia en html ni en diseño web, por lo que la página es un poco simplista, aunque hace lo que se requiere de ella sin muchos problemas excepto ese bloqueo ocasional.

Capítulo 6: Manual de usuario.

En este capítulo se redacta lo necesario que debe conocer alguien que no sabe el comportamiento interno de la estación y solo desea utilizarla, transparentemente del comportamiento interno que la misma realice. A continuación, para las diferentes interfaces, se resume su manual.



TFT.

La pantalla al ser iniciada muestra por pantalla una interfaz que muestra el porcentaje de carga o pasos realizados para finalizar su configuración. Una vez realizada, se pasa a la pantalla de inicio. La primera barra indica que se ha iniciado el sistema, la segunda que se está iniciando la conexión TCP, la tercera que se están generando variables de alto costo computacional y la cuarta que se están iniciando los módulos del sistema.

Figura 1.- Carga.

Pantalla de inicio:

En esta pantalla se puede observar el reloj en la parte superior junto a dos flechas que permiten desplazarnos a las pantallas de medidas. Tras iniciar el sistema la fecha predeterminada son las 0:0:0 del día 1/1/2020. Más abajo podemos ver el botón de valores y el de ajustes que nos mandan a dichas pantallas.

Debajo de Nivel de brillo podemos seleccionar el nivel de brillo de la pantalla, del 1 al 4 en orden de brillo ascendente y automático a la derecha del todo. Presionar el botón de automático es sinónimo de pasar a modo de bajo consumo (LPM o LP). Si presionamos los botones HP automáticamente se selecciona el brillo 4, y seleccionando cualquier otro nivel de brillo se pasa al modo de alto consumo (HPM o HP). Si se presiona el botón de ULP se pasa al modo de muy bajo consumo, que, además de tener brillo automático, se apaga la pantalla si en un tiempo determinado y modificable no es utilizada por el usuario.

En esta pantalla se encuentra el botón de grabar y reproducir audio (Load en la esquina derecha y Play en la esquina izquierda respectivamente).

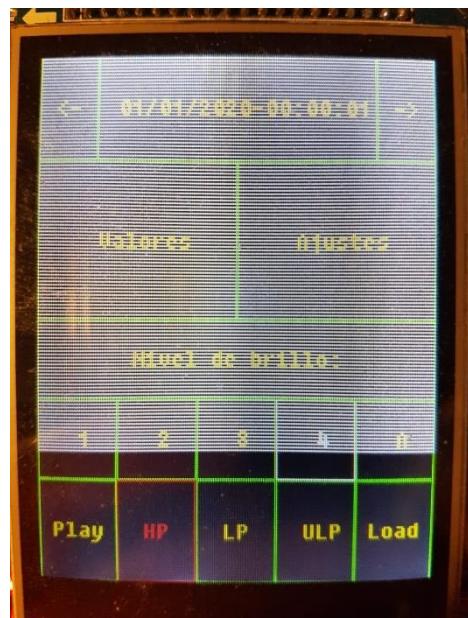


Figura 2.- Inicio.



Figura 3.- Medidas (1).

Pantalla de medidas 1:

En esta pantalla podemos observar las medidas actuales, es decir, las últimas tomadas por la estación. En esta pantalla podemos observar las variables de altitud en metros, velocidad del viento en metros por segundo, humedad en porcentaje relativo, claridad en LUX e índice UV en UVs.

Además, podemos observar el reloj en la misma posición que la pantalla de inicio con las dos flechas de mover la pantalla, por lo que la interfaz superior sigue siendo la misma para cualquiera de las pantallas en la que nos encontramos.



Figura 4.- Medidas (2)



Figura 5.- Ajustes.

Pantalla de ajustes:

En esta pantalla se puede modificar fecha y la hora pulsando los botones de incremento y decremento de la fila correspondiente (+ y - respectivamente). Además, podemos visualizar la IP que utiliza el servidor en el mismo momento que estamos visualizando el valor.

El valor por defecto de la dirección IP del servidor es 192.168.1.120, pero si es modificado, se mostrará el valor modificado para que el usuario sepa en todo momento la dirección a la que referirse.

Para retroceder a la pantalla de inicio basta ocn pulsar cualquier flecha de las situadas en las esquinas superiores.

Pantalla de valores:

En esta pantalla se pueden modificar los valores máximo y mínimo de la estación meteorológica. Tanto de alarma de presión como de alarma de temperatura, valores máximos y mínimos. Pulsando las interfaces de incremento y decremento (+ y - respectivamente) podemos modificar dichos valore. Al final de la pantalla podemos observar los límites mencionados.

Al lado de Pres (que indica presión) vemos dos números, el de la izquierda es el valor mínimo de alarma y el de la derecha el valor máximo.

Al lado de Temp (que indica temperatura) vemos dos números, el de la izquierda es el valor mínimo de alarma y el de la derecha el valor máximo.



Figura 6.- Valores.

Además, si se toca la última línea que muestra la información de los valores mínimos y máximos, vemos que cambia de color, rojo indicando que se representa temperatura y verde que se representa presión, esto puede verse en el servomotor, que indica a su izquierda en su valor mínimo y a la derecha el máximo.

WEB.

La interfaz web consta de 3 partes, una tabla que se actualiza cada 20 segundos con los datos recibidos de la estación meteorológica, una segunda con la fecha de la medida y una tercera con los valores a modificar.

En la figura 7 podemos ver la interfaz WEB, en la última tabla pueden modificarse los siguientes valores:

- Valor mínimo de la alarma de temperatura. (°C)
- Valor máximo de la alarma de temperatura. (°C)
- Valor mínimo de la alarma de presión. (mBar.)
- Valor máximo de la alarma de presión. (mBar.)
- Tiempo que tarda en apagarse la pantalla en ULPM (segundos).

Estacion meteorologica

Datos medios actuales:			
Temperatura:	14.000000	Velocidad del viento:	0.000000
Humedad:	62.799999	Indice UV:	0.000000
Presion:	936.049988	Brillo:	0.000000
Altitud:	663.514160	Longitud:	0.000000
Latitud:	0.000000		

Hora de la ultima muestra:

Ano:	2020	Mes:	1	Dia:	1
Hora:	0	Minuto:	0	Segundos:	24

Magnitudes modificables:

Temperatura min. :	-10
Temperatura max. :	50
Presion min. :	500
Presion max. :	1500
Segundos encendido :	10
<input checked="" type="radio"/> Temperatura	
<input type="radio"/> Presion	
<input type="button" value="Enviar"/>	

Autor: Alberto Palomo Alonso. Sistemas Electronicos Digitales Avanzados. Universidad de Alcala - Escuela politecnica superior.

Figura 7.- Interfaz WEB.

Para configurar nuestro dispositivo que accede vía ethernet, debemos dirigirnos a nuestra configuración de IPv4. En Windows, eso es en la siguiente ruta:

- Configuración de Red e Internet/Cambiar opciones del Adaptador/Click derecho Ethernet/Propiedades (root)/Protocolo de internet versión 4/Propiedades/Usar la siguiente dirección IP.

Por defecto está configurado que se utilice cualquier otra IP que no sea la del servidor, por ejemplo: 192.168.1.10.

Hay que configurar la máscara subred tal que sea: 255.255.255.0

La dirección de la puerta de enlace predeterminada o Gateway debe de ser: 192.168.1.20.

Estos valores pueden ser modificados dentro del código fuente del proyecto.

Para acceder a la web basta con conectar el cable ethernet al dispositivo y acceder a la dirección IP del servidor: 192.168.1.120 por defecto. Una vez entrando nos pedirán usuario y contraseña:

- Usuario: user
- Clave: Alver

Una vez introducido, deberíamos ser capaces de visualizar la interfaz WEB del servidor.

UART.

La interfaz UART es la interfaz USB del dispositivo - placa, con conectar un puerto USB mediante un cable USB-Micro USB al puerto de la tarjeta MINI-DK2 sería suficiente para iniciar la instalación automática del controlador.

Al realizar la instalación, se precisa de un monitor serie para podernos comunicar con un terminal serie, como puede ser Termite. La configuración del puerto serie debe de ser la siguiente:

- 9600 baudios.
- Paridad impar (odd).
- 8 bits por dato.
- 1 bit de stop.
- Append CR. (Sólo CR, el programa está hecho para interpretar sólo el carácter 13.)

Una vez realizada la configuración podemos comunicarnos con los siguientes comandos:

- GIVE [X]: Muestra por pantalla del dato [X] de la lista expuesta a continuación; sin los corchetes:
 - IP
 - TEMPERATURA
 - PRESION
 - VIENTO
 - LUGAR
 - INDICEUV
 - HORA
 - HUMEDAD
 - BRILLOCada uno haciendo referencia a la variable que indica en su propio nombre.
- SET [Y]: Configura la variable [Y] de la lista expuesta a continuación; sin los corchetes:
 - BRILLO
 - HORA
 - MIN TEMP
 - MAX TEMP
 - MIN PRES
 - MAX PRES
 - TEMPERATURA
 - PRESIONDonde brillo es el tiempo que transcurre desde que no se toca la pantalla hasta que se apaga en modo ULPM, min representa mínimo y max representa máximo de los valores de alarma de TEMP, temperatura, y PRES, presión. TEMPERATURA activa la representación de la temperatura en el servomotor y PRESION la de presión en el servomotor.
- KILL: Cuelga el sistema en un while(1), activa el WatchDog.
- ABOUT: Muestra información del creador.
- HELP: Muestra información de la UART.
- HELP SET: Muestra la información del comando GIVE.
- HELP GIVE: Muestra la información del comando SET.

Referencias

<https://github.com/iTzAlver/EstacionMeteorologica.git>

<https://developer.arm.com/ip-products/processors/cortex-m/cortex-m3> [En línea] / aut. ARM.

https://www.amazon.es/Anem%C3%B3metro-Velocidad-Estaci%C3%B3n-Meteorol%C3%B3gica-Arduino/dp/B07BMVYBW9/ref=asc_df_B07BMVYBW9/?tag=googshopes-21&linkCode=df0&hvadid=300930138206&hvpos=1o1&hvnetw=g&hvrand=3634125468818365177&hvptwo=&hvqmt=&hvdev=c&h [En línea] / aut. anemómetro Imagen.

www.intel.com [En línea] / aut. Intel.

www.nxp.com [En línea] / aut. NXP.

www.powermcu.com [En línea] / aut. POWER MCU.

<https://cloud.smartdraw.com/>

www.blackboard.com

https://uah.blackboard.com/ultra/courses/_17817_1/cl/outline

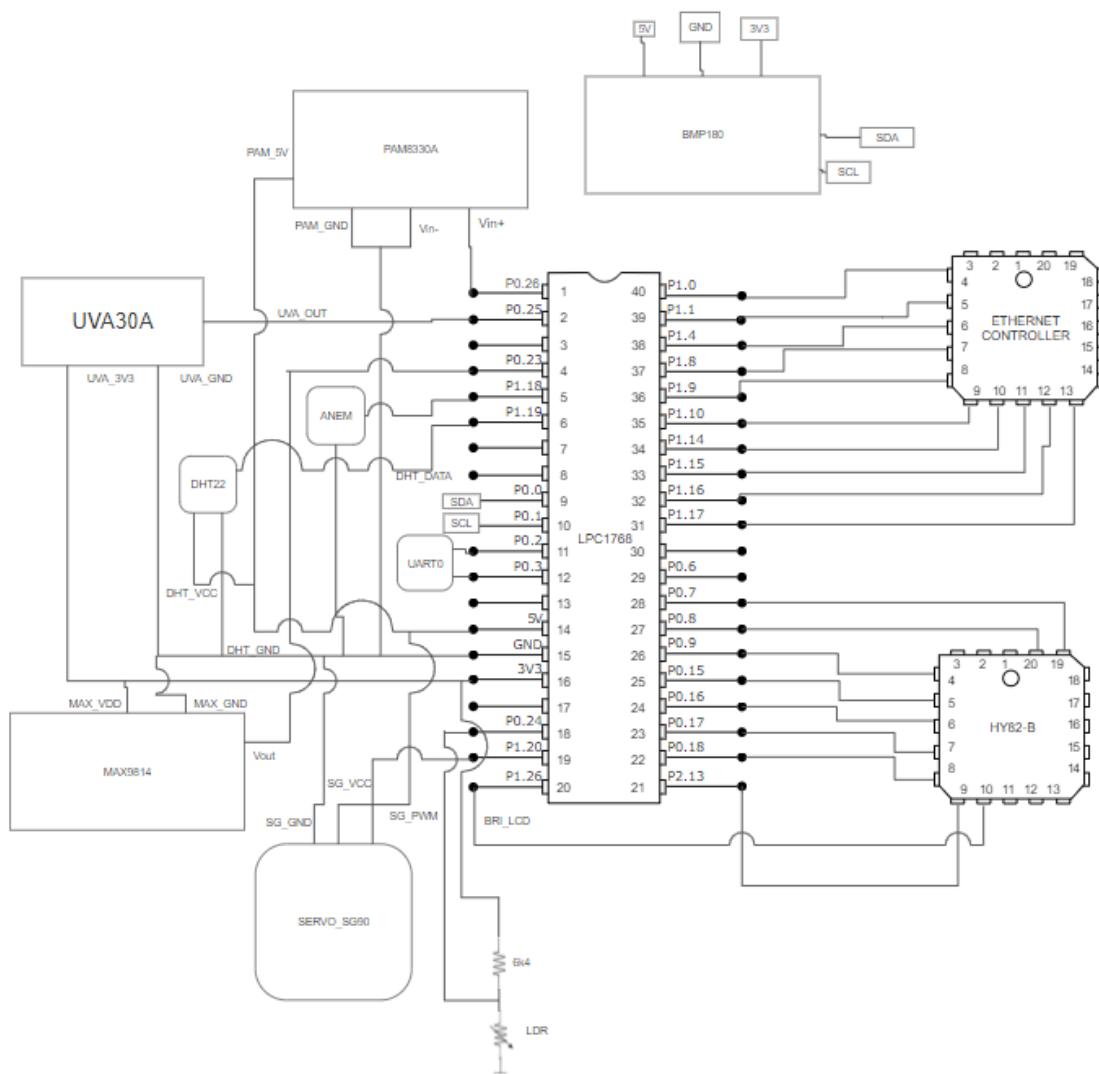
https://github.com/OCFreaks/LPC1768-Tutorial-Examples/blob/master/DHT11_Interfacing/main.c

Anexos.

A continuación, los anexos de la memoria de la estación meteorológica.

Anexo I – Esquemático.

En este anexo se adjunta un pequeño esquemático con la distribución de pines.



Anexo II – Código fuente.

NOTA: La tabulación en el programa Keil uVision4 debería de configurarse a 5 espacios para una lectura cómoda.

Readme.md

ESTACIÓN METEOROLÓGICA

* Este proyecto consiste en una estación meteorológica inteligente que sea capaz de obtener los datos de los sensores acoplados y de interactuar de manera inteligente con el usuario.

* Este proyecto ha sido elaborado por Alberto Palomo Alonso.

* Este proyecto ha sido elaborado para la asignatura S.E.D.A de la UAH.

IMPORTANTE

* Para la lectura del código ajustar el tabulador a 5 espacios/tab.

* Las carpetas están ordenadas por funciones y periféricos.

* Si el proyecto da error en el linker, es problema de la licencia de Keil.

REFERENCIAS

* Los archivos en CMSIS son Firmware proporcionado.

* Los archivos en GLCD es código de API para el controlador ILI9325C.

* El archivo leds.c es código proporcionado por la asignatura S.E.D.A de la UAH como código ejemplo y no ha sufrido modificaciones.

* El archivo menu.c es código proporcionado por la asignatura S.E.D.A de la UAH como código ejemplo y no ha sufrido modificaciones.

* El archivo TouchPanel.c de la sección Menu es código proporcionado por la asignatura S.E.D.A de la uah como ejemplo y ha sufrido modificaciones.

* Las modificaciones en TouchPanel.c son: incluir la librería WDT y la llamada a la función alimentaWDT() para el Watchdog.

* En el archivo statechart.c y statechart.h se han modificado ligeramente funciones y se han implementado.

* El archivo uart.c y uart.h es código proporcionado por la asignatura S.E.D.A de la UAH como código ejemplo y ha sufido ligeras modificaciones.

* El archivo I2Clib.c es código proporcionado por la asignatura S.E.D.A de la UAH como código ejemplo y no ha sufrido modificaciones.

* Leer documentación del proyecto en Git.

Proyecto

* Github:
[EstacionMeterologica] (<https://github.com/iTzAlver/EstacionMeteorologica.git>)

* Gmail: ialver.p@gmail.com

```
>      uah.es
>      arm.com
>      Alberto Palomo
```

Archivos extensión startup.

```
; ****
; * @file      startup_LPC17xx.s
; * @brief     CMSIS Cortex-M3 Core Device Startup File for
; *             NXP LPC17xx Device Series
; * @version   V1.10
; * @date      06. April 2011
; *
; * @note
; * Copyright (C) 2009-2011 ARM Limited. All rights reserved.
; *
; * @par
; * ARM Limited (ARM) is supplying this software for use with Cortex-M
; * processor based microcontrollers. This file can be freely distributed
; * within development tools that are supporting such ARM based processors.
; *
; * @par
; * THIS SOFTWARE IS PROVIDED "AS IS". NO WARRANTIES, WHETHER EXPRESS, IMPLIED
; * OR STATUTORY, INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF
; * MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE APPLY TO THIS SOFTWARE.
; * ARM SHALL NOT, IN ANY CIRCUMSTANCES, BE LIABLE FOR SPECIAL, INCIDENTAL, OR
; * CONSEQUENTIAL DAMAGES, FOR ANY REASON WHATSOEVER.
; *
; * ****
; *----- <<< Use Configuration Wizard in Context Menu >>> -----
;
; <h> Stack Configuration
;   <o> Stack Size (in Bytes) <0x0-0xFFFFFFFF:8>
; </h>
;
Stack_Size      EQU      0x00000200
;
          AREA     STACK, NOINIT, READWRITE, ALIGN=3
Stack_Mem       SPACE    Stack_Size
```

```
_initial_sp

; <h> Heap Configuration
;   <o>  Heap Size (in Bytes) <0x0-0xFFFFFFFF:8>
; </h>

Heap_Size      EQU      0x00000200

          AREA      HEAP, NOINIT, READWRITE, ALIGN=3
          _heap_base
          Heap_Mem      SPACE     Heap_Size
          _heap_limit

          PRESERVE8
          THUMB

; Vector Table Mapped to Address 0 at Reset

          AREA      RESET, DATA, READONLY
          EXPORT    __Vectors

          __Vectors      DCD      _initial_sp           ; Top of Stack
                      DCD      Reset_Handler        ; Reset Handler
                      DCD      NMI_Handler         ; NMI Handler
                      DCD      HardFault_Handler ; Hard Fault Handler
                      DCD      MemManage_Handler ; MPU Fault Handler
                      DCD      BusFault_Handler ; Bus Fault Handler
                      DCD      UsageFault_Handler; Usage Fault Handler
                      DCD      0                   ; Reserved
                      DCD      0                   ; Reserved
                      DCD      0                   ; Reserved
                      DCD      SVC_Handler        ; SVCall Handler
                      DCD      DebugMon_Handler ; Debug Monitor Handler
                      DCD      0                   ; Reserved
                      DCD      PendSV_Handler   ; PendSV Handler
```

```

    DCD      SysTick_Handler           ; SysTick Handler

; External Interrupts

    DCD      WDT_IRQHandler          ; 16: Watchdog Timer
    DCD      TIMER0_IRQHandler       ; 17: Timer0
    DCD      TIMER1_IRQHandler       ; 18: Timer1
    DCD      TIMER2_IRQHandler       ; 19: Timer2
    DCD      TIMER3_IRQHandler       ; 20: Timer3
    DCD      UART0_IRQHandler        ; 21: UART0
    DCD      UART1_IRQHandler        ; 22: UART1
    DCD      UART2_IRQHandler        ; 23: UART2
    DCD      UART3_IRQHandler        ; 24: UART3
    DCD      PWM1_IRQHandler         ; 25: PWM1
    DCD      I2C0_IRQHandler         ; 26: I2C0
    DCD      I2C1_IRQHandler         ; 27: I2C1
    DCD      I2C2_IRQHandler         ; 28: I2C2
    DCD      SPI_IRQHandler          ; 29: SPI
    DCD      SSP0_IRQHandler          ; 30: SSP0
    DCD      SSP1_IRQHandler          ; 31: SSP1
    DCD      PLL0_IRQHandler         ; 32: PLL0 Lock (Main PLL)
    DCD      RTC_IRQHandler          ; 33: Real Time Clock
    DCD      EINT0_IRQHandler        ; 34: External Interrupt 0
    DCD      EINT1_IRQHandler        ; 35: External Interrupt 1
    DCD      EINT2_IRQHandler        ; 36: External Interrupt 2
    DCD      EINT3_IRQHandler        ; 37: External Interrupt 3
    DCD      ADC_IRQHandler          ; 38: A/D Converter
    DCD      BOD_IRQHandler          ; 39: Brown-Out Detect
    DCD      USB_IRQHandler          ; 40: USB
    DCD      CAN_IRQHandler          ; 41: CAN
    DCD      DMA_IRQHandler          ; 42: General Purpose DMA
    DCD      I2S_IRQHandler          ; 43: I2S
    DCD      ENET_IRQHandler         ; 44: Ethernet
    DCD      RIT_IRQHandler          ; 45: Repetitive Interrupt Timer
    DCD      MCPWM_IRQHandler        ; 46: Motor Control PWM
    DCD      QEI_IRQHandler          ; 47: Quadrature Encoder Interface
    DCD      PLL1_IRQHandler         ; 48: PLL1 Lock (USB PLL)
    DCD      USBActivity_IRQHandler   ; 49: USB Activity interrupt to wakeup
    DCD      CANActivity_IRQHandler   ; 50: CAN Activity interrupt to wakeup

```

```
IF      :LNOT::DEF:NO_CRP
AREA    | .ARM.__at_0x02FC|, CODE, READONLY
CRP_Key DCD    0xFFFFFFFF
ENDIF

AREA    | .text|, CODE, READONLY

; Reset Handler

Reset_Handler PROC
    EXPORT Reset_Handler          [WEAK]
    IMPORT SystemInit
    IMPORT __main
    LDR    R0, =SystemInit
    BLX    R0
    LDR    R0, =__main
    BX    R0
ENDP

; Dummy Exception Handlers (infinite loops which can be modified)

NMI_Handler PROC
    EXPORT NMI_Handler           [WEAK]
    B     .
ENDP

HardFault_Handler\
PROC
    EXPORT HardFault_Handler     [WEAK]
    B     .
ENDP

MemManage_Handler\
PROC
    EXPORT MemManage_Handler     [WEAK]
    B     .
ENDP
```

```
BusFault_Handler\  
    PROC  
        EXPORT BusFault_Handler [WEAK]  
        B .  
    ENDP  
  
UsageFault_Handler\  
    PROC  
        EXPORT UsageFault_Handler [WEAK]  
        B .  
    ENDP  
  
SVC_Handler PROC  
    EXPORT SVC_Handler [WEAK]  
    B .  
ENDP  
  
DebugMon_Handler\  
    PROC  
        EXPORT DebugMon_Handler [WEAK]  
        B .  
    ENDP  
  
PendSV_Handler PROC  
    EXPORT PendSV_Handler [WEAK]  
    B .  
ENDP  
  
SysTick_Handler PROC  
    EXPORT SysTick_Handler [WEAK]  
    B .  
ENDP  
  
Default_Handler PROC  
    EXPORT WDT_IRQHandler [WEAK]  
    EXPORT TIMER0_IRQHandler [WEAK]  
    EXPORT TIMER1_IRQHandler [WEAK]  
    EXPORT TIMER2_IRQHandler [WEAK]  
    EXPORT TIMER3_IRQHandler [WEAK]  
    EXPORT UART0_IRQHandler [WEAK]  
    EXPORT UART1_IRQHandler [WEAK]  
    EXPORT UART2_IRQHandler [WEAK]  
    EXPORT UART3_IRQHandler [WEAK]
```

```
EXPORT PWM1_IRQHandler [WEAK]
EXPORT I2C0_IRQHandler [WEAK]
EXPORT I2C1_IRQHandler [WEAK]
EXPORT I2C2_IRQHandler [WEAK]
EXPORT SPI_IRQHandler [WEAK]
EXPORT SSP0_IRQHandler [WEAK]
EXPORT SSP1_IRQHandler [WEAK]
EXPORT PLL0_IRQHandler [WEAK]
EXPORT RTC_IRQHandler [WEAK]
EXPORT EINT0_IRQHandler [WEAK]
EXPORT EINT1_IRQHandler [WEAK]
EXPORT EINT2_IRQHandler [WEAK]
EXPORT EINT3_IRQHandler [WEAK]
EXPORT ADC_IRQHandler [WEAK]
EXPORT BOD_IRQHandler [WEAK]
EXPORT USB_IRQHandler [WEAK]
EXPORT CAN_IRQHandler [WEAK]
EXPORT DMA_IRQHandler [WEAK]
EXPORT I2S_IRQHandler [WEAK]
EXPORT ENET_IRQHandler [WEAK]
EXPORT RIT_IRQHandler [WEAK]
EXPORT MCPWM_IRQHandler [WEAK]
EXPORT QEI_IRQHandler [WEAK]
EXPORT PLL1_IRQHandler [WEAK]
EXPORT USBActivity_IRQHandler [WEAK]
EXPORT CANActivity_IRQHandler [WEAK]
```

```
WDT_IRQHandler
TIMER0_IRQHandler
TIMER1_IRQHandler
TIMER2_IRQHandler
TIMER3_IRQHandler
UART0_IRQHandler
UART1_IRQHandler
UART2_IRQHandler
UART3_IRQHandler
PWM1_IRQHandler
I2C0_IRQHandler
I2C1_IRQHandler
```

Alberto Palomo Alonso.

```
I2C2_IRQHandler  
SPI_IRQHandler  
SSP0_IRQHandler  
SSP1_IRQHandler  
PLL0_IRQHandler  
RTC_IRQHandler  
EINT0_IRQHandler  
EINT1_IRQHandler  
EINT2_IRQHandler  
EINT3_IRQHandler  
ADC_IRQHandler  
BOD_IRQHandler  
USB_IRQHandler  
CAN_IRQHandler  
DMA_IRQHandler  
I2S_IRQHandler  
ENET_IRQHandler  
RIT_IRQHandler  
MCPWM_IRQHandler  
QEI_IRQHandler  
PLL1_IRQHandler  
USBActivity_IRQHandler  
CANActivity_IRQHandler
```

B .

ENDP

ALIGN

; User Initial Stack & Heap

```
IF :DEF:_MICROLIB  
  
EXPORT __initial_sp  
EXPORT __heap_base  
EXPORT __heap_limit
```

```
ELSE

    IMPORT __use_two_region_memory
    EXPORT __user_initial_stackheap

__user_initial_stackheap

    LDR R0, = Heap_Mem
    LDR R1, =(Stack_Mem + Stack_Size)
    LDR R2, = (Heap_Mem + Heap_Size)
    LDR R3, = Stack_Mem
    BX LR

ALIGN

ENDIF

END
```

Archivos extensión C.

```
/****************************************************************************
 * @file      core_cm3.c
 * @brief     CMSIS Cortex-M3 Core Peripheral Access Layer Source File
 * @version   V2.00
 * @date      13. September 2010
 *
 * @note
 * Copyright (C) 2009-2010 ARM Limited. All rights reserved.
 *
 * @par
 * ARM Limited (ARM) is supplying this software for use with Cortex-M
 * processor based microcontrollers. This file can be freely distributed
 * within development tools that are supporting such ARM based processors.
 *
 * @par
 * THIS SOFTWARE IS PROVIDED "AS IS". NO WARRANTIES, WHETHER EXPRESS, IMPLIED
 * OR STATUTORY, INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF
```

```
* MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE APPLY TO THIS SOFTWARE.  
* ARM SHALL NOT, IN ANY CIRCUMSTANCES, BE LIABLE FOR SPECIAL, INCIDENTAL, OR  
* CONSEQUENTIAL DAMAGES, FOR ANY REASON WHATSOEVER.  
*  
*****  
  
#include <stdint.h>  
  
/* define compiler specific symbols */  
  
#if defined ( __CC_ARM )  
    #define __ASM      __asm                         /*!< asm keyword  
for ARM Compiler           */  
    #define __INLINE   __inline                      /*!< inline  
keyword for ARM Compiler   */  
  
#elif defined ( __ICCARM__ )  
    #define __ASM      __asm                         /*!< asm keyword  
for IAR Compiler           */  
    #define __INLINE   inline                        /*!< inline  
keyword for IAR Compiler. Only available in High optimization mode! */  
  
#elif defined ( __GNUC__ )  
    #define __ASM      __asm                         /*!< asm keyword  
for GNU Compiler            */  
    #define __INLINE   inline                        /*!< inline  
keyword for GNU Compiler    */  
  
#elif defined ( __TASKING__ )  
    #define __ASM      __asm                         /*!< asm keyword  
for TASKING Compiler        */  
    #define __INLINE   inline                        /*!< inline  
keyword for TASKING Compiler */  
  
#endif  
  
/* ##### Core Instruction Access ##### */  
  
#if defined ( __CC_ARM ) /*----- RealView Compiler -----*/  
  
/** \brief Reverse byte order (16 bit)  
  
This function reverses the byte order in two unsigned short values.
```

```
\param [in]      value  Value to reverse
\return          Reversed value
*/
#if (_ARMCC_VERSION < 400677)
__ASM uint32_t __REV16(uint32_t value)
{
    rev16 r0, r0
    bx lr
}
#endif /* __ARMCC_VERSION */

/** \brief Reverse byte order in signed short value

    This function reverses the byte order in a signed short value with sign extension to
    integer.

\param [in]      value  Value to reverse
\return          Reversed value
*/
#if (_ARMCC_VERSION < 400677)
__ASM int32_t __REVSH(int32_t value)
{
    revsh r0, r0
    bx lr
}
#endif /* __ARMCC_VERSION */

/** \brief Remove the exclusive lock

    This function removes the exclusive lock which is created by LDREX.

*/
#if (_ARMCC_VERSION < 400000)
__ASM void __CLREX(void)
{
    clrex
}
```

Alberto Palomo Alonso.

```
#endif /* __ARMCC_VERSION */

#if defined (__ICCARM__) /*----- ICC Compiler -----*/
/* obsolete */
#endif /* __ICCARM__ */

#if defined (__GNUC__) /*----- GNU Compiler -----*/
/* obsolete */
#endif /* __GNUC__ */

#if defined (__TASKING__) /*----- TASKING Compiler -----*/
/* obsolete */
#endif /* __TASKING__ */

#endif /* __ARMCC_VERSION */

/* ##### Core Function Access ##### */

#if defined ( __CC_ARM ) /*----- RealView Compiler -----*/

/** \brief Get Control Register

    This function returns the content of the Control Register.

    \return          Control Register value
*/
#if      (__ARMCC_VERSION < 400000)
__ASM uint32_t __get_CONTROL(void)
{
    mrs r0, control
    bx lr
}
#endif /* __ARMCC_VERSION */

/** \brief Set Control Register

    This function writes the given value to the Control Register.

    \param [in]      control  Control Register value to set
*/
#if      (__ARMCC_VERSION < 400000)
__ASM void __set_CONTROL(uint32_t control)
```

```
{  
    msr control, r0  
    bx lr  
}  
#endif /* __ARMCC_VERSION */
```

```
/** \brief Get ISPR Register
```

This function returns the content of the ISPR Register.

```
\return ISPR Register value  
*/  
#if      (__ARMCC_VERSION < 400000)  
__ASM uint32_t __get_IPSR(void)  
{  
    mrs r0, ipsr  
    bx lr  
}  
#endif /* __ARMCC_VERSION */
```

```
/** \brief Get APSR Register
```

This function returns the content of the APSR Register.

```
\return APSR Register value  
*/  
#if      (__ARMCC_VERSION < 400000)  
__ASM uint32_t __get_APSR(void)  
{  
    mrs r0, apsr  
    bx lr  
}  
#endif /* __ARMCC_VERSION */
```

```
/** \brief Get xPSR Register
```

Alberto Palomo Alonso.

This function returns the content of the xPSR Register.

```
\return xPSR Register value
*/
#if      (__ARMCC_VERSION < 400000)
__ASM uint32_t __get_xPSR(void)
{
    mrs r0, xpsr
    bx lr
}
#endif /* __ARMCC_VERSION */
```

/** \brief Get Process Stack Pointer

This function returns the current value of the Process Stack Pointer (PSP).

```
\return PSP Register value
*/
#if      (__ARMCC_VERSION < 400000)
__ASM uint32_t __get_PSP(void)
{
    mrs r0, psp
    bx lr
}
#endif /* __ARMCC_VERSION */
```

/** \brief Set Process Stack Pointer

This function assigns the given value to the Process Stack Pointer (PSP).

```
\param [in] topOfProcStack Process Stack Pointer value to set
*/
#if      (__ARMCC_VERSION < 400000)
__ASM void __set_PSP(uint32_t topOfProcStack)
{
    msr psp, r0
    bx lr
}
```

Alberto Palomo Alonso.

```
}

#endif /* __ARMCC_VERSION */

/** \brief Get Main Stack Pointer

This function returns the current value of the Main Stack Pointer (MSP).

\return MSP Register value

*/
#if      (__ARMCC_VERSION < 400000)
__ASM uint32_t __get_MSP(void)
{
    mrs r0, msp
    bx lr
}
#endif /* __ARMCC_VERSION */

/** \brief Set Main Stack Pointer

This function assigns the given value to the Main Stack Pointer (MSP).

\param [in] topOfMainStack Main Stack Pointer value to set

*/
#if      (__ARMCC_VERSION < 400000)
__ASM void __set_MSP(uint32_t mainStackPointer)
{
    msr msp, r0
    bx lr
}
#endif /* __ARMCC_VERSION */

/** \brief Get Base Priority

This function returns the current value of the Base Priority register.

\return Base Priority register value
```

```
*/  
#if      (__ARMCC_VERSION < 400000)  
__ASM uint32_t __get_BASEPRI(void)  
{  
    mrs r0, basepri  
    bx lr  
}  
#endif /* __ARMCC_VERSION */
```

/** \brief Set Base Priority

This function assigns the given value to the Base Priority register.

```
\param [in]     basePri  Base Priority value to set  
*/  
#if      (__ARMCC_VERSION < 400000)  
__ASM void __set_BASEPRI(uint32_t basePri)  
{  
    msr basepri, r0  
    bx lr  
}  
#endif /* __ARMCC_VERSION */
```

/** \brief Get Priority Mask

This function returns the current state of the priority mask bit from the Priority Mask Register.

```
\return          Priority Mask value  
*/  
#if      (__ARMCC_VERSION < 400000)  
__ASM uint32_t __get_PRIMASK(void)  
{  
    mrs r0, primask  
    bx lr  
}  
#endif /* __ARMCC_VERSION */
```

Alberto Palomo Alonso.

```
/** \brief Set Priority Mask
```

This function assigns the given value to the Priority Mask Register.

```
\param [in]     priMask  Priority Mask
*/
#if      (__ARMCC_VERSION < 400000)
__ASM void __set_PRIMASK(uint32_t priMask)
{
    msr primask, r0
    bx lr
}
#endif /* __ARMCC_VERSION */
```

```
/** \brief Get Fault Mask
```

This function returns the current value of the Fault Mask Register.

```
\return          Fault Mask value
*/
#if      (__ARMCC_VERSION < 400000)
__ASM uint32_t __get_FAULTMASK(void)
{
    mrs r0, faultmask
    bx lr
}
#endif /* __ARMCC_VERSION */
```

```
/** \brief Set the Fault Mask
```

This function assigns the given value to the Fault Mask Register.

```
\param [in]     faultMask  Fault Mask value to set
*/
#if      (__ARMCC_VERSION < 400000)
__ASM void __set_FAULTMASK(uint32_t faultMask)
{
```

```
    msr faultmask, r0
    bx lr
}

#endif /* __ARMCC_VERSION */

#elif (defined (__ICCARM__)) /*----- ICC Compiler -----*/
/* obsolete */
#elif (defined (__GNUC__)) /*----- GNU Compiler -----*/
/* obsolete */
#elif (defined (__TASKING__)) /*----- TASKING Compiler -----*/
/* obsolete */
#endif

/********************* / ***/
* @file      system_LPC17xx.c
* @brief     CMSIS Cortex-M3 Device System Source File for
*            NXP LPC17xx Device Series
* @version   V1.13
* @date      18. April 2012
*
* @note
* Copyright (C) 2009-2012 ARM Limited. All rights reserved.
*
* @par
* ARM Limited (ARM) is supplying this software for use with Cortex-M
* processor based microcontrollers. This file can be freely distributed
* within development tools that are supporting such ARM based processors.
*
* @par
* THIS SOFTWARE IS PROVIDED "AS IS". NO WARRANTIES, WHETHER EXPRESS, IMPLIED
* OR STATUTORY, INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF
* MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE APPLY TO THIS SOFTWARE.
*
* ARM SHALL NOT, IN ANY CIRCUMSTANCES, BE LIABLE FOR SPECIAL, INCIDENTAL, OR
* CONSEQUENTIAL DAMAGES, FOR ANY REASON WHATSOEVER.
*
```

```
*****  
  
#include <stdint.h>  
#include "LPC17xx.h"  
  
/** @addtogroup LPC17xx_System  
 * @ {  
 * /  
  
/*  
//----- <<< Use Configuration Wizard in Context Menu >>> -----  
*/  
  
/*----- Clock Configuration -----  
//  
// <e> Clock Configuration  
//   <h> System Controls and Status Register (SCS)  
//     <o1.4> OSCRANGE: Main Oscillator Range Select  
//           <0=> 1 MHz to 20 MHz  
//           <1=> 15 MHz to 25 MHz  
//     <e1.5> OSCEN: Main Oscillator Enable  
//   </e>  
// </h>  
//  
//   <h> Clock Source Select Register (CLKSRCSEL)  
//     <o2.0..1> CLKSRC: PLL Clock Source Selection  
//           <0=> Internal RC oscillator  
//           <1=> Main oscillator  
//           <2=> RTC oscillator  
//   </h>  
//  
//   <e3> PLL0 Configuration (Main PLL)  
//     <h> PLL0 Configuration Register (PLL0CFG)  
//           <i> F_cco0 = (2 * M * F_in) / N  
//           <i> F_in must be in the range of 32 kHz to 50 MHz  
//           <i> F_cco0 must be in the range of 275 MHz to 550 MHz  
//     <o4.0..14> MSEL: PLL Multiplier Selection
```

```
// <6-32768><#-1>
// <i> M Value
// <o4.16..23> NSEL: PLL Divider Selection
// <1-256><#-1>
// <i> N Value
// </h>
// </e>
//
// <e5> PLL1 Configuration (USB PLL)
// <h> PLL1 Configuration Register (PLL1CFG)
// <i> F_usb = M * F_osc or F_usb = F_ccol / (2 * P)
// <i> F_ccol = F_osc * M * 2 * P
// <i> F_ccol must be in the range of 156 MHz to 320 MHz
// <o6.0..4> MSEL: PLL Multiplier Selection
// <1-32><#-1>
// <i> M Value (for USB maximum value is 4)
// <o6.5..6> PSEL: PLL Divider Selection
// <0=> 1
// <1=> 2
// <2=> 4
// <3=> 8
// <i> P Value
// </h>
// </e>
//
// <h> CPU Clock Configuration Register (CCLKCFG)
// <o7.0..7> CCLKSEL: Divide Value for CPU Clock from PLL0
// <1-256><#-1>
// </h>
//
// <h> USB Clock Configuration Register (USBCLKCFG)
// <o8.0..3> USBSEL: Divide Value for USB Clock from PLL0
// <0-15>
// <i> Divide is USBSEL + 1
// </h>
//
// <h> Peripheral Clock Selection Register 0 (PCLKSEL0)
// <o9.0..1> PCLK_WDT: Peripheral Clock Selection for WDT
// <0=> Pclk = Cclk / 4
```

```
//          <1=> Pclk = Cclk
//
//          <2=> Pclk = Cclk / 2
//
//          <3=> Pclk = Cclk / 8
//
// <o9.2..3>    PCLK_TIMER0: Peripheral Clock Selection for TIMER0
//
//          <0=> Pclk = Cclk / 4
//
//          <1=> Pclk = Cclk
//
//          <2=> Pclk = Cclk / 2
//
//          <3=> Pclk = Cclk / 8
//
// <o9.4..5>    PCLK_TIMER1: Peripheral Clock Selection for TIMER1
//
//          <0=> Pclk = Cclk / 4
//
//          <1=> Pclk = Cclk
//
//          <2=> Pclk = Cclk / 2
//
//          <3=> Pclk = Cclk / 8
//
// <o9.6..7>    PCLK_UART0: Peripheral Clock Selection for UART0
//
//          <0=> Pclk = Cclk / 4
//
//          <1=> Pclk = Cclk
//
//          <2=> Pclk = Cclk / 2
//
//          <3=> Pclk = Cclk / 8
//
// <o9.8..9>    PCLK_UART1: Peripheral Clock Selection for UART1
//
//          <0=> Pclk = Cclk / 4
//
//          <1=> Pclk = Cclk
//
//          <2=> Pclk = Cclk / 2
//
//          <3=> Pclk = Cclk / 8
//
// <o9.12..13>  PCLK_PWM1: Peripheral Clock Selection for PWM1
//
//          <0=> Pclk = Cclk / 4
//
//          <1=> Pclk = Cclk
//
//          <2=> Pclk = Cclk / 2
//
//          <3=> Pclk = Cclk / 8
//
// <o9.14..15>  PCLK_I2C0: Peripheral Clock Selection for I2C0
//
//          <0=> Pclk = Cclk / 4
//
//          <1=> Pclk = Cclk
//
//          <2=> Pclk = Cclk / 2
//
//          <3=> Pclk = Cclk / 8
//
// <o9.16..17>  PCLK_SPI: Peripheral Clock Selection for SPI
//
//          <0=> Pclk = Cclk / 4
//
//          <1=> Pclk = Cclk
//
//          <2=> Pclk = Cclk / 2
//
//          <3=> Pclk = Cclk / 8
//
// <o9.20..21>  PCLK_SSP1: Peripheral Clock Selection for SSP1
```

```
//          <0=> Pclk = Cclk / 4
//
//          <1=> Pclk = Cclk
//
//          <2=> Pclk = Cclk / 2
//
//          <3=> Pclk = Cclk / 8
//
// <o9.22..23> PCLK_DAC: Peripheral Clock Selection for DAC
//
//          <0=> Pclk = Cclk / 4
//
//          <1=> Pclk = Cclk
//
//          <2=> Pclk = Cclk / 2
//
//          <3=> Pclk = Cclk / 8
//
// <o9.24..25> PCLK_ADC: Peripheral Clock Selection for ADC
//
//          <0=> Pclk = Cclk / 4
//
//          <1=> Pclk = Cclk
//
//          <2=> Pclk = Cclk / 2
//
//          <3=> Pclk = Cclk / 8
//
// <o9.26..27> PCLK_CAN1: Peripheral Clock Selection for CAN1
//
//          <0=> Pclk = Cclk / 4
//
//          <1=> Pclk = Cclk
//
//          <2=> Pclk = Cclk / 2
//
//          <3=> Pclk = Cclk / 6
//
// <o9.28..29> PCLK_CAN2: Peripheral Clock Selection for CAN2
//
//          <0=> Pclk = Cclk / 4
//
//          <1=> Pclk = Cclk
//
//          <2=> Pclk = Cclk / 2
//
//          <3=> Pclk = Cclk / 6
//
// <o9.30..31> PCLK_ACF: Peripheral Clock Selection for ACF
//
//          <0=> Pclk = Cclk / 4
//
//          <1=> Pclk = Cclk
//
//          <2=> Pclk = Cclk / 2
//
//          <3=> Pclk = Cclk / 6
//
// </h>
//
// <h> Peripheral Clock Selection Register 1 (PCLKSEL1)
//
// <o10.0..1> PCLK_QEI: Peripheral Clock Selection for the Quadrature Encoder
// Interface
//
//          <0=> Pclk = Cclk / 4
//
//          <1=> Pclk = Cclk
//
//          <2=> Pclk = Cclk / 2
//
//          <3=> Pclk = Cclk / 8
//
// <o10.2..3> PCLK_GPIO: Peripheral Clock Selection for GPIOs
//
//          <0=> Pclk = Cclk / 4
```

```
//          <1=> Pclk = Cclk
//
//          <2=> Pclk = Cclk / 2
//
//          <3=> Pclk = Cclk / 8
//
// <o10.4..5>  PCLK_PCB: Peripheral Clock Selection for the Pin Connect Block
//
//          <0=> Pclk = Cclk / 4
//
//          <1=> Pclk = Cclk
//
//          <2=> Pclk = Cclk / 2
//
//          <3=> Pclk = Cclk / 8
//
// <o10.6..7>  PCLK_I2C1: Peripheral Clock Selection for I2C1
//
//          <0=> Pclk = Cclk / 4
//
//          <1=> Pclk = Cclk
//
//          <2=> Pclk = Cclk / 2
//
//          <3=> Pclk = Cclk / 8
//
// <o10.10..11> PCLK_SSP0: Peripheral Clock Selection for SSP0
//
//          <0=> Pclk = Cclk / 4
//
//          <1=> Pclk = Cclk
//
//          <2=> Pclk = Cclk / 2
//
//          <3=> Pclk = Cclk / 8
//
// <o10.12..13> PCLK_TIMER2: Peripheral Clock Selection for TIMER2
//
//          <0=> Pclk = Cclk / 4
//
//          <1=> Pclk = Cclk
//
//          <2=> Pclk = Cclk / 2
//
//          <3=> Pclk = Cclk / 8
//
// <o10.14..15> PCLK_TIMER3: Peripheral Clock Selection for TIMER3
//
//          <0=> Pclk = Cclk / 4
//
//          <1=> Pclk = Cclk
//
//          <2=> Pclk = Cclk / 2
//
//          <3=> Pclk = Cclk / 8
//
// <o10.16..17> PCLK_UART2: Peripheral Clock Selection for UART2
//
//          <0=> Pclk = Cclk / 4
//
//          <1=> Pclk = Cclk
//
//          <2=> Pclk = Cclk / 2
//
//          <3=> Pclk = Cclk / 8
//
// <o10.18..19> PCLK_UART3: Peripheral Clock Selection for UART3
//
//          <0=> Pclk = Cclk / 4
//
//          <1=> Pclk = Cclk
//
//          <2=> Pclk = Cclk / 2
//
//          <3=> Pclk = Cclk / 8
//
// <o10.20..21> PCLK_I2C2: Peripheral Clock Selection for I2C2
```

```
//          <0=> Pclk = Cclk / 4
//
//          <1=> Pclk = Cclk
//
//          <2=> Pclk = Cclk / 2
//
//          <3=> Pclk = Cclk / 8
//
// <o10.22..23> PCLK_I2S: Peripheral Clock Selection for I2S
//
//          <0=> Pclk = Cclk / 4
//
//          <1=> Pclk = Cclk
//
//          <2=> Pclk = Cclk / 2
//
//          <3=> Pclk = Cclk / 8
//
// <o10.26..27> PCLK_RIT: Peripheral Clock Selection for the Repetitive Interrupt
Timer
//
//          <0=> Pclk = Cclk / 4
//
//          <1=> Pclk = Cclk
//
//          <2=> Pclk = Cclk / 2
//
//          <3=> Pclk = Cclk / 8
//
// <o10.28..29> PCLK_SYSCON: Peripheral Clock Selection for the System Control Block
//
//          <0=> Pclk = Cclk / 4
//
//          <1=> Pclk = Cclk
//
//          <2=> Pclk = Cclk / 2
//
//          <3=> Pclk = Cclk / 8
//
// <o10.30..31> PCLK_MC: Peripheral Clock Selection for the Motor Control PWM
//
//          <0=> Pclk = Cclk / 4
//
//          <1=> Pclk = Cclk
//
//          <2=> Pclk = Cclk / 2
//
//          <3=> Pclk = Cclk / 8
//
// </h>
//
// <h> Power Control for Peripherals Register (PCONP)
//
// <o11.1>     PCTIMO: Timer/Counter 0 power/clock enable
//
// <o11.2>     PCTIM1: Timer/Counter 1 power/clock enable
//
// <o11.3>     PCUART0: UART 0 power/clock enable
//
// <o11.4>     PCUART1: UART 1 power/clock enable
//
// <o11.6>     PCPWM1: PWM 1 power/clock enable
//
// <o11.7>     PCI2C0: I2C interface 0 power/clock enable
//
// <o11.8>     PCSPI: SPI interface power/clock enable
//
// <o11.9>     PCRTC: RTC power/clock enable
//
// <o11.10>    PCSSP1: SSP interface 1 power/clock enable
//
// <o11.12>    PCAD: A/D converter power/clock enable
//
// <o11.13>    PCCAN1: CAN controller 1 power/clock enable
//
// <o11.14>    PCCAN2: CAN controller 2 power/clock enable
```

```

//      <o11.15>      PCGPIO: GPIOs power/clock enable
//      <o11.16>      PCRIT: Repetitive interrupt timer power/clock enable
//      <o11.17>      PCMC: Motor control PWM power/clock enable
//      <o11.18>      PCQEI: Quadrature encoder interface power/clock enable
//      <o11.19>      PCI2C1: I2C interface 1 power/clock enable
//      <o11.21>      PCSSP0: SSP interface 0 power/clock enable
//      <o11.22>      PCTIM2: Timer 2 power/clock enable
//      <o11.23>      PCTIM3: Timer 3 power/clock enable
//      <o11.24>      PCUART2: UART 2 power/clock enable
//      <o11.25>      PCUART3: UART 3 power/clock enable
//      <o11.26>      PCI2C2: I2C interface 2 power/clock enable
//      <o11.27>      PCI2S: I2S interface power/clock enable
//      <o11.29>      PCGPDMA: GP DMA function power/clock enable
//      <o11.30>      PCENET: Ethernet block power/clock enable
//      <o11.31>      PCUSB: USB interface power/clock enable
//      </h>
//
//      <h> Clock Output Configuration Register (CLKOUTCFG)
//      <o12.0..3>      CLKOUTSEL: Selects clock source for CLKOUT
//                      <0=> CPU clock
//                      <1=> Main oscillator
//                      <2=> Internal RC oscillator
//                      <3=> USB clock
//                      <4=> RTC oscillator
//      <o12.4..7>      CLKOUTDIV: Selects clock divider for CLKOUT
//                      <1-16><#-1>
//      <o12.8>          CLKOUT_EN: CLKOUT enable control
//      </h>
//
// </e>
*/

```

```

/** @addtogroup LPC17xx_System_Defines  LPC17xx System Defines
{
*/

```

```

#define CLOCK_SETUP           1

```

```
#define SCS_Val           0x00000020
#define CLKSRCSEL_Val     0x00000001
#define PLL0_SETUP          1
#define PLL0CFG_Val        0x00050063
#define PLL1_SETUP          1
#define PLL1CFG_Val        0x00000023
#define CCLKCFG_Val         0x00000003
#define USBCLKCFG_Val       0x00000000
#define PCLKSEL0_Val        0x00000000
#define PCLKSEL1_Val        0x00000000
#define PCONF_Val            0x042887DE
#define CLKOUTCFG_Val       0x00000000

/*----- Flash Accelerator Configuration -----
// 
// <e> Flash Accelerator Configuration
//   <o1.12..15> FLASHTIM: Flash Access Time
//             <0=> 1 CPU clock (for CPU clock up to 20 MHz)
//             <1=> 2 CPU clocks (for CPU clock up to 40 MHz)
//             <2=> 3 CPU clocks (for CPU clock up to 60 MHz)
//             <3=> 4 CPU clocks (for CPU clock up to 80 MHz)
//             <4=> 5 CPU clocks (for CPU clock up to 100 MHz)
//             <5=> 6 CPU clocks (for any CPU clock)
// </e>
*/
#define FLASH_SETUP          1
#define FLASHCFG_Val         0x00004000

/*
//----- <<< end of configuration section >>> -----
*/

/*-----
Check the register settings
-----*/
#define CHECK_RANGE(val, min, max)          ((val < min) || (val > max))
#define CHECK_RSVD(val, mask)                (val & mask)
```

```
/* Clock Configuration -----*/
#if (CHECK_RSVD((SCS_Val),      ~0x00000030))
    #error "SCS: Invalid values of reserved bits!"
#endif

#if (CHECK_RANGE((CLKSRCSEL_Val), 0, 2))
    #error "CLKSRCSEL: Value out of range!"
#endif

#if (CHECK_RSVD((PLL0CFG_Val),   ~0x00FF7FFF))
    #error "PLL0CFG: Invalid values of reserved bits!"
#endif

#if (CHECK_RSVD((PLL1CFG_Val),   ~0x0000007F))
    #error "PLL1CFG: Invalid values of reserved bits!"
#endif

#if (PLL0_SETUP)           /* if PLL0 is used */
    #if (CCLKCFG_Val < 2)    /* CCLKSEL must be greater than 1 */
        #error "CCLKCFG: CCLKSEL must be greater than 1 if PLL0 is used!"
    #endif
#endif

#if (CHECK_RANGE((CCLKCFG_Val), 0, 255))
    #error "CCLKCFG: Value out of range!"
#endif

#if (CHECK_RSVD((USBCLKCFG_Val), ~0x0000000F))
    #error "USBCLKCFG: Invalid values of reserved bits!"
#endif

#if (CHECK_RSVD((PCLKSEL0_Val),   0x000C0C00))
    #error "PCLKSEL0: Invalid values of reserved bits!"
#endif

#if (CHECK_RSVD((PCLKSEL1_Val),   0x03000300))
    #error "PCLKSEL1: Invalid values of reserved bits!"
#endif
```

```
#if (CHECK_RSVD((PCONP_Val),      0x10100821))
    #error "PCONP: Invalid values of reserved bits!"
#endif

#if (CHECK_RSVD((CLKOUTCFG_Val), ~0x000001FF))
    #error "CLKOUTCFG: Invalid values of reserved bits!"
#endif

/* Flash Accelerator Configuration -----*/
#if (CHECK_RSVD((FLASHCFG_Val), ~0x0000F000))
    #error "FLASHCFG: Invalid values of reserved bits!"
#endif

/*-----
   DEFINES
-----*/

/*-----
   Define clocks
-----*/

#define XTAL      (12000000UL)      /* Oscillator frequency */
#define OSC_CLK   (XTAL)           /* Main oscillator frequency */
#define RTC_CLK   (32768UL)         /* RTC oscillator frequency */
#define IRC_OSC   (4000000UL)        /* Internal RC oscillator frequency */

/* F_cco0 = (2 * M * F_in) / N */
#define __M          (((PLL0CFG_Val      ) & 0x7FFF) + 1)
#define __N          (((PLL0CFG_Val >> 16) & 0x00FF) + 1)
#define __FCCO(__F_IN) ((2ULL * __M * __F_IN) / __N)
#define __CCLK_DIV   (((CCLKCFG_Val      ) & 0x00FF) + 1)

/* Determine core clock frequency according to settings */
#if (PLL0_SETUP)
    #if ((CLKSRCSEL_Val & 0x03) == 1)
        #define __CORE_CLK (__FCCO(OSC_CLK) / __CCLK_DIV)
    #elif ((CLKSRCSEL_Val & 0x03) == 2)
        #define __CORE_CLK (__FCCO(RTC_CLK) / __CCLK_DIV)
    #endif

```

```
#else
#define __CORE_CLK (__FCCO(IRC_OSC) / __CCLK_DIV)
#endif
#else
#if ((CLKSRCSEL_Val & 0x03) == 1)
#define __CORE_CLK (OSC_CLK / __CCLK_DIV)
#elif ((CLKSRCSEL_Val & 0x03) == 2)
#define __CORE_CLK (RTC_CLK / __CCLK_DIV)
#else
#define __CORE_CLK (IRC_OSC / __CCLK_DIV)
#endif
#endif

/***
 * @}
 */

/** @defgroup LPC17xx_System_Public_Variables LPC17xx System Public Variables
 * @{
 */
/*-----*
 * Clock Variable definitions
 *-----*/
uint32_t SystemCoreClock = __CORE_CLK; /*!< System Clock Frequency (Core Clock)*/

/***
 * @}
 */

/** @defgroup LPC17xx_System_Public_Functions LPC17xx System Public Functions
 * @{
 */
/* Update SystemCoreClock variable
 *
 * @param none
 */
```

```

* @return none
*
* @brief Updates the SystemCoreClock with current core Clock
*        retrieved from cpu registers.
*/
void SystemCoreClockUpdate (void)          /* Get Core Clock Frequency */
{
    /* Determine clock frequency according to clock register values */
    if (((LPC_SC->PLL0STAT >> 24) & 3) == 3) { /* If PLL0 enabled and connected */
        switch (LPC_SC->CLKSRCSEL & 0x03) {
            case 0:                         /* Int. RC oscillator => PLL0 */
            case 3:                          /* Reserved, default to Int. RC */
                SystemCoreClock = (IRC_OSC *
                    ((2ULL * ((LPC_SC->PLL0STAT & 0x7FFF) + 1)) /
                    (((LPC_SC->PLL0STAT >> 16) & 0xFF) + 1) /
                    ((LPC_SC->CCLKCFG & 0xFF)+ 1));
                break;
            case 1:                         /* Main oscillator => PLL0 */
                SystemCoreClock = (OSC_CLK *
                    ((2ULL * ((LPC_SC->PLL0STAT & 0x7FFF) + 1)) /
                    (((LPC_SC->PLL0STAT >> 16) & 0xFF) + 1) /
                    ((LPC_SC->CCLKCFG & 0xFF)+ 1));
                break;
            case 2:                         /* RTC oscillator => PLL0 */
                SystemCoreClock = (RTC_CLK *
                    ((2ULL * ((LPC_SC->PLL0STAT & 0x7FFF) + 1)) /
                    (((LPC_SC->PLL0STAT >> 16) & 0xFF) + 1) /
                    ((LPC_SC->CCLKCFG & 0xFF)+ 1));
                break;
        }
    } else {
        switch (LPC_SC->CLKSRCSEL & 0x03) {
            case 0:                         /* Int. RC oscillator => PLL0 */
            case 3:                          /* Reserved, default to Int. RC */
                SystemCoreClock = IRC_OSC / ((LPC_SC->CCLKCFG & 0xFF)+ 1);
                break;
            case 1:                         /* Main oscillator => PLL0 */
                SystemCoreClock = OSC_CLK / ((LPC_SC->CCLKCFG & 0xFF)+ 1);
                break;
            case 2:                         /* RTC oscillator => PLL0 */

```

```
    SystemCoreClock = RTC_CLK / ((LPC_SC->CCLKCFG & 0xFF)+ 1);

    break;
}

}

}

/***
 * Initialize the system
 *
 * @param none
 * @return none
 *
 * @brief Setup the microcontroller system.
 *        Initialize the System.
 */
void SystemInit (void)
{
#if (CLOCK_SETUP)                                /* Clock Setup */

    LPC_SC->SCS      = SCS_Val;

    if (LPC_SC->SCS & (1 << 5)) {                /* If Main Oscillator is enabled */
        while (((LPC_SC->SCS & (1<<6)) == 0);/* Wait for Oscillator to be ready */
    }

    LPC_SC->CCLKCFG  = CCLKCFG_Val;             /* Setup Clock Divider */
    /* Periphral clock must be selected before PLL0 enabling and connecting
     * - according errata.lpc1768-16.March.2010 -
     */
    LPC_SC->PCLKSEL0  = PCLKSEL0_Val;            /* Peripheral Clock Selection */
    LPC_SC->PCLKSEL1  = PCLKSEL1_Val;

    LPC_SC->CLKSRCSEL = CLKSRCSEL_Val;          /* Select Clock Source sysclk / PLL0 */

#endif (PLL0_SETUP)

    LPC_SC->PLL0CFG  = PLL0CFG_Val;             /* configure PLL0 */
    LPC_SC->PLL0FEED  = 0xAA;
    LPC_SC->PLL0FEED  = 0x55;

    LPC_SC->PLL0CON  = 0x01;                     /* PLL0 Enable */
}
```

```

LPC_SC->PLL0FEED = 0xAA;
LPC_SC->PLL0FEED = 0x55;

while (!(LPC_SC->PLL0STAT & (1<<26))); /* Wait for PLOCK0 */



LPC_SC->PLL0CON = 0x03; /* PLL0 Enable & Connect */
LPC_SC->PLL0FEED = 0xAA;
LPC_SC->PLL0FEED = 0x55;

while ((LPC_SC->PLL0STAT & ((1<<25) | (1<<24))) != ((1<<25) | (1<<24))); /* Wait for
PLLC0_STAT & PLLE0_STAT */

#endif

#if (PLL1_SETUP)

LPC_SC->PLL1CFG = PLL1CFG_Val;
LPC_SC->PLL1FEED = 0xAA;
LPC_SC->PLL1FEED = 0x55;

LPC_SC->PLL1CON = 0x01; /* PLL1 Enable */
LPC_SC->PLL1FEED = 0xAA;
LPC_SC->PLL1FEED = 0x55;

while (!(LPC_SC->PLL1STAT & (1<<10))); /* Wait for PLOCK1 */



LPC_SC->PLL1CON = 0x03; /* PLL1 Enable & Connect */
LPC_SC->PLL1FEED = 0xAA;
LPC_SC->PLL1FEED = 0x55;

while ((LPC_SC->PLL1STAT & ((1<< 9) | (1<< 8))) != ((1<< 9) | (1<< 8))); /* Wait for
PLLC1_STAT & PLLE1_STAT */

#else

LPC_SC->USBCLKCFG = USBCLKCFG_Val; /* Setup USB Clock Divider */
#endif

LPC_SC->PCONP = PCONP_Val; /* Power Control for Peripherals */
LPC_SC->CLKOUTCFG = CLKOUTCFG_Val; /* Clock Output Configuration */

#endif

#if (FLASH_SETUP == 1) /* Flash Accelerator Setup */
LPC_SC->FLASHCFG = (LPC_SC->FLASHCFG & ~0x0000F000) | FLASHCFG_Val;
#endif
}

```

```
/**  
 * @}  
 */  
  
/**  
 * @}  
 */  
  
/*-----//  
-----//  
//      @filename      main.c  
//  
//      @version       0.00  
//  
//      @author        Alberto Palomo Alonso  
//  
//  
//  
//      @brief         Código fuente del programa principal.  
//  
//  
//      @category     Principal.  
//  
//  
//      @map          @include  
//  
//      @global       //  
//      @main         //  
//      @end          //  
//  
//  
//-----//  
-----//  
//  
//
```

```
// @include Estos son los archivos utilizados con el código
fuente. //  
//  
//  
//-----**/  
#ifndef HEADER  
#define HEADER  
#include "header.h"  
#endif  
/**-----//  
//  
//  
//-----**/  
// @global Programa principal, variables globales.  
//  
//  
//-----**/  
misDatos_t objDATOS; // Objeto.  
misDatos_t * DATOS = &objDATOS; // Mis datos almacenados  
en la variable objDATOS.  
State_t objESTADO; // Objeto.  
State_t * ESTADO = &objESTADO; // Declarar como extern. (Hey,  
compilador, creeme que hay una variable por ahí que se llama ESTADO)  
Counters_t objCOUNTERS; // Objeto.  
Counters_t * COUNTERS = &objCOUNTERS; // Declarar como extern. (Hey,  
compilador, creeme que hay una variable por ahí que se llama COUNTERS)  
actualizador_t objACTUALIZADOR; // Objeto.  
actualizador_t * ACTUALIZADOR = &objACTUALIZADOR; // Declarar como extern. (Hey,  
compilador, creeme que hay una variable por ahí que se llama ACTUALIZADOR)  
/**-----//  
//  
//-----**/  
// @main Programa principal, inicio after-reset.  
//
```

```
//  
//  
// @ref      __configuraPrograma__ -> configura.h  
//  
//           __mainLoop__          -> statechart.h  
//  
//  
//  
//-----**/  
int main ()  
{  
    __configuraPrograma__();  
    while (1)  
    {  
        __mainLoop__();  
        __mantenerTCP__();  
    }  
}  
/*-----**/  
//  
//  
//-----**/  
//  
// @end      ENDFILE.  
//  
//  
//-----**/  
//  
// @filename Statechart.c  
//  
// @version   0.00  
//
```

Alberto Palomo Alonso.

```
// @author Alberto Palomo Alonso //  
//  
// @brief Código fuente correspondiente a la máquina de estados que  
// compone el menú.  
//  
// @category Principal.  
//  
// @map  
// @include  
// @global  
// @function  
// @end  
//  
//  
//  
//  
//  
//-----//  
//-----//  
// @include Estos son los archivos utilizados en el statechart.  
//-----//  
//-----**/  
#ifndef STATECHART  
#define STATECHART  
#include "Statechart.h"  
#endif  
/*-----//
```

```

//  

//  

//  

// @global      Estas son las variables globales pertenecientes al  

statechart.          //  

//  

//  

//-----**/  

extern uint8_t          Clock[23];  

extern State_t          *ESTADO;  

extern misDatos_t       *DATOS;  

extern actualizador_t  *ACTUALIZADOR;  

extern uint8_t          OWEjecutameExterno;  

extern uint16_t          contadorLUZ;  

modificables_t          MODIFICABLES;  

char buffer[23];  

uint8_t Modo_energetico=0;  

uint8_t Modo_brillo=3;  

uint8_t pressedTouchPanel;  

uint8_t __brilloFade = 0;  

uint8_t __brilloAuto = 0;  

uint8_t Aux8;  

//      ZONA DE PANTALLA DE INICIO.  

screenZone_t zona_0 = { 0 , MAXIMOY , 0 , 0 }; 0 // Marco  

, MAXIMOX , MAXIMOX*0.2 , 0 ; del reloj y botones de alante y atrás.  

screenZone_t zona_1 = { MAXIMOX*0.15 , MAXIMOY*0.2 , 0 , 0 }; 0 // Reloj.  

MAXIMOX*0.7 ,  

screenZone_t zona_2 = { MAXIMOX*0.85 , MAXIMOY*0.2 , 0 , 0 }; 0 // Derecha.  

MAXIMOX*0.15 ,  

screenZone_t zona_3 = { MAXIMOX*0 , MAXIMOY*0.2 , 0 , 0 }; 0 // Izquierda.  

MAXIMOX*0.15 ,  

screenZone_t zona_4 = { MAXIMOX*0 , MAXIMOY*0.3 , 0 , 0 }; MAXIMOY*0.2 , // Primer botón.  

MAXIMOX*0.5 ,  

screenZone_t zona_5 = { MAXIMOX*0.5 , MAXIMOY*0.3 , 0 , 0 }; MAXIMOY*0.2 , // Segundo botón.  

MAXIMOX*0.5 ,  

screenZone_t zona_6 = { MAXIMOX*0 , MAXIMOY*0.15 , 0 , 0 }; MAXIMOY*0.5 , // Brillo info.  

MAXIMOX ,  

screenZone_t zona_7 = { MAXIMOX*0 , MAXIMOY*0.15 , 0 , 0 }; MAXIMOY*0.65 , // Primer botón  

MAXIMOX*0.2 , de brillo.  

screenZone_t zona_8 = { MAXIMOX*0.2 , MAXIMOY*0.15 , 0 , 0 }; MAXIMOY*0.65 , // Segundo botón  

MAXIMOX*0.2 , de brillo.

```

```

screenZone_t zona_9 = { MAXIMOX*0.4 , MAXIMOY*0.65 , de brillo.
MAXIMOX*0.2 , MAXIMOY*0.15 , 0 }; // Tercer botón
de brillo.

screenZone_t zona_10 = { MAXIMOX*0.6 , MAXIMOY*0.65 , MAXIMOX*0.2 , MAXIMOY*0.15 , 0 }; // Cuarto botón
de brillo.

screenZone_t zona_11 = { MAXIMOX*0.8 , MAXIMOY*0.65 , MAXIMOX*0.2 , MAXIMOY*0.15 , 0 }; // Brillo
automático.

screenZone_t zona_12 = { MAXIMOX*0 , MAXIMOY*0.8 , MAXIMOX*0.2 , MAXIMOY*0.2 , 0 }; // Botón de
audio.

screenZone_t zona_13 = { MAXIMOX*0.2 , MAXIMOY*0.8 , MAXIMOX*0.2 , MAXIMOY*0.2 , 0 }; // Volumen = 1.
MAXIMOX*0.2 , MAXIMOY*0.2 , 0 };

screenZone_t zona_14 = { MAXIMOX*0.4 , MAXIMOY*0.8 , MAXIMOX*0.2 , MAXIMOY*0.2 , 0 }; // Volumen = 2.
MAXIMOX*0.2 , MAXIMOY*0.2 , 0 };

screenZone_t zona_15 = { MAXIMOX*0.6 , MAXIMOY*0.8 , MAXIMOX*0.2 , MAXIMOY*0.2 , 0 }; // Volumen = 3.
MAXIMOX*0.2 , MAXIMOY*0.2 , 0 };

screenZone_t zona_16 = { MAXIMOX*0.8 , MAXIMOY*0.8 , MAXIMOX*0.2 , MAXIMOY*0.2 , 0 }; // Botón de load.
MAXIMOX*0.2 , MAXIMOY*0.2 , 0 };

// ZONA DE MEDIDAS.

screenZone_t zona_17 = { MAXIMOX*0 , MAXIMOY*0.2 , MAXIMOX*0.1 , MAXIMOY*0.2 , 0 }; // Información de
página, medidas.
MAXIMOX , MAXIMOY , 0 };

screenZone_t zona_18 = { MAXIMOX*0 , MAXIMOY*0.3 , MAXIMOX*0 , MAXIMOY*0.1 , 0 }; // //
Localización.
MAXIMOX , MAXIMOY , 0 };

screenZone_t zona_19 = { MAXIMOX*0 , MAXIMOY*0.4 , MAXIMOX*0.5 , MAXIMOY*0.15 , 0 }; // Temperatura.
MAXIMOX , MAXIMOY , 0 };

screenZone_t zona_20 = { MAXIMOX*0 , MAXIMOY*0.55 , MAXIMOX*0.5 , MAXIMOY*0.15 , 0 }; // Humedad.
MAXIMOX , MAXIMOY , 0 };

screenZone_t zona_21 = { MAXIMOX*0 , MAXIMOY*0.7 , MAXIMOX*0.5 , MAXIMOY*0.15 , 0 }; // Presión.
MAXIMOX , MAXIMOY , 0 };

screenZone_t zona_22 = { MAXIMOX*0 , MAXIMOY*0.85 , MAXIMOX*0.5 , MAXIMOY*0.15 , 0 }; // IndiceUV.
MAXIMOX , MAXIMOY , 0 };

// ZONAS DE DATOS.

screenZone_t zona_23 = { MAXIMOX*0.4 , MAXIMOY*0.4 , MAXIMOX*0.5 , MAXIMOY*0.15 , 0 }; // Valor de
temperatura.
MAXIMOX , MAXIMOY , 0 };

screenZone_t zona_24 = { MAXIMOX*0.55 , MAXIMOY*0.55 , MAXIMOX*0.5 , MAXIMOY*0.15 , 0 }; // Valor de
humedad.
MAXIMOX , MAXIMOY , 0 };

screenZone_t zona_25 = { MAXIMOX*0.7 , MAXIMOY*0.7 , MAXIMOX*0.5 , MAXIMOY*0.15 , 0 }; // Valor de
presión.
MAXIMOX , MAXIMOY , 0 };

screenZone_t zona_26 = { MAXIMOX*0.85 , MAXIMOY*0.85 , MAXIMOX*0.5 , MAXIMOY*0.15 , 0 }; // Valor de
IndiceUV.
MAXIMOX , MAXIMOY , 0 };

// ZONAS DE AJUSTES.

screenZone_t zona_27 = { MAXIMOX*0 , MAXIMOY*0.2 , MAXIMOX*0.5 , MAXIMOY*0.15 , 0 }; // Horas.
MAXIMOX , MAXIMOY , 0 };

screenZone_t zona_28 = { MAXIMOX*0 , MAXIMOY*0.35 , MAXIMOX*0.5 , MAXIMOY*0.15 , 0 }; // Minutos.
MAXIMOX , MAXIMOY , 0 };

```

```

screenZone_t zona_29 = { MAXIMOX*0 , MAXIMOY*0.15 , 0 , } ; // Segundos.
MAXIMOX*0.5 ,
MAXIMOY*0.65 ,
// Dia.

screenZone_t zona_30 = { MAXIMOX*0 , MAXIMOY*0.15 , 0 , } ; // Dia.
MAXIMOX*0.5 ,
MAXIMOY*0.8 ,
// Slot libre.

screenZone_t zona_27m = { MAXIMOX*0.5 , MAXIMOY*0.15 , 0 , } ; // Resta horas.
MAXIMOX*0.25 ,
MAXIMOY*0.2 ,
// Resta minutos.

screenZone_t zona_28m = { MAXIMOX*0.5 , MAXIMOY*0.15 , 0 , } ; // Resta minutos.
MAXIMOX*0.25 ,
MAXIMOY*0.35 ,
// Resta segundos.

screenZone_t zona_29m = { MAXIMOX*0.5 , MAXIMOY*0.15 , 0 , } ; // Resta segundos.
MAXIMOX*0.25 ,
MAXIMOY*0.5 ,
// Resta dia.

screenZone_t zona_30m = { MAXIMOX*0.5 , MAXIMOY*0.15 , 0 , } ; // Resta dia.
MAXIMOX*0.25 ,
MAXIMOY*0.65 ,
// Resta dia.

screenZone_t zona_27M = { MAXIMOX*0.75 , MAXIMOY*0.15 , 0 , } ; // Suma horas.
MAXIMOX*0.25 ,
MAXIMOY*0.2 ,
// Suma minutos.

screenZone_t zona_28M = { MAXIMOX*0.75 , MAXIMOY*0.15 , 0 , } ; // Suma minutos.
MAXIMOX*0.25 ,
MAXIMOY*0.35 ,
// Suma segundos.

screenZone_t zona_29M = { MAXIMOX*0.75 , MAXIMOY*0.15 , 0 , } ; // Suma segundos.
MAXIMOX*0.25 ,
MAXIMOY*0.5 ,
// Suma dia.

screenZone_t zona_30M = { MAXIMOX*0.75 , MAXIMOY*0.15 , 0 , } ; // Suma dia.
MAXIMOX*0.25 ,
MAXIMOY*0.65 ,
// Suma dia.

// ZONAS DE MEDIDAS 2 (VIENTO)

screenZone_t zona_32 = { MAXIMOX*0 , MAXIMOY*0.2 , 0 , } ; // Velocidad del viento.
MAXIMOX*0.5 ,
MAXIMOY*0.2 ,
// Velocidad del viento.

screenZone_t zona_32n = { MAXIMOX*0.5 , MAXIMOY*0.2 , 0 , } ; // Velocidad del viento.
MAXIMOX*0.5 ,
MAXIMOY*0.2 ,
// Velocidad del viento.

screenZone_t zona_33 = { MAXIMOX*0 , MAXIMOY*0.2 , 0 , } ; // Velocidad del viento.
MAXIMOX ,
MAXIMOY*0.2 ,
// Velocidad del viento.

screenZone_t zona_34 = { MAXIMOX*0 , MAXIMOY*0.2 , 0 , } ; // Cantidad de brillo.
MAXIMOX*0.5 ,
MAXIMOY*0.2 ,
// Cantidad de brillo.

screenZone_t zona_34n = { MAXIMOX*0.5 , MAXIMOY*0.2 , 0 , } ; // Cantidad de brillo.
MAXIMOX*0.5 ,
MAXIMOY*0.2 ,
// Cantidad de brillo.

screenZone_t zona_35 = { MAXIMOX*0 , MAXIMOY*0.2 , 0 , } ; // Cantidad de brillo.
MAXIMOX ,
MAXIMOY*0.2 ,
// Cantidad de brillo.

// Display de barras.

screenZone_t zona_350 = { MAXIMOX*0 , MAXIMOY*0.2 , 0 , } ; // MAXIMOY*0.8
MAXIMOX*0.1 ,
MAXIMOY*0.8 ,
// MAXIMOY*0.8

screenZone_t zona_351 = { MAXIMOX*0.1 , MAXIMOY*0.2 , 0 , } ; // MAXIMOY*0.8
MAXIMOX*0.1 ,
MAXIMOY*0.8 ,
// MAXIMOY*0.8

screenZone_t zona_352 = { MAXIMOX*0.2 , MAXIMOY*0.2 , 0 , } ; // MAXIMOY*0.8
MAXIMOX*0.1 ,
MAXIMOY*0.8 ,
// MAXIMOY*0.8

screenZone_t zona_353 = { MAXIMOX*0.3 , MAXIMOY*0.2 , 0 , } ; // MAXIMOY*0.8
MAXIMOX*0.1 ,
MAXIMOY*0.8 ,
// MAXIMOY*0.8

```

```

screenZone_t zona_354 = { MAXIMOY*0.8 , MAXIMOY*0.4 , MAXIMOY*0.1 , MAXIMOY*0.2 , 0 };
screenZone_t zona_355 = { MAXIMOY*0.8 , MAXIMOY*0.5 , MAXIMOY*0.1 , MAXIMOY*0.2 , 0 };
screenZone_t zona_356 = { MAXIMOY*0.8 , MAXIMOY*0.6 , MAXIMOY*0.1 , MAXIMOY*0.2 , 0 };
screenZone_t zona_357 = { MAXIMOY*0.8 , MAXIMOY*0.7 , MAXIMOY*0.1 , MAXIMOY*0.2 , 0 };
screenZone_t zona_358 = { MAXIMOY*0.8 , MAXIMOY*0.8 , MAXIMOY*0.1 , MAXIMOY*0.2 , 0 };
screenZone_t zona_359 = { MAXIMOY*0.8 , MAXIMOY*0.9 , MAXIMOY*0.1 , MAXIMOY*0.2 , 0 };

// Display de barras.

screenZone_t zona_330 = { MAXIMOY*0.4 , MAXIMOY*0.0 , MAXIMOY*0.1 , MAXIMOY*0.2 , 0 };
screenZone_t zona_331 = { MAXIMOY*0.4 , MAXIMOY*0.1 , MAXIMOY*0.0 , MAXIMOY*0.2 , 0 };
screenZone_t zona_332 = { MAXIMOY*0.4 , MAXIMOY*0.2 , MAXIMOY*0.0 , MAXIMOY*0.2 , 0 };
screenZone_t zona_333 = { MAXIMOY*0.4 , MAXIMOY*0.3 , MAXIMOY*0.0 , MAXIMOY*0.2 , 0 };
screenZone_t zona_334 = { MAXIMOY*0.4 , MAXIMOY*0.4 , MAXIMOY*0.0 , MAXIMOY*0.2 , 0 };
screenZone_t zona_335 = { MAXIMOY*0.4 , MAXIMOY*0.5 , MAXIMOY*0.0 , MAXIMOY*0.2 , 0 };
screenZone_t zona_336 = { MAXIMOY*0.4 , MAXIMOY*0.6 , MAXIMOY*0.0 , MAXIMOY*0.2 , 0 };
screenZone_t zona_337 = { MAXIMOY*0.4 , MAXIMOY*0.7 , MAXIMOY*0.0 , MAXIMOY*0.2 , 0 };
screenZone_t zona_338 = { MAXIMOY*0.4 , MAXIMOY*0.8 , MAXIMOY*0.0 , MAXIMOY*0.2 , 0 };
screenZone_t zona_339 = { MAXIMOY*0.4 , MAXIMOY*0.9 , MAXIMOY*0.0 , MAXIMOY*0.2 , 0 };

// Menú de carga.

screenZone_t zona_lo0 = { MAXIMOY*0.2 , MAXIMOY*0.6 , MAXIMOY*0.1 , MAXIMOY*0.2 , 0 };
screenZone_t zona_lo1 = { MAXIMOY*0.5 , MAXIMOY*0.2 , MAXIMOY*0.6 , MAXIMOY*0.2 , 0 };
screenZone_t zona_lo20 = { MAXIMOY*0.7 , MAXIMOY*0.2 , MAXIMOY*0.2 , MAXIMOY*0.15 , 0 };
screenZone_t zona_lo21 = { MAXIMOY*0.7 , MAXIMOY*0.35 , MAXIMOY*0.2 , MAXIMOY*0.15 , 0 };
screenZone_t zona_lo22 = { MAXIMOY*0.7 , MAXIMOY*0.5 , MAXIMOY*0.2 , MAXIMOY*0.15 , 0 };
screenZone_t zona_lo23 = { MAXIMOY*0.7 , MAXIMOY*0.65 , MAXIMOY*0.2 , MAXIMOY*0.15 , 0 };

/*-----*/
-----//



//




```

```
//  
//          @function      __mainLoop__()  
//  
//  
//          @brief           Esta función es la que se ejecuta en el bucle principal del  
main. Debe contener  
//  
//                                todo el código ejecutable por el loop principal.  
//  
//  
//  
//-----**/  
void    __mainLoop__( void )  
{  
    /**  @LOOP: Primera parte del programa. */  
    alimentaWDT();  
    checkTouchPanel();  
    if ( __brilloAuto && (SysTick->CTRL & 0x10000) )           //      Cada 100 ms si el  
brillo auto está activado.  
    {  
        goto_LUT( DATOS->Brillo, BRILLO2CICLO_LDR , none , &Aux8 , none , none );  
        modificaPulso( PWM6      ,      MODO_CICLO      ,      Aux8      ,      none  
        ,      none      ,      none      );  
    }  
    /**  @LOOP: Máquina de estados LCD. */  
    switch( ESTADO->CHART )  
    {  
        case PANTALLA_INICIO:  
            __pintaInicio__();  
            if ( zoneNewPressed( &zona_2 ) )  
            {  
                ESTADO->CHART = PANTALLA_MEDIDAS1;  
                LCD_Clear(Black);  
            }  
            if ( zoneNewPressed( &zona_3 ) )  
            {  
                ESTADO->CHART = PANTALLA_MEDIDAS2;  
                LCD_Clear(Black);  
            }  
            if ( zoneNewPressed( &zona_5 ) )  
    }
```

```

{
    ESTADO->CHART = PANTALLA_AJUSTES;
    LCD_Clear(Black);
}

if (zoneNewPressed( &zona_4 ))
{
    ESTADO->CHART = PANTALLA_VALORES;
    LCD_Clear(Black);
}

if (zoneNewPressed( &zona_7 ))
{
    __brilloAuto = 0;
    __brilloFade = 0;
    modificaPulso( PWM6 , MODO_CICLO , 1
, none , none , none );
    Modo_brillo = 0;
    Modo_energetico = 0; // HP.
}

if (zoneNewPressed( &zona_8 ))
{
    __brilloAuto = 0;
    __brilloFade = 0;
    modificaPulso( PWM6 , MODO_CICLO , 20
, none , none , none );
    Modo_brillo = 1;
    Modo_energetico = 0; // HP.
}

if (zoneNewPressed( &zona_9 ))
{
    __brilloAuto = 0;
    __brilloFade = 0;
    modificaPulso( PWM6 , MODO_CICLO , 40
, none , none , none );
    Modo_brillo = 2;
    Modo_energetico = 0; // HP.
}

if (zoneNewPressed( &zona_10 ))
{
    modificaPulso( PWM6 , MODO_CICLO , 60
, none , none , none );
    __brilloAuto = 0;
}

```

```

        __brilloFade = 0;
        Modo_brillo = 3;
        Modo_energetico = 0; // HP.

    }

    if (zoneNewPressed( &zona_11))
    {

        __brilloAuto = 1;
        __brilloFade = 0;
        Modo_brillo = 4;
        Modo_energetico = 1; // LP.

    }

    if (zoneNewPressed( &zona_12))
    {

        if (ACTUALIZADOR->Audiorev)
        {

            ACTUALIZADOR->Audiorev = 0;
            __configuraAudio_();
            activarDac();
        }

    }

    if (zoneNewPressed( &zona_16))
    {

        if (ACTUALIZADOR->Audiorev)
        {

            ACTUALIZADOR->Audiorev = 0;
            lanzaUFONO();
        }

    }

}

if (zoneNewPressed( &zona_13))
{

    Modo_energetico = 0; // HP.
    __brilloAuto = 0; // No hay brillo
auto.

    Modo_brillo = 3; // Brillo a tope.

    , none , modificaPulso( PWM6 , MODO_CICLO , 60
    , none , none );
    __brilloFade = 0;

}

```

```
        if (zoneNewPressed(    &zona_14))
        {
            Modo_energetico      =      1;      //      LP.
            Modo_brillo           =      4;      //      Brillo auto.
            __brilloAuto          =      1;      //      Activo el
brillo automático.
            __brilloFade          =      0;      //      No pueda
apagarse la pantalla.

        }

        if (zoneNewPressed(    &zona_15))
        {
            Modo_energetico      =      2;      //      ULP.
            Modo_brillo           =      4;      //      Brillo auto.
            __brilloAuto          =      1;      //      Activo el
brillo automático.
            __brilloFade          =      1;      //      Que pueda
apagarse la pantalla.

        }

        break;

case PANTALLA_MEDIDAS1:
    __pintaMedidas1__();
    if (zoneNewPressed(    &zona_2))
    {
        ESTADO->CHART = PANTALLA_MEDIDAS2;
        LCD_Clear(Black);
    }
    if (zoneNewPressed(    &zona_3))
    {
        ESTADO->CHART = PANTALLA_INICIO;
        LCD_Clear(Black);
    }
    break;

case PANTALLA_MEDIDAS2:
    __pintaMedidas2__();
    if (zoneNewPressed(    &zona_2))
    {
        ESTADO->CHART = PANTALLA_INICIO;
```

```
    LCD_Clear(Black);

}

if (zoneNewPressed( &zona_3))

{

    ESTADO->CHART = PANTALLA_MEDIDAS1;

    LCD_Clear(Black);

}

break;

case PANTALLA_AJUSTES:

    __pintaAjustes__();

if (zoneNewPressed( &zona_2))

{

    ESTADO->CHART = PANTALLA_INICIO;

    LCD_Clear(Black);

}

if (zoneNewPressed( &zona_3))

{

    ESTADO->CHART = PANTALLA_INICIO;

    LCD_Clear(Black);

}

if (zoneNewPressed( &zona_27m))

{

    LPC_RTC->HOUR--;

}

if (zoneNewPressed( &zona_28m))

{

    LPC_RTC->MIN--;

}

if (zoneNewPressed( &zona_29m))

{

    LPC_RTC->SEC--;

}

if (zoneNewPressed( &zona_30m))

{

    LPC_RTC->DOM--;

}

if (zoneNewPressed( &zona_27M))

{
```

```
LPC_RTC->HOUR++;
}

if (zoneNewPressed( &zona_28M ))
{
    LPC_RTC->MIN++;
}

if (zoneNewPressed( &zona_29M ))
{
    LPC_RTC->SEC++;
}

if (zoneNewPressed( &zona_30M ))
{
    LPC_RTC->DOM++;
}

break;

case PANTALLA_LOADING:
break;

case PANTALLA_VALORES:
    __pintaValores__();
    if (zoneNewPressed( &zona_2 ))
    {
        ESTADO->CHART = PANTALLA_INICIO;
        LCD_Clear(Black);
    }
    if (zoneNewPressed( &zona_3 ))
    {
        ESTADO->CHART = PANTALLA_INICIO;
        LCD_Clear(Black);
    }
    if (zoneNewPressed( &zona_27m ))
    {
        MODIFICABLES.Min_servo_t--;
    }
    if (zoneNewPressed( &zona_28m ))
    {
        MODIFICABLES.Max_servo_t--;
    }
    if (zoneNewPressed( &zona_29m ))
```

```
{  
    MODIFICABLES.Min_servo_p -= 10;  
}  
  
if (zoneNewPressed( &zona_30m ))  
{  
    MODIFICABLES.Max_servo_p -= 10;  
}  
  
if (zoneNewPressed( &zona_27M ))  
{  
    MODIFICABLES.Min_servo_t++;  
}  
  
if (zoneNewPressed( &zona_28M ))  
{  
    MODIFICABLES.Max_servo_t++;  
}  
  
if (zoneNewPressed( &zona_29M ))  
{  
    MODIFICABLES.Min_servo_p += 10;  
}  
  
if (zoneNewPressed( &zona_30M ))  
{  
    MODIFICABLES.Max_servo_p += 10;  
}  
  
if (zoneNewPressed( &zona_31 ))  
{  
    MODIFICABLES.Var_medida = 1 - MODIFICABLES.Var_medida;  
}  
  
default:  
    break;  
};  
}  
/*-----//  
//  
//  
//  
@function      __configuraLCD__()  
//
```

```
//  
//  
// @brief Esta función configura el TFT HY32B conectado al driver  
// ILI9325C  
//  
//  
//-----**/  
void __configuraLCD__( void )  
{  
    TP_Init();  
    LCD_Initialization();  
}  
/*-----**/  
//  
//  
//-----**/  
//  
//  
// @function __pintaInicio__()  
//  
//  
// @brief Esta función pinta la pantalla de inicio.  
//  
//  
//-----**/  
void __pintaInicio__( void )  
{  
    squareButton( &zona_1 , (char *)Clock , Yellow , Green );  
    squareButton( &zona_2 , ">" , Yellow , Green );  
    squareButton( &zona_3 , "<" , Yellow , Green );  
    squareButton( &zona_4 , "Valores" , Yellow , Green );  
    squareButton( &zona_5 , "Ajustes" , Yellow , Green );  
    squareButton( &zona_6 , "Nivel de brillo:" , Yellow , Green );  
    switch ( Modo_brillo )  
{
```

```
        case 0:
            squareButton( &zona_7 ,      "1"
White ,      White );
            squareButton( &zona_8 ,      "2"
Yellow ,     Green );
            squareButton( &zona_9 ,      "3"
Yellow ,     Green );
            squareButton( &zona_10 ,     "4"
,      Yellow ,     Green );
            squareButton( &zona_11 ,     "A"
,      Yellow ,     Green );
            break;

        case 1:
            squareButton( &zona_7 ,      "1"
Yellow ,     Green );
            squareButton( &zona_8 ,      "2"
White ,      White );
            squareButton( &zona_9 ,      "3"
Yellow ,     Green );
            squareButton( &zona_10 ,     "4"
,      Yellow ,     Green );
            squareButton( &zona_11 ,     "A"
,      Yellow ,     Green );
            break;

        case 2:
            squareButton( &zona_7 ,      "1"
Yellow ,     Green );
            squareButton( &zona_8 ,      "2"
Yellow ,     Green );
            squareButton( &zona_9 ,      "3"
White ,      White );
            squareButton( &zona_10 ,     "4"
,      Yellow ,     Green );
            squareButton( &zona_11 ,     "A"
,      Yellow ,     Green );
            break;

        case 3:
            squareButton( &zona_7 ,      "1"
Yellow ,     Green );
            squareButton( &zona_8 ,      "2"
Yellow ,     Green );
            squareButton( &zona_9 ,      "3"
Yellow ,     Green );
            squareButton( &zona_10 ,     "4"
,      White ,      White );
            squareButton( &zona_11 ,     "A"
,      Yellow ,     Green );
            break;
```

```

        case 4:
            squareButton( &zona_7 ,      "1"
Yellow ,      Green );
            squareButton( &zona_8 ,      "2"
Yellow ,      Green );
            squareButton( &zona_9 ,      "3"
Yellow ,      Green );
            squareButton( &zona_10 ,     "4"
,      Yellow ,      Green );
            squareButton( &zona_11 ,     "A"
,      White ,      White );
            break;
        }

        squareButton( &zona_12 ,      "Play"
Green );
        switch ( Modo_energetico )
        {
            case 0:
                squareButton( &zona_13 ,      "HP"
,      Red );
                squareButton( &zona_14 ,      "LP"
,      Yellow ,      Green );
                squareButton( &zona_15 ,      "ULP"
Yellow ,      Green );
                break;
            case 1:
                squareButton( &zona_13 ,      "HP"
,      Yellow ,      Green );
                squareButton( &zona_14 ,      "LP"
,      Blue );
                squareButton( &zona_15 ,      "ULP"
Yellow ,      Green );
                break;
            case 2:
                squareButton( &zona_13 ,      "HP"
,      Yellow ,      Green );
                squareButton( &zona_14 ,      "LP"
,      Yellow ,      Green );
                squareButton( &zona_15 ,      "ULP"
White ,      White );
                break;
        }

        squareButton( &zona_16 ,      "Load"
Green );
    }

    /*-----*/

```

```
//  
//  
//  
// @function      __pintaAjustes__()  
//  
//  
// @brief        Esta función pinta la pantalla de ajustes.  
//  
//  
//  
//-----**/  
void  __pintaAjustes__( void )  
{  
    squareButton( &zona_1 ,      (char *)Clock ,      Yellow ,      Green  
);  
    squareButton( &zona_2 ,      "->" ,      Yellow ,  
Green );  
    squareButton( &zona_3 ,      "<" ,      Yellow ,  
Green );  
    squareButton( &zona_27 ,      "Horas" ,      Yellow ,  
Green );  
    squareButton( &zona_28 ,      "Minutos" ,      Yellow ,  
Green );  
    squareButton( &zona_29 ,      "Segundos" ,      Yellow ,  
Green );  
    squareButton( &zona_30 ,      "Dias" ,      Yellow ,  
Green );  
    sprintf((char *)buffer ,      "IP:%d.%d.%d.%d" ,      __IP1B ,      __IP2B  
,      __IP3B ,      __IP4B );  
    squareButton( &zona_31 ,      (char*)buffer ,      Yellow ,  
Green );  
    squareButton( &zona_27m ,      "-" ,      Yellow  
,  
Green );  
    squareButton( &zona_28m ,      "-" ,      Yellow  
,  
Green );  
    squareButton( &zona_29m ,      "-" ,      Yellow  
,  
Green );  
    squareButton( &zona_30m ,      "-" ,      Yellow  
,  
Green );  
    squareButton( &zona_27M ,      "+" ,      Yellow  
,  
Green );  
    squareButton( &zona_28M ,      "+" ,      Yellow  
,  
Green );  
    squareButton( &zona_29M ,      "+" ,      Yellow  
,  
Green );
```

```

        squareButton( &zona_30M , " +" , Yellow
        , Green );
    }

/*-----*/
//



//          @function      __pintaValores__()
//



//          @brief      Esta función pinta la pantalla de valores.
//



//



//



//-----*/
//



void      __pintaValores__( void )
{
    squareButton( &zona_1 , (char *)Clock , Yellow , Green
    );

    squareButton( &zona_2 , "->" , Yellow ,
    Green );
    squareButton( &zona_3 , "<-" , Yellow ,
    Green );
    squareButton( &zona_27 , "Temp.min." , Yellow ,
    Green );
    squareButton( &zona_28 , "Temp.max." , Yellow ,
    Green );
    squareButton( &zona_29 , "Pres.min." , Yellow ,
    Green );
    squareButton( &zona_30 , "Pres.max." , Yellow ,
    Green );
    sprintf((char *)buffer , "Pres [%d,%d] Temp [%d,%d]" ,
    (int)MODIFICABLES.Min_servo_p , (int)MODIFICABLES.Max_servo_p,
    (int)MODIFICABLES.Min_servo_t , (int)MODIFICABLES.Max_servo_t);

    if ( MODIFICABLES.Var_medida )
    {
        squareButton( &zona_31 , (char*)buffer , Green
        , Green );
    }
    else
    {
        squareButton( &zona_31 , (char*)buffer , Red
        , Green );
    }
}

```



```
//  
//          @function      __pintaCargandoSeno__()  
//  
//  
//          @brief      Esta función pinta la pantalla cargando seno.  
//  
//  
//  
//          @function      __pintaCargandoConexion__()  
//  
//  
//          @brief      Esta función pinta la pantalla cargando conexión.  
//  
//  
//-----**/  
void    __pintaCargandoSeno__(void    )  
{  
    squareButton( &zona_lo0      ,      "CARGANDO ..."      ,  
        Blue   ,      Green   );  
  
    squareButton( &zona_lo1      ,      "Creando muestras..." ,      Blue   ,  
        Green  );  
  
    squareBox(    &zona_lo20,      White   );  
}  
/*-----**/  
//  
//  
//  
//-----**/  
void    __pintaCargandoConexion__( void    )  
{  
    squareButton( &zona_lo0      ,      "CARGANDO ..."      ,  
        Blue   ,      Green   );  
  
    squareButton( &zona_lo1      ,      "Buscando conexion TCP..." ,      Blue   ,  
        Green  );  
  
    squareBox(    &zona_lo20,      White   );  
    squareBox(    &zona_lo21,      White   );  
}
```

```
/**-----//  
//  
//  
//  
//  
//  
//  
//  
@function      __pintaCargandoIniciando__()  
//  
//  
//  
//  
@brief        Esta función pinta la pantalla cargando iniciando.  
//  
//  
//  
//-----**/  
void  __pintaCargandoIniciando__( void    )  
{  
    squareButton( &zona_lo0      ,      "CARGANDO ..."      ,  
        Blue   ,      Green   );  
  
    squareButton( &zona_lo1      ,      "Iniciando modulos..."      ,      Blue  
        ,      Green   );  
  
    squareBox(   &zona_lo20,      White   );  
    squareBox(   &zona_lo21,      White   );  
    squareBox(   &zona_lo22,      White   );  
}  
/**-----//  
//  
//  
//  
//  
//  
//  
//  
//  
//  
//  
//  
//  
//  
//-----**/  
void  __pintaCargandoDone__(void    )  
{
```


Alberto Palomo Alonso.

```
squareButton( &zona_1 , (char *)Clock , Yellow , Green );
squareButton( &zona_2 , "->" , Yellow , Green );
squareButton( &zona_3 , "<" , Yellow , Green );
squareButton( &zona_32 , "Temperatura:" , Yellow , Green );
if ( ACTUALIZADOR->TempRev )
{
    sprintf((char*)buffer,"%02f dC", DATOS->Temperatura);
    squareButton( &zona_32n , (char *)buffer , Yellow , Green );
    sprintf((char*)buffer,"%02f mBar.", DATOS->Presion);
    squareButton( &zona_34n , (char *)buffer , Yellow , Green );
    ACTUALIZADOR->TempRev = 0;
} // Digo que toca medir.
switch ( (int)(10*(DATOS->Temperatura - MIN_TEMP)/(MAX_TEMP - MIN_TEMP)) )
{
    case 0:
        squareBox( &zona_330 , Black);
        squareBox( &zona_331 , Black);
        squareBox( &zona_332 , Black);
        squareBox( &zona_333 , Black);
        squareBox( &zona_334 , Black);
        squareBox( &zona_335 , Black);
        squareBox( &zona_336 , Black);
        squareBox( &zona_337 , Black);
        squareBox( &zona_338 , Black);
        squareBox( &zona_339 , Black);
        break;
    case 1:
        squareBox( &zona_330 , White);
        squareBox( &zona_331 , Black);
        squareBox( &zona_332 , Black);
        squareBox( &zona_333 , Black);
        squareBox( &zona_334 , Black);
        squareBox( &zona_335 , Black);
        squareBox( &zona_336 , Black);
        squareBox( &zona_337 , Black);
        squareBox( &zona_338 , Black);
```

```
    squareBox( &zona_339 , Black);
    break;

case 2:
    squareBox( &zona_330 , White);
    squareBox( &zona_331 , White);
    squareBox( &zona_332 , Black);
    squareBox( &zona_333 , Black);
    squareBox( &zona_334 , Black);
    squareBox( &zona_335 , Black);
    squareBox( &zona_336 , Black);
    squareBox( &zona_337 , Black);
    squareBox( &zona_338 , Black);
    squareBox( &zona_339 , Black);
    break;

case 3:
    squareBox( &zona_330 , Yellow);
    squareBox( &zona_331 , Yellow);
    squareBox( &zona_332 , Yellow);
    squareBox( &zona_333 , Black);
    squareBox( &zona_334 , Black);
    squareBox( &zona_335 , Black);
    squareBox( &zona_336 , Black);
    squareBox( &zona_337 , Black);
    squareBox( &zona_338 , Black);
    squareBox( &zona_339 , Black);
    break;

case 4:
    squareBox( &zona_330 , Yellow);
    squareBox( &zona_331 , Yellow);
    squareBox( &zona_332 , Yellow);
    squareBox( &zona_333 , Yellow);
    squareBox( &zona_334 , Black);
    squareBox( &zona_335 , Black);
    squareBox( &zona_336 , Black);
    squareBox( &zona_337 , Black);
    squareBox( &zona_338 , Black);
    squareBox( &zona_339 , Black);
    break;

case 5:
```

```
    squareBox( &zona_330 , Blue);
    squareBox( &zona_331 , Blue);
    squareBox( &zona_332 , Blue);
    squareBox( &zona_333 , Blue);
    squareBox( &zona_334 , Blue);
    squareBox( &zona_335 , Black);
    squareBox( &zona_336 , Black);
    squareBox( &zona_337 , Black);
    squareBox( &zona_338 , Black);
    squareBox( &zona_339 , Black);
    break;

case 6:
    squareBox( &zona_330 , Blue);
    squareBox( &zona_331 , Blue);
    squareBox( &zona_332 , Blue);
    squareBox( &zona_333 , Blue);
    squareBox( &zona_334 , Blue);
    squareBox( &zona_335 , Blue);
    squareBox( &zona_336 , Black);
    squareBox( &zona_337 , Black);
    squareBox( &zona_338 , Black);
    squareBox( &zona_339 , Black);
    break;

case 7:
    squareBox( &zona_330 , Green);
    squareBox( &zona_331 , Green);
    squareBox( &zona_332 , Green);
    squareBox( &zona_333 , Green);
    squareBox( &zona_334 , Green);
    squareBox( &zona_335 , Green);
    squareBox( &zona_336 , Green);
    squareBox( &zona_337 , Black);
    squareBox( &zona_338 , Black);
    squareBox( &zona_339 , Black);
    break;

case 8:
    squareBox( &zona_330 , Green);
    squareBox( &zona_331 , Green);
    squareBox( &zona_332 , Green);
```

```
    squareBox( &zona_333 , Green);
    squareBox( &zona_334 , Green);
    squareBox( &zona_335 , Green);
    squareBox( &zona_336 , Green);
    squareBox( &zona_337 , Green);
    squareBox( &zona_338 , Black);
    squareBox( &zona_339 , Black);
    break;

case 9:
    squareBox( &zona_330 , Red);
    squareBox( &zona_331 , Red);
    squareBox( &zona_332 , Red);
    squareBox( &zona_333 , Red);
    squareBox( &zona_334 , Red);
    squareBox( &zona_335 , Red);
    squareBox( &zona_336 , Red);
    squareBox( &zona_337 , Red);
    squareBox( &zona_338 , Red);
    squareBox( &zona_339 , Black);
    break;

case 10:
    squareBox( &zona_330 , Red);
    squareBox( &zona_331 , Red);
    squareBox( &zona_332 , Red);
    squareBox( &zona_333 , Red);
    squareBox( &zona_334 , Red);
    squareBox( &zona_335 , Red);
    squareBox( &zona_336 , Red);
    squareBox( &zona_337 , Red);
    squareBox( &zona_338 , Red);
    squareBox( &zona_339 , Red);
    break;

default:
    if ( DATOS->Temperatura > MIN_TEMP)
    {
        squareBox( &zona_330 , Red);
        squareBox( &zona_331 , Red);
        squareBox( &zona_332 , Red);
        squareBox( &zona_333 , Red);
```

```
        squareBox( &zona_334 , Red);
        squareBox( &zona_335 , Red);
        squareBox( &zona_336 , Red);
        squareBox( &zona_337 , Red);
        squareBox( &zona_338 , Red);
        squareBox( &zona_339 , Red);

    }

    if ( DATOS->Temperatura < MIN_TEMP)
    {

        squareBox( &zona_330 , Black);
        squareBox( &zona_331 , Black);
        squareBox( &zona_332 , Black);
        squareBox( &zona_333 , Black);
        squareBox( &zona_334 , Black);
        squareBox( &zona_335 , Black);
        squareBox( &zona_336 , Black);
        squareBox( &zona_337 , Black);
        squareBox( &zona_338 , Black);
        squareBox( &zona_339 , Black);

    }

};

squareButton( &zona_34 , "Presion:" , Yellow ,
Green );
switch ( (int)(10*(DATOS->Presion - MIN_PRES)/(MAX_PRES - MIN_PRES))
)

{
    case 0:
        squareBox( &zona_350 , Black);
        squareBox( &zona_351 , Black);
        squareBox( &zona_352 , Black);
        squareBox( &zona_353 , Black);
        squareBox( &zona_354 , Black);
        squareBox( &zona_355 , Black);
        squareBox( &zona_356 , Black);
        squareBox( &zona_357 , Black);
        squareBox( &zona_358 , Black);
        squareBox( &zona_359 , Black);
        break;

    case 1:
```

```
    squareBox( &zona_350 , White);
    squareBox( &zona_351 , Black);
    squareBox( &zona_352 , Black);
    squareBox( &zona_353 , Black);
    squareBox( &zona_354 , Black);
    squareBox( &zona_355 , Black);
    squareBox( &zona_356 , Black);
    squareBox( &zona_357 , Black);
    squareBox( &zona_358 , Black);
    squareBox( &zona_359 , Black);
    break;

case 2:
    squareBox( &zona_350 , White);
    squareBox( &zona_351 , White);
    squareBox( &zona_352 , Black);
    squareBox( &zona_353 , Black);
    squareBox( &zona_354 , Black);
    squareBox( &zona_355 , Black);
    squareBox( &zona_356 , Black);
    squareBox( &zona_357 , Black);
    squareBox( &zona_358 , Black);
    squareBox( &zona_359 , Black);
    break;

case 3:
    squareBox( &zona_350 , Yellow);
    squareBox( &zona_351 , Yellow);
    squareBox( &zona_352 , Yellow);
    squareBox( &zona_353 , Black);
    squareBox( &zona_354 , Black);
    squareBox( &zona_355 , Black);
    squareBox( &zona_356 , Black);
    squareBox( &zona_357 , Black);
    squareBox( &zona_358 , Black);
    squareBox( &zona_359 , Black);
    break;

case 4:
    squareBox( &zona_350 , Yellow);
    squareBox( &zona_351 , Yellow);
    squareBox( &zona_352 , Yellow);
```

```
    squareBox( &zona_353 , Yellow);
    squareBox( &zona_354 , Black);
    squareBox( &zona_355 , Black);
    squareBox( &zona_356 , Black);
    squareBox( &zona_357 , Black);
    squareBox( &zona_358 , Black);
    squareBox( &zona_359 , Black);
    break;

case 5:
    squareBox( &zona_350 , Blue);
    squareBox( &zona_351 , Blue);
    squareBox( &zona_352 , Blue);
    squareBox( &zona_353 , Blue);
    squareBox( &zona_354 , Blue);
    squareBox( &zona_355 , Black);
    squareBox( &zona_356 , Black);
    squareBox( &zona_357 , Black);
    squareBox( &zona_358 , Black);
    squareBox( &zona_359 , Black);
    break;

case 6:
    squareBox( &zona_350 , Blue);
    squareBox( &zona_351 , Blue);
    squareBox( &zona_352 , Blue);
    squareBox( &zona_353 , Blue);
    squareBox( &zona_354 , Blue);
    squareBox( &zona_355 , Blue);
    squareBox( &zona_356 , Black);
    squareBox( &zona_357 , Black);
    squareBox( &zona_358 , Black);
    squareBox( &zona_359 , Black);
    break;

case 7:
    squareBox( &zona_350 , Green);
    squareBox( &zona_351 , Green);
    squareBox( &zona_352 , Green);
    squareBox( &zona_353 , Green);
    squareBox( &zona_354 , Green);
    squareBox( &zona_355 , Green);
```

```
    squareBox( &zona_356 , Green);
    squareBox( &zona_357 , Black);
    squareBox( &zona_358 , Black);
    squareBox( &zona_359 , Black);
    break;

case 8:
    squareBox( &zona_350 , Green);
    squareBox( &zona_351 , Green);
    squareBox( &zona_352 , Green);
    squareBox( &zona_353 , Green);
    squareBox( &zona_354 , Green);
    squareBox( &zona_355 , Green);
    squareBox( &zona_356 , Green);
    squareBox( &zona_357 , Green);
    squareBox( &zona_358 , Black);
    squareBox( &zona_359 , Black);
    break;

case 9:
    squareBox( &zona_350 , Red);
    squareBox( &zona_351 , Red);
    squareBox( &zona_352 , Red);
    squareBox( &zona_353 , Red);
    squareBox( &zona_354 , Red);
    squareBox( &zona_355 , Red);
    squareBox( &zona_356 , Red);
    squareBox( &zona_357 , Red);
    squareBox( &zona_358 , Red);
    squareBox( &zona_359 , Black);
    break;

case 10:
    squareBox( &zona_350 , Red);
    squareBox( &zona_351 , Red);
    squareBox( &zona_352 , Red);
    squareBox( &zona_353 , Red);
    squareBox( &zona_354 , Red);
    squareBox( &zona_355 , Red);
    squareBox( &zona_356 , Red);
    squareBox( &zona_357 , Red);
    squareBox( &zona_358 , Red);
```

```
        squareBox( &zona_359 , Red);
        break;
default:
    if ( DATOS->Presion > MAX_PRES)
    {
        squareBox( &zona_350 , Red);
        squareBox( &zona_351 , Red);
        squareBox( &zona_352 , Red);
        squareBox( &zona_353 , Red);
        squareBox( &zona_354 , Red);
        squareBox( &zona_355 , Red);
        squareBox( &zona_356 , Red);
        squareBox( &zona_357 , Red);
        squareBox( &zona_358 , Red);
        squareBox( &zona_359 , Red);
    }
    if ( DATOS->Presion < MIN_PRES)
    {
        squareBox( &zona_350 , Black);
        squareBox( &zona_351 , Black);
        squareBox( &zona_352 , Black);
        squareBox( &zona_353 , Black);
        squareBox( &zona_354 , Black);
        squareBox( &zona_355 , Black);
        squareBox( &zona_356 , Black);
        squareBox( &zona_357 , Black);
        squareBox( &zona_358 , Black);
        squareBox( &zona_359 , Black);
    }
}
/*-----*/
//          @function      squareButton()
//
```

```
//  
//  
// @brief Dibuja un botón cuadrado, con texto y colores.  
//  
//  
//  
//-----**/  
void squareButton(screenZone_t* zone, char * text, uint16_t textColor, uint16_t  
lineColor)  
{  
    LCD_DrawLine( zone->x, zone->y, zone->x + zone->size_x - 1, zone->y, lineColor);  
    LCD_DrawLine( zone->x, zone->y, zone->x, zone->y + zone->size_y - 1, lineColor);  
    LCD_DrawLine( zone->x, zone->y + zone->size_y - 1, zone->x + zone->size_x - 1,  
zone->y + zone->size_y - 1, lineColor);  
    LCD_DrawLine( zone->x + zone->size_x - 1, zone->y, zone->x + zone->size_x - 1,  
zone->y + zone->size_y - 1, lineColor);  
    GUI_Text(zone->x + zone->size_x/2 - (strlen(text)/2)*8, zone->y + zone->size_y/2  
- 8, (uint8_t*) text, textColor, Black);  
}  
/*-----**/  
//  
//  
//-----**/  
//  
//-----**/  
//  
//-----**/  
// @function squareBox()  
//  
//  
// @brief Dibuja un cuadrado de un color.  
//  
//  
//-----**/  
void squareBox(screenZone_t* zone, uint16_t color)  
{  
    int i;  
    for (i = 0; i < (zone->size_x - 4) ; i++)  
    {  
        LCD_DrawLine( zone->x + i + 2, zone->y + 2, zone->x + i + 2, zone->y +  
zone->size_y - 2, color);  
    }  
}
```

Alberto Palomo Alonso.

```

}

/*-----
-----*/
//



//          @function      checkTouchPanel()
//



//          @brief       Verifica se si ha tocado la pantalla.
//



//          @brief       Verifica si se ha presionado una cierta zona de la
pantalla.
//



void checkTouchPanel(void)
{
    Coordinate* coord;

    coord = Read_Ads7846();

    if (coord > 0) {
        getDisplayPoint(&display, coord, &matrix );
        pressedTouchPanel = 1;
    }
    else
        pressedTouchPanel = 0;
}

/*-----
-----*/
//



//          @function      zoneNewPressed()
//



//          @brief       Verifica si se ha presionado una cierta zona de la
pantalla.
//
```

```
//  
//  
// @zone Zona a comprobar.  
//  
//  
// @return 0 - Si no se ha producido un toque.  
//  
// 1 - Si se ha producido un toque.  
//  
//  
//-----**/  
int8_t zoneNewPressed(screenZone_t * zone)  
{  
    if (pressedTouchPanel == 1) {  
  
        if ((display.x > zone->x) && (display.x < zone->x + zone->size_x) &&  
            (display.y > zone->y) && (display.y < zone->y + zone->size_y))  
        {  
            if (zone->pressed == 0)  
            {  
                zone->pressed = 1;  
                return 1;  
            }  
            return 0;  
        }  
        /* MOD: Esto lo he añadido yo */  
        if (contadorLUZ >= (FREQ_OVERFLOW_SYSTICK *  
MODIFICABLES.TiempoBrillo)) // Si se ha activado el apagar pantalla...  
        {  
            modificaPulso ( PWM6, MODO_CICLO , 60 , ); //  
            none , none , none );  
            La enciendo como si hubiese habido un reset.  
            Modo_brillo = 3;  
            if ( Modo_energetico > 1 )  
            {  
                __brilloAuto = 1;  
                Modo_brillo = 4;  
                if ( Modo_energetico == 2 )  
                {  
  
    }
```

```
        __brilloFade = 1;
    }

}

contadorLUZ = 0;           //      Reseteo el contador de apagar la pantalla.
}

zone->pressed = 0;
return 0;
}

/**-----//  
//  
//  
//  
//  
//  
//-----**/  
  
/******Copyright  
(c)*****  
**  
**                         http://www.powermcu.com  
**  
**-----File Info-----  
--  
** File name:          TouchPanel.c  
** Descriptions:       The TouchPanel application function  
**  
**-----  
--  
** Created by:         AVRman  
** Created date:       2010-11-7  
** Version:            v1.0  
** Descriptions:       The original version  
**
```

```
**-----  
-----  
** Modified by:  
** Modified date:  
** Version:  
** Descriptions:  
**  
*****  
*****/  
  
/* Includes -----*/  
#include "GLCD.h"  
#include "TouchPanel.h"  
  
//-----//  
//      INCLUIDO POR ALBERTO PALOMO.  
#ifndef WDT  
#define WDT  
#include      "WDT.h"  
#endif  
//-----//  
  
/* Private variables -----*/  
Matrix matrix = {17835,-368445,-20450106,515085,6180,-2018789910,-5820405};  
Coordinate display ;  
  
/* DisplaySample LCD struct */  
Coordinate ScreenSample[3];  
  
/* LCD Calibration coordinates */  
Coordinate DisplaySample[3] = {{45,45},{45,270}, {190,190}} ;  
  
/* Private define -----*/  
#define THRESHOLD 80    /* Be careful: this parameter should be tuned for your board  
                         to avoid measure glitches */  
  
*****  
* Function Name : TP_Init
```

```
* Description      : ADS7843 and SPI config
* Input           : None
* Output          : None
* Return          : None
* Attention       : None
*****
void TP_Init(void)
{
    PINSEL_CFG_Type PinCfg;
    SSP_CFG_Type SSP_ConfigStruct;

    /*
     * Initialize SPI pin connect
     * P0.6 - TP_CS - used as GPIO
     * P0.7 - TP_SCK
     * P0.8 - TP_SDO
     * P0.9 - TP_SDI
     * P2.13 - TP_IRQ - used as GPIO
     */
    PinCfg.Funcnum = 2;
    PinCfg.OpenDrain = 0;
    PinCfg.Pinmode = 0;
    PinCfg.Portnum = 0;
    PinCfg.Pinnum = 7;
    PINSEL_ConfigPin(&PinCfg);
    PinCfg.Pinnum = 8;
    PINSEL_ConfigPin(&PinCfg);
    PinCfg.Pinnum = 9;
    PINSEL_ConfigPin(&PinCfg);

    PinCfg.Funcnum = 0;
    PinCfg.Portnum = 0;
    PinCfg.Pinnum = 6;
    PINSEL_ConfigPin(&PinCfg);

    PinCfg.Funcnum = 0;
    PinCfg.Portnum = 2;
    PinCfg.Pinnum = 13;
    PINSEL_ConfigPin(&PinCfg);
```

```
/* P2.13 TP_IRQ is Input */
GPIO_SetDir(TP_IRQ_PORT_NUM, (1<<TP_IRQ_PIN_NUM), 0;
GPIO_SetValue(TP_IRQ_PORT_NUM, (1<<TP_IRQ_PIN_NUM));

/* P0.6 TP_CS is output */
GPIO_SetDir(TP_CS_PORT_NUM, (1<<TP_CS_PIN_NUM), 1;
GPIO_SetValue(TP_CS_PORT_NUM, (1<<TP_CS_PIN_NUM));

/* initialize SSP configuration structure to default */
SSP_ConfigStructInit(&SSP_ConfigStruct);
SSP_ConfigStruct.ClockRate = 500000;
/* Initialize SSP peripheral with parameter given in structure above */
SSP_Init(LPC_SSP1, &SSP_ConfigStruct);

/* Enable SSP peripheral */
SSP_Cmd(LPC_SSP1, ENABLE);
}

/*****************/
* Function Name : DelayUS
* Description   : Delay used for SPI A/D converter
* Input         : - cnt: microseconds
* Output        : None
* Return        : None
* Attention     : None
/*****************/
static void DelayUS(uint32_t cnt)
{
    uint32_t i;
    alimentaWDT(); //      INCLUIDO POR ALBERTO PALOMO.
    for(i = 0;i<cnt;i++)
    {
        uint8_t us = 12; /* Value for one microsecond delay*/
        while (us--)
        {
            ;
        }
    }
}
```

```
}

/*****  
* Function Name : WR_CMD  
* Description    : ADS7843 Command Write  
* Input          : - cmd: command  
* Output         : None  
* Return         : None  
* Attention      : None  
*****/  
  
static uint8_t WR_CMD (uint8_t cmd)  
{  
    /* wait for current SSP activity complete */  
    while (SSP_GetStatus(LPC_SSP1, SSP_STAT_BUSY) == SET);  
  
    SSP_SendData(LPC_SSP1, (uint16_t) cmd);  
  
    while (SSP_GetStatus(LPC_SSP1, SSP_STAT_RXFIFO_NOTEMPTY) == RESET);  
  
    return (SSP_ReceiveData(LPC_SSP1));  
}  
  
/*****  
* Function Name : RD_AD  
* Description    : ADC read (one measure)  
* Input          : None  
* Output         : None  
* Return         : ADS7843 measure  
* Attention      : None  
*****/  
  
static int RD_AD(void)  
{  
    unsigned short buf,temp;  
  
    temp = WR_CMD(0x00);  
    buf = temp << 8;  
    DelayUS(1);  
    temp = WR_CMD(0x00);;  
    buf |= temp;
```

```
    buf >= 3;
    buf &= 0xffff;
    return buf;
}

/*********************************************
* Function Name  : Read_X
* Description    : Configuration of X coordinate and read ADC
* Input          : None
* Output         : None
* Return         : ADS7843 X value read
* Attention      : None
********************************************/

int Read_X(void)
{
    int i;

    TP_CS(0);
    DelayUS(1);
    WR_CMD(CHX);
    DelayUS(1);
    i = RD_AD();
    TP_CS(1);
    return i;
}

/*********************************************
* Function Name  : Read_Y
* Description    : Configuration of Y coordinate and read ADC
* Input          : None
* Output         : None
* Return         : ADS7843 Y value read
* Attention      : None
********************************************/

int Read_Y(void)
{
    int i;

    TP_CS(0);
    DelayUS(1);
    WR_CMD(CHY);
```

```
    DelayUS(1);

    i = RD_AD();

    TP_CS(1);

    return i;

}

/***** Function Name : TP_GetAdXY ****/

* Description      : Read X and Y
* Input            : None
* Output           : None
* Return           : ADS7843 X and Y measure
* Attention        : None
****

void TP_GetAdXY(int *x,int *y)

{
    int adx,ady;

    adx = Read_X();

    DelayUS(1);

    ady = Read_Y();

    *x = adx;

    *y = ady;

}

/***** Function Name : TP_DrawPoint ****/

* Description      : Write a 2x2 pixel red square
* Input            : - Xpos: Row Coordinate
*                   - Ypos: Line Coordinate
* Output           : None
* Return           : None
* Attention        : None
****

void TP_DrawPoint(uint16_t Xpos,uint16_t Ypos)
{
    LCD_SetPoint(Xpos,Ypos,0xf800);

    LCD_SetPoint(Xpos+1,Ypos,0xf800);

    LCD_SetPoint(Xpos,Ypos+1,0xf800);

    LCD_SetPoint(Xpos+1,Ypos+1,0xf800);
```

```
}

/*****  
* Function Name : DrawCross  
* Description   : Draw a white cross  
* Input         : - Xpos: Row Coordinate  
*                 - Ypos: Line Coordinate  
* Output        : None  
* Return        : None  
* Attention     : None  
*****/  
  
void DrawCross(uint16_t Xpos,uint16_t Ypos)  
{  
    LCD_DrawLine(Xpos-15,Ypos,Xpos-2,Ypos,0xffff);  
    LCD_DrawLine(Xpos+2,Ypos,Xpos+15,Ypos,0xffff);  
    LCD_DrawLine(Xpos,Ypos-15,Xpos,Ypos-2,0xffff);  
    LCD_DrawLine(Xpos,Ypos+2,Xpos,Ypos+15,0xffff);  
  
    LCD_DrawLine(Xpos-15,Ypos+15,Xpos-7,Ypos+15,RGB565CONVERT(184,158,131));  
    LCD_DrawLine(Xpos-15,Ypos+7,Xpos-15,Ypos+15,RGB565CONVERT(184,158,131));  
  
    LCD_DrawLine(Xpos-15,Ypos-15,Xpos-7,Ypos-15,RGB565CONVERT(184,158,131));  
    LCD_DrawLine(Xpos-15,Ypos-7,Xpos-15,Ypos-15,RGB565CONVERT(184,158,131));  
  
    LCD_DrawLine(Xpos+7,Ypos+15,Xpos+15,Ypos+15,RGB565CONVERT(184,158,131));  
    LCD_DrawLine(Xpos+15,Ypos+7,Xpos+15,Ypos+15,RGB565CONVERT(184,158,131));  
  
    LCD_DrawLine(Xpos+7,Ypos-15,Xpos+15,Ypos-15,RGB565CONVERT(184,158,131));  
    LCD_DrawLine(Xpos+15,Ypos-15,Xpos+15,Ypos-7,RGB565CONVERT(184,158,131));  
}  
  
/*****  
* Function Name : Read_Ads7846  
* Description   : Make nine TouchPanel measures, evaluate if there are too  
*                 much noise and return the more mean of the more similar measures  
* Input         : None  
* Output        : None  
* Return        : 0 - if no touch detected or a two noise measure  
*                 * Coordinate - if touch detected with X,Y measures.  
*****
```

```
* Attention           : None
*****
Coordinate *Read_Ads7846(void)
{
    static Coordinate screen;
    int m0,m1,m2,TP_X[1],TP_Y[1],temp[3];
    uint8_t count=0;
    int buffer[2][9]={{0},{0}};
    do                                /* Reapeat for 9 measures if touch*/
    {
        TP_GetAdXY(TP_X,TP_Y);
        buffer[0][count]=TP_X[0];
        buffer[1][count]=TP_Y[0];
        count++;
    }
    while(!TP_INT_IN&& count<9); /* TP_INT_IN is low level active if touch */

    if(count==9) /* if 9 measures available */
    {
        /* Obtain three means of three X mesures each */
        temp[0]=(buffer[0][0] + buffer[0][1] + buffer[0][2])/3;
        temp[1]=(buffer[0][3] + buffer[0][4] + buffer[0][5])/3;
        temp[2]=(buffer[0][6] + buffer[0][7] + buffer[0][8])/3;

        /* Calculate the distance of X centroids */
        m0=temp[0]-temp[1];
        m1=temp[1]-temp[2];
        m2=temp[2]-temp[0];

        /* Obtain de absolute value */
        m0=m0>0?m0:(-m0);
        m1=m1>0?m1:(-m1);
        m2=m2>0?m2:(-m2);

        /* return 0 if three X distances are bigger than THESHOLD. This means no touch */
        if( m0>THRESHOLD && m1>THRESHOLD && m2>THRESHOLD ) return 0;

        /* Choose the two more similar centroids and obteain the mean. Choose the best X
measures */
        if(m0<m1)
```

```
{  
    if (m2<m0)  
        screen.x=(temp[0]+temp[2])/2;  
    else  
        screen.x=(temp[0]+temp[1])/2;  
}  
  
else if (m2<m1)  
    screen.x=(temp[0]+temp[2])/2;  
else  
    screen.x=(temp[1]+temp[2])/2;  
  
/* Obtain three means of three Y measures each */  
temp[0]=(buffer[1][0] + buffer[1][1] + buffer[1][2])/3;  
temp[1]=(buffer[1][3] + buffer[1][4] + buffer[1][5])/3;  
temp[2]=(buffer[1][6] + buffer[1][7] + buffer[1][8])/3;  
  
/* Calculate the distance of Y centroids */  
m0=temp[0]-temp[1];  
m1=temp[1]-temp[2];  
m2=temp[2]-temp[0];  
  
/* Obtain de absolute value */  
m0=m0>0?m0:(-m0);  
m1=m1>0?m1:(-m1);  
m2=m2>0?m2:(-m2);  
  
/* return 0 if three Y distances are bigger than THRESHOLD. This means no touch */  
if (m0>THRESHOLD && m1>THRESHOLD && m2>THRESHOLD) return 0;  
  
/* Choose the two more similar centroids and obtain the mean. Choose the best Y  
measures */  
if (m0<m1)  
{  
    if (m2<m0)  
        screen.y=(temp[0]+temp[2])/2;  
    else  
        screen.y=(temp[0]+temp[1])/2;  
}  
else if (m2<m1)  
    screen.y=(temp[0]+temp[2])/2;
```

```
    else
        screen.y=(temp[1]+temp[2])/2;

    /* Fill the return struct if a touch detected and return 0 if no tocuch */
    return &screen;
}

return 0;
}

*****  

* Function Name : setCalibrationMatrix
* Description   : Calculate the calibration metrix
* Input         : None
* Output        : None
* Return        : Calibration matrix
*
* Attention      : None
*****  

uint8_t setCalibrationMatrix( Coordinate * displayPtr,
                             Coordinate * screenPtr,
                             Matrix * matrixPtr)
{

    uint8_t retTHRESHOLD = 0 ;

    /* Divider = (X0-X2)*(Y1-Y2) - (X1-X2)*(Y0-Y2) */
    matrixPtr->Divider = ((screenPtr[0].x - screenPtr[2].x) * (screenPtr[1].y -
screenPtr[2].y)) -
((screenPtr[1].x - screenPtr[2].x) * (screenPtr[0].y -
screenPtr[2].y)) ;

    if( matrixPtr->Divider == 0 )
    {
        retTHRESHOLD = 1;
    }
    else
    {
        /* An = ((XD0-XD2)*(Y1-Y2) - (XD1-XD2)*(Y0-Y2)) */
        matrixPtr->An = ((displayPtr[0].x - displayPtr[2].x) * (screenPtr[1].y -
screenPtr[2].y)) -
```

```

        ((displayPtr[1].x - displayPtr[2].x) * (screenPtr[0].y -
screenPtr[2].y)) ;

/* Bn = ((X0-X2)*(XD1-XD2) - (XD0-XD2)*(X1-X2)) */

matrixPtr->Bn = ((screenPtr[0].x - screenPtr[2].x) * (displayPtr[1].x -
displayPtr[2].x)) -
((displayPtr[0].x - displayPtr[2].x) * (screenPtr[1].x -
screenPtr[2].x)) ;

/* Cn = (Y0*(X2*XD1 - X1*XD2) + Y1*(X0*XD2 - X2*XD0) + Y2*(X1*XD0 - X0*XD1)) */

matrixPtr->Cn = (screenPtr[2].x * displayPtr[1].x - screenPtr[1].x *
displayPtr[2].x) * screenPtr[0].y +
(screenPtr[0].x * displayPtr[2].x - screenPtr[2].x *
displayPtr[0].x) * screenPtr[1].y +
(screenPtr[1].x * displayPtr[0].x - screenPtr[0].x *
displayPtr[1].x) * screenPtr[2].y ;

/* Dn = ((YD0-YD2)*(Y1-Y2) - (YD1-YD2)*(Y0-Y2)) */

matrixPtr->Dn = ((displayPtr[0].y - displayPtr[2].y) * (screenPtr[1].y -
screenPtr[2].y)) -
((displayPtr[1].y - displayPtr[2].y) * (screenPtr[0].y -
screenPtr[2].y)) ;

/* En = ((X0-X2)*(YD1-YD2)-(YD0-YD2)*(X1-X2)) */

matrixPtr->En = ((screenPtr[0].x - screenPtr[2].x) * (displayPtr[1].y -
displayPtr[2].y)) -
((displayPtr[0].y - displayPtr[2].y) * (screenPtr[1].x -
screenPtr[2].x)) ;

/* Fn = (Y0*(X2*YD1-X1*YD2) + Y1*(X0*YD2-X2*YD0) + Y2*(X1*YD0-X0*YD1)) */

matrixPtr->Fn = (screenPtr[2].x * displayPtr[1].y - screenPtr[1].x *
displayPtr[2].y) * screenPtr[0].y +
(screenPtr[0].x * displayPtr[2].y - screenPtr[2].x *
displayPtr[0].y) * screenPtr[1].y +
(screenPtr[1].x * displayPtr[0].y - screenPtr[0].x *
displayPtr[1].y) * screenPtr[2].y ;
}

return( retTHRESHOLD ) ;
}

*****  

* Function Name : getDisplayPoint  

* Description   : Translate AD measures to XY coordinates with calibration matrix  

* Input         : None  

* Output        : None  

* Return        : 0 - No valid touch

```

```
*           1 - Valid touch
*
*           Coordinate struct with X and Y
*
* Attention      : None
*****
uint8_t getDisplayPoint(Coordinate * displayPtr,
                        Coordinate * screenPtr,
                        Matrix * matrixPtr )
{
    uint8_t retTHRESHOLD = 0 ;

    if( matrixPtr->Divider != 0 )
    {
        /* X = An*x + Bn*y + Cn */
        displayPtr->x = ( (matrixPtr->An * screenPtr->x) +
                           (matrixPtr->Bn * screenPtr->y) +
                           matrixPtr->Cn
                           ) / matrixPtr->Divider ;
        /* Y = Dn*x + En*y + Fn */
        displayPtr->y = ( (matrixPtr->Dn * screenPtr->x) +
                           (matrixPtr->En * screenPtr->y) +
                           matrixPtr->Fn
                           ) / matrixPtr->Divider ;
    }
    else
    {
        retTHRESHOLD = 1;
    }
    return(retTHRESHOLD);
}

*****
* Function Name  : TouchPanel_Calibrate
* Description    : Draw three crosses, receive the calibration touches and
*                   calculate calibration matrix
* Input          : None
* Output         : None
* Return         : None
* Attention      : None
*****
```

Alberto Palomo Alonso.

```
void TouchPanel_Calibrate(void)
{
    uint8_t i;
    Coordinate * Ptr;
    for(i=0;i<3;i++)
    {
        LCD_Clear(Black);
        GUI_Text(10,10,"Touch crosshair to calibrate",0xffff,Black);
        DelayUS( 500 * 1000 );
        DrawCross(DisplaySample[i].x,DisplaySample[i].y);
        do
        {
            Ptr=Read_Ads7846();
        }
        while( Ptr == (void*)0 );
        ScreenSample[i].x= Ptr->x; ScreenSample[i].y= Ptr->y;
    }
    setCalibrationMatrix( &DisplaySample[0],&ScreenSample[0],&matrix ) ;
    LCD_Clear(Black);
}

/*****************/
*****END FILE*****
*****/
```



```
/**-----//-
//          @filename      configura.c
//          @version       0.00
//          @author        Alberto Palomo Alonso
//          @brief         Código que configura y llama a las funciones de
//                      configuración para hacer un Setup del programa.//
//          @
//          //
```

Alberto Palomo Alonso.

```

//  

//  

// @ref      __configuraPWM__          -> PWM.h  

//  

//           modificaPulso          -> PWM.h  

//  

//           __configuraLCD__        -> Statechart.h  

//  

//  

//  

//-----*/  

void __configuraPrograma__( void )  

{  

    __configuraLCD__();  

    LCD_Clear(Black);  

    __pintaCargandoInicio__();  

    __iniciaVariables__();  

    LCD_Clear(Black);  

    __pintaCargandoSeno__();  

    // Añadir generación de variables de alto costo computacional.  

    LCD_Clear(Black);  

    __pintaCargandoConexion__();  

    __configuraWEB__();  

    LCD_Clear(Black);  

    __pintaCargandoIniciando__();  

    __configuraSysTick__();  

    __configuraTimer0__();  

    __configuraLDR__();  

    __configuraUVA30A__();  

    __configuraUFONO__();  

    __configuraRTC__();  

    __configuraPWM__ ( Fpwm , ACTIVOS_2_1 | ACTIVOS_6_1 );  

    modificaPulso ( PWM2, MODO_SERVO , none , 90  

    , MINIMO_SERVO , MAXIMO_SERVO );  

    modificaPulso ( PWM6, MODO_CICLO , 50 , none  

    , none , none );  

    __configuraWDT__();  

    __configuraDAC__();  

    __configuraDMA__();  

    __configuraOW__();  

    __configuraAnemometro__();  

}

```

```
    __configuraUART0__          ();
//    __configuraUART3__          ();
    __configuraI2C__            ();
#ifndef DEBUG
//      TouchPanel_Calibrate();
#endif
LCD_Clear(Black);
__pintaCargandoDone__  ();
LCD_Clear(Black);
ESTADO->CHART = PANTALLA_INICIO;
}
/*-----
-----*/
//
//                                //
//                                @funcion __iniciaVariables__()
//                                //
//                                //
//                                @brief Esta función inicia las variables del sistema para que
tengan un momento inicial.        //
//                                //
//                                //
//-----*/
void __iniciaVariables__()
{
    ESTADO->CHART = PANTALLA_LOADING;
    DATOS->Temperatura      = 0;
    DATOS->Humedad          = 0;
    DATOS->Presion           = 0;
    DATOS->VelViento         = 0;
    DATOS->IndiceUV          = 0;
    DATOS->Lugar.Altura       = 0;
    DATOS->Lugar.Longitud     = 0;
    DATOS->Lugar.Latitud      = 0;

    MODIFICABLES.Max_servo_t = (float)MAXIMO_TEMPERATURA;
    MODIFICABLES.Min_servo_t = (float)MINIMO_TEMPERATURA;
```

Alberto Palomo Alonso.

Alberto Palomo Alonso.

```
float presion;

/**-----//  
//  
//  
//-----**/  
// @function      __configuraI2C__()  
//  
//  
// @brief         Esta función es la que configura el protocolo I2C.  
//  
//  
//-----**/  
void  __configuraI2C__  (      void      )  
{  
    __calibraBMP();  
}  
void __calibraBMP()  
{  
    I2CSendAddr(  BMP_ADD ,      WRITE  );  
    I2CSendByte(  AC1      );  
    I2CSendAddr(  BMP_ADD ,      READ   );  
    COEF.ac1     =      I2CGetByte(  SACK   )      << 8;  
    COEF.ac1     |=      I2CGetByte(  NACK   );  
    I2CSendStop();  
  
    I2CSendAddr(  BMP_ADD ,      WRITE  );  
    I2CSendByte(  AC2      );  
    I2CSendAddr(  BMP_ADD ,      READ   );  
    COEF.ac2     =      I2CGetByte(  SACK   )      << 8;  
    COEF.ac2     |=      I2CGetByte(  NACK   );  
    I2CSendStop();  
  
    I2CSendAddr(  BMP_ADD ,      WRITE  );  
    I2CSendByte(  AC3      );  
    I2CSendAddr(  BMP_ADD ,      READ   );  
    COEF.ac3     =      I2CGetByte(  SACK   )      << 8;
```

```
COEF.ac3 |= I2CGetByte( NACK );
I2CSendStop();

I2CSendAddr( BMP_ADD, WRITE );
I2CSendByte( AC4 );
I2CSendAddr( BMP_ADD, READ );
COEF.ac4 = I2CGetByte( SACK ) << 8;
COEF.ac4 |= I2CGetByte( NACK );
I2CSendStop();

I2CSendAddr( BMP_ADD, WRITE );
I2CSendByte( AC5 );
I2CSendAddr( BMP_ADD, READ );
COEF.ac5 = I2CGetByte( SACK ) << 8;
COEF.ac5 |= I2CGetByte( NACK );
I2CSendStop();

I2CSendAddr( BMP_ADD, WRITE );
I2CSendByte( AC6 );
I2CSendAddr( BMP_ADD, READ );
COEF.ac6 = I2CGetByte( SACK ) << 8;
COEF.ac6 |= I2CGetByte( NACK );
I2CSendStop();

I2CSendAddr( BMP_ADD, WRITE );
I2CSendByte( B1 );
I2CSendAddr( BMP_ADD, READ );
COEF.b1 = I2CGetByte( SACK ) << 8;
COEF.b2 |= I2CGetByte( NACK );
I2CSendStop();

I2CSendAddr( BMP_ADD, WRITE );
I2CSendByte( B2 );
I2CSendAddr( BMP_ADD, READ );
COEF.b2 = I2CGetByte( SACK ) << 8;
COEF.b2 |= I2CGetByte( NACK );
I2CSendStop();

I2CSendAddr( BMP_ADD, WRITE );
```

Alberto Palomo Alonso.

```

}

/**-----//-----*/
//          //          //

//          @function      pedirDatoReg()
//          //
//          //
//          //          //
//          @brief           Pide un dato al sensor I2C de 16 bits especificando el
registro.          //
//          //
//          //          //
//          //          //
//          //          //          *
//          void    pedirDatoReg   (      uint8_t REG      )
{
    irRegistro(      REG      );
    pedirDato      ();
    procesarDato(  PRESION_BMP      );
}
/**-----//-----*/
//          //          //
//          //          //
//          @function      pedirDato()
//          //
//          //
//          //          //
//          @brief           Pide un dato al sensor I2C de 16 bits.
//          //
//          //
//          //          //
//          //          //          *
//          void    pedirDato   (      void      )
{
    I2CSendAddr(  BMP_ADD ,      READ      );          //
    la dirección en modo lectura.          //
//          //          //          //          //          Mandar

```

Alberto Palomo Alonso.

```
//  
//  
// @ref           Extraido de  
// https://github.com/BoschSensortec/BMP180_driver; referido por la datasheet.  
//  
//  
//  
//-----**/  
uint16_t obtenerDatos (      uint8_t REG      )  
{  
    uint16_t      RETVAL;  
  
    I2CSendAddr(  BMP_ADD ,      WRITE   );  
    I2CSendByte(  REG      );  
    I2CSendAddr(  BMP_ADD ,      READ    );  
    RETVAL =      I2CGetByte(  SACK    )      << 8;  
    RETVAL |=      I2CGetByte(  NACK    );  
  
    I2CSendStop();  
  
    return RETVAL;  
}  
  
void mandaDatos (      uint8_t REG      ,      uint8_t DATA)  
{  
    I2CSendAddr(  BMP_ADD ,      WRITE   );  
    I2CSendByte(  REG      );  
    I2CSendByte(  DATA    );  
    I2CSendStop();  
}  
  
void medirBMP()  
{  
    int i;  
    long UT, UP, X1, X2, X3, B3, B5, B6, T, p;  
    unsigned long B4, B7;  
    mandaDatos (      0xF4      ,      0x2E    );  
    //Espera 4.7ms.  
    for ( i = 0; i < 1000; i++)  
    {  
        I2Cdelay();  
    }  
    //Espera activa corta!  
    UT = obtenerDatos (      0xF6);
```

```
mandaDatos(0xF4, 0x34);  
// Esperar 4.7ms.  
  
for (i = 0; i < 1000; i++)  
{  
    I2Cdelay();  
}  
  
// Espera activa corta!  
  
UP = obtenerDatos(0xF6);  
  
X1 = (UT - COEF.ac6) * COEF.ac5 / 32768;  
X2 = COEF.mc * 2048 / (X1 + COEF.md);  
B5 = X1 + X2;  
T = ((B5 + 8) >> 4);  
  
B6 = B5 - 4000;  
X1 = (COEF.b2 * ((B6 * B6) >> 12)) >> 11;  
X2 = (COEF.ac2 * B6) >> 11;  
X3 = X1 + X2;  
B3 = ((COEF.ac1 * 4 + X3) + 2) / 4;  
X1 = (COEF.ac3 * B6) >> 13;  
X2 = (COEF.b1 * ((B6 * B6) >> 12)) >> 16;  
X3 = (X1 + X2 + 2) >> 2;  
B4 = COEF.ac4 * (unsigned long)(X3 + 32768) >> 15;  
B7 = ((unsigned long)UP - B3) * (50000);  
  
if (B7 < 0x80000000)  
{  
    p = (B7 * 2) / B4;  
}  
else  
{  
    p = (B7 / B4) * 2;  
}  
  
X1 = (p >> 8) * (p >> 8);  
X1 = (X1 * 3038 >> 16);  
X2 = (-7357 * p) >> 16;  
p = p + ((X1 + X2 + 3791) >> 4);  
// temperatura = (float)(28.0/107.0)*((float)T)/10;  
// presion = (float)(936.0/1150.0)*(float)p;
```

```
temperatura    = ((float)T)/10;
presion        = (float)p;
procesarDatos(0);
}

/*
-----*/
//  

//  

//  

//          @end      ENDFILE.  

//  

//  

//  

//-----*/  

/*****  

/*Funciones de control del bus I2C*/  

/* SDA=P0.0 y SCL=P0.1 */  

/*****  

#ifndef I2C  

#define I2C  

#include "I2C.h"  

#endif  

void I2Cdelay(void)                                //  

    Retardo minimo de 4.7 us  

{  

    unsigned char i;  

    for( i=0 ; i < DELAYTOTAL ; i++ );      //  

    Modificar limite para garantizar los tiempos (Bus standar -->F_max=100kHz)  

}  

//Genera un pulso de reloj (1 ciclo)
void pulso_SCL(void)
{
    LPC_GPIO0->PIOSET=(1<<SCL); // Genera pulso de reloj (nivel alto)
    I2Cdelay();
    LPC_GPIO0->PIOCLR=(1<<SCL); // Nivel bajo
    I2Cdelay();
}
```

```
void I2CSendByte(unsigned char byte)
{
    unsigned char i;
    for(i=0;i<8;i++){

        if (byte &0x80)  LPC_GPIO0->FIOSET=(1<<SDA); // envia cada bit,
comenzando por el MSB

        else  LPC_GPIO0->FIOCLR=(1<<SDA);

        byte = byte <<1;                      // Siguiente bit

        pulso_SCL();

    }

    // Leer ACK que envia el Slave (el Master ha de enviar un pulso de reloj)
    // CONFIGURAR PIN SDA COMO ENTRADA;
    // espera ACK(config. pin como entrada)

    LPC_GPIO0->FIODIR&=~(1<<SDA);

    pulso_SCL();

    // CONFIGURA PIN SDA COMO SALIDA;
    LPC_GPIO0->FIODIR|=(1<<SDA);           // Dejamos SDA de nuevo como salida
}

//Función que envía START + Byte de dirección del Slave (con bit LSB iniciando R/W)
void I2CSendAddr(unsigned char addr, unsigned char rw)
{
    //CONFIGURAR PINS SDA, SCL COMO SALIDAS;
    // Por si se nos olvidada en la conf. general.

    LPC_GPIO0->FIODIR|=(1<<SDA) | (1<<SCL);

    LPC_GPIO0->FIOSET|=(1<<SDA) | (1<<SCL);           // SDA y SCL a nivel alto
para garantizar el

    // reposo del bus + tiempo.

    I2Cdelay();

    LPC_GPIO0->FIOCLR|=(1<<SDA);
//condicion de START: Bajar SDA y luego SCL

    I2Cdelay();

    LPC_GPIO0->FIOCLR|=(1<<SCL);

    I2Cdelay();
```

```
I2CSendByte((addr=addr<<1) + rw); //envia byte de direccion
                                         //addr, direccion (7bits)
                                         //rw=1, lectura
                                         //rw=0, escritura

}

// Función para leer un Byte del Slave. El Master envía al final de la lectura
// el bit ACK o NACK (si es último byte leído) que se pasa como argumento de la función.
unsigned char I2CGetByte(unsigned char ACK)
{
    // ACK = 0, para cualquier byte que no sea el ultimo.
    // ACK = 1 (NACK), despues de leer el ultimo byte
    unsigned char i, byte;
    //CONFIGURAR PIN SDA COMO ENTRADA; //configura pin SDA como entrada
    LPC_GPIO0->FIODIR&=~(1<<SDA);
    for(i=0;i<8;i++){                                //lee un bit comenzando por
el MSB

        LPC_GPIO0->FIOSET=(1<<SCL); //mientras SCL=1
        I2Cdelay();
        byte=byte<<1;
        if(LPC_GPIO0->FIOPIN&(1<<SDA)) byte++;           //Si leemos
"1" sumamos para introducir el "1"
        LPC_GPIO0->FIOCLR=(1<<SCL);                      //Si leemos
"0" solo desplazamos (se introduce un "0")
        I2Cdelay();
    }

    //CONFIGURAR PIN SDA COMO SALIDA;
    // Master envía un ACK por cada byte leido.
    LPC_GPIO0->FIODIR|=(1<<SDA);

    if(ACK) LPC_GPIO0->FIOSET=(1<<SDA);                // ACK o (NACK) es función del
último byte leído
    else LPC_GPIO0->FIOCLR=(1<<SDA);

    pulso_SCL();                                         // Pulso de reloj para su
envío

    return (byte);
}
```

}

```
void I2CSendStop(void)
{
    LPC_GPIO0->FIOCLR=(1<<SDA);
    I2Cdelay();
    LPC_GPIO0->FIOSET=(1<<SCL); // Subir SCL, y después SDA!! para dejar
el bus en reposo
    I2Cdelay();
    LPC_GPIO0->FIOSET=(1<<SDA);
    I2Cdelay();
}
```

```
/**-
 * @filename Anemometro.c
 */
// @version 2.00
//
// @author Alberto Palomo Alonso
//
// @brief Este es el programa donde se encuentran las funciones
correspondientes al
// anemómetro de la estación.
//
// @category Medida.
//
// @map @include
//
// @variables
//
```

Alberto Palomo Alonso.

```
//  
//  
  
//  
// @function __configuraAnemometro()  
//  
//  
//  
//-----**/  
void __configuraAnemometro()  
{  
    LPC_PINCON->PINMODE3     &=  ~(PULL_UP << (2*2));           // PULL_UP  
    LPC_PINCON->PINMODE3     |=   PULL_UP << (2*2);           // PULL_UP  
    LPC_PINCON->PINSEL3      &=  ~((CAPTURE_FUNCION) << (2*2)); //  
    CAPTURE1.0  
    LPC_PINCON->PINSEL3      |=   (CAPTURE_FUNCION) << (2*2); //  
    CAPTURE1.0  
    LPC_TIM1->CTCR          |=   CTCR_MASCARA;                //  
    Flanco de subida.  
    LPC_TIM1->CCR            |=   CCR_MASCARA_EN;              //  
    Inicio con interrupción.  
    LPC_SC->PCONP           |=   TIMER1_BIT;                  //  
    Activo el módulo del timer 1.  
    LPC_TIM1->PR             =   0;  
    // Sin prescaler.  
    LPC_TIM1->TCR            |=   RESET_TIMER_TCR;            //  
    Reseteo el contador.  
    LPC_TIM1->TCR            &=  ~RESET_TIMER_TCR;            // Reseteo  
    el contador.  
    LPC_TIM1->TCR            |=   0x1;  
    // Que cuente.  
    LPC_PINCON->PINSEL3      |=   0x3     <<     4;  
    // Entrada como CAP1.0.  
    NVIC_EnableIRQ(TIMER1_IRQn);  
}  
/*-----**/  
//  
//  
//  
// @HANDLER mideAnemometro()  
//  
//  
//
```

```

//           @brief      Es la medidas del anemómetro, se suicida al acabar y es
reanimada por el reanimador cada 5s. //

//



//-----**/


void mideAnemometro()
{
    /**
     * @WARNING: Esto me parece un poco sucio,
     * pero es la única manera de que no se generen
     * interrupciones espúrias
     * utilizando únicamente recursos software. Idealmente
     * no debería usar delays y
     * varias interrupciones por
     * flanco debido al ruido. Esto se arregla con un condensador
     * a masa en la entrada, pero es
     * perder recursos hardware y he decidido perderlos
     * mediante software.*/
}

int i;

for(i = 0; i < 30000; i++) {}

/** @TODO Poner condensadores entre Vin y masa
del capture 1.0 para evitar doble int en flancos ascendentes.*/



LPC_TIM1->IR = 1 << 4; // Desactivo la
interrupción.

CAPcont++; // Incremento el contador de pulsos.

if ( CAPcont >= PULSOS_VUELTA ) /* @WARNING: Se
generan __UN__ interrupciones por pulso y 2*PULSOS_VUELTA por vuelta.*/
{
    CLKbuff[1] = CLKbuff[0]; // Almaceno el valor anterior.

    CLKbuff[0] = LPC_TIM1->CR0; // Cargo el valor
actual.

    CAPcont = 0; // Reseteo
el contador de pulsos.

    aux_viento =
    Ftick*(float)PI*(float)DIAMETRO_ANEMOMETRO/((float)100*(float)((uint32_t)CLKbuff[0] - (uint32_t)CLKbuff[1])); // Metros / segundo.

    SLAYERcont++; // Hay
warmup, aumento el slayer.

    if ( SLAYERcont == WARMUP_CICLOS )
    {
        LPC_TIM1->CCR |= ~CCR_MASCARA_DIS; // Slay capture.
OJO: QUE HAY QUE REVIVIRLO.
    }
}

```

Alberto Palomo Alonso.

```
//          @category      Periférico.  
//  
//  
//          @map           @include  
//  
//          @VARIABLES  
//  
//          @funcion  
//  
//          @end  
//  
//  
//  
//  
//  
//-----//  
//  
//-----//  
//  
//-----//  
//  
//-----//  
//          @include      Estos son los archivos utilizados en el código de  
LDR.          //  
//  
//  
//-----//  
//-----**/  
#ifndef LDR  
#define LDR  
#include      "LDR.h"  
#endif  
/*-----//  
-----//  
//  
//  
//-----//  
//          @variables    Variables del fichero.  
//  
//  
//
```

Alberto Palomo Alonso.

```

//-----*/
```

extern misDatos_t * DATOS;

float BUFFER_BRILLO = 0;

float BUFFER_UVA = 0;

extern actualizador_t * ACTUALIZADOR;

extern uint8_t YaPuedesMedir;

uint32_t contador;

uint8_t AUDIO[MUESTRAS_AUDIO];

```
/*-----*/
```

//

//

```
//-----*/
```

//

// @funcion __configuraLDR()

```
//-----*/
```

//

//

// @brief Esta función configura el ADC y el LDR

```
//-----*/
```

//

//

//

//-----*/

void __configuraLDR()

{

LPC_SC->PCONP |= PCONP_ADC_ON; // Enciendo el
ADClock.

LPC_PINCON->PINSEL1 |= PINSEL_ADC01; // AD0.1

LPC_PINCON->PINMODE1 &= ~PINMODE_ADC01; // AD0.1

LPC_ADC->ADCR |= BRUST_PIN // Modo
ráfaga.

 | SEL_CANAL1 // AD0.1
activado.

 | ADC_POWER //
Empiezo ENCENDIENDO el ADC.

 | CLK_DIV_MAX; // Clkdiv
hace que Fadc = Fclk/256, inferior al umbral de 13MHz. (Ojo: clkdiv = 0 implica que no
funcione en placa).

LPC_ADC->ADINTEN |= SEL_CANAL1; // Genera
interrupción el canal 1. (Debería ser el penúltimo)

ACTUALIZADOR->LDRrev = 1; // Inicia para
activar.

LPC_ADC->ADINTEN &= ~(SEL_CANAL_GLOBAL); // Apago la interrupción
global.

Alberto Palomo Alonso.

Alberto Palomo Alonso.

```

//ACTUALIZADOR->LDR = 1;

// Señal al LCD para que muestre por pantalla.

break;

case 0:

    AUDIO[contador] = (uint8_t)((0xFF) & LPC_ADC->ADDR0 >>
(4+4)); // El ADC es de 12 bits y las muestras de 8 bits, por lo que hay que
reducir los 4 LSB.

    if (contador++ >= MUESTRAS_AUDIO - 1)

    {

        contador = 0;

        // Reseteo el contador.

        LPC_TIM1->MCR = 0; // No interrumpe el MRO.

        ACTUALIZADOR->Audiorev = 1; // Señalizo el fin del audio.

        recuperarContexto(); // Recupero el contexto del ADC.

    }

    break;

}

}

/*-----*/
// @end ENDFILE.

// -----**/


/*-----*/
// @filename uFono.c //
// @version 1.00 //
// @author Alberto Palomo Alonso //

```

Alberto Palomo Alonso.

```
//  
  
//          @variables      Variables del fichero.  
//  
//  
//  
//-----**/  
extern uint8_t Timer2_MODO;  
uint8_t YaPuedesMedir = 1;  
uint32_t      ADC_ConfigBuffer;  
uint32_t    ADC_IntenBuffer;  
/*-----**/  
//  
//  
//  
//          @funcion      __configuraUFONO__()  
//  
//  
//  
//          @ref           Configura todo lo necesario para la lectura de audio.  
//  
//  
//  
//          @WARNING       Utiliza variables de bloqueo.  
//  
//  
//  
//-----**/  
void __configuraUFONO__()  
{  
//      LPC_PINCON->PINSEL1 |= ~ (0x3 << (2*PIN_UFONO));  
LPC_PINCON->PINSEL1 |= (FUNC_ADC << (2*PIN_UFONO));  
  
//      NEW:  
LPC_TIM1->MR0 = Fclk*TsAudio - 1; // Cada MR0 se genera una  
interrupci n de leer el audio.  
LPC_TIM1->TCR = 0x2; // Reset al contador.  
LPC_TIM1->TCR = 0x1; // Activo contador.
```

Alberto Palomo Alonso.

Alberto Palomo Alonso.

Alberto Palomo Alonso.

Alberto Palomo Alonso.

```

__configuraLDR__();

}

LPC_PINCON->PINSEL1 |= PINSEL_ADC02; // AD0.2

LPC_PINCON->PINMODE1 &= ~PINMODE_ADC02; // AD0.2
LPC_ADC->ADCR |= SEL_CANAL2; // AD0.2

}

/*-----*/
// @end ENDFILE.

// ----- */
// @filename OneWire.c
// @version 4.01
// @author Alberto Palomo Alonso
// @brief Código que configura el protocolo monohilo del sensor de
// temperatura y humedad.
// @category Opcional.
// @
// 
```

Alberto Palomo Alonso.

```
uint8_t         compruebaRespuesta    (      void      );
uint8_t leerByte           (      void      );
//      Externo:
extern misDatos_t          *      DATOS;
uint32_t        TRAZA     [100];
uint32_t    HOLD      [100];
int p;
uint8_t Checksum;
/**-----*/
//
```

```
//
```

```
//      @function      __configuraOW__()
//
```

```
//
```

```
//
```

```
//      @brief      Configura los pines y los recursos utilizados para el
// protocolo OneWire.
//
```

```
//
```

```
//
```

```
//-----**/
void __configuraOW__()
{
    inicializaT3();
    reiniciaCuenta();
}

/**-----*/
//
```

```
//
```

```
//
```

```
//      @function      mideTemperatura()
//
```

```
//
```

```
//
```

```
//      @brief      Configura los pines y los recursos utilizados para el
// protocolo OneWire.
//
```

```
//
```

```
//
```

```
//-----
-----**/
void mideTemperatura ( void )
{
    int i;
    uint8_t Check[4] = {0,0,0,0};
    uint32_t Rx = 0;
    uint8_t Checksum_Recibido = 0;
    p = 0;
    Checksum = 0;
    /** @state: Estado en el que mandamos la señal de petición. */
    CONFIG_OUT;
    CLEAR_PIN;
    _delayUS(18000);
    CONFIG_IN;
    /** @state: Esperamos la respuesta.
    */
    if ( compruebaRespuesta() )
    {
        //      ERROR A AL ESPERAR LA RESPUESTA DEL SENSOR...
        return;
    }
    /** @state: Leemos los 5 bytes... */
    for(i = 0; i < 4; i++)
    {
        Rx |= (leerByte() << (3-i)*8);
    }
    Checksum_Recibido = leerByte();
    /** @state: Procesamos Rx. */
    Check[0] = ((Rx & (0xFF000000)) >> 6*4);
    Check[1] = ((Rx & (0x00FF0000)) >> 4*4);
    Check[2] = ((Rx & (0x0000FF00)) >> 2*4);
    Check[3] = ((Rx & (0x000000FF)) >> 0*4);
    Checksum = Check[0] + Check[1] + Check[2] + Check[3] ;

    if( Checksum == Checksum_Recibido )
    {
        DATOS->Humedad = (float) (((Rx >> 16) & 0xFFFF)/10.0);
        DATOS->Temperatura = (float) (((Rx >> 00) & 0xFFFF)/10.0);
    }
}
```

```
}

/**-----//



//



//



//



@function      reiniciaCuenta()          //



//



@brief         Reinicia el contador del timer 3.          //



//



//



@ret           Retorna el valor de la cuenta antes de reiniciarla.          //



//



//



//-----**/



uint32_t reiniciaCuenta()



{
    uint32_t retval = US_AHORA;

    LPC_TIM3->TCR = 2;      //      Reinicia timer.

    LPC_TIM3->TCR = 1;      //      Activa cuentas.

    return retval;
}

/**-----//



//



//



//



@function      inicializaT3()          //



//



@brief         Configura el timer 3 para utilizarlo para el OneWire.          //



//



//
```

Alberto Palomo Alonso.

```
//          @ret           Retorna 0 si todo ha ido bien, 1 si no.
//          //  
//  
//-----  
-----**/  
uint8_t compruebaRespuesta()  
{  
    uint32_t      Tiempo = 0;  
    /**      @state: Esperamos que el sensor responda con un pull down... */  
    reiniciaCuenta();  
    while (      ENTRADA &&      US_AHORA < 45)           //      Si la entrada  
está a nivel alto y no han pasado 45 us...  
    {  
        //      Mantente en espera.  
    }  
    Tiempo = reiniciaCuenta();           //      Me  
quedo con cuentos us han pasado.  
    TRAZA[p++] = Tiempo; //!!!!  
    if (      Tiempo < 5 || Tiempo > 45)           //      Si el  
márgen de pull down del sensor no es el adecuado.  
    {  
        return 1;  
        //      Exit error code.  
    }  
    /**      @state: Esperamos la respuesta del sensor... */  
    reiniciaCuenta();  
    while (      ENTRADA == 0    &&      US_AHORA < 100)           //      Tiempo nivel  
bajo...  
    {  
  
    }  
    Tiempo = reiniciaCuenta();  
    TRAZA[p++] = Tiempo; //!!!!  
    if (      Tiempo < 70 || Tiempo > 90)           //      Si el  
tiempo de pull down no es adecuado...  
    {  
        return 1;  
        //      Exit error code.  
    }  
    reiniciaCuenta();  
    while (      ENTRADA          &&      US_AHORA < 100)           //      Tiempo  
nivel alto...  
}
```

```
{  
  
}  
Tiempo = reiniciaCuenta();  
TRAZA[p++] = Tiempo; //!!!!!  
if ( Tiempo < 70 || Tiempo > 90) // Si el  
tiempo de pull down no es adecuado...  
{  
    return 1;  
    // Exit error code.  
}  
return 0;  
// Respuesta correcta.  
}  
/*-----*/  
//  
//  
//  
//  
@function leerByte()  
//  
//  
//  
@brief Lee 8 bits en ráfaga del sensor.  
//  
//  
//  
//-----**/  
uint8_t leerByte( void )  
{  
    int i;  
    uint8_t Rx=0;  
    uint32_t Tiempo = 0;  
    for (i = 0; i < 8; i++)  
    {  
        reiniciaCuenta();  
        while ( ENTRADA == 0 && US_AHORA < 100) //  
Mientras la entrada valga 0...  
        {  
            // Mantenernos esperando.  
    }
```

```

        }

        Tiempo = reiniciaCuenta();

        TRAZA[p++] = Tiempo; //!!!!!

        if (Tiempo < 40 || Tiempo > 67) // Si el tiempo está fuera del margen.

        {
            return 0;
        // Error al leer el bit, retorna 0.

    }

    Tiempo = 0;

    reiniciaCuenta();

    while (      ENTRADA && US_AHORA < 100) // Mientras la entrada valga 1...

    {
        // Mantenernos esperando.

    }

    Tiempo = reiniciaCuenta();

    TRAZA[p++] = Tiempo; //!!!!!

    if (      Tiempo > 60 && Tiempo < 80) // Si entra en el margen del 1...

    {
        Rx |= 1 << (7-i);
        // Añadimos un 1.

    }

    else

    {
        if ( Tiempo < 10 || Tiempo > 100) // Si se ha salido del margen...

        {
            return 0;
        // Error al leer el bit, retorna 0.

    }

}

return Rx;
}
/*-----*/
//
```

```
//          @end          ENDFILE.  
//  
//  
//  
//-----**/  
  
/*-----//  
-----*/  
//          @filename      PWM.c  
//  
//          @version       0.00  
//  
//          @author        Alberto Palomo Alonso  
//  
//  
//  
//          @brief         Este es el programa donde están definidas las funciones a  
utilizar en el módulo  
//  
//          PWM dedicado al proyecto de Sistemas electrónicos  
digitales avanzados (UAH - EPS).  
//  
//  
//  
//  
//          @category     Opcional.  
//  
//  
//  
//          @map          @include  
//  
//          @variables  
//  
//          @function  
//  
//          @end  
//  
//  
//  
//          |---|  |           |   | \  /|  
//
```

Alberto Palomo Alonso.

```
/**-----  
-----*/  
//  
//  
//  
// @function      __configuraPWM__()  
//  
//  
//  
// @brief         Configura el PWM en función de la frecuencia a utilizar y  
que PWM se quieren usar.  
//  
//           El primer byte activa las del puerto 2 y las del  
segundo las del otro puerto en orden  
//  
//           ascendente.  
//  
//  
// @FrecuenciaPWM      Frecuencia a la que se desea ajustar el  
ciclo PWM. En Hz.  
//  
// @CualesPWM      Máscara que define los PWM a configurar, los  
6 primeros bits corresponden al  
//  
//           corresponden al puerto 1. En orden ascendente  
//           puerto 2. Del 9º bit al 14º bit  
//  
//           desde del PWM1.1 al PWM1.6  
//  
//  
//  
//  
//  
//-----**/  
void __configuraPWM__(float  FrecuenciaPWM ,      uint16_t      CuallesPWM      )  
{  
    LPC_SC->PCONP |= PCONP_MASK;  
    // Enciendo el PWM.  
    LPC_PWM1->MRO = ((uint32_t)(Ftick/FrecuenciaPWM) - 1);  
    // Configuro la frecuencia.  
    LPC_PWM1->TCR |= TCR_MASK;  
    // Activo el PWM.  
    LPC_PWM1->MCR &= ~TODO_1_32;  
    // Reset al MCR.  
    LPC_PWM1->MCR |= 0x2;  
    // Ahora el MRO resetea el contador.  
    LPC_PINCON->PINSEL4 &= ~0xFFF;  
    // Reset en pines PWM puerto 2.
```

```
LPC_PINCON->PINSEL3 &= ~0x33CF30;
// Reset en pines PWM puerto 1.

for (COUNTERS->i = 0; COUNTERS->i < 6; (COUNTERS->i)++)
// Para el puerto 2: seleccionados.

{

    if ( (CualesPWM >> COUNTERS->i) & ~0xFFFF)
    // Miro si está seleccionado el iésimo.

    {

        LPC_PINCON->PINSEL4 |= (FUNC1      << (2*COUNTERS->i)); // Pongo la función 1 en los pines PWM.

        LPC_PWM1->PCR |= (0x1 << (COUNTERS->i + 0x9));
        // Pongo la función de enable output en el PWM.

    }

}

/** @REMARK: Esto se configuraría como open drain sobre todo por no perder potencia, pero prefiero aseugrar con pull.*/

for (COUNTERS->i = 6; COUNTERS->i < 12; (COUNTERS->i)++)
// Para el otro puerto: seleccionados.

{

    if ( (CualesPWM >> (COUNTERS->i + 2)) & ~0xFFFF)
    // Miro si está seleccionado el iésimo.

    {

        LPC_PWM1->PCR |= (0x1 << (COUNTERS->i - 0x6 + 0x9));
        // Pongo la función de enable output en el PWM.

        switch (COUNTERS->i)
        // Pongo la función 2 en los pines PWM.

        {

            case 6:

                LPC_PINCON->PINSEL3 |= FUNC2 << 2*2;
                // LPC_PINCON->PINMODE3 |= OPEN_DRAIN << 2*2;
                break;

            case 7:

                LPC_PINCON->PINSEL3 |= FUNC2 << 2*4;
                // LPC_PINCON->PINMODE3 |= OPEN_DRAIN << 2*4;
                break;

            case 8:

                LPC_PINCON->PINSEL3 |= FUNC2 << 2*5;
                // LPC_PINCON->PINMODE3 |= OPEN_DRAIN << 2*5;
                break;

            case 9:

                LPC_PINCON->PINSEL3 |= FUNC2 << 2*7;
                // LPC_PINCON->PINMODE3 |= OPEN_DRAIN << 2*7;
                break;

        }

    }

}
```

```

        case 10:
            LPC_PINCON->PINSEL3 |= FUNC2 << 2*8;
        // LPC_PINCON->PINMODE3 |= OPEN_DRAIN << 2*8;
            break;

        case 11:
            LPC_PINCON->PINSEL3 |= FUNC2 << 2*10;
        // LPC_PINCON->PINMODE3 |= OPEN_DRAIN << 2*8;
            break;

        default:
            break;
    }

}

COUNTERS->i = 0;
// Dejo el contador a 0.

}

/*-----
-----*/
//



// @function modificaPulso()
//



// @brief Configura el pulso de PWM en función del ciclo de trabajo
o de valores de oscilación
// si se encuentra en modo servo.
//



// @PWMin Selecciona el PWM1.n a modificar.
//



// @Modo Selecciona si modo servo (valores) o modo ciclo (en
porcentaje).
//



// @Ciclo Selecciona el ciclo de trabajo para el modo ciclo.
//



// @Grados Selecciona los grados a rotar el servo en modo
servo.
//



// @Minimo Selecciona el mínimo valor de Ton del ciclo PWM. En
segundos.
//



// @Maximo Selecciona el máximo valor de Ton del ciclo PWM. En
segundos.
//
```

```
//  
//  
//  
//  
//  
//  
//  
//  
//  
//  
//-----**/  
void modificaPulso( uint32_t PWMn , uint8_t Modo , uint8_t Ciclo  
, uint8_t Grados, float Minimo , float Maximo )  
{  
    if (PWMn > 3)  
    {  
        PWMn += 6; // Debido a la asimétrica distribución de los  
registros.  
    }  
  
    Minimo *= KMN; // Definitivamente habia algo mal.  
    Maximo *= KMX; // Estos servos utilizan la potencia del pulso  
y no precisamente su duración.  
  
    /** @REMARK: La potencia entregada no es la debida. En la datasheet  
especifica pulsos del rango de 5V, se ofrece uno  
de 3.3V, se ha podido usar un transistor, pero deduzco que  
estos servos utilizan la potencia de la señal  
    PWM para obtener el ángulo y modificando los tiempos  
podemos entregar más potencia de señal.*/  
  
    switch(Modo )  
    {  
        case MODO_CICLO: // Escribo en LPC_PWM1->MRn el valor  
correspondiente al porcentaje de MR0; < 1.  
            *(_IO uint32_t *)((uint32_t)&(LPC_PWM1->MR0) +  
(uint32_t)(0x4*PWMn)) = (uint32_t)((float)(LPC_PWM1->MR0)*((float)Ciclo/(float)100));  
            break;  
        case MODO_SERVO: // Escribo en LPC_PWM1->MRn el valor  
correspondiente al tiempo Ton en función del grado.  
            *(_IO uint32_t *)((uint32_t)&(LPC_PWM1->MR0) +  
(uint32_t)(0x4*PWMn)) = (uint32_t)((Ftick*(Minimo + (Maximo -  
Minimo)*(float)(Grados/(float)(180)))- (float)1));
```

Alberto Palomo Alonso.

```
//          @map           @include
//
//          @variables
//
//          @funcion
//
//          @end
//
//          //
//          //
//          //
//          //
//          //
//          //
//          //-----//-----//
//          //
//          //
//          //-----//-----//
//          //-----**/
//          @include           Estos son los archivos utilizados en el código de
LDR.                                         //          //
//          //
//          //-----//-----**/
#ifndef DAC
#define DAC
#include      "DAC.h"
#endif
/**-----//-----*/
//          //
//          //
//          //-----//-----//
//          @variables           Variables del fichero.
//          //          //
//          //
//          //-----//-----**/
extern actualizador_t *      ACTUALIZADOR;
/**-----//-----*/
```

```
//  
//  
  
//          @funcion      __configuraDAC__()  
//  
//  
//          @brief       Función de configuración del DAC y su timer (Timer2).  
//  
//  
//          @REMARK:     Para activar un periodo del DAC (2 segundos)  
//  
//  
//  
//-----**/  
  
void __configuraDAC__()  
{  
    LPC_GPIO3->FIODIR |= (1 << LECTURA_AUDIO) | (1 <<  
    ESCRITURA_AUDIO);  
    // Leds de lectura / escritura de audio.  
  
    LPC_GPIO3->FIOCLR = (1 << LECTURA_AUDIO);  
    //Turn ON LED1  
  
    LPC_GPIO3->FIOCLR = (1 << ESCRITURA_AUDIO);  
    //Turn ON LED2  
}  
/*-----*/  
  
//  
//  
//          @funcion      activarDac()  
//  
//  
//          @brief       Señal de activar el timer del DAC.  
//  
//  
//          @REMARK:     Utiliza DMA.  
//
```

```
//  
//  
//-----**/  
  
void activarDac()  
{  
    /** @TODO: DMA */  
  
    LPC_GPDMACh0->DMACConfig |= 1;  
    // Activo el DMA.  
  
    LPC_TIM1->MCR = (1 <<3) | (1  
    << 4); // Activo la interrupción por MR1 y reset por MR1.  
  
    LPC_TIM1->MR1 = (Fclk*DURACION_AUDIO) - 1;  
    // Valor de MR1.  
  
    LPC_TIM1->TCR = 0x2;  
    // Reset del timer.  
  
    LPC_TIM1->TCR = 0x1;  
    // El timer cuenta.  
  
    ACTUALIZADOR->Audiorev = 0;  
    // Señalizo el bloqueo de audio.  
  
    LPC_GPIO3->FIOSET = (1 <<  
    ESCRITURA_AUDIO); // Señalizo escritura de audio.  
}  
//-----//  
  
//  
//  
//-----//  
  
//          @funcion      activarDac() //  
//  
//  
//          @brief        Señal de activar el timer del DAC. //  
//  
//  
//          @REMARK:      Activador del DAC. //  
//  
//  
//  
//-----**/  
  
void desactivarDac()  
{
```

Alberto Palomo Alonso.

```

LPC_GPDMACh0->DMACCConfig      &=      ~0x1;
//      Desactivo el DMA.

ACTUALIZADOR->Audiorev        =        1;
//      Señalizo el fin del DAC.

LPC_GPIO3->FIOCLR             =          (      1      <<      3);
//      Señalizo escritura de audio.

LPC_TIM1->MCR                 &=      ~(7      <<      3);
//      Desactivo la interrupción por MR1 y reset tras MR1.

LPC_DAC->DACR                 =        0;
//      No hay señal de salida.

}

/**-----*/
//



//



//      @end      ENDFILE.

//



//



//



//-----*/
//



//-----*/
//



//      @filename      LUT.c
//



//      @version       0.00
//



//      @author        Alberto Palomo Alonso
//



//



//      @brief         Este es el programa donde se encuentran las look up tables
// que optimizan
//                                //



//                                el uso de cpu sacrificando memoria, pero como uso
// una SD la memoria no es
//                                //



//                                un problema grande.
//



//



//      @category      Opcional.
//



//
```

Alberto Palomo Alonso.

```
//  
//  
// @map @include  
//  
// @variables //  
// @LUT  
//  
// @function  
//  
// @end  
//  
//  
//  
//-----//  
//  
//  
//-----//  
//  
//-----//  
//  
//-----//  
//-----*/  
  
// @include Includes pertenecientes al módulo del anemómetro.  
//  
//  
//-----//  
//-----*/  
  
#ifndef LUT  
#define LUT  
  
#include "LUT.h"  
  
#endif  
/*-----//  
-----*/  
  
//  
//  
//  
//-----//  
//-----*/  
  
// @variables Variables del fichero.  
//  
//  
//-----//  
//-----*/
```

Alberto Palomo Alonso.

```

// @brief Esta función es la que mira las LUT y obtiene el dato que
queremos de la base
// de datos.

//
//



//-----**/


void goto_LUT( float variable , uint8_t LUTn , float * ret_flotante , uint8_t * ret_int8
, uint16_t * ret_int16 , uint32_t * ret_int32)
{

    switch( LUTn )

    {

        case BRILLO_LDR:

            *ret_flotante = Brillo_LDR[ (uint8_t)((variable -
LDRRESISTENCIA_MIN) / (LDRRESISTENCIA_MAX - LDRRESISTENCIA_MIN)) *Brillo_LDR[0]) + 1];

            break;

        case BRILLO2CICLO_LDR:

            *ret_int8      = Brillo2ciclo_LDR[ (uint8_t)((variable -
BRILLO_MIN) / (BRILLO_MAX - BRILLO_MIN))
*Brillo2ciclo_LDR[0]) + 1];

            break;

        case INDICE_UVA:

            *ret_flotante = variable; // El output DC corresponde al
indice, es muy sencillo traducirlo, se recomienda no llamar a esta función en este modo.

            break;

        case BRILLO_LDR_NOLUT:

            *ret_flotante = -(1.0102)*variable + 102.0204;

            if (*ret_flotante < 0)

            {

                *ret_flotante = 0;

            }

        default:

            break;

    }

}

//-----**/


//



//-----**/


//
```

Alberto Palomo Alonso.

```
//          @end      ENDFILE.  
//  
//  
//  
//-----**/  
  
/*-----//  
-----*/  
//          @filename    DMA.c  
//  
//          @version     0.00  
//  
//          @author      Alberto Palomo Alonso  
//  
//  
//  
//          @brief       Código fuente que contiene las funciones para audio (DMA).  
//  
//  
//          @category    Periférico.  
//  
//  
//          @map        @include  
//  
//          @variables  
//  
//          @funcion  
//  
//          @end  
//  
//  
//  
//  
//-----//  
-----//  
//  
//
```

```
//          @include      Estos son los archivos utilizados en el código de
LDR.                                         //  
//  
//  
//-----**/  
#ifndef DMA  
#define DMA  
#include      "DMA.h"  
#endif  
/*-----*/  
//  
//  
//  
//          @variables      Variables del fichero.  
//  
//  
//  
//-----**/  
extern actualizador_t *      ACTUALIZADOR;  
extern uint8_t AUDIO[MUESTRAS_AUDIO];  
DMA_t    LLIO;  
uint8_t sinusoide[N_samples_wave];  
/*-----*/  
//  
//  
//  
//          @funcion      __configuraDMA__()  
//  
//  
//  
//          @brief       Función de configuración del DMA.  
//  
//  
//
```

```

//          @REMARK:           Para activar un periodo del DAC (2 segundos)
//                                //
//                                //
//-----**/
void __configuraDMA__ (void)
{
    int i;
    for(i=0; i < N_samples_wave; i++)
    {
        sinusoide[i] = (int)(127 + 127*sin(2*PI*i/N_samples_wave)); // DACR bit
6-15 VALUE (valor ya desplazado!!!)

    }
    LPC_PINCON->PINSEL1 |= (2<<20);
    // enable AOUT (P0.26) pin

    __configuraTono__();
}

/**-----*/
//          //
//          //
//-----*/
//          @funcion      __configuraTono__()
//          //
//          //
//          @brief       Función de configuración del tono.
//          //
//          //
//-----**/
void __configuraTono__()
{
    // Linked CH0

    LLI0.source      = (uint32_t) &sinusoide[0];
    LLI0.destination = (uint32_t) &(LPC_DAC->DACR) + 1;
    LLI0.next        = (uint32_t) &LLI0;
    LLI0.control     = 1<<26 | 0<<21 | 0<<18 | N_samples_wave; //Transfersize=
N_samples_wave, SWidth=8bits, DWidth=8bits, Source Increment

```

```

LPC_SC->PCONP |= (1<<29);
// Power DMA

LPC_GPDMA->DMACConfig = 1; // enable the GPDMA
controller

LPC_GPDMA->DMACSync = (1<<6); // enable synchro logic for all reqs

LPC_GPDMA->DMACCSrcAddr = (uint32_t) &sinusoide[0];
LPC_GPDMA->DMACCDestAddr = (uint32_t) &(LPC_DAC->DSCR) + 1;
LPC_GPDMA->DMACCLLI = (uint32_t) &LLI0; // linked lists for ch0

LPC_GPDMA->DMACCControl = N_samples_wave // transfer size (0 - 11) = 32
muestras /ciclo

= 1

bit
0
increment
mode
    | (0 << 12) // source burst size (12 - 14) = 1
    | (0 << 15) // destination burst size (15 - 17)
    | (0 << 18) // source width (18 - 20) = 32 bit
CAMBIADO
    | (0 << 21) // destination width (21 - 23) = 32
NO CAMBIADO
    | (0 << 24) // source AHB select (24) = AHB 0
    | (0 << 25) // destination AHB select (25) = AHB
0
    | (1 << 26) // source increment (26) = increment
    | (0 << 27) // destination increment (27) = no
increment
    | (0 << 28) // mode select (28) = access in user
mode
    | (0 << 29) // (29) = access not bufferable
    | (0 << 30) // (30) = access not cacheable
    | (0 << 31); // terminal count interrupt disabled

LPC_GPDMA->DMACCConfig = 0
// channel enabled (0)

| (0 << 1) // source peripheral (1 - 5) = none
destination peripheral (6 - 10) = DAC
control (11 - 13) = MEM to PERF
control (11 - 13) = MEM to PERF
| (1 << 11) // flow
control (11 - 13) = MEM to PERF
control (11 - 13) = MEM to PERF
| (0 << 14) // (14)
= mask out error interrupt
| (0 << 15) // (15)
= mask out terminal count interrupt
| (0 << 16) // (16)
= no locked transfers

```

```

        | (0 << 18);                                // (27)
= no HALT

//F_out (salida del DAC)

LPC_DAC->DACCNTVAL = Fclk/N_samples_wave/Ftono -1; // (Ts DAC = F_out/N_samples
< Tsetup DAC = luseg. !!!!)

/* DMA, timer running, dbuff */
LPC_DAC->DACCTRL = 1<<3 | 1<<2 | 1<<1;
ACTUALIZADOR->Audiorev = 1;

}

/**-----*/
//



//



//funcion      __configuraAudio__()

//



//


//



//brief      Función de configuración del audio.
//



//



//-----*/
//



void __configuraAudio__()
{
    // Linked CH0

    LLI0.source      = (uint32_t) AUDIO;
    LLI0.destination = (uint32_t) &(LPC_DAC->DACR) + 1;
    LLI0.next        = (uint32_t) &LLI0;
    LLI0.control     = 1<<26 | 0<<21 | 0<<18 | MUESTRAS_AUDIO;           //
Incremento origen, MUESTRAS_AUDIO muestras, 8 bits todo.

    LPC_SC->PCONP      |=      (1<<29);
    // Enciendo el DMA.

    LPC_GPDMA->DMACConfig = 1;                      // Activo el
controlador del DMA.

    LPC_GPDMA->DMACSync   =      (1<<6);          // Sincronización
de registros.

```

```

LPC_GPDMACh0->DMACCSrcAddr = (uint32_t) AUDIO; // Empieza
en AUDIO.

LPC_GPDMACh0->DMACCDestAddr = (uint32_t) &(LPC_DAC->DSCR) + 1; // Ve a
DSCR + 2.

LPC_GPDMACh0->DMACCLLI = (uint32_t) &LLI0; // linked lists for ch0

LPC_GPDMACh0->DMACCControl = MUESTRAS_AUDIO // transfer size (0 - 11) = 32
muestras /ciclo

| (0 << 12) // source burst size (12 - 14) = 1
| (0 << 15) // destination burst size (15 - 17)
= 1

| (0 << 18) // source width (18 - 20) = 32 bit
| (2 << 21) // destination width (21 - 23) = 32
bit

| (0 << 24) // source AHB select (24) = AHB 0
| (0 << 25) // destination AHB select (25) = AHB
0

| (1 << 26) // source increment (26) = increment
| (0 << 27) // destination increment (27) = no
increment

| (0 << 28) // mode select (28) = access in user
mode

| (0 << 29) // (29) = access not bufferable
| (0 << 30) // (30) = access not cacheable
| (0 << 31); // terminal count interrupt disabled

LPC_GPDMACh0->DMACCConfig = 0
// channel enabled (0)

| (0 << 1) // source peripheral (1 - 5) = none
destination peripheral (6 - 10) = DAC

| (7 << 6) // control (11 - 13) = MEM to PERF
control (11 - 13) = MEM to PERF

| (1 << 11) // flow
= mask out error interrupt

| (0 << 14) // (14)
= mask out terminal count interrupt

| (0 << 15) // (15)
= mask out locked transfers

| (0 << 16) // (16)
= no locked transfers

| (0 << 18); // (27)
= no HALT

//F_out (salida del DAC)

LPC_DAC->DACCNTVAL = (Fclk/4000) - 1; // (Ts DAC = TsAudio < Tsetup DAC =
1useg. !!!!)

/* DMA, timer running, dbuff */
LPC_DAC->DACCTRL = 1<<3 | 1<<2 | 1<<1;

```

Alberto Palomo Alonso.

```
ACTUALIZADOR->Audiorev = 1;
}

/*
-----//
```

//

```
//
```

//

```
//
```

//

```
-----*/
```



```
/*
-----//
```

// @filename WDT.c //

// @version 0.00 //

// @author Alberto Palomo Alonso //

//

```
//
```

// @brief Código fuente del configurado del WDT. //

//

```
//
```

// @category Interno. //

//

```
//
```

// @map @include //

Alberto Palomo Alonso.

```
/*
-----//  
//      @filename      UART0.c  
//  
//      @version       2.01  
//  
//      @author        Alberto Palomo Alonso  
//  
//  
//  
//      @brief         Este es el programa que recoge la transmisión por UART0.  
//  
//  
//  
//      @category      Opcional.  
//  
//  
//  
//      @map           @include  
//  
//  
//      @variables  
//  
//      @function  
//  
//      @end  
//  
//  
//  
//-----//  
//  
//      @include      Includes pertenecientes a la transmisión asíncrona.  
//  
//  
//  
//-----**/  
#ifndef UART0
```

```
#define UART0

#include      "UART0.h"

#endif

/**-----*/
//



//



//



//      @variables          Variables del fichero.
//



//



//



//      Variables globales y externas.

char UART0_BUFFER_TX[CADMAX + 1];

extern char bufferx[30];

extern misDatos_t * DATOS;

extern uint8_t Clock[23];

uint8_t EstadoUART0 = UART_TX;

extern modificables_t MODIFICABLES;

uint8_t Inmortal = 0;

/**-----*/
//



//



//



//      @function      __configuraUART0__()

//



//



//      @brief           Esta función es la que configura el UART0 para transmisión
//      y recepción.
//



//



//



//      void  __configuraUART0__( void )          //      Configurado a
9600 baudios.

{

    uart0_init(9600);
```

```
tx_cadena_UART0(      "Hola.\n"      );
}

/**-----*/
//



//



//          @function      procesarComando()
//



//



//          @brief      Esta función manda el UART0_BUFFER_TX a la salida TX del
//UART0.
//



//          @input      char * Buff:   El buffer donde está contenido el comando.
//



//



//          @ret      Devuelve 1 si ha sido exitoso y 0 si no se ha
//obtenido el comando.
//



//



//-----*/
//



uint8_t procesarComando(      char      *      Buff      )

{
    uint8_t      retval = 0;

    switch( EstadoUART0 )

    {

        case UART_TX:

            /**      SECCIÓN PARA LOS COMANDOS DE TIPO 0: ABOUT*/
            if      (      !strcmp(      Buff      ,      COM0      )      )

            {

                retval = 1;

                strcpy(UART0_BUFFER_TX      ,      "\n Autor: \t Alberto
Palomo Alonso \n Version: \t 2.1.0 \n Sistemas Electronicos Digitales Avanzados \t UAH
\n"      );

            }

            /**      SECCIÓN PARA LOS COMANDOS DE TIPO 1: GIVE*/
            if      (      !strcmp(      Buff      ,      COM10)      )

            {
```

```
        if ( !Inmortal )
        {
                strcpy (      UART0_BUFFER_TX
"\nSUGAR n.n\n");
                Inmortal = 1;
                retval = 1;
        }
}

if (      !strcmp(      Buff      ,      COM1)  )
{
        sprintf(      UART0_BUFFER_TX      ,
"\nIP:
%d.%d.%d.%d \n", __IP1B, __IP2B, __IP3B, __IP4B);
        retval = 1;
}

if (      !strcmp(      Buff      ,      COM12)  )
{
        sprintf(      UART0_BUFFER_TX      ,
"\nTEMPERATURA: %f °C\n", DATOS->Temperatura);

        retval = 1;
}

if (      !strcmp(      Buff      ,      COM13)  )
{
        sprintf(      UART0_BUFFER_TX      ,
"\nPRESION: %f
mBar.\n", DATOS->Presion);

        retval = 1;
}

if (      !strcmp(      Buff      ,      COM14)  )
{
        sprintf(      UART0_BUFFER_TX      ,
"\nVELOCIDAD
DEL VIENTO: %f m./s.\n", DATOS->VelViento);

        retval = 1;
}

if (      !strcmp(      Buff      ,      COM15)  )
{
        sprintf(      UART0_BUFFER_TX      ,
"\nX: NA
\nY: NA \nZ: %f m.\n", DATOS->Lugar.Altura);

        retval = 1;
}

if (      !strcmp(      Buff      ,      COM16)  )
{
        sprintf(      UART0_BUFFER_TX      ,
"\nINDICE UV:
%f\n", DATOS->IndiceUV);
```

```
        retval = 1;
    }

    if      (      !strcmp(      Buff      ,      COM17)  )
    {

        strcpy (      UART0_BUFFER_TX      ,      (const char
*)Clock);

        retval = 1;
    }

    if      (      !strcmp(      Buff      ,      COM18)  )
    {

        sprintf (      UART0_BUFFER_TX      ,      "\nHUMEDAD: %f
\n", 0.01*DATOS->Humedad);

        retval = 1;
    }

    if      (      !strcmp(      Buff      ,      COM19)  )
    {

        sprintf (      UART0_BUFFER_TX      ,      "\nBRILLO: %f
LUX.\n", DATOS->Brillo);

        retval = 1;
    }

    /**
     * SECCIÓN PARA LOS COMANDOS DE TIPO 3: KILL*/
    if      (      !strcmp(      Buff      ,      COM3      )  )
    {

        while ( !Inmortal );

        strcpy(UART0_BUFFER_TX      ,      "Demasiado dulce como
para matarlo, mejor para otra ocasion...\n");

        retval = 1;
    }

    /**
     * SECCIÓN PARA LOS COMANDOS DE TIPO 4: HELP */
    if      (      !strcmp(      Buff      ,      COM4      )  )
    {

        strcpy(UART0_BUFFER_TX      ,      "Informacion:\n\n
ABOUT: Muestra info. del sistema.\n GIVE: Proporciona el dato deseado.\n KILL: Cuelga el
programa.\n SET: Modifica variables.\n"      );

        retval = 1;
    }

    if      (      !strcmp(      Buff      ,      COM41      )  )
    {

        strcpy(UART0_BUFFER_TX      ,      "\nGIVE + [IP,
TEMPERATURA, PRESION, BRILLO, LUGAR, VIENTO, INDICEUV, HORA, HUMEDAD]\n"      );

        retval = 1;
    }

    if      (      !strcmp(      Buff      ,      COM42      )  )
    {
```

```
{  
    strcpy( UART0_BUFFER_TX , "\nSET + [BRILLO,  
HORA, MIN TEMP, MAX TEMP, MIN PRES, MAX PRES, TEMPERATURA, PRESION]\n" );  
  
    retval = 1;  
}  
  
/** CONTROL DE ERROR:  
if ( !retval )  
{  
    strcpy( UART0_BUFFER_TX , "Error: comando no definido,  
escriba 'HELP' para ver la lista.\n" );  
}  
  
/** SECCIÓN PARA LOS COMANDOS DE TIPO 2: SET  
if ( !strcmp( Buff , COM20 ) )  
{  
    strcpy( UART0_BUFFER_TX , "Introduzca los  
segundos de brillo: \n" );  
  
    retval = 1;  
  
    EstadoUART0 = UART_RX_BRILLO;  
}  
  
if ( !strcmp( Buff , COM21 ) )  
{  
    strcpy( UART0_BUFFER_TX , "Introduzca la fecha  
separada por espacios: \n" );  
  
    retval = 1;  
  
    EstadoUART0 = UART_RX_HORA;  
}  
  
if ( !strcmp( Buff , COM22 ) )  
{  
    strcpy( UART0_BUFFER_TX , "Introduzca el valor  
minimo de temperatura: \n" );  
  
    retval = 1;  
  
    EstadoUART0 = UART_RX_MINT;  
}  
  
if ( !strcmp( Buff , COM23 ) )  
{  
    strcpy( UART0_BUFFER_TX , "Introduzca el valor  
maximo de temperatura: \n" );  
  
    retval = 1;  
  
    EstadoUART0 = UART_RX_MAXT;  
}  
  
if ( !strcmp( Buff , COM24 ) )  
{
```

```

        strcpy( UART0_BUFFER_TX , "Introduzca el valor
minimo de presion: \n");
        retval = 1;
        EstadoUART0 = UART_RX_MINP;
    }

    if ( !strcmp( Buff , COM25 ) )
    {
        strcpy( UART0_BUFFER_TX , "Introduzca el valor
maximo de presion: \n");
        retval = 1;
        EstadoUART0 = UART_RX_MAXP;
    }

    if ( !strcmp( Buff , COM27 ) )
    {
        strcpy( UART0_BUFFER_TX , "Ahora medimos
presion... \n" );
        retval = 1;
        MODIFICABLES.Var_medida = 1;
    }

    if ( !strcmp( Buff , COM26 ) )
    {
        strcpy( UART0_BUFFER_TX , "Ahora medimos
temperatura... \n" );
        retval = 1;
        MODIFICABLES.Var_medida = 0;
    }

    break;

case UART_RX_BRILLO:
    sscanf( bufferx , "%d" , &MODIFICABLES.TiempoBrillo);
    strcpy( UART0_BUFFER_TX , "Tiempo de hold
cambiado.\n" );
    EstadoUART0 = UART_TX;
    break;

case UART_RX_MINT:
    sscanf( bufferx , "%f" , &MODIFICABLES.Min_servo_t);
    strcpy( UART0_BUFFER_TX , "Cota minima de
temperatura cambiada.\n" );
    EstadoUART0 = UART_TX;
    break;

case UART_RX_MAXT:
    sscanf( bufferx , "%f" , &MODIFICABLES.Max_servo_t);

```

```

        strcpy( UART0_BUFFER_TX , "Cota maxima de
temperatura cambiada.\n" );
        EstadoUART0 = UART_TX;
        break;

    case UART_RX_MINP:
        sscanf( bufferx , "%f" , &MODIFICABLES.Min_servo_p );
        strcpy( UART0_BUFFER_TX , "Cota minima de
presion cambiada.\n" );
        EstadoUART0 = UART_TX;
        break;

    case UART_RX_MAXP:
        sscanf( bufferx , "%f" , &MODIFICABLES.Max_servo_p );
        strcpy( UART0_BUFFER_TX , "Cota maxima de
presion cambiada.\n" );
        EstadoUART0 = UART_TX;
        break;

    case UART_RX_HORA:
        sscanf( bufferx , "%d %d %d %d %d" , (int
*)&LPC_RTC->DOM , (int *)&LPC_RTC->MONTH , (int *)&LPC_RTC->YEAR , (int *)&LPC_RTC->HOUR ,
(int *)&LPC_RTC->MIN , (int *)&LPC_RTC->SEC );
        strcpy( UART0_BUFFER_TX , "Hora cambiada...\\n"
);
        EstadoUART0 = UART_TX;
        break;

    }

tx_cadena_UART0(UART0_BUFFER_TX);

/**      ZONA RETURN      */

return retval;
}

/*-----
-----//



//



//



//



//-----**/

```

```
/* uart.c

 * contiene las funciones:

1  UART0_IRQHandler(void)
2  tx_cadena_UART0(char *ptr)
3  uart0_set_baudrate(unsigned int baudrate)
4  uart0_init(int baudrate)

 */

#include <LPC17xx.h>
#include "uart.h"

char bufferx[30];      // Buffer de recepción
char *ptr_rx; // puntero de recepción
char rx_completa;// Flag de recepción de cadena completa que se activa al recibir
CR(0x0D)
char *ptr_tx;           // puntero de transmisión
char tx_completa;       // Flag de transmisión de cadena completa

/*
 * UART0 interrupt handler
 */
void UART0_IRQHandler(void) {

    switch(LPC_UART0->IIR&0x0E) {

        case 0x04:                                /* RBR,
Receiver Buffer Ready */
            *ptr_rx=LPC_UART0->RBR;                /* lee el dato recibido y lo
almacena */
            if (*ptr_rx++ ==13)                      // Caracter return --> Cadena
completa
            {
                *ptr_rx=0;                         /* Añadimos el
carácter null para tratar los datos recibidos como una cadena*/
                rx_completa = 1; /* rx completa */
                procesarComando( bufferx );
                ptr_rx=bufferx; /* puntero al
inicio del buffer para nueva recepción */
            }
    }
}
```

```
        }

        break;

    case 0x02:                                /* THRE, Transmit
Holding Register empty */

        if (*ptr_tx!=0) LPC_UART0->THR=*ptr_tx++; /* carga un nuevo dato para
ser transmitido */

        else tx_completa=1;

        break;

    }

}

// Función para enviar una cadena de texto
// El argumento de entrada es la dirección de la cadena, o
// directamente la cadena de texto entre comillas
void tx_cadena_UART0(char *cadena)
{
    ptr_tx=cadena;
    tx_completa=0;
LPC_UART0->THR=*ptr_tx++; // IMPORTANTE: Introducir un carácter al comienzo para
iniciar TX o
                                            // activar flag interrupción por
registro transmisor vacío

static int uart0_set_baudrate(unsigned int baudrate) {
    int errorStatus = -1; // Failure

    // UART clock (FCCO / PCLK_UART0)
    // unsigned int uClk = SystemCoreClock / 4;
    unsigned int uClk =SystemCoreClock/4;
    unsigned int calcBaudrate = 0;
    unsigned int temp = 0;

    unsigned int mulFracDiv, dividerAddFracDiv;
    unsigned int divider = 0;
    unsigned int mulFracDivOptimal = 1;
    unsigned int dividerAddOptimal = 0;
    unsigned int dividerOptimal = 0;
```

```
unsigned int relativeError = 0;
unsigned int relativeOptimalError = 100000;

uClk = uClk >> 4; /* div by 16 */

/*
 * The formula is :
 * BaudRate= uClk * (mulFracDiv/(mulFracDiv+dividerAddFracDiv) / (16 * DLL)
 *
 * The value of mulFracDiv and dividerAddFracDiv should comply to the following
expressions:
 * 0 < mulFracDiv <= 15, 0 <= dividerAddFracDiv <= 15
 */
for (mulFracDiv = 1; mulFracDiv <= 15; mulFracDiv++) {
    for (dividerAddFracDiv = 0; dividerAddFracDiv <= 15; dividerAddFracDiv++) {
        temp = (mulFracDiv * uClk) / (mulFracDiv + dividerAddFracDiv);

        divider = temp / baudrate;
        if ((temp % baudrate) > (baudrate / 2))
            divider++;

        if (divider > 2 && divider < 65536) {
            calcBaudrate = temp / divider;

            if (calcBaudrate <= baudrate) {
                relativeError = baudrate - calcBaudrate;
            } else {
                relativeError = calcBaudrate - baudrate;
            }

            if (relativeError < relativeOptimalError) {
                mulFracDivOptimal = mulFracDiv;
                dividerAddOptimal = dividerAddFracDiv;
                dividerOptimal = divider;
                relativeOptimalError = relativeError;
                if (relativeError == 0)
                    break;
            }
        }
    }
}
```

```
}

    if (relativeError == 0)
        break;
}

if (relativeOptimalError < ((baudrate * UART_ACCEPTED_BAUDRATE_ERROR) / 100)) {

    LPC_UART0->LCR |= DLAB_ENABLE;           // importante poner a 1
    LPC_UART0->DLM = (unsigned char) ((dividerOptimal >> 8) & 0xFF);
    LPC_UART0->DLL = (unsigned char) dividerOptimal;
    LPC_UART0->LCR &= ~DLAB_ENABLE;          // importante poner a 0

    LPC_UART0->FDR = ((mulFracDivOptimal << 4) & 0xF0) | (dividerAddOptimal & 0x0F);

    errorStatus = 0; //< Success
}

return errorStatus;
}

void uart0_init(int baudrate) {

    LPC_PINCON->PINSEL0 |= (1<<4) | (1<<6); // Change P0.2 and P0.3 mode to TXD0 and RXD0
    ptr_rx = bufferx;

    LPC_UART0->LCR &= ~STOP_1_BIT & ~PARITY_NONE; // Set 8N1 mode (8 bits/dato, sin
    pariad, y 1 bit de stop)

    LPC_UART0->LCR |= CHAR_8_BIT | 1 << 3; /* @CHANGED: CHAR_8_BIT AÑADIDO POR
    ALBERTO PALOMO.*/

    uart0_set_baudrate(baudrate); // Set the baud rate

    LPC_UART0->IER = THRE_IRQ_ENABLE|RBR_IRQ_ENABLE; // Enable UART TX and RX interrupt
    (for LPC17xx UART)

    NVIC_EnableIRQ(UART0_IRQn); // Enable the UART interrupt (for Cortex-CM3 NVIC)

}
```

```
/**-----//  
// @filename HTTP_SOURCE.c //  
// @version 0.00 //  
// @author Alberto Palomo Alonso //  
//  
// @brief Código que configura la página WEB. //  
//  
// @category Opcional. //  
// @map @include //  
// @funcion //  
// @end //  
//  
// @include //  
//-----//  
//  
//-----*/
```

```
#ifndef HTTPSOURCE
#define HTTPSOURCE
#include      "HTTP_SOURCE.h"
#endif

void __configuraWEB__()
{
    init_TcpNet(); //      Inicializamos TcpNet(RTL.h).
}

void __mantenerTCP__()
{
    main_TcpNet();
}

/**-----//-----
//-----*/
//          //

//          @end      ENDFILE.

//          //
//          //
//-----*/
//-----**/


/*
 *      RL-ARM - TCPnet
 *-----*
 *      Name:      NET_CONFIG.C
 *      Purpose: Configuration of RL TCPnet by user
 *      Rev.:      V4.72
 *-----*
 *      This code is part of the RealView Run-Time Library.
 *      Copyright (c) 2004-2013 KEIL - An ARM Company. All rights reserved.
 *-----*/
#include <Net_Config.h>
```

```
#include      "miGlobal.h"

----- <<< Use Configuration Wizard in Context Menu >>> -----
//  
  
// <h>System Definitions  
// ======  
// <i> Global TCPnet System definitions  
//     <s.15>Local Host Name  
//         <i> This is the name under which embedded host  
//             can be accessed on a local area network.  
//             <i> Default: "mcb2300"  
  
#define LHOST_NAME      "MiniDK2"  
  
//     <o>Memory Pool size <1536-262144:4><#/4>  
//         <i> This is the size of a memory pool in bytes. Buffers for  
//             TCPnet packets are allocated from this memory pool.  
//             <i> Default: 8000 bytes  
  
#define MEM_SIZE        3000  
  
//     <o>Tick Timer interval <10=> 10 ms <20=> 20 ms <25=> 25 ms  
//                     <40=> 40 ms <50=> 50 ms <100=> 100 ms  
//                     <200=> 200 ms  
//         <i> System Tick Timer interval for software timers  
//         <i> Default: 100 ms  
  
#define TICK_INTERVAL   100  
  
// </h>  
// <e>Ethernet Network Interface  
// ======  
// <i> Enable or disable Ethernet Network Interface  
  
#define ETH_ENABLE      1  
  
//     <h>MAC Address  
// ======  
//     <i> Local Ethernet MAC Address  
//         <i> Value FF:FF:FF:FF:FF:FF is not allowed.  
//         <i> It is an ethernet Broadcast MAC address.  
//             <o>Address byte 1 <0x00-0xff:2>  
//                 <i> LSB is an ethernet Multicast bit.
```

```
//      <i> Must be 0 for local MAC address.  
//      <i> Default: 0x00  
  
#define _MAC1          0xE0  
  
  
//      <o>Address byte 2 <0x00-0xff>  
//      <i> Default: 0x30  
  
#define _MAC2          0xF8  
  
  
//      <o>Address byte 3 <0x00-0xff>  
//      <i> Default: 0x6C  
  
#define _MAC3          0x46  
  
  
//      <o>Address byte 4 <0x00-0xff>  
//      <i> Default: 0x00  
  
#define _MAC4          0x35  
  
  
//      <o>Address byte 5 <0x00-0xff>  
//      <i> Default: 0x00  
  
#define _MAC5          0x45  
  
  
//      <o>Address byte 6 <0x00-0xff>  
//      <i> Default: 0x01  
  
#define _MAC6          0xBC  
  
  
//  </h>  
//  <h>IP Address  
//  ======  
//  <i> Local Static IP Address  
//  <i> Value 255.255.255.255 is not allowed.  
//  <i> It is a Broadcast IP address.  
//  <o>Address byte 1 <0-255>  
//  <i> Default: 192  
  
#define _IP1           __IP1B  
  
  
//  <o>Address byte 2 <0-255>  
//  <i> Default: 168  
  
#define _IP2           __IP2B  
  
  
//  <o>Address byte 3 <0-255>
```

```
//      <i> Default: 0
#define _IP3          __IP3B

//      <o>Address byte 4 <0-255>
//      <i> Default: 100
#define _IP4          __IP4B

//      </h>
//      <h>Subnet mask
//      =====
//      <i> Local Subnet mask
//      <o>Mask byte 1 <0-255>
//      <i> Default: 255
#define _MSK1         __MASK1B

//      <o>Mask byte 2 <0-255>
//      <i> Default: 255
#define _MSK2         __MASK2B

//      <o>Mask byte 3 <0-255>
//      <i> Default: 255
#define _MSK3         __MASK3B

//      <o>Mask byte 4 <0-255>
//      <i> Default: 0
#define _MSK4         __MASK4B

//      </h>
//      <h>Default Gateway
//      =====
//      <i> Default Gateway IP Address
//      <o>Address byte 1 <0-255>
//      <i> Default: 192
#define _GW1          __GW1B

//      <o>Address byte 2 <0-255>
//      <i> Default: 168
#define _GW2          __GW2B
```

```
//      <o>Address byte 3 <0-255>
//      <i> Default: 0
#define _GW3          __GW3B

//      <o>Address byte 4 <0-255>
//      <i> Default: 254
#define _GW4          __GW4B

//      </h>
//      <h>Primary DNS Server
//      =====
//      <i> Primary DNS Server IP Address
//      <o>Address byte 1 <0-255>
//      <i> Default: 194
#define _pDNS1        192

//      <o>Address byte 2 <0-255>
//      <i> Default: 25
#define _pDNS2        168

//      <o>Address byte 3 <0-255>
//      <i> Default: 2
#define _pDNS3        5

//      <o>Address byte 4 <0-255>
//      <i> Default: 129
#define _pDNS4        1

//      </h>
//      <h>Secondary DNS Server
//      =====
//      <i> Secondary DNS Server IP Address
//      <o>Address byte 1 <0-255>
//      <i> Default: 194
#define _sDNS1        194

//      <o>Address byte 2 <0-255>
//      <i> Default: 25
#define _sDNS2        25
```

```
//      <o>Address byte 3 <0-255>
//      <i> Default: 2
#define _sDNS3          2

//      <o>Address byte 4 <0-255>
//      <i> Default: 130
#define _sDNS4          130

//  </h>
//  <h>ARP Definitions
//  =====
//  <i> Address Resolution Protocol Definitions
//  <o>Cache Table size <5-100>
//  <i> Number of cached hardware/IP addresses
//  <i> Default: 10
#define ARP_TABSIZE     10

//  <o>Cache Timeout in seconds <5-255>
//  <i> A timeout for a cached hardware/IP addresses
//  <i> Default: 150
#define ARP_TIMEOUT     150

//  <o>Number of Retries <0-20>
//  <i> Number of Retries to resolve an IP address
//  <i> before ARP module gives up
//  <i> Default: 4
#define ARP_MAXRETRY    4

//  <o>Resend Timeout in seconds <1-10>
//  <i> A timeout to resend the ARP Request
//  <i> Default: 2
#define ARP_RESEND      2

//  <q>Send Notification on Address changes
//  <i> When this option is enabled, the embedded host
//  <i> will send a Gratuitous ARP notification at startup,
//  <i> or when the device IP address has changed.
//  <i> Default: Disabled
```

```
#define ARP_NOTIFY      0

//  </h>
//  <e>IGMP Group Management
//  =====
//  <i> Enable or disable Internet Group Management Protocol
#define IGMP_ENABLE      0

//  <o>Membership Table size <2-50>
//  <i> Number of Groups this host can join
//  <i> Default: 5
#define IGMP_TABSIZE     5

//  </e>
//  <q>NetBIOS Name Service
//  =====
//  <i> When this option is enabled, the embedded host can be
//  <i> accessed by his name on the local LAN using NBNS protocol.
#define NBNS_ENABLE      0

//  <e>Dynamic Host Configuration
//  =====
//  <i> When this option is enabled, local IP address, Net Mask
//  <i> and Default Gateway are obtained automatically from
//  <i> the DHCP Server on local LAN.
#define DHCP_ENABLE       0

//  <s.40>Vendor Class Identifier
//  <i> This value is optional. If specified, it is added
//  <i> to DHCP request message, identifying vendor type.
//  <i> Default: ""
#define DHCP_VCID        ""

//  <q>Bootfile Name
//  <i> This value is optional. If enabled, the Bootfile Name
//  <i> (option 67) is also requested from DHCP server.
//  <i> Default: disabled
#define DHCP_BOOTF        0
```

```
//      <q>NTP Servers
//      <i> This value is optional. If enabled, a list of NTP Servers
//      <i> (option 42) is also requested from DHCP server.
//      <i> Default: disabled
#define DHCP_NTPSRV      0

//  </e>
// </e>

// <e>PPP Network Interface
// =====
// <i> Enable or disable PPP Network Interface
#define PPP_ENABLE      0

//      <h>IP Address
// =====
//      <i> Local Static IP Address
//      <o>Address byte 1 <0-255>
//      <i> Default: 192
#define _IP1P           192

//      <o>Address byte 2 <0-255>
//      <i> Default: 168
#define _IP2P           168

//      <o>Address byte 3 <0-255>
//      <i> Default: 125
#define _IP3P           125

//      <o>Address byte 4 <0-255>
//      <i> Default: 1
#define _IP4P           1

//  </h>
//      <h>Subnet mask
// =====
//      <i> Local Subnet mask
//      <o>Mask byte 1 <0-255>
//      <i> Default: 255
```

```
#define _MSK1P          255

//      <o>Mask byte 2 <0-255>
//      <i> Default: 255

#define _MSK2P          255

//      <o>Mask byte 3 <0-255>
//      <i> Default: 255

#define _MSK3P          255

//      <o>Mask byte 4 <0-255>
//      <i> Default: 0

#define _MSK4P          0

//      </h>

//      <h>Primary DNS Server
//      =====

//      <i> Primary DNS Server IP Address
//      <o>Address byte 1 <0-255>
//      <i> Default: 194

#define _pDNS1P         194

//      <o>Address byte 2 <0-255>
//      <i> Default: 25

#define _pDNS2P         25

//      <o>Address byte 3 <0-255>
//      <i> Default: 2

#define _pDNS3P         2

//      <o>Address byte 4 <0-255>
//      <i> Default: 129

#define _pDNS4P         129

//      </h>

//      <h>Secondary DNS Server
//      =====

//      <i> Secondary DNS Server IP Address
//      <o>Address byte 1 <0-255>
```

```
//      <i> Default: 194
#define _sDNS1P          194

//      <o>Address byte 2 <0-255>
//      <i> Default: 25
#define _sDNS2P          25

//      <o>Address byte 3 <0-255>
//      <i> Default: 2
#define _sDNS3P          2

//      <o>Address byte 4 <0-255>
//      <i> Default: 130
#define _sDNS4P          130

//      </h>
//      <e>Logon Authentication
//      =====
//      <i> Enable or disable user authentication
#define PPP_AUTHEN        1

//      <q>Unsecured password (PAP)
//      <i>Allow or use Password Authentication Protocol.
#define PPP_PAPEN         1

//      <q>Secured password (CHAP-MD5)
//      <i>Request or use Challenge Handshake Authentication
//      <i>Protocol with MD5 digest algorithm.
#define PPP_CHAPEN        1

//      </e>
//      <q>Obtain Client IP address automatically
//      =====
//      <i> This option only applies when PPP Dial-up is used to dial
//      <i> to remote PPP Server. If checked, network connection
//      <i> dynamically obtains an IP address from remote PPP Server.
#define PPP_GETIP          1

//      <q>Use Default Gateway on remote Network
```

```
// =====
// <i> This option only applies when both Ethernet and PPP Dial-up
// <i> are used. If checked, data that cannot be sent to local LAN
// <i> is forwarded to Dial-up network instead.

#define PPP_DEFGW      1

// <o>Async Control Character Map <0x0-0xffffffff>
// <i> A bit-map of control characters 0-31, which are
// <i> transmitted escaped as a 2 byte sequence.
// <i> For XON/XOFF set this value to: 0x000A 0000
// <i> Default: 0x00000000

#define PPP_ACCM      0x00000000

// <o>LCP Echo Interval in seconds <0-3600>
// <i> If no frames are received within this interval, PPP sends an
// <i> Echo Request and expects an Echo Response from the peer.
// <i> If the response is not received, the link is terminated.
// <i> A value of 0 disables the LCP Echo test.
// <i> Default: 30

#define PPP_ECHOTOUT   30

// <o>Number of Retries <0-20>
// <i> How many times PPP will try to retransmit data
// <i> before giving up. Increase this value for links
// <i> with low baud rates or high latency.
// <i> Default: 3

#define PPP_MAXRETRY   3

// <o>Retry Timeout in seconds <1-10>
// <i> If no response received within this time frame,
// <i> PPP module will try to resend the data again.
// <i> Default: 2

#define PPP_RETRYTOUT   2

// </e>
// <e>SLIP Network Interface
// =====
// <i> Enable or disable SLIP Network Interface

#define SLIP_ENABLE     0
```

```
// <h>IP Address  
// =====  
// <i> Local Static IP Address  
// <o>Address byte 1 <0-255>  
// <i> Default: 192  
#define _IP1S 192  
  
// <o>Address byte 2 <0-255>  
// <i> Default: 168  
#define _IP2S 168  
  
// <o>Address byte 3 <0-255>  
// <i> Default: 225  
#define _IP3S 225  
  
// <o>Address byte 4 <0-255>  
// <i> Default: 1  
#define _IP4S 1  
  
// </h>  
// <h>Subnet mask  
// =====  
// <i> Local Subnet mask  
// <o>Mask byte 1 <0-255>  
// <i> Default: 255  
#define _MSK1S 255  
  
// <o>Mask byte 2 <0-255>  
// <i> Default: 255  
#define _MSK2S 255  
  
// <o>Mask byte 3 <0-255>  
// <i> Default: 255  
#define _MSK3S 255  
  
// <o>Mask byte 4 <0-255>  
// <i> Default: 0  
#define _MSK4S 0
```

```
//      </h>

//      <h>Primary DNS Server
//      =====
//      <i> Primary DNS Server IP Address
//          <o>Address byte 1 <0-255>
//          <i> Default: 194
#define _pDNS1S      194

//          <o>Address byte 2 <0-255>
//          <i> Default: 25
#define _pDNS2S      25

//          <o>Address byte 3 <0-255>
//          <i> Default: 2
#define _pDNS3S      2

//          <o>Address byte 4 <0-255>
//          <i> Default: 129
#define _pDNS4S      129

//      </h>

//      <h>Secondary DNS Server
//      =====
//      <i> Secondary DNS Server IP Address
//          <o>Address byte 1 <0-255>
//          <i> Default: 194
#define _sDNS1S      194

//          <o>Address byte 2 <0-255>
//          <i> Default: 25
#define _sDNS2S      25

//          <o>Address byte 3 <0-255>
//          <i> Default: 2
#define _sDNS3S      2

//          <o>Address byte 4 <0-255>
//          <i> Default: 130
```

```
#define _sDNS4S      130

// </h>
// <q>Use Default Gateway on remote Network
// =====
// <i> This option only applies when both Ethernet and SLIP Dial-up
// <i> are used. If checked, data that cannot be sent to local LAN
// <i> is forwarded to Dial-up network instead.

#define SLIP_DEFGW    1

// </e>
// <e>UDP Sockets
// =====
// <i> Enable or disable UDP Sockets
#define UDP_ENABLE     1

// <o>Number of UDP Sockets <1-20>
// <i> Number of available UDP sockets
// <i> Default: 5
#define UDP_NUMSOCKS   2

// </e>
// <e>TCP Sockets
// =====
// <i> Enable or disable TCP Sockets
#define TCP_ENABLE      1

// <o>Number of TCP Sockets <1-20>
// <i> Number of available TCP sockets
// <i> Default: 5
#define TCP_NUMSOCKS   12

// <o>Number of Retries <0-20>
// <i> How many times TCP module will try to retransmit data
// <i> before giving up. Increase this value for high-latency
// <i> and low_throughput networks.
// <i> Default: 5
#define TCP_MAXRETRY    5
```

```
// <o>Retry Timeout in seconds <1-10>
// <i> If data frame not acknowledged within this time frame,
// <i> TCP module will try to resend the data again.
// <i> Default: 4
#define TCP_RETRYTOUT 4

// <o>Default Connect Timeout in seconds <1-600>
// <i> Default TCP Socket Keep Alive timeout. When it expires
// <i> with no TCP data frame send, TCP Connection is closed.
// <i> Default: 120
#define TCP_DEFOUT 120

// <o>Maximum Segment Size <536-1460>
// <i> The Maximum Segment Size specifies the maximum
// <i> number of bytes in the TCP segment's Data field.
// <i> Default: 1460
#define TCP_MAXSEGSZ 1460

// <o>Receive Window Size <536-65535>
// <i> Receive Window Size specifies the size of data,
// <i> that the socket is able to buffer in flow-control mode.
// <i> Default: 4380
#define TCP_RECWINSZ 4380

/* TCP fixed timeouts */
#define TCP_INIT_RETRY_TOUT 1      /* TCP initial Retransmit period in sec. */
#define TCP_SYN_RETRY_TOUT 2      /* TCP SYN frame retransmit period in sec. */
#define TCP_CONTRY 7              /* Number of retries to establish a conn. */

// </e>
// <e>HTTP Server
// =====
// <i> Enable or disable HTTP Server
#define HTTP_ENABLE 1

// <o>Number of HTTP Sessions <1-10>
// <i> Number of simultaneously active HTTP Sessions.
// <i> Default: 3
#define HTTP_NUMSESS 6
```

```
// <o>Port Number <1-65535>
// <i> Listening port number.
// <i> Default: 80
#define HTTP_PORTNUM 80

// <s.50>Server-Id header
// <i> This value is optional. If specified, it overrides
// <i> the default HTTP Server header from the library.
// <i> Default: ""
#define HTTP_SRVID ""

// <e>Enable User Authentication
// <i> When enabled, the user will have to authenticate
// <i> himself by username and password before accessing
// <i> any page on this Embedded WEB server.
#define HTTP_ENAUTH 1

// <s.20>Authentication Realm
// <i> Default: "Embedded WEB Server"
#define HTTP_AUTHREALM "Embedded WEB Server"

// <s.15>Authentication Username
// <i> Default: "admin"
#define HTTP_AUTHUSER "user"

// <s.15>Authentication Password
// <i> Default: ""
#define HTTP_AUTHPASSW "Alver"

// </e>
// </e>
// <e>Telnet Server
// =====
// <i> Enable or disable Telnet Server
#define TNET_ENABLE 0

// <o>Number of Telnet Connections <1-10>
// <i> Number of simultaneously active Telnet Connections.
```

```
// <i> Default: 1
#define TNET_NUMSESS 2

// <o>Port Number <1-65535>
// <i> Listening port number.
// <i> Default: 23
#define TNET_PORTNUM 23

// <o>Idle Connection Timeout in seconds <0-3600>
// <i> When timeout expires, the connection is closed.
// <i> A value of 0 disables disconnection on timeout.
// <i> Default: 120
#define TNET_IDLETOUT 120

// <q>Disable Echo
// <i> When disabled, the server will not echo
// <i> characters it receives.
// <i> Default: Not disabled
#define TNET_NOECHO 0

// <e>Enable User Authentication
// <i> When enabled, the user will have to authenticate
// <i> himself by username and password before access
// <i> to the system is allowed.
#define TNET_ENAUTH 1

// <s.15>Authentication Username
// <i> Default: "admin"
#define TNET_AUTHUSER "admin"

// <s.15>Authentication Password
// <i> Default: ""
#define TNET_AUTHPASSW ""

// </e>
// </e>
// <e>TFTP Server
// =====
// <i> Enable or disable TFTP Server
```

```
#define TFTP_ENABLE      0

//  <o>Number of TFTP Sessions <1-10>
//  <i> Number of simultaneously active TFTP Sessions
//  <i> Default: 1
#define TFTP_NUMSESS     1

//  <o>Port Number <1-65535>
//  <i> Listening port number.
//  <i> Default: 69
#define TFTP_PORTNUM     69

//  <q>Enable Firewall Support
//  <i> Use the same Port Number to receive
//  <i> requests and send answers to clients.
//  <i> Default: Not Enabled
#define TFTP_ENFWALL     0

//  <o>Inactive Session Timeout in seconds <5-120>
//  <i> When timeout expires TFTP Session is closed.
//  <i> Default: 15
#define TFTP_DEFTOUT     15

//  <o>Number of Retries <1-10>
//  <i> How many times TFTP Server will try to
//  <i> retransmit the data before giving up.
//  <i> Default: 4
#define TFTP_MAXRETRY    4

// </e>
// <e>TFTP Client
// =====
// <i> Enable or disable TFTP Client
#define TFTPC_ENABLE      0

//  <o>Block Size <128=>128   <256=>256   <512=>512
//                  <1024=>1024 <1428=>1428
//  <i> Size of transfer block in bytes.
//  <i> Default: 512
```

```
#define TFTPC_BLOCKSZ 512

// <o>Number of Retries <1-10>
// <i> How many times TFTP Client will try to
// <i> retransmit the data before giving up.
// <i> Default: 4
#define TFTPC_MAXRETRY 4

// <o>Retry Timeout <2=>200 ms <5=>500 ms <10=>1 sec
// <20=>2 sec <50=>5 sec <100=>10 sec
// <i> If data frame not acknowledged within this time frame,
// <i> TFTP Client will try to resend the data again.
// <i> Default: 500 ms
#define TFTPC_RETRYTO 5

// </e>
// <e>FTP Server
// =====
// <i> Enable or disable FTP Server
#define FTP_ENABLE 0

// <o>Number of FTP Sessions <1-10>
// <i> Number of simultaneously active FTP Sessions
// <i> Default: 1
#define FTP_NUMSESS 3

// <o>Port Number <1-65535>
// <i> Listening port number.
// <i> Default: 21
#define FTP_PORTNUM 21

// <s.50>Welcome Message
// <i> This value is optional. If specified,
// <i> it overrides the default welcome message.
// <i> Default: ""
#define FTP_WELMSG ""

// <o>Idle Session Timeout in seconds <0-3600>
// <i> When timeout expires, the connection is closed.
```

```
// <i> A value of 0 disables disconnection on timeout.  
// <i> Default: 120  
  
#define FTP_IDLEOUT 120  
  
// <e>Enable User Authentication  
// <i> When enabled, the user will have to authenticate  
// <i> himself by username and password before access  
// <i> to the system is allowed.  
  
#define FTP_ENAUTH 1  
  
// <s.15>Authentication Username  
// <i> Default: "admin"  
  
#define FTP_AUTHUSER "admin"  
  
// <s.15>Authentication Password  
// <i> Default: ""  
  
#define FTP_AUTHPASSW ""  
  
// </e>  
// </e>  
// <e>FTP Client  
// ======  
// <i> Enable or disable FTP Client  
  
#define FTPC_ENABLE 0  
  
// <o>Response Timeout in seconds <1-120>  
// <i> This is a time for FTP Client to wait for a response from  
// <i> the Server. If timeout expires, Client aborts operation.  
// <i> Default: 10  
  
#define FTPC_DEFOUT 10  
  
// <q>Passive mode (PASV)  
// <i> The client initiates a data connection to the server.  
// <i> Default: Not passive (Active)  
  
#define FTPC_PASVMODE 0  
  
// </e>  
// <e>DNS Client  
// ======
```

```
// <i> Enable or disable DNS Client
#define DNS_ENABLE      0

//      <o>Cache Table size <5-100>
//      <i> Number of cached DNS host names/IP addresses
//      <i> Default: 20
#define DNS_TABSIZE     20

// </e>

// <e>SMTP Client
// =====
// <i> Enable or disable SMTP Client
#define SMTP_ENABLE      0

//      <o>Response Timeout in seconds <5-120>
//      <i> This is a time for SMTP Client to wait for a response from
//      <i> SMTP Server. If timeout expires, Client aborts operation.
//      <i> Default: 20
#define SMTP_DEFTOUT     20

// </e>

// <e>SNMP Agent
// =====
// <i> Enable or disable SNMP Agent
#define SNMP_ENABLE       0

//      <s.15>Community Name
//      <i> Defines where an SNMP message is destined for.
//      <i> Default: "public"
#define SNMP_COMMUNITY   "public"

//      <o>Port Number <1-65535>
//      <i> Listening port number.
//      <i> Default: 161
#define SNMP_PORTNUM     161

//      <o>Trap Port Number <1-65535>
//      <i> Port number for Trap operations.
//      <i> Default: 162
```

```
#define SNMP_TRAPPRT 162

// <h>Trap Server
// =====
// <i> Trap Server IP Address
// <o>Address byte 1 <0-255>
// <i> Default: 192
#define SNMP_TRAPIP1 192

// <o>Address byte 2 <0-255>
// <i> Default: 168
#define SNMP_TRAPIP2 168

// <o>Address byte 3 <0-255>
// <i> Default: 0
#define SNMP_TRAPIP3 0

// <o>Address byte 4 <0-255>
// <i> Default: 100
#define SNMP_TRAPIP4 1

// </h>
// </e>
// <e>SNTP Client
// =====
// <i> Enable or disable SNTP Client
#define SNTP_ENABLE 0

// <q>Broadcast Mode
// =====
// <i> Enable this option, if you have NTP/SNTP server
// <i> on LAN, which is broadcasting NTP time messages.
// <i> Disable this option to access public NTP server.
// <i> Default: disabled
#define SNTP_BCMODE 0

// <h>NTP Server
// =====
// <i> Server IP Address
```

```
// <o>Address byte 1 <0-255>
// <i> Default: 217
#define SNTP_SRVIP1    217

// <o>Address byte 2 <0-255>
// <i> Default: 79
#define SNTP_SRVIP2    79

// <o>Address byte 3 <0-255>
// <i> Default: 179
#define SNTP_SRVIP3    179

// <o>Address byte 4 <0-255>
// <i> Default: 106
#define SNTP_SRVIP4    106

// </h>
// </e>
// <e>BSD Socket Interface
// =====
// <i> Enable or disable Berkeley Socket Programming Interface
#define BSD_ENABLE      0

// <o>Number of BSD Sockets <1-20>
// <i> Number of available Berkeley Sockets
// <i> Default: 2
#define BSD_NUMSOCKS   2

// <o>Number of Streaming Server Sockets <0-20>
// <i> Defines a number of Streaming (TCP) Server sockets,
// <i> that listen for an incoming connection from the client.
// <i> Default: 1
#define BSD_SRVSOCKS   1

// <o>Receive Timeout in seconds <0-600>
// <i> A timeout for socket receive in blocking mode.
// <i> Timeout value of 0 means indefinite timeout.
// <i> Default: 20
#define BSD_RCVTOOUT   20
```

```
// <q>Hostname Resolver  
// <i> Enable or disable Berkeley style hostname resolver.  
#define BSD_GETHOSTEN 0  
  
// </e>  
//----- <<< end of configuration section >>> -----  
  
/*-----  
 * Fatal Error Handler  
 *-----*/  
  
void sys_error (ERROR_CODE code) {  
    /* This function is called when a fatal error is encountered. The normal */  
    /* program execution is not possible anymore. Add your crytical error .*/  
    /* handler code here. */  
  
    switch (code) {  
        case ERR_MEM_ALLOC:  
            /* Out of memory. */  
            break;  
  
        case ERR_MEM_FREE:  
            /* Trying to release non existing memory block. */  
            break;  
  
        case ERR_MEM_CORRUPT:  
            /* Memory Link pointer is Corrupted. */  
            /* More data written than the size of allocated mem block. */  
            break;  
  
        case ERR_MEM_LOCK:  
            /* Locked Memory management function (alloc/free) re-entered. */  
            /* RTX multithread protection malfunctioning, not implemented */  
            /* or interrupt disable is not functioning correctly. */  
            break;  
  
        case ERR_UDP_ALLOC:  
            /* Out of UDP Sockets. */  
    }  
}
```

```
break;

case ERR_TCP_ALLOC:
    /* Out of TCP Sockets. */
    break;

case ERR_TCP_STATE:
    /* TCP State machine in undefined state. */
    break;
}

/* End-less loop */
while (1);

}

/*
 *      TCPnet Config Functions
 */
#define __NET_CONFIG__

#include <Net_lib.c>

/*
 * end of file
 */
/*
**-----//**
//          @filename      HTTP_CGI.c
//          @version       3.00
//
//          @author        Alberto Palomo Alonso
//          @brief         Código que contiene las llamadas a las funciones de CGI.
//
```

Alberto Palomo Alonso.

```
//  
//  
//-----**/  
extern misDatos_t      *      DATOS;  
extern modificables_t MODIFICABLES;  
/*-----**/  
//  
//  
//  
//  
// @function      cgi_process_var  
//  
// @brief         Utilizado para el método GET.  
//  
//  
//  
//-----**/  
void cgi_process_var  (      U8*      qs)  
{  
    U8 * var;  
    var = (U8 *)alloc_mem(40);  
  
    do  
    {  
        qs = http_get_var(qs , var, 40);  
        if( var[0] )  
        {  
            if (str_scomp( var , (U8 *)"tmin="))  
            {  
                sscanf( (const char *)&var[5] , "%f" ,  
&MODIFICABLES.Min_servo_t);  
            }  
            if (str_scomp( var , (U8 *)"tmax="))  
            {  
                sscanf( (const char *)&var[5] , "%f" ,  
&MODIFICABLES.Max_servo_t);  
            }  
        }  
    }  
}
```



```
-----*/  
void cgi_process_data (      U8          tipo      ,      U8      *      qs      ,  
    U16          longitud)  
{  
    //      NO UTILIZADO,  NO HAY PETICIONES EN ESTA VERSIÓN.  
}  
/*-----//  
//  
//  
//  
//  
//  
//  
//  
//  
//  
//  
//  
//  
//  
//  
//  
//  
//  
//  
//  
//  
//  
//  
//  
//  
//-----*/
```

```

U16 cgi_func ( U8 *bufflen, U32 *env, U8 *buff
, U16
{
    U32 longitud;

    switch( env[0] )
    {
        case TEMPERATURA:
            longitud = sprintf ( (char*)buff , (const char
*)&env[4]
, DATOS->Temperatura );
            break;
        case PRESION:
            longitud = sprintf ( (char*)buff , (const char
*)&env[4]
, DATOS->Presion );
            break;
        case HUMEDAD:
            longitud = sprintf ( (char*)buff , (const char
*)&env[4]
, DATOS->Humedad );
            break;
        case BRILLO:
            longitud = sprintf ( (char*)buff , (const char
*)&env[4]
, DATOS->Brillo );
            break;
        case ALTITUD:
            longitud = sprintf ( (char*)buff , (const char
*)&env[4]
, DATOS->Lugar.Altura );
            break;
        case LATITUD:
            longitud = sprintf ( (char*)buff , (const char
*)&env[4]
, DATOS->Lugar.Latitud );
            break;
        case LONGITUD:
            longitud = sprintf ( (char*)buff , (const char
*)&env[4]
, DATOS->Lugar.Longitud );
            break;
        case INDICEUV:
            longitud = sprintf ( (char*)buff , (const char
*)&env[4]
, DATOS->IndiceUV );
            break;
        case VELOCIDAD:
            longitud = sprintf ( (char*)buff , (const char
*)&env[4]
, DATOS->VelViento );
            break;
        case ANYO:
    }
}

```



```
/*-----  
 *      RL-ARM - TCPnet  
 *-----  
 *      Name:      EMAC_LPC17xx.c  
 *      Purpose:   Driver for NXP LPC1768 EMAC Ethernet Controller  
 *      Rev.:     V4.20  
 *-----  
 *      This code is part of the RealView Run-Time Library.  
 *      Copyright (c) 2004-2011 KEIL - An ARM Company. All rights reserved.  
 *-----  
 *      Modified to be used with LAN8720 of MiniDk2 Board.  
 *-----*/  
  
#include <Net_Config.h>  
#include "EMAC_LPC17xx_LAN8720.h"           // LAN8720  
#include <LPC17xx.h>                      /* LPC17xx definitions */  
  
//#define _10MBIT_  
//#define _100MBIT_  
  
typedef void (*IAP)(U32 *cmd, U32 *res);  
IAP iap_entry = (IAP)0x1FFF1FF1;  
  
/* The following macro definitions may be used to select the speed  
of the physical link:  
  
_10MBIT_    - connect at 10 MBit only  
_100MBIT_   - connect at 100 MBit only  
  
By default an autonegotiation of the link speed is used. This may take  
longer to connect, but it works for 10MBit and 100MBit physical links. */  
  
/* Net_Config.c */  
extern U8 own_hw_addr[];
```

```
/* EMAC local DMA Descriptors. */

static          RX_Desc Rx_Desc[NUM_RX_FRAG];
static __align(8) RX_Stat Rx_Stat[NUM_RX_FRAG]; /* Must be 8-Byte aligned */
static          TX_Desc Tx_Desc[NUM_TX_FRAG];
static          TX_Stat Tx_Stat[NUM_TX_FRAG];

/* EMAC local DMA buffers. */

static U32 rx_buf[NUM_RX_FRAG][ETH_FRAG_SIZE>>2];
static U32 tx_buf[NUM_TX_FRAG][ETH_FRAG_SIZE>>2];

/*-----
 *      EMAC Ethernet Driver Functions
 *-----
 * Required functions for Ethernet driver module:
 * a. Polling mode: - void init_etherne()
 *                  - void send_frame (OS_FRAME *frame)
 *                  - void poll_etherne (void)
 * b. Interrupt mode: - void init_etherne ()
 *                      - void send_frame (OS_FRAME *frame)
 *                      - void int_enable_eth ()
 *                      - void int_disable_eth ()
 *                      - interrupt function
 *-----*/
/* Local Function Prototypes */

static void rx_descr_init (void);
static void tx_descr_init (void);
static void write_PHY (U32 PhyReg, U16 Value);
static U16 read_PHY (U32 PhyReg);

/*----- comando setMac -----*/
unsigned char setMac (unsigned short * p_mac,unsigned short mode)
{
    int i;

    if(mode==0){
        for(i=3;i<6;i++)

```

```
    own_hw_adr[i]=(unsigned char)*p_mac++;

    own_hw_adr[0]=MAC_CID_0;
    own_hw_adr[1]=MAC_CID_1;
    own_hw_adr[2]=MAC_CID_2;

}

else

{

    for(i=0;i<6;i++)

        own_hw_adr[i]=(unsigned char)*p_mac++;

}

int_disable_eth ();

/* Disable Rx and Tx*/

LPC_EMAC->MAC1&= ~MAC1_REC_EN;

/* Disable Rx */

LPC_EMAC->Command&= ~CR_RX_EN & ~CR_TX_EN; /* Enable Receive */

//#define CR_RX_EN           0x00000001 /* Enable Receive */ */

//#define CR_TX_EN           0x00000002 /* Enable Transmit */ */

/* Set the Ethernet MAC Address registers */

LPC_EMAC->SA0 = ((U32)own_hw_adr[5] << 8) | (U32)own_hw_adr[4];

LPC_EMAC->SA1 = ((U32)own_hw_adr[3] << 8) | (U32)own_hw_adr[2];

LPC_EMAC->SA2 = ((U32)own_hw_adr[1] << 8) | (U32)own_hw_adr[0];

/* Reset all EMAC internal modules. */

LPC_EMAC->MAC1|= (MAC1_RES_TX | MAC1_RES_MCS_TX | MAC1_RES_RX |

                    MAC1_RES_MCS_RX | MAC1_SIM_RES | MAC1_SOFT_RES);

LPC_EMAC->Command|= (CR_REG_RES | CR_TX_RES | CR_RX_RES | CR_PASS_RUNT_FRM);

int_enable_eth ();

return 1; //cambiar por una comprobación de si se ha comido la mac

}

----- init_etherent -----*/
```

```
void init_ethernet (void)
{
    /* Initialize the EMAC ethernet controller. */

    static U32 regv,tout,id1,id2;

    /* Power Up the EMAC controller. */
    LPC_SC->PCONP |= 0x40000000;

    /* Enable P1 Ethernet Pins. */
    LPC_PINCON->PINSEL2 = 0x50150105;

    /* LPC176x devices, no MDIO, MDC remap. */
    LPC_PINCON->PINSEL3 = (LPC_PINCON->PINSEL3 & ~0x0000000F) | 0x00000005; //MDIO y MCLK
operativos

    /* Reset all EMAC internal modules. */
    LPC_EMAC->MAC1      = MAC1_RES_TX | MAC1_RES_MCS_TX | MAC1_RES_RX |
                           MAC1_RES_MCS_RX | MAC1_SIM_RES | MAC1_SOFT_RES;
    LPC_EMAC->Command = CR_REG_RES | CR_TX_RES | CR_RX_RES;

    /* A short delay after reset. */
    for (tout = 0; tout < 0x100; tout++) ; //LAN8720

    /* Initialize MAC control registers. */
    LPC_EMAC->MAC1 = MAC1_PASS_ALL; //accepts control frames: care
    LPC_EMAC->MAC2 = MAC2_CRC_EN | MAC2_PAD_EN;
    LPC_EMAC->MAXF = ETH_MAX_FLEN; //1536
    LPC_EMAC->CLRT = CLRT_DEF;
    LPC_EMAC->IPGR = IPGR_DEF;

    //LPC_EMAC->Command = CR_RMII | CR_PASS_RUNT_FRM; //DP83848C
    LPC_EMAC->Command = CR_RMII | CR_PASS_RUNT_FRM | CR_PASS_RX_FILT; // LAN8720
                           //CR_MII must be enabled or eth will not work

    /* Enable and Reset Reduced MII interface. */
    LPC_EMAC->MCFG = MCFG_CLK_DIV64 | MCFG_RES_MII; // LAN8720
    for (tout = 0; tout < 0x0100; tout++) ; // LAN8720
    LPC_EMAC->MCFG = MCFG_CLK_DIV64; // LAN8720
```

```
/* Reset Reduced MII Logic. */
LPC_EMAC->SUPP = SUPP_RES_RMII;
for (tout = 0; tout < 0x100; tout++) ; //LAN8720
LPC_EMAC->SUPP = 0;

/* Put the PHY in reset mode */
write_PHY (PHY_REG_BMCR, 0x8000);

/* Wait for hardware reset to end. */
for (tout = 0; tout < 0x100000; tout++)
{
    regv = read_PHY (PHY_REG_BMCR);
    if (!(regv & 0x8800))
        break; /* Reset complete, device not Power Down. */
}

/* Read PHY ID. */
id1 = read_PHY (PHY_REG_IDR1);
id2 = read_PHY (PHY_REG_IDR2); //the 4 LSB may vary depending on the silicon rev num

/* Check if this is a LAN8720_ID PHY. */
if (((id1 << 16) | (id2 & 0xFFFF)) == LAN8720_ID)
{
    /* Configure the PHY device */
#if defined (_10MBIT_)
    /* Connect at 10Mbit */
    write_PHY (PHY_REG_BMCR, PHY_FULLD_10M);
#elif defined (_100MBIT_)
    /* Connect at 100Mbit */
    write_PHY (PHY_REG_BMCR, PHY_FULLD_100M);
#else
    /* Use autonegotiation about the link speed. */
    write_PHY (PHY_REG_BMCR, PHY_AUTO_NEG);
    /* Wait to complete Auto_Negotiation. */
    for (tout = 0; tout < 0x100000; tout++)
    {
        regv = read_PHY (PHY_REG_BMSR);
        if (regv & 0x0020)
            break; /* Autonegotiation Complete. */
    }

```

```
}

#endif

}

/* Check the link status. */

for (tout = 0; tout < 0x10000; tout++)

{

    regv = read_PHY (PHY_REG_BMSR); //LAN8720_ID

    if (regv & 0x0004) //LAN8720_ID

        break; /* Link is on. */

}

regv = read_PHY (PHY_REG_STS); //LAN8720

/* Configure Full/Half Duplex mode. */

if (regv & 0x0010)

{

    /* Full duplex is enabled. */

    LPC_EMAC->MAC2 |= MAC2_FULL_DUP;

    LPC_EMAC->Command |= CR_FULL_DUP;

    LPC_EMAC->IPGT = IPGT_FULL_DUP;

}

else

{

    /* Half duplex mode. */

    LPC_EMAC->IPGT = IPGT_HALF_DUP;

}

/* Configure 100MBit/10MBit mode. */

if (regv & 0x0008)

{

    /* 100MBit mode. */

    LPC_EMAC->SUPP = SUPP_SPEED;

}

else

{

    /* 10MBit mode. */

    LPC_EMAC->SUPP = 0;

}
```

```
/* Set the Ethernet MAC Address registers */
LPC_EMAC->SA0 = ((U32)own_hw_adr[5] << 8) | (U32)own_hw_adr[4];
LPC_EMAC->SA1 = ((U32)own_hw_adr[3] << 8) | (U32)own_hw_adr[2];
LPC_EMAC->SA2 = ((U32)own_hw_adr[1] << 8) | (U32)own_hw_adr[0];

/* Initialize Tx and Rx DMA Descriptors */
rx_descr_init ();
tx_descr_init ();

/* Receive Broadcast, Multicast and Perfect Match Packets */
LPC_EMAC->RxFilterCtrl = RFC_MCAST_EN | RFC_BCAST_EN | RFC_PERFECT_EN;

/* Enable EMAC interrupts. */
LPC_EMAC->IntEnable = INT_RX_DONE | INT_TX_DONE;

/* Reset all interrupts */
LPC_EMAC->IntClear = 0xFFFF;

/* Enable receive and transmit mode of MAC Ethernet core */
LPC_EMAC->Command |= (CR_RX_EN | CR_TX_EN);
LPC_EMAC->MAC1 |= MAC1_REC_EN;
}

/*----- int_enable_eth -----*/
void int_enable_eth (void) {
    /* Ethernet Interrupt Enable function. */
    NVIC_EnableIRQ(ENET IRQn);
}

/*----- int_disable_eth -----*/
void int_disable_eth (void) {
    /* Ethernet Interrupt Disable function. */
    NVIC_DisableIRQ(ENET IRQn);
}
```

```
/*----- send_frame -----*/  
  
void send_frame (OS_FRAME *frame) {  
    /* Send frame to EMAC ethernet controller */  
    U32 idx,len;  
    U32 *sp,*dp;  
  
    idx = LPC_EMAC->TxProduceIndex;  
    sp = (U32 *)&frame->data[0];  
    dp = (U32 *)Tx_Desc[idx].Packet;  
  
    /* Copy frame data to EMAC packet buffers. */  
    for (len = (frame->length + 3) >> 2; len; len--) {  
        *dp++ = *sp++;  
    }  
  
    Tx_Desc[idx].Ctrl = (frame->length-1) | (TCTRL_INT | TCTRL_LAST);  
  
    /* Start frame transmission. */  
    if (++idx == NUM_TX_FRAG) idx = 0;  
    LPC_EMAC->TxProduceIndex = idx;  
}  
  
/*----- interrupt_ethernet -----*/  
  
void ENET_IRQHandler (void) {  
    /* EMAC Ethernet Controller Interrupt function. */  
    OS_FRAME *frame;  
    U32 idx,int_stat,RxLen,info;  
    U32 *sp,*dp;  
  
    while ((int_stat = (LPC_EMAC->IntStatus & LPC_EMAC->IntEnable)) != 0) {  
        LPC_EMAC->IntClear = int_stat;  
        if (int_stat & INT_RX_DONE) {  
            /* Packet received, check if packet is valid. */  
            idx = LPC_EMAC->RxConsumeIndex;  
            while (idx != LPC_EMAC->RxProduceIndex) {
```

```
    info = Rx_Stat[idx].Info;
    if (!(info & RINFO_LAST_FLAG)) {
        goto rel;
    }

    RxLen = (info & RINFO_SIZE) - 3;
    if (RxLen > ETH_MTU || (info & RINFO_ERR_MASK)) {
        /* Invalid frame, ignore it and free buffer. */
        goto rel;
    }

    /* Flag 0x80000000 to skip sys_error() call when out of memory. */
    frame = alloc_mem (RxLen | 0x80000000);
    /* if 'alloc_mem()' has failed, ignore this packet. */
    if (frame != NULL) {
        dp = (U32 *)&frame->data[0];
        sp = (U32 *)Rx_Desc[idx].Packet;
        for (RxLen = (RxLen + 3) >> 2; RxLen; RxLen--) {
            *dp++ = *sp++;
        }
        put_in_queue (frame);
    }
    rel:   if (++idx == NUM_RX_FRAG) idx = 0;
    /* Release frame from EMAC buffer. */
    LPC_EMAC->RxConsumeIndex = idx;
}
}

if (int_stat & INT_TX_DONE) {
    /* Frame transmit completed. */
}
}

/*----- rx_descr_init -----*/
static void rx_descr_init (void) {
    /* Initialize Receive Descriptor and Status array. */
    U32 i;
```

```
for (i = 0; i < NUM_RX_FRAG; i++) {  
    Rx_Desc[i].Packet = (U32)&rx_buf[i];  
    Rx_Desc[i].Ctrl = RCTRL_INT | (ETH_FRAG_SIZE-1);  
    Rx_Stat[i].Info = 0;  
    Rx_Stat[i].HashCRC = 0;  
}  
  
/* Set EMAC Receive Descriptor Registers. */  
LPC_EMAC->RxDescriptor = (U32)&Rx_Desc[0];  
LPC_EMAC->RxStatus = (U32)&Rx_Stat[0];  
LPC_EMAC->RxDescriptorNumber = NUM_RX_FRAG-1;  
  
/* Rx Descriptors Point to 0 */  
LPC_EMAC->RxConsumeIndex = 0;  
}  
  
/*----- tx_descr_init -----*/  
  
static void tx_descr_init (void) {  
    /* Initialize Transmit Descriptor and Status array. */  
    U32 i;  
  
    for (i = 0; i < NUM_TX_FRAG; i++) {  
        Tx_Desc[i].Packet = (U32)&tx_buf[i];  
        Tx_Desc[i].Ctrl = 0;  
        Tx_Stat[i].Info = 0;  
    }  
  
    /* Set EMAC Transmit Descriptor Registers. */  
    LPC_EMAC->TxDescriptor = (U32)&Tx_Desc[0];  
    LPC_EMAC->TxStatus = (U32)&Tx_Stat[0];  
    LPC_EMAC->TxDescriptorNumber = NUM_TX_FRAG-1;  
  
    /* Tx Descriptors Point to 0 */  
    LPC_EMAC->TxProduceIndex = 0;  
}
```

```
/*----- output_Mdio -----*/  
  
#define delay()      __nop(); __nop(); __nop();  
  
//static void output_Mdio (U32 val, U32 n) {  
//  /* Output a value to the MII PHY management interface. */  
  
//  for (val <= (32 - n); n; val <= 1, n--) {  
//    if (val & 0x80000000) {  
//      LPC_GPIO2->FIOSET = MDIO;  
//    }  
//    else {  
//      LPC_GPIO2->FIOCLR = MDIO;  
//    }  
//    delay ();  
//    LPC_GPIO2->FIOSET = MDC;  
//    delay ();  
//    LPC_GPIO2->FIOCLR = MDC;  
//  }  
//}  
  
/*----- turnaround_Mdio -----*/  
  
//static void turnaround_Mdio (void) {  
//  /* Turnaround MDO is tristated. */  
  
//  LPC_GPIO2->FIODIR &= ~MDIO;  
//  LPC_GPIO2->FIOSET = MDC;  
//  delay ();  
//  LPC_GPIO2->FIOCLR = MDC;  
//  delay ();  
//}  
  
/*----- input_Mdio -----*/  
  
//static U32 input_Mdio (void) {  
//  /* Input a value from the MII PHY management interface. */  
//  U32 i, val = 0;
```

```
// for (i = 0; i < 16; i++) {
//     val <= 1;
//     LPC_GPIO2->FIOSET = MDC;
//     delay ();
//     LPC_GPIO2->FIOCLR = MDC;
//     if (LPC_GPIO2->FIOPIN & MDIO) {
//         val |= 1;
//     }
// }
// return (val);
//}

/*----- write_PHY -----*/
static void write_PHY (U32 PhyReg, U16 Value) {
    /* Write a data 'Value' to PHY register 'PhyReg'. */
    U32 tout;

    /* Hardware MII Management for LPC176x devices. */ // LAN8720
    LPC_EMAC->MADR = LAN8720_DEF_ADR | PhyReg;    //The device and registernumber that will
    be accessed
    LPC_EMAC->MWTD = Value;                      //The value to write in the register

    /* Wait until operation completed */
    for (tout = 0; tout < MII_WR_TOUT; tout++) {
        if (((LPC_EMAC->MIND & MIND_BUSY) == 0) {
            break;
        }
    }
}

/*----- read_PHY -----*/
static U16 read_PHY (U32 PhyReg) {
    /* Read a PHY register 'PhyReg'. */
    U32 tout, val;
```

```
LPC_EMAC->MADR = LAN8720_DEF_ADR | PhyReg;  
LPC_EMAC->MCMD = MCMD_READ; //Set bit in the command register to execute a single  
read  
  
/* Wait until operation completed */  
  
for (tout = 0; tout < MII_RD_TOUT; tout++) {  
  
    if ((LPC_EMAC->MIND & MIND_BUSY) == 0) {  
  
        break;  
    }  
}  
  
LPC_EMAC->MCMD = 0;  
val = LPC_EMAC->MRDD;  
return (val);  
}  
  
/*-----  
 * end of file  
 *-----*/  
  
/*-----  
 *      Generated by FCARM FILE CONVERTER V2.41  
 *-----  
 *      Name:      WEB.C  
 *      Purpose:   HTTP Web page, generated by the user.  
 *      Note:      This is a generated file, do not modify !!!  
 *-----  
 *      This code is part of the RealView Run-Time Library.  
 *      Copyright (c) 2004-2013 ARM Germany GmbH. All rights reserved.  
 *-----*/  
  
#include <Net_Config.h>  
  
/* Last-Modified: Sun, 19 Jan 2020 16:53:00 GMT */  
const U32 FileMD = 1579452780;  
  
/*-----*/
```

```
static const U8 Web[2221] = {

/*-- File: index.cgi, 2221 bytes --*/

0x01,0x3C,0x21,0x44,0x4F,0x43,0x54,0x59,0x50,0x45,0x20,0x68,0x74,0x6D,0x6C,
0x3E,0x20,0x3C,0x68,0x74,0x6D,0x6C,0x3E,0x20,0x3C,0x68,0x65,0x61,0x64,0x3E,
0x20,0x3C,0x6D,0x65,0x74,0x61,0x20,0x63,0x6F,0x6E,0x74,0x65,0x6E,0x74,0x3D,
0x22,0x74,0x65,0x78,0x74,0x2F,0x68,0x74,0x6D,0x6C,0x3B,0x20,0x63,0x68,0x61,
0x72,0x73,0x65,0x74,0x3D,0x55,0x54,0x46,0x2D,0x38,0x22,0x20,0x68,0x74,0x74,
0x70,0x2D,0x65,0x71,0x75,0x69,0x76,0x3D,0x22,0x63,0x6F,0x6E,0x74,0x65,0x6E,
0x74,0x2D,0x74,0x79,0x70,0x65,0x22,0x3E,0x01,0x20,0x3C,0x6D,0x65,0x74,0x61,
0x20,0x68,0x74,0x74,0x70,0x2D,0x65,0x71,0x75,0x69,0x76,0x3D,0x22,0x72,0x65,
0x66,0x72,0x65,0x73,0x68,0x22,0x20,0x63,0x6F,0x6E,0x74,0x65,0x6E,0x74,0x3D,
0x22,0x32,0x3B,0x20,0x75,0x72,0x6C,0x3D,0x68,0x74,0x70,0x3A,0x2F,
0x2F,0x31,0x39,0x32,0x2E,0x31,0x36,0x38,0x2E,0x31,0x2E,0x31,0x32,0x30,0x2F,
0x69,0x6E,0x64,0x65,0x78,0x2E,0x63,0x67,0x69,0x22,0x3E,0x20,0x3C,0x74,0x69,
0x74,0x6C,0x65,0x3E,0x45,0x53,0x54,0x41,0x43,0x49,0x4F,0x4E,0x20,0x4D,0x45,
0x54,0x45,0x4F,0x52,0x4F,0x4C,0x4F,0x47,0x49,0x43,0x41,0x3C,0x2F,0x74,0x69,
0x74,0x6C,0x65,0x3E,0x01,0x20,0x3C,0x2F,0x68,0x65,0x61,0x64,0x3E,0x20,0x3C,
0x62,0x6F,0x64,0x79,0x20,0x73,0x74,0x79,0x6C,0x65,0x3D,0x22,0x62,0x61,0x63,
0x6B,0x67,0x72,0x6F,0x75,0x6E,0x64,0x2D,0x63,0x6F,0x6C,0x72,0x3A,0x72,
0x67,0x62,0x28,0x32,0x30,0x30,0x2C,0x32,0x30,0x30,0x2C,0x32,0x30,0x30,0x29,
0x3B,0x22,0x3E,0x20,0x3C,0x68,0x31,0x20,0x61,0x6C,0x69,0x67,0x6E,0x3D,0x22,
0x63,0x65,0x6E,0x74,0x65,0x72,0x22,0x3E,0x45,0x73,0x74,0x61,0x63,0x69,0x6F,
0x6E,0x20,0x6D,0x65,0x74,0x65,0x6F,0x72,0x6F,0x6C,0x6F,0x67,0x69,0x63,0x61,
0x3C,0x2F,0x68,0x31,0x3E,0x20,0x3C,0x62,0x72,0x20,0x2F,0x3E,0x01,0x20,0x3C,
0x74,0x61,0x62,0x6C,0x65,0x20,0x61,0x6C,0x69,0x67,0x6E,0x3D,0x22,0x63,0x65,
0x6E,0x74,0x65,0x72,0x22,0x20,0x62,0x6F,0x72,0x64,0x65,0x72,0x3D,0x22,0x31,
0x22,0x3E,0x20,0x3C,0x63,0x61,0x70,0x74,0x69,0x6F,0x6E,0x3E,0x44,0x61,0x74,
0x6F,0x73,0x20,0x6D,0x65,0x64,0x69,0x6F,0x73,0x20,0x61,0x63,0x74,0x61,
0x6C,0x65,0x73,0x3A,0x3C,0x2F,0x63,0x61,0x70,0x74,0x69,0x6F,0x6E,0x3E,0x20,
0x3C,0x74,0x72,0x3E,0x20,0x3C,0x74,0x64,0x3E,0x54,0x65,0x6D,0x70,0x65,0x72,
0x61,0x74,0x75,0x72,0x61,0x3A,0x3C,0x2F,0x74,0x64,0x64,0x3E,0x20,0x3C,0x74,0x64,
0x3E,0x02,0x74,0x20,0x22,0x20,0x25,0x66,0x01,0x20,0x3C,0x2F,0x74,0x64,0x3E,
0x20,0x3C,0x74,0x64,0x3E,0x56,0x65,0x6C,0x6F,0x63,0x69,0x64,0x61,0x64,0x20,
0x64,0x65,0x6C,0x20,0x76,0x69,0x65,0x6E,0x74,0x6F,0x3A,0x3C,0x2F,0x74,0x64,
0x3E,0x20,0x3C,0x74,0x64,0x3E,0x02,0x76,0x20,0x22,0x20,0x25,0x66,0x01,0x20,
0x3C,0x2F,0x74,0x64,0x3E,0x20,0x3C,0x74,0x72,0x3E,0x20,0x3C,0x74,0x72,
0x3E,0x20,0x3C,0x74,0x64,0x3E,0x48,0x75,0x6D,0x65,0x64,0x61,0x64,0x3A,0x3C,
```

0x2F, 0x74, 0x64, 0x3E, 0x20, 0x3C, 0x74, 0x64, 0x3E, 0x02, 0x68, 0x20, 0x22, 0x20, 0x25,
0x66, 0x01, 0x20, 0x3C, 0x2F, 0x74, 0x64, 0x3E, 0x20, 0x3C, 0x74, 0x64, 0x3E, 0x49, 0x6E,
0x64, 0x69, 0x63, 0x65, 0x20, 0x55, 0x56, 0x3A, 0x3C, 0x2F, 0x74, 0x64, 0x3E, 0x20, 0x3C,
0x74, 0x64, 0x3E, 0x02, 0x69, 0x20, 0x22, 0x20, 0x25, 0x66, 0x01, 0x20, 0x3C, 0x2F, 0x74,
0x64, 0x3E, 0x20, 0x3C, 0x2F, 0x74, 0x72, 0x3E, 0x20, 0x3C, 0x74, 0x72, 0x3E, 0x20, 0x3C,
0x74, 0x64, 0x3E, 0x50, 0x72, 0x65, 0x73, 0x69, 0x6F, 0x6E, 0x3A, 0x3C, 0x2F, 0x74, 0x64,
0x3E, 0x20, 0x3C, 0x74, 0x64, 0x3E, 0x02, 0x70, 0x20, 0x22, 0x20, 0x25, 0x66, 0x01, 0x20,
0x3C, 0x2F, 0x74, 0x64, 0x3E, 0x20, 0x3C, 0x74, 0x64, 0x3E, 0x42, 0x72, 0x69, 0x6C, 0x6C,
0x6F, 0x3A, 0x3C, 0x2F, 0x74, 0x64, 0x3E, 0x20, 0x3C, 0x74, 0x64, 0x3E, 0x02, 0x62, 0x20,
0x22, 0x20, 0x25, 0x66, 0x01, 0x20, 0x3C, 0x2F, 0x74, 0x64, 0x3E, 0x20, 0x3C, 0x2F, 0x74,
0x72, 0x3E, 0x20, 0x3C, 0x2F, 0x74, 0x61, 0x62, 0x6C, 0x65, 0x3E, 0x20, 0x3C, 0x74, 0x61,
0x62, 0x6C, 0x65, 0x20, 0x61, 0x6C, 0x69, 0x67, 0x6E, 0x3D, 0x22, 0x63, 0x65, 0x6E, 0x74,
0x65, 0x72, 0x22, 0x20, 0x62, 0x6F, 0x72, 0x64, 0x65, 0x72, 0x3D, 0x22, 0x31, 0x22, 0x3E,
0x20, 0x3C, 0x74, 0x72, 0x3E, 0x20, 0x3C, 0x74, 0x64, 0x3E, 0x41, 0x6C, 0x74, 0x69, 0x74,
0x75, 0x64, 0x3A, 0x3C, 0x2F, 0x74, 0x64, 0x3E, 0x20, 0x3C, 0x74, 0x64, 0x3E, 0x02, 0x61,
0x20, 0x22, 0x20, 0x25, 0x66, 0x01, 0x20, 0x3C, 0x2F, 0x74, 0x64, 0x3E, 0x20, 0x3C, 0x74,
0x64, 0x3E, 0x4C, 0x6F, 0x6E, 0x67, 0x69, 0x74, 0x75, 0x64, 0x3A, 0x3C, 0x2F, 0x74, 0x64,
0x3E, 0x20, 0x3C, 0x74, 0x64, 0x3E, 0x02, 0x78, 0x20, 0x22, 0x20, 0x25, 0x66, 0x01, 0x20,
0x3C, 0x2F, 0x74, 0x64, 0x3E, 0x20, 0x3C, 0x74, 0x64, 0x3E, 0x02, 0x79,
0x75, 0x64, 0x3A, 0x3C, 0x2F, 0x74, 0x64, 0x3E, 0x20, 0x3C, 0x74, 0x64, 0x3E, 0x02, 0x79,
0x20, 0x22, 0x20, 0x25, 0x66, 0x01, 0x20, 0x3C, 0x2F, 0x74, 0x64, 0x3E, 0x20, 0x3C, 0x2F,
0x74, 0x72, 0x3E, 0x20, 0x3C, 0x2F, 0x74, 0x61, 0x62, 0x6C, 0x65, 0x3E, 0x20, 0x3C, 0x62,
0x72, 0x3E, 0x3C, 0x2F, 0x62, 0x72, 0x3E, 0x20, 0x3C, 0x74, 0x61, 0x62, 0x6C, 0x65, 0x20,
0x61, 0x6C, 0x69, 0x67, 0x6E, 0x3D, 0x22, 0x63, 0x65, 0x6E, 0x74, 0x65, 0x72, 0x22, 0x20,
0x62, 0x6F, 0x72, 0x64, 0x65, 0x72, 0x3D, 0x22, 0x31, 0x22, 0x3E, 0x20, 0x3C, 0x63, 0x61,
0x70, 0x74, 0x69, 0x6F, 0x6E, 0x3E, 0x48, 0x6F, 0x72, 0x61, 0x20, 0x64, 0x65, 0x20, 0x6C,
0x61, 0x20, 0x75, 0x6C, 0x74, 0x69, 0x6D, 0x61, 0x20, 0x6D, 0x75, 0x65, 0x73, 0x74, 0x72,
0x61, 0x3A, 0x3C, 0x2F, 0x63, 0x61, 0x70, 0x74, 0x69, 0x6F, 0x6E, 0x3E, 0x01, 0x20, 0x3C,
0x74, 0x72, 0x3E, 0x20, 0x3C, 0x74, 0x64, 0x3E, 0x41, 0x6E, 0x79, 0x6F, 0x3A, 0x3C, 0x2F,
0x74, 0x64, 0x3E, 0x20, 0x3C, 0x74, 0x64, 0x3E, 0x02, 0x41, 0x20, 0x22, 0x20, 0x25, 0x64,
0x01, 0x20, 0x3C, 0x2F, 0x74, 0x64, 0x3E, 0x20, 0x3C, 0x74, 0x64, 0x3E, 0x4D, 0x65, 0x73,
0x3A, 0x3C, 0x2F, 0x74, 0x64, 0x3E, 0x20, 0x3C, 0x74, 0x64, 0x3E, 0x02, 0x4D, 0x20, 0x22,
0x20, 0x25, 0x64, 0x01, 0x20, 0x3C, 0x2F, 0x74, 0x64, 0x3E, 0x20, 0x3C, 0x74, 0x64, 0x3E,
0x44, 0x69, 0x61, 0x3A, 0x3C, 0x2F, 0x74, 0x64, 0x3E, 0x20, 0x3C, 0x74, 0x64, 0x3E, 0x02,
0x44, 0x20, 0x22, 0x20, 0x25, 0x64, 0x01, 0x20, 0x3C, 0x2F, 0x74, 0x64, 0x3E, 0x20, 0x3C,
0x74, 0x72, 0x3E, 0x20, 0x3C, 0x74, 0x64, 0x3E, 0x48, 0x6F, 0x72, 0x61, 0x3A, 0x3C, 0x2F,
0x74, 0x64, 0x3E, 0x20, 0x3C, 0x74, 0x64, 0x3E, 0x02, 0x48, 0x20, 0x22, 0x20, 0x25, 0x64,
0x01, 0x20, 0x3C, 0x2F, 0x74, 0x64, 0x3E, 0x20, 0x3C, 0x74, 0x64, 0x3E, 0x4D, 0x69, 0x6E,
0x75, 0x74, 0x6F, 0x3A, 0x3C, 0x2F, 0x74, 0x64, 0x3E, 0x20, 0x3C, 0x74, 0x64, 0x3E, 0x02,

0x54, 0x20, 0x22, 0x20, 0x25, 0x64, 0x01, 0x20, 0x3C, 0x2F, 0x74, 0x64, 0x3E, 0x20, 0x3C,
0x74, 0x64, 0x3E, 0x53, 0x65, 0x67, 0x75, 0x6E, 0x64, 0x6F, 0x73, 0x3A, 0x3C, 0x2F, 0x74,
0x64, 0x3E, 0x20, 0x3C, 0x74, 0x64, 0x3E, 0x02, 0x53, 0x20, 0x22, 0x20, 0x25, 0x64, 0x01,
0x20, 0x3C, 0x2F, 0x74, 0x64, 0x3E, 0x20, 0x3C, 0x2F, 0x74, 0x72, 0x3E, 0x20, 0x3C, 0x2F,
0x74, 0x61, 0x62, 0x6C, 0x65, 0x3E, 0x20, 0x3C, 0x62, 0x72, 0x3E, 0x3C, 0x2F, 0x62, 0x72,
0x3E, 0x20, 0x3C, 0x62, 0x72, 0x3E, 0x3C, 0x2F, 0x62, 0x72, 0x3E, 0x20, 0x3C, 0x74, 0x61,
0x62, 0x6C, 0x65, 0x20, 0x73, 0x74, 0x69, 0x6C, 0x65, 0x3D, 0x22, 0x77, 0x69, 0x64, 0x74,
0x68, 0x3A, 0x20, 0x31, 0x30, 0x30, 0x25, 0x22, 0x20, 0x62, 0x6F, 0x72, 0x64, 0x65, 0x72,
0x3D, 0x22, 0x31, 0x22, 0x20, 0x61, 0x6C, 0x69, 0x67, 0x6E, 0x3D, 0x22, 0x63, 0x65, 0x6E,
0x74, 0x65, 0x72, 0x22, 0x3E, 0x20, 0x3C, 0x74, 0x62, 0x6F, 0x64, 0x79, 0x3E, 0x20, 0x3C,
0x74, 0x72, 0x3E, 0x20, 0x3C, 0x74, 0x64, 0x3E, 0x01, 0x20, 0x3C, 0x68, 0x31, 0x20, 0x73,
0x74, 0x79, 0x6C, 0x65, 0x3D, 0x22, 0x20, 0x74, 0x65, 0x78, 0x74, 0x2D, 0x61, 0x6C, 0x69,
0x67, 0x6E, 0x3A, 0x20, 0x63, 0x65, 0x6E, 0x74, 0x65, 0x72, 0x3B, 0x22, 0x3E, 0x20, 0x4D,
0x61, 0x67, 0x6E, 0x69, 0x74, 0x64, 0x65, 0x73, 0x20, 0x6D, 0x6F, 0x64, 0x69, 0x66,
0x69, 0x63, 0x61, 0x62, 0x6C, 0x65, 0x73, 0x3A, 0x20, 0x3C, 0x2F, 0x68, 0x31, 0x3E, 0x20,
0x3C, 0x66, 0x6F, 0x72, 0x6D, 0x20, 0x6D, 0x65, 0x74, 0x68, 0x6F, 0x64, 0x3D, 0x22, 0x47,
0x45, 0x54, 0x22, 0x20, 0x61, 0x63, 0x74, 0x69, 0x6F, 0x6E, 0x3D, 0x22, 0x69, 0x6E, 0x64,
0x65, 0x78, 0x2E, 0x63, 0x67, 0x69, 0x22, 0x3E, 0x01, 0x20, 0x3C, 0x62, 0x72, 0x3E, 0x20,
0x54, 0x65, 0x6D, 0x70, 0x65, 0x72, 0x61, 0x74, 0x75, 0x72, 0x61, 0x20, 0x6D, 0x69, 0x6E,
0x2E, 0x20, 0x3A, 0x20, 0x3C, 0x69, 0x6E, 0x70, 0x75, 0x74, 0x20, 0x73, 0x69, 0x7A, 0x65,
0x3D, 0x22, 0x31, 0x30, 0x20, 0x76, 0x61, 0x6C, 0x75, 0x65, 0x3D, 0x22, 0x2D, 0x31,
0x30, 0x22, 0x20, 0x6E, 0x61, 0x6D, 0x65, 0x3D, 0x22, 0x74, 0x6D, 0x69, 0x6E, 0x22, 0x20,
0x74, 0x79, 0x70, 0x65, 0x3D, 0x22, 0x74, 0x65, 0x78, 0x74, 0x22, 0x3E, 0x20, 0x3C, 0x62,
0x72, 0x3E, 0x20, 0x54, 0x65, 0x6D, 0x70, 0x65, 0x72, 0x61, 0x74, 0x75, 0x72, 0x61, 0x20,
0x6D, 0x61, 0x78, 0x2E, 0x20, 0x3A, 0x01, 0x20, 0x3C, 0x69, 0x6E, 0x70, 0x75, 0x74, 0x20,
0x73, 0x69, 0x7A, 0x65, 0x3D, 0x22, 0x31, 0x30, 0x22, 0x20, 0x76, 0x61, 0x6C, 0x75, 0x65,
0x3D, 0x22, 0x35, 0x30, 0x20, 0x6E, 0x61, 0x6D, 0x65, 0x3D, 0x22, 0x70, 0x6D, 0x69,
0x6E, 0x22, 0x20, 0x74, 0x79, 0x70, 0x65, 0x3D, 0x22, 0x74, 0x65, 0x78, 0x74, 0x22, 0x3E,
0x20, 0x3C, 0x62, 0x72, 0x3E, 0x20, 0x50, 0x72, 0x65, 0x73, 0x69, 0x6F, 0x6E, 0x20, 0x6D,
0x61, 0x78, 0x2E, 0x20, 0x3A, 0x01, 0x20, 0x3C, 0x69, 0x6E, 0x70, 0x75, 0x74, 0x20, 0x73,
0x69, 0x7A, 0x65, 0x3D, 0x22, 0x31, 0x30, 0x22, 0x20, 0x76, 0x61, 0x6C, 0x75, 0x65, 0x3D,
0x22, 0x35, 0x30, 0x20, 0x6E, 0x61, 0x6D, 0x65, 0x3D, 0x22, 0x70, 0x6D, 0x69,
0x6E, 0x22, 0x20, 0x74, 0x79, 0x70, 0x65, 0x3D, 0x22, 0x74, 0x65, 0x78, 0x74, 0x22,
0x3E, 0x20, 0x3C, 0x62, 0x72, 0x3E, 0x20, 0x53, 0x65, 0x67, 0x75, 0x6E, 0x64, 0x6F, 0x73,

```
0x20,0x65,0x6E,0x63,0x65,0x6E,0x64,0x69,0x64,0x6F,0x20,0x3A,0x01,0x20,0x3C,
0x69,0x6E,0x70,0x75,0x74,0x20,0x73,0x69,0x7A,0x65,0x3D,0x22,0x31,0x30,0x22,
0x20,0x76,0x61,0x6C,0x75,0x65,0x3D,0x22,0x31,0x30,0x22,0x20,0x6E,0x61,0x6D,
0x65,0x3D,0x22,0x62,0x73,0x65,0x63,0x22,0x20,0x74,0x79,0x70,0x65,0x3D,0x22,
0x74,0x65,0x78,0x74,0x22,0x3E,0x01,0x20,0x3C,0x62,0x72,0x3E,0x20,0x3C,0x69,
0x6E,0x70,0x75,0x74,0x20,0x76,0x61,0x6C,0x75,0x65,0x3D,0x22,0x73,0x69,0x22,
0x20,0x74,0x79,0x70,0x65,0x3D,0x22,0x72,0x61,0x64,0x69,0x6F,0x22,0x20,0x6E,
0x61,0x6D,0x65,0x3D,0x22,0x76,0x61,0x72,0x74,0x22,0x3E,0x20,0x54,0x65,0x6D,
0x70,0x65,0x72,0x61,0x74,0x75,0x72,0x61,0x3C,0x62,0x72,0x3E,0x01,0x20,0x3C,
0x69,0x6E,0x70,0x75,0x74,0x20,0x76,0x61,0x6C,0x75,0x65,0x3D,0x22,0x73,0x69,
0x22,0x20,0x74,0x79,0x70,0x65,0x3D,0x22,0x72,0x61,0x64,0x69,0x6F,0x22,0x20,
0x6E,0x61,0x6D,0x65,0x3D,0x22,0x76,0x61,0x72,0x70,0x22,0x3E,0x20,0x50,0x72,
0x65,0x73,0x69,0x6F,0x6E,0x3C,0x62,0x72,0x3E,0x20,0x3C,0x62,0x72,0x3E,0x20,
0x3C,0x69,0x6E,0x70,0x74,0x20,0x76,0x61,0x6C,0x75,0x65,0x3D,0x22,0x45,
0x6E,0x76,0x69,0x61,0x72,0x22,0x20,0x74,0x79,0x70,0x65,0x3D,0x22,0x73,0x75,
0x62,0x6D,0x69,0x74,0x22,0x3E,0x20,0x3C,0x2F,0x66,0x6F,0x72,0x6D,0x3E,0x20,
0x3C,0x2F,0x74,0x64,0x3E,0x01,0x20,0x3C,0x2F,0x74,0x72,0x3E,0x20,0x3C,0x2F,
0x74,0x62,0x6F,0x64,0x79,0x3E,0x20,0x3C,0x2F,0x74,0x61,0x62,0x6C,0x65,0x3E,
0x20,0x3C,0x62,0x72,0x3E,0x3C,0x2F,0x62,0x72,0x3E,0x20,0x3C,0x62,0x72,0x3E,
0x3C,0x2F,0x62,0x72,0x3E,0x20,0x3C,0x62,0x72,0x3E,0x20,0x3C,0x62,0x72,0x3E,
0x20,0x3C,0x62,0x72,0x3E,0x3C,0x2F,0x62,0x72,0x3E,0x20,0x3C,0x74,0x61,0x62,
0x6C,0x65,0x20,0x61,0x6C,0x69,0x67,0x6E,0x3D,0x22,0x63,0x65,0x6E,0x74,0x65,
0x72,0x22,0x3E,0x20,0x3C,0x74,0x72,0x3E,0x01,0x20,0x3C,0x74,0x64,0x3E,0x41,
0x75,0x74,0x6F,0x72,0x3A,0x20,0x41,0x6C,0x62,0x65,0x72,0x74,0x6F,0x20,0x50,
0x61,0x6C,0x6F,0x6D,0x6F,0x20,0x41,0x6C,0x6F,0x6E,0x73,0x6F,0x2E,0x3C,0x2F,
0x74,0x64,0x3E,0x20,0x3C,0x74,0x64,0x3E,0x53,0x69,0x73,0x74,0x65,0x6D,0x61,
0x73,0x20,0x45,0x6C,0x65,0x63,0x74,0x72,0x6F,0x6E,0x69,0x63,0x6F,0x73,0x20,
0x44,0x69,0x67,0x69,0x74,0x61,0x6C,0x65,0x73,0x20,0x41,0x76,0x61,0x6E,0x7A,
0x61,0x64,0x6F,0x73,0x2E,0x3C,0x2F,0x74,0x64,0x3E,0x01,0x20,0x3C,0x74,0x64,
0x3E,0x55,0x6E,0x69,0x76,0x65,0x72,0x73,0x69,0x64,0x61,0x64,0x20,0x64,0x65,
0x20,0x41,0x6C,0x63,0x61,0x6C,0x61,0x20,0x2D,0x20,0x45,0x73,0x63,0x75,0x65,
0x6C,0x61,0x20,0x70,0x6F,0x6C,0x69,0x74,0x65,0x63,0x6E,0x69,0x63,0x61,0x20,
0x73,0x75,0x70,0x65,0x72,0x69,0x6F,0x72,0x2E,0x3C,0x2F,0x74,0x64,0x3E,0x20,
0x3C,0x2F,0x74,0x72,0x3E,0x20,0x3C,0x2F,0x74,0x61,0x62,0x6C,0x65,0x3E,0x20,
0x3C,0x2F,0x62,0x6F,0x64,0x79,0x3E,0x20,0x3C,0x2F,0x68,0x74,0x6D,0x6C,0x3E,
0x00,
};

/*-----*/
```

```
const HTTP_FILE FileTab[2] = {  
    { 0xECD3FEF0, &Web[0] },  
    { 0x00000000, &Web[2221] }  
};  
  
/*-----  
 * end of file  
-----*/  
  
/**-----  
-----/  
// @filename RTC.c //  
// @version 0.00 //  
//  
// @author Alberto Palomo Alonso //  
//  
//  
// @brief Código fuente del configurador y manejador del RTC.  
//  
//  
// @category Opcional.  
//  
//  
// @map @include  
//  
// @variables //  
// @function //  
// @HANDLER //  
// @end //
```

```
//  
//  
//  
//  
//-----//  
//  
//  
  
//          @include      Estos son los archivos utilizados con el código del  
RTC.                                         //  
  
//  
//  
//-----//  
-----**/  
  
#ifndef RTC  
  
#define RTC  
  
#include "RTC.h"  
  
#endif  
  
#ifndef STRING  
  
#define STRING  
  
#include <string.h>  
  
#endif  
  
#ifndef STDIO  
  
#define STDIO  
  
#include     <stdio.h>  
  
#endif  
  
#ifndef SYSTEMSYMBOLS  
  
#define SYSTEMSYMBOLS  
  
#include     "Systemsymbols.h"  
  
#endif  
  
/*-----//  
-----//  
  
//  
//  
  
//          @variables      Variables del fichero.  
//  
//  
//
```

```

//-----
-----**/
uint8_t Clock[23];

extern Counters_t      *      COUNTERS;

/*-----*/
-----//



//



//



//



//      @function      __configuraRTC__()

//



//



//      @brief  Función que configura el RTC como un contador que interrumpe cada
segundo.          //



//



//      @Tiempo Puntero a variable donde se almacena el tiempo actual.
//



//



//



//-----*/
-----**/


void __configuraRTC__( void )
{
    LPC_SC->PCONP      |= RTCMASK;           // Activo el RTC.

    LPC_RTC->CCR        = CALIBRACION_RTC;   // Calibro el RTC.

    LPC_RTC->CALIBRATION = CALIBRATION_VALUE;

    LPC_RTC->CCR        = 0x1;

    LPC_RTC->CIIR       |= INT_SEGUNDOS;     // Interrupción del
RTC cada segundo.

    LPC_RTC->YEAR        = 2020;              // Configuro el
registro

    LPC_RTC->MONTH       = 1;                  // que tiene
en cuenta

    LPC_RTC->DOM         = 1;                  // los dias,
meses,

    LPC_RTC->HOUR        = 0;                  // minutos y
segundos

    LPC_RTC->MIN         = 0;                  // del RTC.

    LPC_RTC->SEC         = 0;                  //



    NVIC_EnableIRQ( RTC IRQn);                 // Habilito la
interrumpción del RTC.
}

```

Alberto Palomo Alonso.

```
/*
-----//  
//          @filename      Timers.c  
//  
//          @version       7.00  
//  
//          @author        Alberto Palomo Alonso  
//  
//  
//          @brief         Código que configura y programa los manejadores de los  
// timers.  
//  
//          @category      Principal.  
//  
//  
//          @map           @include  
//  
//          @variables  
//  
//          @funcion  
//  
//          @end  
//  
//  
//  
//  
//          @include  
//-----//  
//  
//          @include  
//          Estos son los archivos utilizados para los timers.  
//  
//  
//
```

```
-----*/  
#ifndef TIMERS  
#define TIMERS  
#include      "Timers.h"  
#endif  
/*-----*/  
//  
//  
  
//  
// @variables      Variables del fichero.  
//  
//  
//  
//-----*/  
uint8_t          TIM0_ticks      =      0;  
uint8_t          Timer2_MODO     =      MODO_SALIDA;  
uint32_t         CAP11_BUFF     =      0;  
// Contador.  
uint16_t         contadorLUZ    =      0;  
// Externos.  
extern uint8_t   __brilloAuto;  
extern uint8_t   __brilloFade;  
extern uint8_t   YaPuedesMedir;  
extern Counters_t * COUNTERS;  
extern misDatos_t * DATOS;  
extern actualizador_t * ACTUALIZADOR;  
extern uint8_t   * AUDIO;  
extern uint8_t   * CAPcont;  
extern modificables_t MODIFICABLES;  
/*-----*/  
//  
//  
  
//  
// @function      __configuraSysTick__()  
//
```

```

//  

//  

//          @brief      Configura el systick para desbordar cada 100 ms.  

//  

//  

//  

//-----**/  

void __configuraSysTick__()  

{  

    SysTick->LOAD = (SystemCoreClock / FREQ_OVERFLOW_SYSTICK) - 1;      //      SysTick  

configurado a desbordar cada 100 ms para TcpNet.  

    SysTick->CTRL = MASCARA_CTRL_SYSTICK;  

//      Fcpu como clock y no activo la interrupción del SysTickTimer.  

    SysTick_Config( SystemCoreClock / FREQ_OVERFLOW_SYSTICK );  

}  

/*-----**/  

//  

//  

//  

//-----**/  

//  

//  

//  

//          @function      __configuraTimer0__()  

//  

//  

//  

//          @brief      Configura el Timer0 para nterrumpir cada Ts0 segundos.  

//  

//  

//  

//  

//-----**/  

void __configuraTimer0__()  

{  

    LPC_SC->PCONP |= 0x1 << 22 | 0x1 << 23 | 1 << 16;      //      Todos los  

timer.  

    LPC_SC->PCONP |= TIMER0_BIT;                                //      Activo  

el módulo del timer 0.  

    LPC_TIM0->MCR = TIMER0_MCR_MASK;                            //      Activo  

el ISR y reseteo TC.  

    LPC_TIM0->PR = 0;                                         //  

Sin prescaler.  

    LPC_TIM0->TCR |= ACTIVAR_TIMER;                            //      Activo  

el timer.  

    LPC_TIM0->MR0 = Ftick * Ts0 - 1;                          //      Cargo  

para que interrumpa cada 0.5s.
}

```

Alberto Palomo Alonso.

```
//  
//  
//          @brief      Manejador de la interrupción del SysTick. Cada 100 ms se  
realizan acciones.  
//  
//  
//  
//-----**/  
void SysTick_Handler()  
{  
  
    timer_tick();  
  
    if (contadorLUZ < FREQ_OVERFLOW_SYSTICK *  
(MODIFICABLES.TiempoBrillo))  
    {  
  
        contadorLUZ++;  
    }  
  
    else  
    {  
  
        if (_brilloFade)  
// Si pasan 60s y el brillo automático está desactivado...  
        {  
  
            _brilloAuto = 0;  
            _brilloFade = 0;  
  
            none , modificaPulso( PWM6 , MODO_CICLO , 1 ,  
            none , none , none ); // Apago la pantalla.  
        }  
    }  
}  
/*-----**/  
//  
//  
//  
//          @brief      Manejador de la interrupción del Timer0. Reanima el  
muestreo de los sensores.  
//  
//  
//
```

```

//-----
-----**/
//      Bloque 1:      Apoyo del timer 1:
//
//                                              Temperatura + Humedad + Vel.Viento.

static void      _subAnemoTempe()
{
    LPC_TIM0->IR =     LPC_TIM0->IR;                                // Borro
interrupción.

    if( ! (TIM0_ticks % (uint8_t)CsCAP) )                                // Si toca
muestrear captures...

    {
        LPC_TIM1->CCR |=     CCR_MASCARA_EN;                          // Genera
interrupción el CAP1.0, ojo que se mata así en el timer 1.

        LPC_TIM1->CCR |=     OW_CCR_MASCARA_EN;                        // Genera
interrupción el CAP1.1, ojo que se mata así en el timer 1.

        mideTemperatura();                                            // Le digo
a la placa que lanze la señal de request.

        medirBMP();                                                 // Leo el
sensor de presión atmosférica.

        if ( !ACTUALIZADOR->AnemometroRev && YaPuedesMedir)          // Si el
actualizador está a 0 (Es decir, no hay datos capturados).

        {
            DATOS->VelViento = 0;                                     // No hay viento.

            ACTUALIZADOR->Anemometro = 1;                            // Ya está
medido, es 0 m/s.

        }
        ACTUALIZADOR->AnemometroRev = 0;                            // Digo que ya he
medido.

        ACTUALIZADOR->TempRev =      1;

    }

}

//      Bloque 2:      ADC burst:
//
//                                              UVA + LDR.

static void      _subBurst()
{
    if( ! (TIM0_ticks % (uint8_t)CsADC) )                                // LDR +
UVA van el BURST.

    {
        if( ACTUALIZADOR->LDRrev && YaPuedesMedir )                  // Es
bloqueable por el audio.

        {
            LPC_SC->PCONP |=     PCONP_ADC_ON; // Enciendo el ADC.

            ACTUALIZADOR->LDRrev =      0;           // Aviso
que no he medido aún.

```

```

LPC_ADC->ADCR             &=      ~ADC_START;    //      Ojito
que es modo ráfaga, no hay start.

LPC_ADC->ADCR             |=      BRUST_PIN;     //      Ráfaga.

}

}

//      Actualizo el servo.

void __subServo(      void      )
{
    if (      !MODIFICABLES.Var_medida      )

    {

        if (DATOS->Temperatura >= MODIFICABLES.Max_servo_t)

        {

            modificaPulso      (      PWM2,      MODO_SERVO      ,      none
,      180      ,      MINIMO_SERVO      ,      MAXIMO_SERVO      );
            if (ACTUALIZADOR->Audiorev)
            {

                ACTUALIZADOR->Audiorev = 0;
                __configuraTono__();
                activarDac();
            }

        }

        if (DATOS->Temperatura <= MODIFICABLES.Min_servo_t)

        {

            modificaPulso      (      PWM2,      MODO_SERVO      ,      none
,      0      ,      MINIMO_SERVO      ,      MAXIMO_SERVO      );
            if (ACTUALIZADOR->Audiorev)
            {

                ACTUALIZADOR->Audiorev = 0;
                __configuraAudio__();
                activarDac();
            }

        }

        modificaPulso      (      PWM2,      MODO_SERVO      ,      none
(180*(DATOS->Temperatura - MIN_TEMP)/(MAX_TEMP - MIN_TEMP))
MINIMO_SERVO      ,      MAXIMO_SERVO      );
    }

    else

    {

        if (DATOS->Presion >= MODIFICABLES.Max_servo_p)
        {

```

Alberto Palomo Alonso.

```
//  
//  
// @ref Ir a Anemómetro.c (comparte con OneWire.c)  
//  
//  
//  
//-----**/  
void TIMER1_IRQHandler()  
{  
    uint8_t SWART = (uint8_t)(LPC_TIM1->IR);  
  
    if (SWART & CAP10_IR)  
    {  
        mideAnemometro();  
    }  
    if (SWART & MR1_IR)  
    {  
        desactivarDAC();  
    }  
    LPC_TIM1->IR = LPC_TIM1->IR; // No pierdo nada en  
asegurarme que se cierra el timer.  
}  
/*-----**/  
//  
//  
//  
// @HANDLER TIMER2_IRQHandler()  
//  
//  
//  
// @brief N/A  
//  
//  
//-----**/  
void TIMER2_IRQHandler()  
{  
    // NO USADO.
```

```
}

/**-----//



// @HANDLER      TIMER3_IRQHandler()
//



// @brief        Timer de apoyo para el monohilo.
//



// @
// USADO POR EL MONOHILO.

void TIMER3_IRQHandler()
{
    // NO USADO.

}

/**-----//



// @end        ENDFILE.

//



// @
//



*****Copyright
(c) *****



**



**          http://www.powermcu.com
```

```
**  
-----File Info-----  
-----  
** File name: diskio.c  
** Descriptions: The FATFS Diskio  
**  
**-----  
-----  
** Created by: AVRman  
** Created date: 2010-11-4  
** Version: v1.0  
** Descriptions: The original version  
**  
**-----  
-----  
** Modified by:  
** Modified date:  
** Version:  
** Descriptions:  
**  
*****/*  
*****/  
  
/* Includes -----*/  
#include "diskio.h"  
#include "SPI_MSD_Driver.h"  
  
/* Private variables -----*/  
extern MSD_CARDINFO CardInfo;  
  
DSTATUS disk_initialize (  
    BYTE drv /* Physical drive number (0..) */  
)  
{  
    int Status;  
  
    switch (drv)  
    {  
        case 0 :  
    }  
}
```

```
    Status = MSD_Init();

    if(Status == 0)
        return 0;
    else
        return STA_NOINIT;

    case 1 :
        return STA_NOINIT;

    case 2 :
        return STA_NOINIT;
    }

    return STA_NOINIT;
}

/*-----*/
/* Return Disk Status */

DSTATUS disk_status (
    BYTE drv             /* Physical drive nmuber (0..) */
)
{
    switch (drv)
    {
        case 0 :

            /* translate the reslut code here */

            return 0;

        case 1 :

            /* translate the reslut code here */

            return 0;

        case 2 :
```

```
/* translate the reslut code here */

return 0;

default:

break;

}

return STA_NOINIT;
}

/*
 * Read Sector(s)
 */

DRESULT disk_read (
    BYTE drv,           /* Physical drive nmuber (0..) */
    BYTE *buff,          /* Data buffer to store read data */
    DWORD sector,       /* Sector address (LBA) */
    BYTE count           /* Number of sectors to read (1..255) */
)

{
    int Status;
    if( !count )
    {

        return RES_PARERR; /* count<=0 */
    }

switch (drv)
{

case 0:
    if(count==1)           /* 1,sector*/
    {
        Status = MSD_ReadSingleBlock( sector ,buff );
    }
    else                   /* >1,sector*/

```

```
{  
    Status = MSD_ReadMultiBlock( sector , buff ,count);  
}  
  
if(Status == 0)  
{  
    return RES_OK;  
}  
  
else  
{  
    return RES_ERROR;  
}  
  
  
case 1:  
    break;  
  
case 2:  
    break;  
  
default:  
    break;  
  
}  
  
return RES_ERROR;  
}  
  
  
/*-----*/  
/* Write Sector(s) */  
  
#if _READONLY == 0  
DRESULT disk_write (   
    BYTE drv, /* Physical drive nmuber (0..) */  
    const BYTE *buff, /* Data to be written */  
    DWORD sector, /* Sector address (LBA) */  
    BYTE count /* Number of sectors to write (1..255) */  
)  
{
```

```
int Status;
if( !count )
{
    return RES_PARERR; /* count<=0 Error */
}

switch (drv)
{
case 0:
    if(count==1)           /* 1, sector */
    {
        Status = MSD_WriteSingleBlock( sector , (uint8_t *)(&buff[0]) );
    }
    else                   /* >1, sectors */
    {
        Status = MSD_WriteMultiBlock( sector , (uint8_t *)(&buff[0]) , count );
    }
    if(Status == 0)
    {
        return RES_OK;
    }
    else
    {
        return RES_ERROR;
    }
case 2:
    break;
default :
    break;
}
return RES_ERROR;
}

#endif /* _READONLY */

/*-----*/
/* Miscellaneous Functions */
```

```
DRESULT disk_ioctl (
    BYTE drv,             /* Physical drive nmuber (0..) */
    BYTE ctrl,            /* Control code */
    void *buff            /* Buffer to send/receive control data */

)

{
    if (drv)
    {

        return RES_PARERR; /* ÅÅÌ²Ù×÷£¬·ñÔò·µ»Ø²ÎÊý'íîó */

    }

    MSD_GetCardInfo(&CardInfo);

    switch (ctrl)
    {

        case CTRL_SYNC :

            return RES_OK;

        case GET_SECTOR_COUNT :

            *(DWORD*)buff = CardInfo.Capacity/CardInfo.BlockSize;
            return RES_OK;

        case GET_BLOCK_SIZE :

            *(WORD*)buff = CardInfo.BlockSize;
            return RES_OK;

        case CTRL_POWER :

            break;

        case CTRL_LOCK :

            break;

        case CTRL_EJECT :

            break;

        /* MMC/SDC command */
        case MMC_GET_TYPE :

            break;
    }
}
```

```
case MMC_GET_CSD :
    break;

case MMC_GET_CID :
    break;

case MMC_GET_OCR :
    break;

case MMC_GET_SDSTAT :
    break;
}

return RES_PARERR;
}

/* µÃµ½íÃµpCalendar, ñÊµµÃ½ ÁºÈØÉÚ, ÈÇDWORD get_fattime (void) Äæ±ä»» */
/*-----*/
/* User defined function to give a current time to fatfs module */
/* 31-25: Year(0-127 org.1980), 24-21: Month(1-12), 20-16: Day(1-31) */
/* 15-11: Hour(0-23), 10-5: Minute(0-59), 4-0: Second(0-29 *2) */
DWORD get_fattime (void)
{
    return 0;
}

***** END FILE *****

/*
FatFs - FAT file system module R0.08a          (C) ChaN, 2010
-----/
```

```
/ FatFs module is a generic FAT file system module for small embedded systems.  
/  
/ This is a free software that opened for education, research and commercial  
/ developments under license policy of following terms.  
/  
/ Copyright (C) 2010, ChaN, all right reserved.  
/  
/ * The FatFs module is a free software and there is NO WARRANTY.  
/ * No restriction on use. You can use, modify and redistribute it for  
/ personal, non-profit or commercial products UNDER YOUR RESPONSIBILITY.  
/ * Redistributions of source code must retain the above copyright notice.  
/  
/-----/  
/ Feb 26,'06 R0.00 Prototype.  
/  
/ Apr 29,'06 R0.01 First stable version.  
/  
/ Jun 01,'06 R0.02 Added FAT12 support.  
/  
/ Removed unbuffered mode.  
/  
/ Fixed a problem on small (<32M) partition.  
/ Jun 10,'06 R0.02a Added a configuration option (_FS_MINIMUM).  
/  
/ Sep 22,'06 R0.03 Added f_rename().  
/  
/ Changed option _FS_MINIMUM to _FS_MINIMIZE.  
/ Dec 11,'06 R0.03a Improved cluster scan algorithm to write files fast.  
/  
/ Fixed f_mkdir() creates incorrect directory on FAT32.  
/  
/ Feb 04,'07 R0.04 Supported multiple drive system.  
/  
/ Changed some interfaces for multiple drive system.  
/  
/ Changed f_mountdrv() to f_mount().  
/  
/ Added f_mkfs().  
/ Apr 01,'07 R0.04a Supported multiple partitions on a physical drive.  
/  
/ Added a capability of extending file size to f_lseek().  
/  
/ Added minimization level 3.  
/  
/ Fixed an endian sensitive code in f_mkfs().  
/ May 05,'07 R0.04b Added a configuration option _USE_NTFLAG.  
/  
/ Added FSInfo support.  
/  
/ Fixed DBCS name can result FR_INVALID_NAME.  
/  
/ Fixed short seek (<= csize) collapses the file object.  
/
```

Alberto Palomo Alonso.

```
/ Aug 25,'07 R0.05 Changed arguments of f_read(), f_write() and f_mkfs().  
/  
/ Fixed f_mkfs() on FAT32 creates incorrect FSInfo.  
/  
/ Fixed f_mkdir() on FAT32 creates incorrect directory.  
/  
/ Feb 03,'08 R0.05a Added f_truncate() and f_utime().  
/  
/ Fixed off by one error at FAT sub-type determination.  
/  
/ Fixed btr in f_read() can be mistruncated.  
/  
/ Fixed cached sector is not flushed when create and close without  
write.  
/  
/  
/ Apr 01,'08 R0.06 Added fputc(), fputs(), fprintf() and fgets().  
/  
/ Improved performance of f_lseek() on moving to the same or following  
cluster.  
/  
/  
/ Apr 01,'09 R0.07 Merged Tiny-FatFs as a buffer configuration option. (_FS_TINY)  
/  
/ Added long file name support.  
/  
/ Added multiple code page support.  
/  
/ Added re-entrancy for multitask operation.  
/  
/ Added auto cluster size selection to f_mkfs().  
/  
/ Added rewind option to f_readdir().  
/  
/ Changed result code of critical errors.  
/  
/ Renamed string functions to avoid name collision.  
/  
/ Apr 14,'09 R0.07a Separated out OS dependent code on reentrant cfg.  
/  
/ Added multiple sector size support.  
/  
/ Jun 21,'09 R0.07c Fixed f_unlink() can return FR_OK on error.  
/  
/ Fixed wrong cache control in f_lseek().  
/  
/ Added relative path feature.  
/  
/ Added f_chdir() and f_chdrive().  
/  
/ Added proper case conversion to extended char.  
/  
/ Nov 03,'09 R0.07e Separated out configuration options from ff.h to ffconf.h.  
/  
/ Fixed f_unlink() fails to remove a sub-dir on _FS_RPATH.  
/  
/ Fixed name matching error on the 13 char boundary.  
/  
/ Added a configuration option, _LFN_UNICODE.  
/  
/ Changed f_readdir() to return the SFN with always upper case on non-  
LFN cfg.  
/  
/  
/ May 15,'10 R0.08 Added a memory configuration option. (_USE_LFN = 3)  
/  
/ Added file lock feature. (_FS_SHARE)  
/  
/ Added fast seek feature. (_USE_FASTSEEK)  
/  
/ Changed some types on the API, XCHAR->TCHAR.  
/  
/ Changed fname member in the FILINFO structure on Unicode cfg.  
/  
/ String functions support UTF-8 encoding files on Unicode cfg.
```

Alberto Palomo Alonso.

```
/ Aug 16,'10 R0.08a Added f_getcwd(). (_FS_RPATH = 2)
/
    Added sector erase feature. (_USE_ERASE)
/
    Moved file lock semaphore table from fs object to the bss.
/
    Fixed a wrong directory entry is created on non-LFN cfg when the
given name contains ;'.
/
    Fixed f_mkfs() creates wrong FAT32 volume.

-----*/



#include "ff.h"                      /* FatFs configurations and declarations */
#include "diskio.h"                  /* Declarations of low level disk I/O functions */

-----


/*-----*
Module Private Definitions

-----*/



#if _FATFS != 8255
#error Wrong include file (ff.h).
#endif


/* Definitions on sector size */

#if _MAX_SS != 512 && _MAX_SS != 1024 && _MAX_SS != 2048 && _MAX_SS != 4096
#error Wrong sector size.
#endif

#if _MAX_SS != 512
#define SS(fs) ((fs)->ssize) /* Multiple sector size */
#else
#define SS(fs) 512U           /* Fixed sector size */
#endif


/* Reentrancy related */

#if _FS_REENTRANT
#if _USE_LFN == 1
#error Static LFN work area must not be used in re-entrant configuration.
#endif
#define ENTER_FF(fs)          { if (!lock_fs(fs)) return FR_TIMEOUT; }

```

```
#define LEAVE_FF(fs, res)      { unlock_fs(fs, res); return res; }

#else

#define ENTER_FF(fs)

#define LEAVE_FF(fs, res)      return res

#endif

#define ABORT(fs, res)          { fp->flag |= FA__ERROR; LEAVE_FF(fs, res); }

/* File shareing feature */

#if _FS_SHARE

#if _FS_READONLY

#error _FS_SHARE must be 0 on read-only cfg.

#endif

typedef struct {

    FATFS *fs;                      /* File ID 1, volume (NULL:blank entry) */

    DWORD clu;                      /* File ID 2, directory */

    WORD idx;                       /* File ID 3, directory index */

    WORD ctr;                       /* File open counter, 0:none,
0x01..0xFF:read open count, 0x100:write mode */

} FILESEM;

#endif

/* Misc definitions */

#define LD_CLUST(dir) (((DWORD)LD_WORD(dir+DIR_FstClusHI)<<16) |
LD_WORD(dir+DIR_FstClusLO))

#define ST_CLUST(dir,cl) {ST_WORD(dir+DIR_FstClusLO, cl); ST_WORD(dir+DIR_FstClusHI,
(DWORD)cl>>16);}

/* Character code support macros */

#define IsUpper(c)   (((c)>='A') && ((c)<='Z'))

#define IsLower(c)   (((c)>='a') && ((c)<='z'))

#define IsDigit(c)   (((c)>='0') && ((c)<='9'))

#if _DF1S           /* Code page is DBCS */

#define _DF2S /* Two 1st byte areas */

#define IsDBCS1(c)   (((BYTE)(c) >= _DF1S && (BYTE)(c) <= _DF1E) || ((BYTE)(c) >= _DF2S
&& (BYTE)(c) <= _DF2E))


```

```
#else /* One 1st byte area */

#define IsDBCS1(c) ((BYTE)(c) >= _DF1S && (BYTE)(c) <= _DF1E)
#endif

#ifndef _DS3S /* Three 2nd byte areas */

#define IsDBCS2(c) (((BYTE)(c) >= _DS1S && (BYTE)(c) <= _DS1E) || ((BYTE)(c) >= _DS2S
&& (BYTE)(c) <= _DS2E) || ((BYTE)(c) >= _DS3S && (BYTE)(c) <= _DS3E))

#else /* Two 2nd byte areas */

#define IsDBCS2(c) (((BYTE)(c) >= _DS1S && (BYTE)(c) <= _DS1E) || ((BYTE)(c) >= _DS2S
&& (BYTE)(c) <= _DS2E))

#endif

#else /* Code page is SBCS */

#define IsDBCS1(c) 0
#define IsDBCS2(c) 0

#endif /* _DF1S */

/* Name status flags */

#define NS 11 /* Offset of name status byte */
#define NS LOSS 0x01 /* Out of 8.3 format */
#define NS LFN 0x02 /* Force to create LFN entry */
#define NS LAST 0x04 /* Last segment */
#define NS BODY 0x08 /* Lower case flag (body) */
#define NS EXT 0x10 /* Lower case flag (ext) */
#define NS DOT 0x20 /* Dot entry */

/* FAT sub-type boundaries */

/* Note that the FAT spec by Microsoft says 4085 but Windows works with 4087! */

#define MIN_FAT16 4086 /* Minimum number of clusters for FAT16 */
#define MIN_FAT32 65526 /* Minimum number of clusters for FAT32 */

/* FatFs refers the members in the FAT structures as byte array instead of
structure member because there are incompatibility of the packing option
between compilers. */
```

#define BS_jmpBoot	0
#define BS_OEMName	3
#define BPB_BytsPerSec	11
#define BPB_SecPerClus	13
#define BPB_RsvdSecCnt	14
#define BPB_NumFATs	16
#define BPB_RootEntCnt	17
#define BPB_TotSec16	19
#define BPB_Media	21
#define BPB_FATSz16	22
#define BPB_SecPerTrk	24
#define BPB_NumHeads	26
#define BPB_HiddSec	28
#define BPB_TotSec32	32
#define BS_DrvNum	36
#define BS_BootSig	38
#define BS_VolID	39
#define BS_VolLab	43
#define BS_FilSysType	54
#define BPB_FATSz32	36
#define BPB_ExtFlags	40
#define BPB_FSVer	42
#define BPB_RootClus	44
#define BPB_FSIInfo	48
#define BPB_BkBootSec	50
#define BS_DrvNum32	64
#define BS_BootSig32	66
#define BS_VolID32	67
#define BS_VolLab32	71
#define BS_FilSysType32	82
#define FSI_LeadSig	0
#define FSI_StrucSig	484
#define FSI_Free_Count	488
#define FSI_Nxt_Free	492
#define MBR_Table	446
#define BS_55AA	510
#define DIR_Name	0
#define DIR_Attr	11

```
#define DIR_NTres           12
#define DIR_CrtTime          14
#define DIR_CrtDate          16
#define DIR_FstClusHI         20
#define DIR_WrtTime           22
#define DIR_WrtDate           24
#define DIR_FstClusLO         26
#define DIR_FileSize          28
#define LDIR_Ord               0
#define LDIR_Attr              11
#define LDIR_Type              12
#define LDIR_Checksum          13
#define LDIR_FstClusLO         26

/*-----*/
/* Work area */

#if _VOLUMES
static
FATFS *FatFs[_VOLUMES];      /* Pointer to the file system objects (logical drives) */
#else
#error Number of drives must not be 0.
#endif

static
WORD Fsid;                  /* File system mount ID */

#if _FS_RPATH
static
BYTE CurrVol;               /* Current drive */
#endif

#if _FS_SHARE
static
FILESEMFiles[_FS_SHARE];    /* File lock semaphores */
#endif
```

```
#if _USE_LFN == 0                                /* No LFN */
#define DEF_NAMEBUF          BYTE sfn[12]
#define INIT_BUF(dobj)        (dobj).fn = sfn
#define FREE_BUF()

#elif _USE_LFN == 1                                /* LFN with static LFN working buffer */
static WCHAR LfnBuf[_MAX_LFN+1];
#define DEF_NAMEBUF          BYTE sfn[12]
#define INIT_BUF(dobj)        { (dobj).fn = sfn; (dobj).lfn = LfnBuf; }
#define FREE_BUF()

#elif _USE_LFN == 2                                /* LFN with dynamic LFN working buffer on the stack */
#define DEF_NAMEBUF          BYTE sfn[12]; WCHAR lbuf[_MAX_LFN+1]
#define INIT_BUF(dobj)        { (dobj).fn = sfn; (dobj).lfn = lbuf; }
#define FREE_BUF()

#elif _USE_LFN == 3                                /* LFN with dynamic LFN working buffer on the heap */
#define DEF_NAMEBUF          BYTE sfn[12]; WCHAR *lfn
#define INIT_BUF(dobj)        { lfn = ff_memalloc((_MAX_LFN + 1) * 2); \
                           if (!lfn) LEAVE_FF((dobj).fs,
FR_NOT_ENOUGH_CORE); \
                           (dobj).lfn = lfn;   (dobj).fn =
sfn; }
#define FREE_BUF()           ff_memfree(lfn)

#else
#error Wrong LFN configuration.
#endif

/*-----*
Module Private Functions
-----*/
```

```
/* String functions */  
/*-----*/  
  
/* Copy memory to memory */  
  
static  
void mem_cpy (void* dst, const void* src, UINT cnt) {  
    BYTE *d = (BYTE*)dst;  
    const BYTE *s = (const BYTE*)src;  
  
#if _WORD_ACCESS == 1  
    while (cnt >= sizeof(int)) {  
        *(int*)d = *(int*)s;  
        d += sizeof(int); s += sizeof(int);  
        cnt -= sizeof(int);  
    }  
#endif  
    while (cnt--)  
        *d++ = *s++;  
}  
  
/* Fill memory */  
  
static  
void mem_set (void* dst, int val, UINT cnt) {  
    BYTE *d = (BYTE*)dst;  
  
    while (cnt--)  
        *d++ = (BYTE)val;  
}  
  
/* Compare memory to memory */  
  
static  
int mem_cmp (const void* dst, const void* src, UINT cnt) {  
    const BYTE *d = (const BYTE *)dst, *s = (const BYTE *)src;  
    int r = 0;  
  
    while (cnt-- && (r = *d++ - *s++) == 0) ;  
    return r;  
}
```

```
/* Check if chr is contained in the string */
static

int chk_chr (const char* str, int chr) {
    while (*str && *str != chr) str++;
    return *str;
}

/*-----*/
/* Request/Release grant to access the volume */
/*-----*/
#if _FS_REENTRANT

static

int lock_fs (
    FATFS *fs           /* File system object */
)
{
    return ff_req_grant(fs->sobj);
}

static

void unlock_fs (
    FATFS *fs,           /* File system object */
    FRESULT res          /* Result code to be returned */
)
{
    if (res != FR_NOT_ENABLED &&
        res != FR_INVALID_DRIVE &&
        res != FR_INVALID_OBJECT &&
        res != FR_TIMEOUT) {
        ff_rel_grant(fs->sobj);
    }
}
#endif
```

```
/*
 *-----*/
/* File shareing control functions */
/*-----*/
#if _FS_SHARE

static

FRESULT chk_lock ( /* Check if the file can be accessed */

    DIR* dj,           /* Directory object pointing the file to be checked */
    int acc            /* Desired access (0:Read, 1:Write, 2:Delete/Rename) */
)
{
    UINT i, be;

    /* Search file semaphore table */

    for (i = be = 0; i < _FS_SHARE; i++) {
        if (Files[i].fs) { /* Existing entry */

            if (Files[i].fs == dj->fs &&          /* Check if the file matched
with an open file */

                Files[i].clu == dj->sclust &&
                Files[i].idx == dj->index) break;

            } else { /* Blank entry */

                be++;
            }
        }

        if (i == _FS_SHARE) /* The file is not opened */

            return (be || acc == 2) ? FR_OK : FR_TOO_MANY_OPEN_FILES; /* Is there a
blank entry for new file? */

        /* The file has been opened. Reject any open against writing file and all write
mode open */

        return (acc || Files[i].ctr == 0x100) ? FR_LOCKED : FR_OK;
    }
}

static

int enq_lock ( /* Check if an entry is available for a new file */

    FATFS* fs           /* File system object */
)
{
    UINT i;
```

```
for (i = 0; i < _FS_SHARE && Files[i].fs; i++) ;

return (i == _FS_SHARE) ? 0 : 1;
}

static

UINT inc_lock (           /* Increment file open counter and returns its index (0:int error)
 */
    DIR* dj,           /* Directory object pointing the file to register or increment */
    int acc            /* Desired access mode (0:Read, !0:Write) */

)
{
    UINT i;

    for (i = 0; i < _FS_SHARE; i++) {      /* Find the file */

        if (Files[i].fs == dj->fs &&
            Files[i].clu == dj->sclust &&
            Files[i].idx == dj->index) break;
    }

    if (i == _FS_SHARE) {                  /* Not opened. Register it as new.

 */
        for (i = 0; i < _FS_SHARE && Files[i].fs; i++) ;

        if (i == _FS_SHARE) return 0; /* No space to register (int err) */

        Files[i].fs = dj->fs;
        Files[i].clu = dj->sclust;
        Files[i].idx = dj->index;
        Files[i].ctr = 0;
    }

    if (acc && Files[i].ctr) return 0; /* Access violation (int err) */

    Files[i].ctr = acc ? 0x100 : Files[i].ctr + 1;      /* Set semaphore value */

    return i + 1;
}
```

```
static

FRESULT dec_lock ( /* Decrement file open counter */

    UINT i /* Semaphore index */

)

{

    WORD n;

    FRESULT res;

    if (--i < _FS_SHARE) {

        n = Files[i].ctr;

        if (n == 0x100) n = 0;

        if (n) n--;

        Files[i].ctr = n;

        if (!n) Files[i].fs = 0;

        res = FR_OK;

    } else {

        res = FR_INT_ERR;

    }

    return res;

}

static

void clear_lock ( /* Clear lock entries of the volume */

    FATFS *fs

)

{

    UINT i;

    for (i = 0; i < _FS_SHARE; i++) {

        if (Files[i].fs == fs) Files[i].fs = 0;

    }

}

#endif

/*-----*/
```

```
/* Change window offset */  
/*-----*/  
  
static  
FRESULT move_window (  
    FATFS *fs, /* File system object */  
    DWORD sector /* Sector number to make appearance in the fs->win[] */  
) /* Move to zero only writes back dirty window */  
{  
    DWORD wsect;  
  
    wsect = fs->winsect;  
    if (wsect != sector) /* Changed current window */  
#if !_FS_READONLY  
        if (fs->wflag) { /* Write back dirty window if needed */  
            if (disk_write(fs->drv, fs->win, wsect, 1) != RES_OK)  
                return FR_DISK_ERR;  
            fs->wflag = 0;  
            if (wsect < (fs->fatbase + fs->fsize)) { /* In FAT area */  
                BYTE nf;  
                for (nf = fs->n_fats; nf > 1; nf--) { /* Reflect the  
change to all FAT copies */  
                    wsect += fs->fsize;  
                    disk_write(fs->drv, fs->win, wsect, 1);  
                }  
            }  
        }  
#endif  
        if (sector) {  
            if (disk_read(fs->drv, fs->win, sector, 1) != RES_OK)  
                return FR_DISK_ERR;  
            fs->winsect = sector;  
        }  
    }  
  
    return FR_OK;  
}
```

```
/*-----*/
/* Clean-up cached data */
/*-----*/

#if !_FS_READONLY

static

FRESULT sync ( /* FR_OK: successful, FR_DISK_ERR: failed */

    FATFS *fs      /* File system object */

)
{

    FRESULT res;

    res = move_window(fs, 0);

    if (res == FR_OK) {

        /* Update FSInfo sector if needed */

        if (fs->fs_type == FS_FAT32 && fs->fsi_flag) {

            fs->winsect = 0;

            mem_set(fs->win, 0, 512);

            ST_WORD(fs->win+BS_55AA, 0xAA55);
            ST_DWORD(fs->win+FSI_LeadSig, 0x41615252);
            ST_DWORD(fs->win+FSI_StrucSig, 0x61417272);
            ST_DWORD(fs->win+FSI_Free_Count, fs->free_clust);
            ST_DWORD(fs->win+FSI_Nxt_Free, fs->last_clust);
            disk_write(fs->drv, fs->win, fs->fsi_sector, 1);
            fs->fsi_flag = 0;
        }

        /* Make sure that no pending write process in the physical drive */

        if (disk_ioctl(fs->drv, CTRL_SYNC, (void*) 0) != RES_OK)

            res = FR_DISK_ERR;
    }
}

return res;
}

#endif
```

```
/*-----*/
/* Get sector# from cluster# */  
/*-----*/  
  
DWORD clust2sect ( /* !=0: Sector number, 0: Failed - invalid cluster# */  
    FATFS *fs, /* File system object */  
    DWORD clst /* Cluster# to be converted */  
)  
{  
    clst -= 2;  
    if (clst >= (fs->n_fatent - 2)) return 0; /* Invalid cluster# */  
    return clst * fs->csize + fs->database;  
}  
  
/*-----*/  
/* FAT access - Read value of a FAT entry */  
/*-----*/  
  
DWORD get_fat ( /* 0xFFFFFFFF:Disk error, 1:Internal error, Else:Cluster status */  
    FATFS *fs, /* File system object */  
    DWORD clst /* Cluster# to get the link information */  
)  
{  
    UINT wc, bc;  
    BYTE *p;  
  
    if (clst < 2 || clst >= fs->n_fatent) /* Check range */  
        return 1;  
  
    switch (fs->fs_type) {  
        case FS_FAT12 :  
            bc = (UINT)clst; bc += bc / 2;
```

```
        if (move_window(fs, fs->fatbase + (bc / SS(fs)))) break;
        wc = fs->win[bc % SS(fs)]; bc++;
        if (move_window(fs, fs->fatbase + (bc / SS(fs)))) break;
        wc |= fs->win[bc % SS(fs)] << 8;
        return (clst & 1) ? (wc >> 4) : (wc & 0xFFFF);

    case FS_FAT16 :
        if (move_window(fs, fs->fatbase + (clst / (SS(fs) / 2)))) break;
        p = &fs->win[clst * 2 % SS(fs)];
        return LD_WORD(p);

    case FS_FAT32 :
        if (move_window(fs, fs->fatbase + (clst / (SS(fs) / 4)))) break;
        p = &fs->win[clst * 4 % SS(fs)];
        return LD_DWORD(p) & 0xFFFFFFFF;
    }

    return 0xFFFFFFFF; /* An error occurred at the disk I/O layer */
}

/*-----
/* FAT access - Change value of a FAT entry
/*-----*/
#endif !FS_READONLY

FRESULT put_fat (
    FATFS *fs,      /* File system object */
    DWORD clst,     /* Cluster# to be changed in range of 2 to fs->n_fatent - 1 */
    DWORD val       /* New value to mark the cluster */
)
{
    UINT bc;
    BYTE *p;
    FRESULT res;
```

```
if (clst < 2 || clst >= fs->n_fatent) { /* Check range */
    res = FR_INT_ERR;

} else {

    switch (fs->fs_type) {

        case FS_FAT12 :

            bc = clst; bc += bc / 2;

            res = move_window(fs, fs->fatbase + (bc / SS(fs)));
            if (res != FR_OK) break;
            p = &fs->win[bc % SS(fs)];
            *p = (clst & 1) ? ((*p & 0x0F) | ((BYTE)val << 4)) : (BYTE)val;
            bc++;
            fs->wflag = 1;
            res = move_window(fs, fs->fatbase + (bc / SS(fs)));
            if (res != FR_OK) break;
            p = &fs->win[bc % SS(fs)];
            *p = (clst & 1) ? (BYTE)(val >> 4) : ((*p & 0xF0) | ((BYTE)(val >> 8) & 0x0F));
            break;

        case FS_FAT16 :

            res = move_window(fs, fs->fatbase + (clst / (SS(fs) / 2)));
            if (res != FR_OK) break;
            p = &fs->win[clst * 2 % SS(fs)];
            ST_WORD(p, (WORD)val);
            break;

        case FS_FAT32 :

            res = move_window(fs, fs->fatbase + (clst / (SS(fs) / 4)));
            if (res != FR_OK) break;
            p = &fs->win[clst * 4 % SS(fs)];
            val |= LD_DWORD(p) & 0xFFFFFFFF;
            ST_DWORD(p, val);
            break;

        default :

            res = FR_INT_ERR;
    }

    fs->wflag = 1;
}
```

```
        return res;
    }

#endif /* !_FS_READONLY */

/*
 * FAT handling - Remove a cluster chain
 */
#endif /* !_FS_READONLY */

#if !_FS_READONLY
static

FRESULT remove_chain (
    FATFS *fs,                      /* File system object */
    DWORD clst,                      /* Cluster# to remove a chain from */
)
{
    FRESULT res;
    DWORD nxt;

#if _USE_ERASE
    DWORD scl = clst, ecl = clst, resion[2];
#endif

    if (clst < 2 || clst >= fs->n_fatent) {      /* Check range */
        res = FR_INT_ERR;
    } else {
        res = FR_OK;
        while (clst < fs->n_fatent) {                /* Not a last link? */
            nxt = get_fat(fs, clst);                  /* Get cluster status
*/
            if (nxt == 0) break;                         /* Empty cluster? */
            if (nxt == 1) { res = FR_INT_ERR; break; }   /* Internal error? */
            if (nxt == 0xFFFFFFFF) { res = FR_DISK_ERR; break; } /* Disk
error? */
            res = put_fat(fs, clst, 0);                 /* Mark the cluster
"empty" */
            if (res != FR_OK) break;
            if (fs->free_clust != 0xFFFFFFFF) { /* Update FSInfo */
                fs->free_clust++;
            }
        }
    }
}
```

```

        fs->fsi_flag = 1;

    }

#endif _USE_ERASE

    if (ecl + 1 == nxt) { /* Next cluster is contiguous */

        ecl = nxt;

    } else { /* End of contiguous clusters */
        resion[0] = clust2sect(fs, scl);
        /* Start sector */

        resion[1] = clust2sect(fs, ecl) + fs->csizen - 1; /* End
sector */

        disk_ioctl(fs->drv, CTRL_ERASE_SECTOR, resion);
        /* Erase the block */

        scl = ecl = nxt;
    }

#endif

    clst = nxt; /* Next cluster */
}

}

return res;
}

#endif

/*-----*/
/* FAT handling - Stretch or Create a cluster chain */
/*-----*/

#endif !_FS_READONLY

static

DWORD create_chain ( /* 0:No free cluster, 1:Internal error, 0xFFFFFFFF:Disk error,
>=2>New cluster# */

    FATFS *fs, /* File system object */
    DWORD clst /* Cluster# to stretch. 0 means create a new chain.
*/
)
{
    DWORD cs, ncl, scl;
    FRESULT res;
}

```

```

if (clst == 0) {                                /* Create a new chain */

    scl = fs->last_clust;                      /* Get suggested start point */

    if (!scl || scl >= fs->n_fatent) scl = 1;

}

else {                                         /* Stretch the current chain */

    cs = get_fat(fs, clst);                     /* Check the cluster status */

    if (cs < 2) return 1;                         /* It is an invalid cluster */

    if (cs < fs->n_fatent) return cs;           /* It is already followed by next
cluster */

    scl = clst;
}

ncl = scl;                                     /* Start cluster */

for (;;) {

    ncl++;                                       /* Next cluster */

    if (ncl >= fs->n_fatent) {                  /* Wrap around */

        ncl = 2;

        if (ncl > scl) return 0;                 /* No free cluster */

    }

    cs = get_fat(fs, ncl);                       /* Get the cluster status */

    if (cs == 0) break;                          /* Found a free cluster */

    if (cs == 0xFFFFFFFF || cs == 1)/* An error occurred */

        return cs;

    if (ncl == scl) return 0;                   /* No free cluster */
}

res = put_fat(fs, ncl, 0x0FFFFFFF); /* Mark the new cluster "last link" */

if (res == FR_OK && clst != 0) {

    res = put_fat(fs, clst, ncl); /* Link it to the previous one if needed */

}

if (res == FR_OK) {

    fs->last_clust = ncl;                      /* Update FSINFO */

    if (fs->free_clust != 0xFFFFFFFF) {

        fs->free_clust--;
        fs->fsi_flag = 1;
    }
} else {

    ncl = (res == FR_DISK_ERR) ? 0xFFFFFFFF : 1;
}

```

```
}

    return ncl;           /* Return new cluster number or error code */

}

#endif /* !_FS_READONLY */



/*-----*/
/* Directory handling - Set directory index */
/*-----*/



static
FRESULT dir_sdi (
    DIR *dj,           /* Pointer to directory object */
    WORD idx            /* Directory index number */
)
{
    DWORD clst;
    WORD ic;

    dj->index = idx;
    clst = dj->sclust;
    if (clst == 1 || clst >= dj->fs->n_fatent) /* Check start cluster range */
        return FR_INT_ERR;
    if (!clst && dj->fs->fs_type == FS_FAT32) /* Replace cluster# 0 with root
cluster# if in FAT32 */
        clst = dj->fs->dirbase;

    if (clst == 0) {      /* Static table (root-dir in FAT12/16) */
        dj->clust = clst;
        if (idx >= dj->fs->n_rootdir)          /* Index is out of range */
            return FR_INT_ERR;
        dj->sect = dj->fs->dirbase + idx / (SS(dj->fs) / 32);      /* Sector# */
    }
    else {                /* Dynamic table (sub-dirs or root-dir in FAT32) */
        ic = SS(dj->fs) / 32 * dj->fs->cszie;           /* Entries per cluster */
        while (idx >= ic) {     /* Follow cluster chain */

```

```

        clst = get_fat(dj->fs, clst);                                /* Get next
cluster */

        if (clst == 0xFFFFFFFF) return FR_DISK_ERR; /* Disk error */

        if (clst < 2 || clst >= dj->fs->n_fatent) /* Reached to end of
table or int error */

            return FR_INT_ERR;

        idx -= ic;

    }

    dj->clust = clst;

    dj->sect = clust2sect(dj->fs, clst) + idx / (SS(dj->fs) / 32); /*

Sector# */

}

dj->dir = dj->fs->win + (idx % (SS(dj->fs) / 32)) * 32; /* Ptr to the entry in
the sector */

return FR_OK; /* Seek succeeded */
}

/*-----*/
/* Directory handling - Move directory index next */
/*-----*/

static

HRESULT dir_next ( /* FR_OK: Succeeded, FR_NO_FILE: End of table, FR_DENIED: EOT and
could not stretch */

    DIR *dj,                      /* Pointer to directory object */
    int stretch                    /* 0: Do not stretch table, 1: Stretch table if needed */
)
{
    DWORD clst;
    WORD i;

    i = dj->index + 1;

    if (!i || !dj->sect) /* Report EOT when index has reached 65535 */

        return FR_NO_FILE;

    if (!(i % (SS(dj->fs) / 32))) { /* Sector changed? */

```

```

        dj->sect++;
                                /* Next sector */

        if (dj->clust == 0) { /* Static table */
            if (i >= dj->fs->n_rootdir) /* Report EOT when end of table */
                return FR_NO_FILE;
        }
        else { /* Dynamic table */
            if (((i / (SS(dj->fs) / 32)) & (dj->fs->csize - 1)) == 0) {
/* Cluster changed? */
                clst = get_fat(dj->fs, dj->clust);
/* Get next cluster */
                if (clst <= 1) return FR_INT_ERR;
                if (clst == 0xFFFFFFFF) return FR_DISK_ERR;
                if (clst >= dj->fs->n_fatent) {
/* When it reached end of dynamic table */
#endif !FS_READONLY
                BYTE c;
                if (!stretch) return FR_NO_FILE;
/* When do not stretch, report EOT */
                clst = create_chain(dj->fs, dj->clust);
/* Stretch cluster chain */
                if (clst == 0) return FR_DENIED;
                if (clst == 1) return FR_INT_ERR;
                if (clst == 0xFFFFFFFF) return FR_DISK_ERR;
/* Clean-up stretched table */
                if (move_window(dj->fs, 0)) return FR_DISK_ERR;
                mem_set(dj->fs->win, 0, SS(dj->fs));
                dj->fs->winsect = clust2sect(dj->fs, clst); /* Cluster start sector */
                for (c = 0; c < dj->fs->csize; c++) {
/* Fill the new cluster with 0 */
                    dj->fs->wflag = 1;
                    if (move_window(dj->fs, 0)) return
FR_DISK_ERR;
                    dj->fs->winsect++;
                }
                dj->fs->winsect -= c;
/* Rewind window address */
#else
                return FR_NO_FILE; /* Report EOT */
#endif
#endif

```

```
        }

        dj->clust = clst;                                /* Initialize
data for new cluster */

        dj->sect = clust2sect(dj->fs, clst);

    }

}

dj->index = i;
dj->dir = dj->fs->win + (i % (SS(dj->fs) / 32)) * 32;

return FR_OK;
}

/*-----*/
/* LFN handling - Test/Pick/Fit an LFN segment from/to directory entry */
/*-----*/
#if _USE_LFN

static

const BYTE LfnOfs[] = {1,3,5,7,9,14,16,18,20,22,24,28,30}; /* Offset of LFN chars in the
directory entry */

static

int cmp_lfn (                                         /* 1:Matched, 0:Not matched */
    WCHAR *lfnbuf,                                     /* Pointer to the LFN to be compared */
    BYTE *dir                                         /* Pointer to the directory entry containing a part
of LFN */
)
{
    UINT i, s;
    WCHAR wc, uc;

    i = ((dir[LDIR_Ord] & 0xBF) - 1) * 13;          /* Get offset in the LFN buffer */
    s = 0; wc = 1;
    do {
        uc = LD_WORD(dir+LfnOfs[s]); /* Pick an LFN character from the entry */

```

```

        if (wc) { /* Last char has not been processed */

            wc = ff_wtoupper(uc); /* Convert it to upper case */

            if (i >= _MAX_LFN || wc != ff_wtoupper(lfnbuf[i++])) /*

Compare it */

                return 0; /* Not matched */

        } else {

            if (uc != 0xFFFF) return 0; /* Check filler */

        }

        } while (++s < 13); /* Repeat until all chars in the

entry are checked */

        if ((dir[LDIR_Ord] & 0x40) && wc && lfnbuf[i]) /* Last segment matched but

different length */

            return 0;

        return 1; /* The part of LFN matched */

    }

static

int pick_lfn (/* 1: Succeeded, 0: Buffer overflow */

    WCHAR *lfnbuf, /* Pointer to the Unicode-LFN buffer */

    BYTE *dir /* Pointer to the directory entry */

)

{
    UINT i, s;

    WCHAR uc, wc;

    i = ((dir[LDIR_Ord] & 0x3F) - 1) * 13; /* Offset in the LFN buffer */

    s = 0; wc = 1;

    do {

        uc = LD_WORD(dir+LfnOfs[s]); /* Pick an LFN character from the

entry */

        if (wc) { /* Last char has not been processed */

            if (i >= _MAX_LFN) return 0; /* Buffer overflow? */

            lfnbuf[i++] = wc = uc; /* Store it */

        } else {

            if (uc != 0xFFFF) return 0; /* Check filler */

        }

    }
}

```

```

        }

    } while (++s < 13);                                /* Read all character
in the entry */

    if (dir[LDIR_Ord] & 0x40) {                         /* Put terminator if it is
the last LFN part */

        if (i >= _MAX_LFN) return 0;                      /* Buffer overflow? */

        lfnbuf[i] = 0;

    }

    return 1;
}

#endif !_FS_READONLY

static

void fit_lfn (
    const WCHAR *lfnbuf,   /* Pointer to the LFN buffer */
    BYTE *dir,             /* Pointer to the directory entry */
    BYTE ord,              /* LFN order (1-20) */
    BYTE sum               /* SFN sum */
)
{
    UINT i, s;
    WCHAR wc;

    dir[LDIR_Chksum] = sum;                            /* Set check sum */
    dir[LDIR_Attr] = AM_LFN;                           /* Set attribute. LFN entry */
    dir[LDIR_Type] = 0;
    ST_WORD(dir+LDIR_FstClusLO, 0);

    i = (ord - 1) * 13;                                /* Get offset in the LFN buffer */
    s = wc = 0;
    do {
        if (wc != 0xFFFF) wc = lfnbuf[i++]; /* Get an effective char */
        ST_WORD(dir+LfnOfs[s], wc); /* Put it */
        if (!wc) wc = 0xFFFF;          /* Padding chars following last char */
    } while (++s < 13);

    if (wc == 0xFFFF || !lfnbuf[i]) ord |= 0x40; /* Bottom LFN part is the start of
LFN sequence */
}

```

```
    dir[LDIR_Ord] = ord;                                /* Set the LFN order */
}

#endif
#endif

/*
 * Create numbered name
 */
/*
@if _USE_LFN
void gen_numname (
    BYTE *dst,                      /* Pointer to generated SFN */
    const BYTE *src,                 /* Pointer to source SFN to be modified */
    const WCHAR *lfn,                /* Pointer to LFN */
    WORD seq                         /* Sequence number */
)
{
    BYTE ns[8], c;
    UINT i, j;

    mem_cpy(dst, src, 11);

    if (seq > 5) { /* On many collisions, generate a hash number instead of
sequential number */
        do seq = (seq >> 1) + (seq << 15) + (WORD)*lfn++; while (*lfn);
    }

    /* itoa */
    i = 7;
    do {
        c = (seq % 16) + '0';
        if (c > '9') c += 7;
        ns[i--] = c;
        seq /= 16;
    } while (seq);
    ns[i] = '~';
}
```

```
/* Append the number */

for (j = 0; j < i && dst[j] != ' '; j++) {
    if (IsDBCS1(dst[j])) {
        if (j == i - 1) break;
        j++;
    }
}
do {
    dst[j++] = (i < 8) ? ns[i++] : ' ';
} while (j < 8);
}

#endif

/*-----*/
/* Calculate sum of an SFN */
/*-----*/

#if _USE_LFN
static
BYTE sum_sfn (
    const BYTE *dir           /* Ptr to directory entry */
)
{
    BYTE sum = 0;
    UINT n = 11;

    do sum = (sum >> 1) + (sum << 7) + *dir++; while (--n);

    return sum;
}
#endif

/*-----*/
/* Directory handling - Find an object in the directory */
/*-----*/
```

```

static

FRESULT dir_find (
    DIR *dj           /* Pointer to the directory object linked to the file name
*/
)
{
    FRESULT res;
    BYTE c, *dir;

#if _USE_LFN
    BYTE a, ord, sum;
#endif

    res = dir_sdi(dj, 0);                      /* Rewind directory object */

    if (res != FR_OK) return res;

#if _USE_LFN
    ord = sum = 0xFF;
#endif

    do {
        res = move_window(dj->fs, dj->sect);
        if (res != FR_OK) break;

        dir = dj->dir;                         /* Ptr to the directory entry
of current index */

        c = dir[DIR_Name];
        if (c == 0) { res = FR_NO_FILE; break; }   /* Reached to end of table */

#if _USE_LFN /* LFN configuration */
        a = dir[DIR_Attr] & AM_MASK;
        if (c == 0xE5 || ((a & AM_VOL) && a != AM_LFN)) { /* An entry without
valid data */
            ord = 0xFF;
        } else {
            if (a == AM_LFN) {                      /* An LFN entry is found */
                if (dj->lfn) {
                    if (c & 0x40) {                  /* Is it start of LFN
sequence? */
                        sum = dir[LDIR_Chksum];
                        c &= 0xBF; ord = c;      /* LFN start order */
                        dj->lfn_idx = dj->index;
                    }
                    /* Check validity of the LFN entry and compare it
with given name */
                }
            }
        }
    }
}

```

```

        ord = (c == ord && sum == dir[LDIR_Chksum] &&
        cmp_lfn(dj->lfn, dir)) ? ord - 1 : 0xFF;
    }

} else {                                              /* An SFN entry is
found */

    if (!ord && sum == sum_sfn(dir)) break;          /* LFN
matched? */

    ord = 0xFF; dj->lfn_idx = 0xFFFF;                /* Reset LFN sequence
*/
    if (!(dj->fn[NS] & NS LOSS) && !mem_cmp(dir, dj->fn, 11))
break; /* SFN matched? */

}

#endif /* Non LFN configuration */

if (!(dir[DIR_Attr] & AM_VOL) && !mem_cmp(dir, dj->fn, 11)) /* Is it a
valid entry? */

break;

#endif

res = dir_next(dj, 0);                      /* Next entry */
} while (res == FR_OK);

return res;
}

/*-----*/
/* Read an object from the directory */
/*-----*/
#if _FS_MINIMIZE <= 1

static
FRESULT dir_read (
    DIR *dj           /* Pointer to the directory object that pointing the entry
to be read */
)
{
    FRESULT res;
    BYTE c, *dir;

#if _USE_LFN
    BYTE a, ord = 0xFF, sum = 0xFF;
#endif
}

```

```

res = FR_NO_FILE;
while (dj->sect) {
    res = move_window(dj->fs, dj->sect);
    if (res != FR_OK) break;
    dir = dj->dir;                                /* Ptr to the directory entry
of current index */
    c = dir[DIR_Name];
    if (c == 0) { res = FR_NO_FILE; break; }        /* Reached to end of table */
#ifndef _USE_LFN /* LFN configuration */
    a = dir[DIR_Attr] & AM_MASK;
    if (c == 0xE5 || (_FS_RPATH && c == '.') || ((a & AM_VOL) && a != AM_LFN)) { /* An entry without valid data */
        ord = 0xFF;
    } else {
        if (a == AM_LFN) {                               /* An LFN entry is found */
            if (c & 0x40) {                            /* Is it start of LFN
sequence? */
                sum = dir[LDIR_Chksum];
                c &= 0xBF; ord = c;
                dj->lfn_idx = dj->index;
            }
            /* Check LFN validity and capture it */
            ord = (c == ord && sum == dir[LDIR_Chksum] && pick_lfn(dj->lfn, dir)) ? ord - 1 : 0xFF;
        } else {                                       /* An SFN entry is
found */
            if (ord || sum != sum_sfn(dir))          /* Is there a valid
LFN? */
                dj->lfn_idx = 0xFFFF;                  /* It has no LFN. */
            break;
        }
    }
#else /* Non LFN configuration */
    if (c != 0xE5 && (_FS_RPATH || c != '.') && !(dir[DIR_Attr] & AM_VOL)) /* Is it a valid entry? */
        break;
#endif
    res = dir_next(dj, 0);                          /* Next entry */
    if (res != FR_OK) break;
}

if (res != FR_OK) dj->sect = 0;

```

```
    return res;
}

#endif

/*
 * Register an object to the directory
 */
/*-----*/
#if !_FS_READONLY
static

FRESULT dir_register /* FR_OK:Successful, FR_DENIED:No free entry or too many SFN
collision, FR_DISK_ERR:Disk error */

    DIR *dj           /* Target directory with object name to be created
*/
)
{
    FRESULT res;

    BYTE c, *dir;

#if _USE_LFN /* LFN configuration */

    WORD n, ne, is;

    BYTE sn[12], *fn, sum;

    WCHAR *lfn;

    fn = dj->fn; lfn = dj->lfn;
    mem_cpy(sn, fn, 12);

    if (_FS_RPATH && (sn[NS] & NS_DOT))          /* Cannot create dot entry */
        return FR_INVALID_NAME;

    if (sn[NS] & NS LOSS) {                         /* When LFN is out of 8.3 format,
generate a numbered name */

        fn[NS] = 0; dj->lfn = 0;                   /* Find only SFN */

        for (n = 1; n < 100; n++) {
            gen_numname(fn, sn, lfn, n); /* Generate a numbered name */

            res = dir_find(dj);           /* Check if the name
collides with existing SFN */

            if (res != FR_OK) break;
        }

        if (n == 100) return FR_DENIED;             /* Abort if too many
collisions */
    }
}

```

```

        if (res != FR_NO_FILE) return res; /* Abort if the result is other than
'not collided' */

        fn[NS] = sn[NS]; dj->lfn = lfn;

    }

    if (sn[NS] & NS_LFN) { /* When LFN is to be created, reserve an SFN
+ LFN entries. */

        for (ne = 0; lfn[ne]; ne++) ;

        ne = (ne + 25) / 13;

    } else { /* Otherwise reserve only an
SFN entry. */

        ne = 1;
    }

/* Reserve contiguous entries */

res = dir_sdi(dj, 0);

if (res != FR_OK) return res;

n = is = 0;

do {

    res = move_window(dj->fs, dj->sect);

    if (res != FR_OK) break;

    c = *dj->dir; /* Check the entry status */

    if (c == 0xE5 || c == 0) { /* Is it a blank entry? */

        if (n == 0) is = dj->index; /* First index of the contiguous
entry */

        if (++n == ne) break; /* A contiguous entry that required count is
found */

    } else {

        n = 0; /* Not a blank entry. Restart
to search */
    }

    res = dir_next(dj, 1); /* Next entry with table stretch */

} while (res == FR_OK);

if (res == FR_OK && ne > 1) /* Initialize LFN entry if needed */

    res = dir_sdi(dj, is);

if (res == FR_OK) {

    sum = sum_sfns(dj->fn); /* Sum of the SFN tied to the LFN */

    ne--;
    do { /* Store LFN entries in
bottom first */

        res = move_window(dj->fs, dj->sect);

        if (res != FR_OK) break;
    }
}

```

```

        fit_lfn(dj->lfn, dj->dir, (BYTE)ne, sum);
        dj->fs->wflag = 1;
        res = dir_next(dj, 0); /* Next entry */
    } while (res == FR_OK && --ne);
}

}

#else /* Non LFN configuration */

res = dir_sdi(dj, 0);
if (res == FR_OK) {
    do { /* Find a blank entry for the SFN */
        res = move_window(dj->fs, dj->sect);
        if (res != FR_OK) break;
        c = *dj->dir;
        if (c == 0xE5 || c == 0) break; /* Is it a blank entry? */
        res = dir_next(dj, 1); /* Next entry with table
stretch */
    } while (res == FR_OK);
}
#endif

if (res == FR_OK) /* Initialize the SFN entry */
{
    res = move_window(dj->fs, dj->sect);
    if (res == FR_OK) {
        dir = dj->dir;
        mem_set(dir, 0, 32); /* Clean the entry */
        mem_cpy(dir, dj->fn, 11); /* Put SFN */
#if _USE_LFN
        dir[DIR_NTres] = *(dj->fn+NS) & (NS_BODY | NS_EXT); /* Put NT flag
*/
#endif
    }
    dj->fs->wflag = 1;
}

return res;
}

#endif /* !_FS_READONLY */

```

```

/*-----*/
/* Remove an object from the directory */
/*-----*/

#if !_FS_READONLY && !_FS_MINIMIZE

static

FRESULT dir_remove ( /* FR_OK: Successful, FR_DISK_ERR: A disk error */

    DIR *dj           /* Directory object pointing the entry to be
removed */

)
{
    FRESULT res;

#if _USE_LFN /* LFN configuration */

    WORD i;

    i = dj->index; /* SFN index */

    res = dir_sdi(dj, (WORD)((dj->lfn_idx == 0xFFFF) ? i : dj->lfn_idx)); /* Goto
the SFN or top of the LFN entries */

    if (res == FR_OK) {

        do {

            res = move_window(dj->fs, dj->sect);

            if (res != FR_OK) break;

            *dj->dir = 0xE5; /* Mark the entry "deleted"
*/
        }

        dj->fs->wflag = 1;

        if (dj->index >= i) break; /* When reached SFN, all entries of
the object has been deleted. */

        res = dir_next(dj, 0); /* Next entry */

    } while (res == FR_OK);

    if (res == FR_NO_FILE) res = FR_INT_ERR;
}

#else /* Non LFN configuration */

    res = dir_sdi(dj, dj->index);

    if (res == FR_OK) {

        res = move_window(dj->fs, dj->sect);

        if (res == FR_OK) {
            *dj->dir = 0xE5; /* Mark the entry "deleted"
*/
        }

        dj->fs->wflag = 1;
    }
}

```

```
}

#endif

return res;
}

#endif /* !_FS_READONLY */

/*-----*/
/* Pick a segment and create the object name in directory form */
/*-----*/

static
FRESULT create_name (
    DIR *dj,                      /* Pointer to the directory object */
    const TCHAR **path             /* Pointer to pointer to the segment in the path string */
)
{
#ifdef _EXCvt
    static const BYTE excvt[] = _EXCvt; /* Upper conversion table for extended chars */
#endif

#endif /* _USE_LFN */ /* LFN configuration */

    BYTE b, cf;
    WCHAR w, *lfn;
    UINT i, ni, si, di;
    const TCHAR *p;

    /* Create LFN in Unicode */

    si = di = 0;
    p = *path;
    lfn = dj->lfn;
    for (;;) {
        w = p[si++];                  /* Get a character */
        if (w < ' ' || w == '/' || w == '\\') break; /* Break on end of segment */
        if (di >= _MAX_LFN)           /* Reject too long name */
            return FR_INVALID_NAME;
    }
}
```

```

#endif !_LFN_UNICODE

    w &= 0xFF;

    if (IsDBCS1(w)) {                                /* Check if it is a DBC 1st
byte (always false on SBCS cfg) */

        b = (BYTE)p[si++];                            /* Get 2nd byte */

        if (!IsDBCS2(b))

            return FR_INVALID_NAME;                  /* Reject invalid sequence */

        w = (w << 8) + b;                           /* Create a DBC */

    }

    w = ff_convert(w, 1);                            /* Convert ANSI/OEM to Unicode */

    if (!w) return FR_INVALID_NAME;                /* Reject invalid code */

#endif

if (w < 0x80 && chk_chr("\":>\?|\x7F", w)) /* Reject illegal chars for
LFN */

    return FR_INVALID_NAME;

lfn[di++] = w;                                    /* Store the Unicode char */

}

*pPath = &p[si];                                 /* Return pointer to
the next segment */

cf = (w < ' ') ? NS_LAST : 0;                  /* Set last segment flag if end of path */

#endif _FS_RPATH

if ((di == 1 && lfn[di-1] == '.') || /* Is this a dot entry? */

    (di == 2 && lfn[di-1] == '.' && lfn[di-2] == '.'))

    lfn[di] = 0;

for (i = 0; i < 11; i++)

    dj->fn[i] = (i < di) ? '.' : ' ';

dj->fn[i] = cf | NS_DOT;                         /* This is a dot entry */

return FR_OK;

}

#endif

while (di) {                                       /* Strip trailing spaces and
dots */

    w = lfn[di-1];

    if (w != ' ' && w != '.') break;

    di--;

}

if (!di) return FR_INVALID_NAME;                /* Reject nul string */

lfn[di] = 0;                                     /* LFN is created */

/* Create SFN in directory form */

```

```

mem_set(dj->fn, ' ', 11);

for (si = 0; lfn[si] == ' ' || lfn[si] == '.'; si++) ;      /* Strip leading
spaces and dots */

if (si) cf |= NS_LOSS | NS_LFN;

while (di && lfn[di - 1] != '.') di--;           /* Find extension (di<=si: no
extension) */

b = i = 0; ni = 8;

for (;;) {

    w = lfn[si++];                                /* Get an LFN char */

    if (!w) break;                                /* Break on end of the LFN */

    if (w == ' ' || (w == '.' && si != di)) {    /* Remove spaces and dots */

        cf |= NS_LOSS | NS_LFN; continue;

    }

    if (i >= ni || si == di) {                    /* Extension or end of SFN */

        if (ni == 11) {                            /* Long extension */

            cf |= NS_LOSS | NS_LFN; break;

        }

        if (si != di) cf |= NS_LOSS | NS_LFN;      /* Out of 8.3 format

*/
    }

    if (si > di) break;                          /* No extension */

    si = di; i = 8; ni = 11;                      /* Enter extension section */

    b <= 2; continue;

}

if (w >= 0x80) {                                /* Non ASCII char */

#define _EXCWT

    w = ff_convert(w, 0);                      /* Unicode -> OEM code */

    if (w) w = excvt[w - 0x80];    /* Convert extended char to upper

(SBCS) */

#else

    w = ff_convert(ff_wtoupper(w), 0);    /* Upper converted Unicode ->

OEM code */

#endif

    cf |= NS_LFN;                            /* Force create LFN entry */

}

if (_DF1S && w >= 0x100) {                  /* Double byte char (always false on

SBCS cfg) */

    if (i >= ni - 1) {

        cf |= NS_LOSS | NS_LFN; i = ni; continue;
}

```

```

        }

        dj->fn[i++] = (BYTE)(w >> 8);

    } else {                                     /* Single byte char */

        if (!w || chk_chr("+,;=[]", w)) {      /* Replace illegal chars for
SFN */

            w = '_'; cf |= NS_LOSS | NS_LFN; /* Lossy conversion */

        } else {

            if (IsUpper(w)) {                  /* ASCII large capital */

                b |= 2;

            } else {

                if (IsLower(w)) {             /* ASCII small capital */

                    b |= 1; w -= 0x20;

                }

            }

        }

        dj->fn[i++] = (BYTE)w;
    }

}

if (dj->fn[0] == 0xE5) dj->fn[0] = 0x05; /* If the first char collides with
deleted mark, replace it with 0x05 */

if (ni == 8) b <= 2;

if ((b & 0x0C) == 0x0C || (b & 0x03) == 0x03)           /* Create LFN entry when
there are composite capitals */

    cf |= NS_LFN;

if (!(cf & NS_LFN)) {                                     /* When LFN is in 8.3
format without extended char, NT flags are created */

    if ((b & 0x03) == 0x01) cf |= NS_EXT; /* NT flag (Extension has
only small capital) */

    if ((b & 0x0C) == 0x04) cf |= NS_BODY; /* NT flag (Filename has only
small capital) */

}

dj->fn[NS] = cf; /* SFN is created */

return FR_OK;

#else /* Non-LFN configuration */

BYTE b, c, d, *sfm;
UINT ni, si, i;

```

```

const char *p;

/* Create file name in directory form */
sfn = dj->fn;
mem_set(sfn, ' ', 11);
si = i = b = 0; ni = 8;
p = *path;

#if _FS_RPATH

if (p[si] == '.') { /* Is this a dot entry? */
    for (;;) {
        c = (BYTE)p[si++];
        if (c != '.' || si >= 3) break;
        sfn[i++] = c;
    }
    if (c != '/' && c != '\\' && c > ' ') return FR_INVALID_NAME;
    *path = &p[si];
/* Return pointer to the next segment */

    sfn[NS] = (c <= ' ') ? NS_LAST | NS_DOT : NS_DOT; /* Set last segment
flag if end of path */

    return FR_OK;
}

#endif

for (;;) {
    c = (BYTE)p[si++];
    if (c <= ' ' || c == '/' || c == '\\') break; /* Break on end of
segment */
    if (c == '.' || i >= ni) {
        if (ni != 8 || c != '.') return FR_INVALID_NAME;
        i = 8; ni = 11;
        b <= 2; continue;
    }
    if (c >= 0x80) { /* Extended char? */
        b |= 3; /* Eliminate NT flag
*/
#endif

#ifdef _EXCVT
    c = excvt[c-0x80]; /* Upper conversion (SBCS) */
#else
#if !_DF1S /* ASCII only cfg */
    return FR_INVALID_NAME;
#endif
#endif
#endif
#endif

```

```

        }

        if (IsDBCS1(c)) {                                /* Check if it is a DBC 1st
byte (always false on SBCS cfg) */

            d = (BYTE)p[si++];                          /* Get 2nd byte */

            if (!IsDBCS2(d) || i >= ni - 1)           /* Reject invalid DBC */

                return FR_INVALID_NAME;

            sfn[i++] = c;
            sfn[i++] = d;

        } else {                                         /* Single byte code */

            if (chk_chr("\\""+,:;<=>\?[]|\x7F", c))      /* Reject illegal chrs
for SFN */

                return FR_INVALID_NAME;

            if (IsUpper(c)) {                            /* ASCII large capital? */

                b |= 2;

            } else {

                if (IsLower(c)) {                      /* ASCII small capital? */

                    b |= 1; c -= 0x20;

                }

            }

            sfn[i++] = c;

        }

    }

    *path = &p[si];                                    /* Return pointer to
the next segment */

    c = (c <= ' ') ? NS_LAST : 0;                  /* Set last segment flag if end of path */

    if (!i) return FR_INVALID_NAME;                  /* Reject nul string */

    if (sfn[0] == 0xE5) sfn[0] = 0x05;             /* When first char collides with 0xE5,
replace it with 0x05 */

    if (ni == 8) b <= 2;
    if ((b & 0x03) == 0x01) c |= NS_EXT; /* NT flag (Name extension has only small
capital) */

    if ((b & 0x0C) == 0x04) c |= NS_BODY;          /* NT flag (Name body has only small
capital) */

    sfn[NS] = c;                                    /* Store NT flag, File name is created */

    return FR_OK;
#endif
}

```

```
/*-----*/
/* Get file information from directory entry */
/*-----*/
#if _FS_MINIMIZE <= 1
static
void get_fileinfo (           /* No return code */
    DIR *dj,                  /* Pointer to the directory object */
    FILINFO *fno               /* Pointer to the file information to be filled */
)
{
    UINT i;
    BYTE nt, *dir;
    TCHAR *p, c;

    p = fno->fname;
    if (dj->sect) {
        dir = dj->dir;
        nt = dir[DIR_NTres];          /* NT flag */
        for (i = 0; i < 8; i++) {     /* Copy name body */
            c = dir[i];
            if (c == ' ') break;
            if (c == 0x05) c = (TCHAR) 0xE5;
            if (_USE_LFN && (nt & NS_BODY) && IsUpper(c)) c += 0x20;
#if _LFN_UNICODE
            if (IsDBCS1(c) && i < 7 && IsDBCS2(dir[i+1]))
                c = (c << 8) | dir[++i];
            c = ff_convert(c, 1);
            if (!c) c = '?';
#endif
            *p++ = c;
        }
        if (dir[8] != ' ') {           /* Copy name extension */
            *p++ = '.';
            for (i = 8; i < 11; i++) {
                c = dir[i];

```

```

        if (c == ' ') break;

        if (_USE_LFN && (nt & NS_EXT) && IsUpper(c)) c += 0x20;

#if _LFN_UNICODE

        if (IsDBCS1(c) && i < 10 && IsDBCS2(dir[i+1]))
            c = (c << 8) | dir[++i];

        c = ff_convert(c, 1);

        if (!c) c = '?';

#endif

*p++ = c;

}

}

fno->fattrib = dir[DIR_Attr]; /* Attribute */

fno->fsize = LD_DWORD(dir+DIR_FileSize); /* Size */

fno->fdate = LD_WORD(dir+DIR_WrtDate); /* Date */

fno->ftime = LD_WORD(dir+DIR_WrtTime); /* Time */

}

*p = 0; /* Terminate SFN str by a \0 */

```

```

#endif _USE_LFN

if (fno->lfname && fno->lfsize) {

    TCHAR *tp = fno->lfname;
    WCHAR w, *lfn;

    i = 0;

    if (dj->sect && dj->lfn_idx != 0xFFFF) /* Get LFN if available */
        lfn = dj->lfn;

    while ((w = *lfn++) != 0) /* Get an LFN char */

#endif !_LFN_UNICODE

    conversion */

    if (!w) { i = 0; break; } /* Could not convert,
no LFN */

    if (_DF1S && w >= 0x100) /* Put 1st byte if it
is a DBC (always false on SBCS cfg) */
        tp[i++] = (TCHAR)(w >> 8);

#endif

    if (i >= fno->lfsize - 1) { i = 0; break; } /* Buffer
overflow, no LFN */

    tp[i++] = (TCHAR)w;

}

```

```
        tp[i] = 0;      /* Terminate the LFN str by a \0 */

    }

#endif
}

#endif /* _FS_MINIMIZE <= 1 */

/*-----*/
/* Follow a file path
 *-----*/

static

FRESULT follow_path ( /* FR_OK(0): successful, !=0: error code */

    DIR *dj,           /* Directory object to return last directory and
found object */

    const TCHAR *path   /* Full-path string to find a file or directory */

)

{

    FRESULT res;

    BYTE *dir, ns;

    /*-----*/

#if _FS_RPATH

    if (*path == '/' || *path == '\\') { /* There is a heading separator */

        path++; dj->sclust = 0;           /* Strip it and start from the root
dir */

    } else {                                /* No heading
separator */

        dj->sclust = dj->fs->cdir; /* Start from the current dir */

    }

#else

    if (*path == '/' || *path == '\\') /* Strip heading separator if exist */

        path++;

    dj->sclust = 0;                      /* Start from the root
dir */

#endif

    if ((UINT)*path < ' ')                /* Nul path means the start
directory itself */

        res = dir_sdi(dj, 0);

    /*-----*/
}
```

```

dj->dir = 0;

} else {                                     /* Follow path */

    for (;;) {

        res = create_name(dj, &path); /* Get a segment */

        if (res != FR_OK) break;

        res = dir_find(dj);           /* Find it */

        ns = *(dj->fn+NS);

        if (res != FR_OK) {          /* Failed to find the
object */

            if (res != FR_NO_FILE) break; /* Abort if any hard error
occurred */

            /* Object not found */

            if (_FS_RPATH && (ns & NS_DOT)) { /* If dot entry is not
exit */

                dj->sclust = 0; dj->dir = 0; /* It is the root dir
*/
                res = FR_OK;

                if (!(ns & NS_LAST)) continue;

            } else {
/* Could not find the object */

                if (!(ns & NS_LAST)) res = FR_NO_PATH;

            }
            break;
        }

        if (ns & NS_LAST) break;          /* Last segment match.
Function completed. */

        dir = dj->dir;                  /* There is
next segment. Follow the sub directory */

        if (!(dir[DIR_Attr] & AM_DIR)) { /* Cannot follow because it
is a file */

            res = FR_NO_PATH; break;

        }

        dj->sclust = LD_CLUST(dir);

    }

}

return res;
}

```

```

/*-----*/
/* Load boot record and check if it is an FAT boot record */
/*-----*/

static
BYTE check_fs (           /* 0:The FAT BR, 1:Valid BR but not an FAT, 2:Not a BR, 3:Disk
error */

    FATFS *fs,          /* File system object */

    DWORD sect           /* Sector# (lba) to check if it is an FAT boot record or not */

)
{
    if (disk_read(fs->drv, fs->win, sect, 1) != RES_OK) /* Load boot record */

        return 3;

    if (LD_WORD(&fs->win[BS_55AA]) != 0xAA55           /* Check record signature
(always placed at offset 510 even if the sector size is >512) */

        return 2;

    if ((LD_DWORD(&fs->win[BS_FilSysType]) & 0xFFFF) == 0x544146) /* Check "FAT"
string */

        return 0;

    if ((LD_DWORD(&fs->win[BS_FilSysType32]) & 0xFFFF) == 0x544146)

        return 0;

    return 1;
}

/*-----*/
/* Check if the file system object is valid or not */
/*-----*/

static
HRESULT chk_mounted ( /* FR_OK(0): successful, !=0: any error occurred */

    const TCHAR **path,      /* Pointer to pointer to the path name (drive number) */

    FATFS **rfs,           /* Pointer to pointer to the found file system object */

    BYTE chk_wp             /* !=0: Check media write protection for write
access */

)
{

```

```

    BYTE fmt, b, *tbl;
    UINT vol;
    DSTATUS stat;
    DWORD bsect, fasize, tsect, sysect, nclst, szbfat;
    WORD nrsv;
    const TCHAR *p = *path;
    FATFS *fs;

    /* Get logical drive number from the path name */
    vol = p[0] - '0';                                /* Is there a drive number?
*/
    if (vol <= 9 && p[1] == ':') {                    /* Found a drive number, get and
strip it */
        p += 2; *path = p;                            /* Return pointer to the path
name */
    } else {                                         /* No drive number is
given */
        #if _FS_RPATH
            vol = CurrVol;                          /* Use current drive */
        #else
            vol = 0;                               /* Use drive 0 */
        #endif
    }

    /* Check if the logical drive is valid or not */
    if (vol >= _VOLUMES)                           /* Is the drive number valid? */
        return FR_INVALID_DRIVE;
    *rfs = fs = FatFs[vol];                         /* Return pointer to the
corresponding file system object */
    if (!fs) return FR_NOT_ENABLED;                  /* Is the file system object
available? */

ENTER_FF(fs);                                     /* Lock file system */

    if (fs->fs_type) {                            /* If the logical drive has
been mounted */
        stat = disk_status(fs->drv);
        if (!(stat & STA_NOINIT)) {                /* and the physical drive is kept
initialized (has not been changed), */
            #if !_FS_READONLY
                if (chk_wp && (stat & STA_PROTECT)) /* Check write protection if
needed */
                    return FR_WRITE_PROTECTED;
            }
        }
    }
}

```

```

#endif

        return FR_OK;                                /* The file system object is
valid */

    }

}

/* The logical drive must be mounted. */

/* Following code attempts to mount a volume. (analyze BPB and initialize the fs
object) */

fs->fs_type = 0;                                /* Clear the file system
object */

fs->drv = (BYTE)LD2PD(vol);                     /* Bind the logical drive and a
physical drive */

stat = disk_initialize(fs->drv);                /* Initialize low level disk I/O layer */

if (stat & STA_NOINIT)                          /* Check if the initialization
succeeded */

    return FR_NOT_READY;                         /* Failed to initialize due to no
media or hard error */

#if _MAX_SS != 512                               /* Get disk sector size
(variable sector size cfg only) */

    if (disk_ioctl(fs->drv, GET_SECTOR_SIZE, &fs->ssize) != RES_OK)

        return FR_DISK_ERR;

#endif

#if !_FS_READONLY

    if (chk_wp && (stat & STA_PROTECT)) /* Check disk write protection if needed */

        return FR_WRITE_PROTECTED;

#endif

/* Search FAT partition on the drive. Supports only generic partitionings, FDISK
and SFD. */

fmt = check_fs(fs, bsect = 0);                   /* Check sector 0 if it is a VBR */

if (fmt == 1) {                                    /* Not an FAT-VBR, the
disk may be partitioned */

    /* Check the partition listed in top of the partition table */

tbl = &fs->win[MBR_Table + LD2PT(vol) * 16]; /* Partition table */

    if (tbl[4]) {                                /* Check the partition existing? */

        bsect = LD_DWORD(&tbl[8]);               /* Partition offset in LBA */

        fmt = check_fs(fs, bsect);                 /* Check the partition */

    }
}

if (fmt == 3) return FR_DISK_ERR;

```

```

        if (fmt) return FR_NO_FILESYSTEM;                                /* No FAT
volume is found */

/* Following code initializes the file system object */

        if (LD_WORD(fs->win+BPB_BytsPerSec) != SS(fs))           /* (BPB_BytsPerSec
must be equal to the physical sector size) */
                return FR_NO_FILESYSTEM;

        fasize = LD_WORD(fs->win+BPB_FATSz16);                      /* Number of
sectors per FAT */
        if (!fasize) fasize = LD_DWORD(fs->win+BPB_FATSz32);
        fs->fsize = fasize;

        fs->n_fats = b = fs->win[BPB_NumFATs];                      /* Number of
FAT copies */
        if (b != 1 && b != 2) return FR_NO_FILESYSTEM;                  /* (Must be 1 or 2) */
        fasize *= b;
        /* Number of sectors for FAT area */

        fs->csize = b = fs->win[BPB_SecPerClus];                      /* Number of sectors
per cluster */
        if (!b || (b & (b - 1))) return FR_NO_FILESYSTEM;             /* (Must be power of 2) */

        fs->n_rootdir = LD_WORD(fs->win+BPB_RootEntCnt);            /* Number of root directory
entries */
        if (fs->n_rootdir % (SS(fs) / 32)) return FR_NO_FILESYSTEM;   /* */
(BPB_RootEntCnt must be sector aligned) */

        tsect = LD_WORD(fs->win+BPB_TotSec16);                        /* Number of
sectors on the volume */
        if (!tsect) tsect = LD_DWORD(fs->win+BPB_TotSec32);

        nrsv = LD_WORD(fs->win+BPB_RsvdSecCnt);                      /* Number of
reserved sectors */
        if (!nrsv) return FR_NO_FILESYSTEM;                            /* */
(BPB_RsvdSecCnt must not be 0) */

/* Determine the FAT sub type */

        sysect = nrsv + fasize + fs->n_rootdir / (SS(fs) / 32);      /* RSV+FAT+DIR */
        if (tsect < sysect) return FR_NO_FILESYSTEM;                  /* (Invalid volume size) */
        nclst = (tsect - sysect) / fs->csize;                         /* Number of
clusters */
        if (!nclst) return FR_NO_FILESYSTEM;                            /* (Invalid volume
size) */

```

```

fmt = FS_FAT12;

if (nclst >= MIN_FAT16) fmt = FS_FAT16;
if (nclst >= MIN_FAT32) fmt = FS_FAT32;

/* Boundaries and Limits */

fs->n_fatent = nclst + 2;                                /* Number of FAT entries */

fs->database = bsect + sysect;                            /* Data start sector */
fs->fatbase = bsect + nrsv;                             /* FAT start sector */
fs->dirbase = LD_DWORD(fs->win+BPB_RootClus);        /* Root directory start cluster */

szbfat = fs->n_fatent * 4;                               /* (Required FAT size) */

} else {
    if (!fs->n_rootdir) return FR_NO_FILESYSTEM;          /* (BPB_RootEntCnt must not be 0) */
    fs->dirbase = fs->fatbase + fasize;                  /* Root directory start sector */
    szbfat = (fmt == FS_FAT16) ?                           /* (Required FAT size) */
        fs->n_fatent * 2 : fs->n_fatent * 3 / 2 + (fs->n_fatent & 1);
}

if (fs->fsize < (szbfat + (SS(fs) - 1)) / SS(fs)) /* (FAT size must not be less than FAT sectors) */

return FR_NO_FILESYSTEM;

#endif !_FS_READONLY

/* Initialize cluster allocation information */

fs->free_clust = 0xFFFFFFFF;
fs->last_clust = 0;

/* Get fsinfo if available */

if (fmt == FS_FAT32) {
    fs->fsi_flag = 0;
    fs->fsi_sector = bsect + LD_WORD(fs->win+BPB_FSIInfo);
    if (disk_read(fs->drv, fs->win, fs->fsi_sector, 1) == RES_OK &&
        LD_WORD(fs->win+BS_55AA) == 0xAA55 &&
        LD_DWORD(fs->win+FSI_LeadSig) == 0x41615252 &&
        LD_DWORD(fs->win+FSI_StrucSig) == 0x61417272) {
}

```

```
        fs->last_clust = LD_DWORD(fs->win+FSI_Nxt_Free);
        fs->free_clust = LD_DWORD(fs->win+FSI_Free_Count);
    }

}

#endif

fs->fs_type = fmt;           /* FAT sub-type */
fs->id = ++Fsid;            /* File system mount ID */
fs->winsect = 0;             /* Invalidate sector cache */
fs->wflag = 0;

#if _FS_RPATH
    fs->cdir = 0;          /* Current directory (root dir) */
#endif

#if _FS_SHARE
    clear_lock(fs);
#endif

return FR_OK;
}

/*-----
 * Check if the file/dir object is valid or not
 *-----
 */

static
HRESULT validate ( /* FR_OK(0): The object is valid, !=0: Invalid */
    FATFS *fs,           /* Pointer to the file system object */
    WORD id               /* Member id of the target object to be checked */
)
{
    if (!fs || !fs->fs_type || fs->id != id)
        return FR_INVALID_OBJECT;

    ENTER_FF(fs);          /* Lock file system */

    if (disk_status(fs->drv) & STA_NOINIT)
        return FR_NOT_READY;
```

```
    return FR_OK;
}

/*-----*/
Public Functions
-----*/



/*-----*/
/* Mount/Unmount a Logical Drive */
/*-----*/

FRESULT f_mount (
    BYTE vol,           /* Logical drive number to be mounted/unmounted */
    FATFS *fs           /* Pointer to new file system object (NULL for unmount) */
)
{
    FATFS *rfs;

    if (vol >= _VOLUMES)          /* Check if the drive number is valid */
        return FR_INVALID_DRIVE;
    rfs = FatFs[vol];             /* Get current fs object */

    if (rfs) {
#if _FS_SHARE
        clear_lock(rfs);
#endif
#if _FS_REENTRANT
        /* Discard sync object of the
         * current volume */
        if (!ff_del_syncobj(rfs->sobj)) return FR_INT_ERR;
#endif
        rfs->fs_type = 0;           /* Clear old fs object */
    }
}
```

```
}

if (fs) {
    fs->fs_type = 0;                                /* Clear new fs object */
#if _FS_REENTRANT                               /* Create sync object for the new
volume */

    if (!ff_cre_syncobj(vol, &fs->sobj)) return FR_INT_ERR;
#endif

}

FatFs[vol] = fs;                                    /* Register new fs object */

return FR_OK;
}

/*-----*/
/* Open or Create a File                         */
/*-----*/

FRESULT f_open (
    FIL *fp,                                     /* Pointer to the blank file object */
    const TCHAR *path,      /* Pointer to the file name */
    BYTE mode          /* Access mode and file open mode flags */
)
{
    FRESULT res;
    DIR dj;
    BYTE *dir;
    DEF_NAMEBUF;

    fp->fs = 0;                                  /* Clear file object */

#if !_FS_READONLY
    mode &= FA_READ | FA_WRITE | FA_CREATE_ALWAYS | FA_OPEN_ALWAYS | FA_CREATE_NEW;
    res = chk_mounted(&path, &dj.fs, (BYTE)(mode & ~FA_READ));
#else
    mode &= FA_READ;

```

```

    res = chk_mounted(&path, &dj.fs, 0);

#endif

INIT_BUF(dj);

if (res == FR_OK)

    res = follow_path(&dj, path); /* Follow the file path */

dir = dj.dir;

#if !_FS_READONLY      /* R/W configuration */

if (res == FR_OK) {

    if (!dir)      /* Current dir itself */

        res = FR_INVALID_NAME;

#endif _FS_SHARE

else

    res = chk_lock(&dj, (mode & ~FA_READ) ? 1 : 0);

#endif

}

/* Create or Open a file */

if (mode & (FA_CREATE_ALWAYS | FA_OPEN_ALWAYS | FA_CREATE_NEW)) {

    DWORD dw, cl;

    if (res != FR_OK) {                                     /* No file, create new
*/}

        if (res == FR_NO_FILE)                                /* There is no file to open,
create a new entry */

#if _FS_SHARE

        res = enq_lock(dj.fs) ? dir_register(&dj) : FR_TOO_MANY_OPEN_FILES;

#else

        res = dir_register(&dj);

#endif

    mode |= FA_CREATE_ALWAYS;                               /* File is created */

    dir = dj.dir;                                         /* New entry */

}

else {                                                 /* Any object
is already existing */

    if (mode & FA_CREATE_NEW) {                         /* Cannot create new */

        res = FR_EXIST;

    } else {

        if (dir[DIR_Attr] & (AM_RDO | AM_DIR))          /* Cannot
overwrite it (R/O or DIR) */

            res = FR_DENIED;

    }

}

```

```

        }

        if (res == FR_OK && (mode & FA_CREATE_ALWAYS)) { /* Truncate it if
overwrite mode */

            dw = get_fattime();                                /* Created
time */

            ST_DWORD(dir+DIR_CrtTime, dw);

            dir[DIR_Attr] = 0;                                /* Reset
attribute */

            ST_DWORD(dir+DIR_FileSize, 0);                      /* size = 0 */

            cl = LD_CLUST(dir);                               /* Get start
cluster */

            ST_CLUST(dir, 0);                                /* cluster = 0 */

        }

        dj.fs->wflag = 1;

        if (cl) {                                         /*

Remove the cluster chain if exist */

            dw = dj.fs->winsect;

            res = remove_chain(dj.fs, cl);

            if (res == FR_OK) {

                dj.fs->last_clust = cl - 1; /* Reuse the cluster
hole */

                res = move_window(dj.fs, dw);

            }

        }

    }

}

else { /* Open an existing file */

    if (res == FR_OK) {                                /* Follow
succeeded */

        if (dir[DIR_Attr] & AM_DIR) {                  /* It is a directory */

            res = FR_NO_FILE;

        } else {

            if ((mode & FA_WRITE) && (dir[DIR_Attr] & AM_RDO)) /* R/O
violation */

                res = FR_DENIED;

        }

    }

    if (res == FR_OK) {

        if (mode & FA_CREATE_ALWAYS)                  /* Set file change flag if
created or overwritten */

            mode |= FA__WRITTEN;

        fp->dir_sect = dj.fs->winsect;                 /* Pointer to the
directory entry */
    }
}

```

```

        fp->dir_ptr = dir;

#if _FS_SHARE

    fp->lockid = inc_lock(&dj, (mode & ~FA_READ) ? 1 : 0);
    if (!fp->lockid) res = FR_INT_ERR;

#endif

}

#else /* R/O configuration */

if (res == FR_OK) { /* Follow succeeded */

    if (!dir) { /* Current dir itself */
        res = FR_INVALID_NAME;
    } else {
        if (dir[DIR_Attr] & AM_DIR) /* It is a directory */
            res = FR_NO_FILE;
    }
}

#endif /* R/O configuration */

FREE_BUF();

if (res == FR_OK) {

    fp->flag = mode; /* File access mode */

    fp->org_clust = LD_CLUST(dir); /* File start cluster */

    fp->fsize = LD_DWORD(dir+DIR_FileSize); /* File size */

    fp->fptr = 0; /* File pointer */

    fp->dsect = 0;

#if _USE_FASTSEEK

    fp->cltbl = 0; /* No cluster link map table */

#endif

    fp->fs = dj.fs; fp->id = dj.fs->id; /* Validate file object */
}

LEAVE_FF(dj.fs, res);
}

/*-----*/

```

```

/* Read File */  

/*-----*/  
  

FRESULT f_read (  

    FIL *fp, /* Pointer to the file object */  

    void *buff, /* Pointer to data buffer */  

    UINT btr, /* Number of bytes to read */  

    UINT *br /* Pointer to number of bytes read */  

)  

{  

    FRESULT res;  

    DWORD clst, sect, remain;  

    UINT rcnt, cc;  

    BYTE csect, *rbuf = buff;  
  

    *br = 0; /* Initialize byte counter */  
  

    res = validate(fp->fs, fp->id); /* Check  
validity of the object */  

    if (res != FR_OK) LEAVE_FF(fp->fs, res);  

    if (fp->flag & FA__ERROR) /* Check abort  
flag */  

        LEAVE_FF(fp->fs, FR_INT_ERR);  

    if (!(fp->flag & FA_READ)) /* Check  
access mode */  

        LEAVE_FF(fp->fs, FR_DENIED);  

    remain = fp->fsize - fp->fptr;  

    if (btr > remain) btr = (UINT)remain; /* Truncate btr by  
remaining bytes */  
  

    /* Repeat until all data transferred */  

    for ( ; btr;  

        rbuf += rcnt, fp->fptr += rcnt, *br += rcnt, btr -= rcnt) {  

        if ((fp->fptr % SS(fp->fs)) == 0) /* On the sector  
boundary? */  

            csect = (BYTE)(fp->fptr / SS(fp->fs) & (fp->fs->csize - 1));  

/* Sector offset in the cluster */  

        if (!csect) /* On  
the cluster boundary? */  

            clst = (fp->fptr == 0) ? /* On the top  
of the file? */  

                fp->org_clust : get_fat(fp->fs, fp->curr_clust);

```

```

        if (clst <= 1) ABORT(fp->fs, FR_INT_ERR);

        if (clst == 0xFFFFFFFF) ABORT(fp->fs, FR_DISK_ERR);

        fp->curr_clust = clst;                                /* Update
current cluster */

    }

    sect = clust2sect(fp->fs, fp->curr_clust); /* Get current sector
*/
    if (!sect) ABORT(fp->fs, FR_INT_ERR);

    sect += csect;

    cc = btr / SS(fp->fs);                                /* When
remaining bytes >= sector size, */

    if (cc) {
        /* Read maximum contiguous sectors directly */

        if (csect + cc > fp->fs->csize)                      /* Clip at
cluster boundary */

            cc = fp->fs->csize - csect;

        if (disk_read(fp->fs->drv, rbuff, sect, (BYTE)cc) !=

RES_OK)

            ABORT(fp->fs, FR_DISK_ERR);

#if !_FS_READONLY && _FS_MINIMIZE <= 2                  /* Replace one of the
read sectors with cached data if it contains a dirty sector */

#endif

#endif

#endif

if (fp->fs->wflag && fp->fs->winsect - sect < cc)

    mem_cpy(rbuff + ((fp->fs->winsect - sect) * SS(fp-
>fs)), fp->fs->win, SS(fp->fs));

#else

if ((fp->flag & FA_DIRTY) && fp->dsect - sect < cc)

    mem_cpy(rbuff + ((fp->dsect - sect) * SS(fp->fs)),

fp->buf, SS(fp->fs));

#endif

#endif

rcnt = SS(fp->fs) * cc;                                  /*

Number of bytes transferred */

continue;

}

#endif

#endif

if (_FS_TINY)

#if !_FS_READONLY

if (fp->flag & FA_DIRTY) {                                /* Write
sector I/O buffer if needed */

    if (disk_write(fp->fs->drv, fp->buf, fp->dsect, 1) !=

RES_OK)

        ABORT(fp->fs, FR_DISK_ERR);

    fp->flag &= ~FA_DIRTY;
}
#endif

```

```

        if (fp->dsect != sect) {                                /* Fill sector
buffer with file data */
            if (disk_read(fp->fs->drv, fp->buf, sect, 1) != RES_OK)
                ABORT(fp->fs, FR_DISK_ERR);
        }
#endif
        fp->dsect = sect;
    }

    rcnt = SS(fp->fs) - (fp->fptr % SS(fp->fs)); /* Get partial sector data
from sector buffer */

    if (rcnt > btr) rcnt = btr;

#ifndef _FS_TINY
    if (move_window(fp->fs, fp->dsect))                      /* Move sector window
*/
        ABORT(fp->fs, FR_DISK_ERR);

    mem_cpy(rbuff, &fp->fs->win[fp->fptr % SS(fp->fs)], rcnt); /* Pick
partial sector */
#else
    mem_cpy(rbuff, &fp->buf[fp->fptr % SS(fp->fs)], rcnt);      /* Pick
partial sector */
#endif
}

LEAVE_FF(fp->fs, FR_OK);
}

#endif !_FS_READONLY
/*-----
/* Write File
/*-----*/
FRESULT f_write (
    FIL *fp,                                     /* Pointer to the file object */
    const void *buff,                            /* Pointer to the data to be written */
    UINT btw,                                    /* Number of bytes to write */
    UINT *bw                                     /* Pointer to number of bytes written */
)
{
    FRESULT res;

```

```

        DWORD clst, sect;
        UINT wcnt, cc;
        const BYTE *wbuff = buff;
        BYTE csect;

        *bw = 0;           /* Initialize byte counter */

        res = validate(fp->fs, fp->id);                      /* Check
validity of the object */

        if (res != FR_OK) LEAVE_FF(fp->fs, res);

        if (fp->flag & FA__ERROR)                                /* Check abort
flag */
            LEAVE_FF(fp->fs, FR_INT_ERR);

        if (!(fp->flag & FA_WRITE))                            /* Check
access mode */

            LEAVE_FF(fp->fs, FR_DENIED);

        if (fp->fsiz + btw < fp->fsiz) btw = 0;                /* File size cannot reach 4GB
*/
    }

    for ( ; btw; )                                              /*

Repeat until all data transferred */

        wbuff += wcnt, fp->fptr += wcnt, *bw += wcnt, btw -= wcnt) {

        if ((fp->fptr % SS(fp->fs)) == 0) {                  /* On the sector
boundary? */

            csect = (BYTE)(fp->fptr / SS(fp->fs) & (fp->fs->csize - 1));
            /* Sector offset in the cluster */

            if (!csect) {                                       /* On
the cluster boundary? */

                if (fp->fptr == 0) {                           /* On the top
of the file? */

                    clst = fp->org_clust;                     /* Follow from
the origin */

                    if (clst == 0)                               /* When
there is no cluster chain, */

                        fp->org_clust = clst = create_chain(fp->fs,
0);      /* Create a new cluster chain */

                    } else {                                 /* Middle or end of the file */

                        clst = create_chain(fp->fs, fp->curr_clust);
                        /* Follow or stretch cluster chain */

                    }
                }

                if (clst == 0) break;                         /* Could not
allocate a new cluster (disk full) */

                if (clst == 1) ABORT(fp->fs, FR_INT_ERR);
                if (clst == 0xFFFFFFFF) ABORT(fp->fs, FR_DISK_ERR);
            }
        }
    }
}

```

```

                fp->curr_clust = clst;                                /* Update
current cluster */

}

#endif _FS_TINY

        if (fp->fs->winsect == fp->dsect && move_window(fp->fs, 0))
/* Write back data buffer prior to following direct transfer */

                ABORT(fp->fs, FR_DISK_ERR);

#else

        if (fp->flag & FA_DIRTY) {                         /* Write back data buffer
prior to following direct transfer */

                if (disk_write(fp->fs->drv, fp->buf, fp->dsect, 1) !=

RES_OK)

                        ABORT(fp->fs, FR_DISK_ERR);

                fp->flag &= ~FA_DIRTY;
}
#endif

sect = clust2sect(fp->fs, fp->curr_clust); /* Get current sector
*/
if (!sect) ABORT(fp->fs, FR_INT_ERR);

sect += csect;

cc = btw / SS(fp->fs);                                /* When
remaining bytes >= sector size, */

if (cc) {
/* Write maximum contiguous sectors directly */

        if (csect + cc > fp->fs->csize)                      /* Clip at
cluster boundary */

                cc = fp->fs->csize - csect;

        if (disk_write(fp->fs->drv, wbuff, sect, (BYTE)cc) !=

RES_OK)

                ABORT(fp->fs, FR_DISK_ERR);
}

#endif _FS_TINY

        if (fp->fs->winsect - sect < cc) { /* Refill sector cache
if it gets dirty by the direct write */

                mem_cpy(fp->fs->win, wbuff + ((fp->fs->winsect -
sect) * SS(fp->fs)), SS(fp->fs));
fp->fs->wflag = 0;
}
#else

        if (fp->dsect - sect < cc) {                         /* Refill sector cache
if it gets dirty by the direct write */

                mem_cpy(fp->buf, wbuff + ((fp->dsect - sect) *
SS(fp->fs)), SS(fp->fs));
fp->flag &= ~FA_DIRTY;
}
#endif

```

```

        wcnt = SS(fp->fs) * cc;                                /*
Number of bytes transferred */

        continue;

    }

#endif _FS_TINY

    if (fp->fptr >= fp->fsize) {                           /* Avoid silly buffer
filling at growing edge */

        if (move_window(fp->fs, 0)) ABORT(fp->fs, FR_DISK_ERR);

        fp->fs->winsect = sect;
    }

#else

    if (fp->dsect != sect) {                                /* Fill sector
buffer with file data */

        if (fp->fptr < fp->fsize &&

            disk_read(fp->fs->drv, fp->buf, sect, 1) != RES_OK)

            ABORT(fp->fs, FR_DISK_ERR);
    }

#endif

    fp->dsect = sect;
}

wcnt = SS(fp->fs) - (fp->fptr % SS(fp->fs)); /* Put partial sector into
file I/O buffer */

if (wcnt > btw) wcnt = btw;

#endif _FS_TINY

if (move_window(fp->fs, fp->dsect))                      /* Move sector window
*/
    ABORT(fp->fs, FR_DISK_ERR);

mem_cpy(&fp->fs->win[fp->fptr % SS(fp->fs)], wbuff, wcnt); /* Fit partial
sector */

fp->fs->wflag = 1;

#else

mem_cpy(&fp->buf[fp->fptr % SS(fp->fs)], wbuff, wcnt);      /* Fit partial
sector */

fp->flag |= FA_DIRTY;
#endif

}

if (fp->fptr > fp->fsize) fp->fsize = fp->fptr;          /* Update file size if needed
*/
fp->flag |= FA_WRITTEN;                                     /* Set file
change flag */

LEAVE_FF(fp->fs, FR_OK);

```

```
}

/*-----*/
/* Synchronize the File Object */
/*-----*/

FRESULT f_sync (
    FIL *fp           /* Pointer to the file object */
)
{
    FRESULT res;
    DWORD tim;
    BYTE *dir;

    res = validate(fp->fs, fp->id);           /* Check validity of the object */
    if (res == FR_OK) {
        if (fp->flag & FA__WRITTEN) { /* Has the file been written? */
            #if !_FS_TINY /* Write-back dirty buffer */
                if (fp->flag & FA__DIRTY) {
                    if (disk_write(fp->fs->drv, fp->buf, fp->dsect, 1) != RES_OK)
                        LEAVE_FF(fp->fs, FR_DISK_ERR);
                    fp->flag &= ~FA__DIRTY;
                }
            #endif
        }
        /* Update the directory entry */
        res = move_window(fp->fs, fp->dir_sect);
        if (res == FR_OK) {
            dir = fp->dir_ptr;
            dir[DIR_Attr] |= AM_ARC;
            /* Set archive bit */
            ST_DWORD(dir+DIR_FileSize, fp->filesize);          /* Update file size */
            ST_CLUST(dir, fp->org_clust);                      /* Update start cluster */
            tim = get_fattime();
            /* Update updated time */
            ST_DWORD(dir+DIR_WrtTime, tim);
        }
    }
}
```

```
        fp->flag &= ~FA__WRITTEN;
        fp->fs->wflag = 1;
        res = sync(fp->fs);
    }
}

LEAVE_FF(fp->fs, res);
}

#endif /* !_FS_READONLY */

/*-----*/
/* Close File */
/*-----*/

FRESULT f_close (
    FIL *fp           /* Pointer to the file object to be closed */
)
{
    FRESULT res;

#ifdef _FS_READONLY
    FATFS *fs = fp->fs;
    res = validate(fs, fp->id);
    if (res == FR_OK) fp->fs = 0; /* Discard file object */
    LEAVE_FF(fs, res);
#else
    res = f_sync(fp);           /* Flush cached data */
#ifdef _FS_SHARE
    if (res == FR_OK) {          /* Decrement open counter */
#ifdef _FS_REENTRANT
        res = validate(fp->fs, fp->id);
        if (res == FR_OK) {
            res = dec_lock(fp->lockid);

```

```
        unlock_fs(fp->fs, FR_OK);
    }

#else
    res = dec_lock(fp->lockid);
#endif
}

#endif
}

if (res == FR_OK) fp->fs = 0; /* Discard file object */
return res;
#endif
}

/*-----
 * Current Drive/Directory Handlings
 *-----
 */

#if _FS_RPATH >= 1

FRESULT f_chdrive (
    BYTE drv             /* Drive number */
)
{
    if (drv >= _VOLUMES) return FR_INVALID_DRIVE;

    CurrVol = drv;

    return FR_OK;
}

FRESULT f_chdir (
    const TCHAR *path     /* Pointer to the directory path */
)
{
    FRESULT res;
```

```
DIR dj;
DEF_NAMEBUF;

res = chk_mounted(&path, &dj.fs, 0);
if (res == FR_OK) {
    INIT_BUF(dj);
    res = follow_path(&dj, path);           /* Follow the path */
    FREE_BUF();
    if (res == FR_OK) {                     /* Follow completed */
        if (!dj.dir) {
            dj.fs->cdir = dj.sclust;      /* Start directory itself */
        } else {
            if (dj.dir[DIR_Attr] & AM_DIR) /* Reached to the
directory */
                dj.fs->cdir = LD_CLUST(dj.dir);
            else
                res = FR_NO_PATH;         /* Reached but a file */
        }
    }
    if (res == FR_NO_FILE) res = FR_NO_PATH;
}

LEAVE_FF(dj.fs, res);
}

#endif _FS_RPATH >= 2
FRESULT f_getcwd (
    TCHAR *path,    /* Pointer to the directory path */
    UINT sz_path   /* Size of path */
)
{
    FRESULT res;
    DIR dj;
    UINT i, n;
    DWORD ccl;
    TCHAR *tp;
    FILINFO fno;
```

```

DEF_NAMEBUF;

*path = 0;

res = chk_mounted((const TCHAR**)&path, &dj.fs, 0); /* Get current volume */

if (res == FR_OK) {

    INIT_BUF(dj);

    i = sz_path;           /* Bottom of buffer (dir stack base) */

    dj.sclust = dj.fs->cdir;           /* Start to follow upper dir
from current dir */

    while ((ccl = dj.sclust) != 0) {     /* Repeat while current dir is a
sub-dir */

        res = dir_sdi(&dj, 1);           /* Get parent dir */

        if (res != FR_OK) break;

        res = dir_read(&dj);

        if (res != FR_OK) break;

        dj.sclust = LD_CLUST(dj.dir); /* Goto parent dir */

        res = dir_sdi(&dj, 0);

        if (res != FR_OK) break;

        do {                                /* Find the
entry links to the child dir */

            res = dir_read(&dj);

            if (res != FR_OK) break;

            if (ccl == LD_CLUST(dj.dir)) break; /* Found the entry */

            res = dir_next(&dj, 0);

        } while (res == FR_OK);

        if (res == FR_NO_FILE) res = FR_INT_ERR; /* It cannot be 'not
found'. */

        if (res != FR_OK) break;
    }

    #if _USE_LFN

        fno.lfname = path;

        fno.lfsize = i;
    #endif

    get_fileinfo(&dj, &fno);           /* Get the dir name and push
it to the buffer */

    tp = fno.fname;

    if (_USE_LFN && *path) tp = path;

    for (n = 0; tp[n]; n++) ;

    if (i < n + 3) {

        res = FR_NOT_ENOUGH_CORE; break;
    }

    while (n) path[--i] = tp[--n];
}

```

```

        path[--i] = '/';
    }

    tp = path;

    if (res == FR_OK) {
        *tp++ = '0' + CurrVol;                      /* Put drive number */
        *tp++ = ':';
        if (i == sz_path) {                         /* Root-dir */
            *tp++ = '/';
        } else {                                     /* Sub-dir */
            do {                                     /* Add stacked path str */
                *tp++ = path[i++];
            } while (i < sz_path);
        }
    }

    *tp = 0;
    FREE_BUF();
}

LEAVE_FF(dj.fs, res);
}

#endif /* _FS_RPATH >= 2 */
#endif /* _FS_RPATH >= 1 */

#endif _FS_MINIMIZE <= 2
/*-----*/
/* Seek File R/W Pointer
/*-----*/

FRESULT f_lseek (
    FIL *fp,          /* Pointer to the file object */
    DWORD ofs         /* File pointer from top of file */
)
{
    FRESULT res;

    res = validate(fp->fs, fp->id);           /* Check validity of the object */
}

```

```

    if (res != FR_OK) LEAVE_FF(fp->fs, res);

    if (fp->flag & FA__ERROR) /* Check abort flag */

        LEAVE_FF(fp->fs, FR_INT_ERR);

#endif _USE_FASTSEEK

    if (fp->cltbl) { /* Fast seek */

        DWORD cl, pcl, ncl, tcl, dsc, tlen, *tbl = fp->cltbl;
        BYTE csc;

        tlen = *tbl++;

        if (ofs == CREATE_LINKMAP) { /* Create link map table */

            cl = fp->org_clust;
            if (cl) {

                do { /* Get a fragment and store the top
and length */

                    pcl = cl; ncl = 0;
                    do { /* Get a fragment and store the top
and length */

                        pcl = cl; ncl++;

                        cl = get_fat(fp->fs, cl);
                        if (cl <= 1) ABORT(fp->fs, FR_INT_ERR);
                        if (cl == 0xFFFFFFFF) ABORT(fp->fs,
FR_DISK_ERR);

                        } while (cl == pcl + 1);
                        *tbl++ = ncl; *tbl++ = tcl;
                        tlen -= 2;

                    } while (cl < fp->fs->n_fatent);
                }
                *tbl = 0; /* Terminate table */
            }
        }

    } else { /* Fast seek */

        if (ofs > fp->fsiz) /* Clip offset at the file size */

            ofs = fp->fsiz;
        fp->fptr = ofs; /* Set file pointer */

        if (ofs) {

            dsc = (ofs - 1) / SS(fp->fs);
            cl = dsc / fp->fs->csize;
            for (;;) {

```

```

        ncl = *tbl++;
        if (!ncl) ABORT(fp->fs, FR_INT_ERR);
        if (cl < ncl) break;
        cl -= ncl; tbl++;
    }

    fp->curr_clust = cl + *tbl;
    csc = (BYTE)(dsc & (fp->fs->csize - 1));
    dsc = clust2sect(fp->fs, fp->curr_clust);
    if (!dsc) ABORT(fp->fs, FR_INT_ERR);
    dsc += csc;
    if (fp->fptr % SS(fp->fs) && dsc != fp->dsect) {

#ifdef !_FS_TINY
#ifdef !_FS_READONLY
        if (fp->flag & FA_DIRTY) { /* Flush dirty
            buffer if needed */
            if (disk_write(fp->fs->drv, fp->buf, fp-
>dsect, 1) != RES_OK)
                ABORT(fp->fs, FR_DISK_ERR);
            fp->flag &= ~FA_DIRTY;
        }
#endif
#endif
        if (disk_read(fp->fs->drv, fp->buf, dsc, 1) != RES_OK)
            ABORT(fp->fs, FR_DISK_ERR);
#endif
        fp->dsect = dsc;
    }
}

} else
#endif /* !_FS_READONLY */

/* Normal Seek */
{
    DWORD clst, bcs, nsect, ifptr;

    if (ofs > fp->fsize) /* In read-only mode,
clip offset with the file size */
#ifdef !_FS_READONLY
    && !(fp->flag & FA_WRITE)
#endif
}

```



```

        clst = get_fat(fp->fs, clst); /* Follow
cluster chain if not in write mode */

        if (clst == 0xFFFFFFFF) ABORT(fp->fs, FR_DISK_ERR);

        if (clst <= 1 || clst >= fp->fs->n_fatent)
ABORT(fp->fs, FR_INT_ERR);

        fp->curr_clust = clst;
        fp->fptr += bcs;
        ofs -= bcs;

    }

    fp->fptr += ofs;

    if (ofs % SS(fp->fs)) {
        nsect = clust2sect(fp->fs, clst); /* Current
sector */

        if (!nsect) ABORT(fp->fs, FR_INT_ERR);
        nsect += ofs / SS(fp->fs);
    }
}

if (fp->fptr % SS(fp->fs) && nsect != fp->dsect) {

#endif !_FS_TINY
#endif !_FS_READONLY

        if (fp->flag & FA_DIRTY) { /* Flush dirty buffer
if needed */

            if (disk_write(fp->fs->drv, fp->buf, fp->dsect, 1) !=

RES_OK)

                ABORT(fp->fs, FR_DISK_ERR);

            fp->flag &= ~FA_DIRTY;
        }
#endif

        if (disk_read(fp->fs->drv, fp->buf, nsect, 1) != RES_OK)

            ABORT(fp->fs, FR_DISK_ERR);
#endif

        fp->dsect = nsect;
    }

#endif !_FS_READONLY

        if (fp->fptr > fp->fs->size) /* Set change flag if the
file size is extended */

            fp->fs->size = fp->fptr;
            fp->flag |= FA_WRITTEN;
    }

#endif
}

```

```
LEAVE_FF(fp->fs, res);

}

#endif /* _FS_H */

/*-----*
 * Create a Directroy Object
 *-----*/
FRESULT f_opendir (
    DIR *dj,                      /* Pointer to directory object to create */
    const TCHAR *path             /* Pointer to the directory path */
)
{
    FRESULT res;
    DEF_NAMEBUF;

    res = chk_mounted(&path, &dj->fs, 0);
    if (res == FR_OK) {
        INIT_BUF(*dj);
        res = follow_path(dj, path);           /* Follow the path to the
directory */
        FREE_BUF();
        if (res == FR_OK) {                    /* Follow
completed */
            if (dj->dir) {                  /* It is not
the root dir */
                if (dj->dir[DIR_Attr] & AM_DIR) { /* The object is a
directory */
                    dj->sclust = LD_CLUST(dj->dir);
                } else {                      /* The
object is not a directory */
                    res = FR_NO_PATH;
                }
            }
            if (res == FR_OK) {
                dj->id = dj->fs->id;
                res = dir_sdi(dj, 0);          /* Rewind dir */
            }
        }
    }
}
```

```
        }

        if (res == FR_NO_FILE) res = FR_NO_PATH;

    }

    LEAVE_FF(dj->fs, res);

}

/*-----*/
/* Read Directory Entry in Sequence */
/*-----*/

FRESULT f_readdir (
    DIR *dj,                      /* Pointer to the open directory object */
    FILINFO *fno                   /* Pointer to file information to return */
)
{
    FRESULT res;
    DEF_NAMEBUF;

    res = validate(dj->fs, dj->id);           /* Check validity of the
object */

    if (res == FR_OK) {
        if (!fno) {

            res = dir_sdi(dj, 0);             /* Rewind the directory
object */

        } else {
            INIT_BUF(*dj);
            res = dir_read(dj);             /* Read an directory
item */

        if (res == FR_NO_FILE) {             /* Reached end of dir */
            dj->sect = 0;
            res = FR_OK;
        }
        if (res == FR_OK) {                 /* A valid entry is
found */
            get_fileinfo(dj, fno);         /* Get the object information
*/
        }
    }
}
```

```
        res = dir_next(dj, 0);           /* Increment index for next
 */
        if (res == FR_NO_FILE) {
            dj->sect = 0;
            res = FR_OK;
        }
    }
    FREE_BUF();
}
}

LEAVE_FF(dj->fs, res);
}

#endif /* _FS_MINIMIZE */

/*-----*/
/* Get File Status
/*-----*/
FRESULT f_stat (
    const TCHAR *path,      /* Pointer to the file path */
    FILINFO *fno           /* Pointer to file information to return */
)
{
    FRESULT res;
    DIR dj;
    DEF_NAMEBUF;

    res = chk_mounted(&path, &dj.fs, 0);
    if (res == FR_OK) {
        INIT_BUF(dj);
        res = follow_path(&dj, path); /* Follow the file path */
        if (res == FR_OK)             /* Follow completed */
            if (dj.dir)              /* Found an object */
                get_fileinfo(&dj, fno);
            else                      /* It is root dir */
                res = FR_INVALID_NAME;
    }
}
```

```
    }

    FREE_BUF();

}

LEAVE_FF(dj.fs, res);

}

#endif /* _FS_READONLY */

/*-----*/
/* Get Number of Free Clusters */
/*-----*/

FRESULT f_getfree (
    const TCHAR *path,      /* Pointer to the logical drive number (root dir) */
    DWORD *nclst,          /* Pointer to the variable to return number of free
clusters */
    FATFS **fatfs          /* Pointer to pointer to corresponding file system object
to return */
)
{
    FRESULT res;

    DWORD n, clst, sect, stat;
    UINT i;
    BYTE fat, *p;

    /* Get drive number */
    res = chk_mounted(&path, fatfs, 0);
    if (res == FR_OK) {
        /* If free_clust is valid, return it without full cluster scan */
        if ((*fatfs)->free_clust <= (*fatfs)->n_fatent - 2) {
            *nclst = (*fatfs)->free_clust;
        } else {
            /* Get number of free clusters */
            fat = (*fatfs)->fs_type;
            n = 0;
            if (fat == FS_FAT12) {
                clst = 2;
```

```

        do {
            stat = get_fat(*fatfs, clst);
            if (stat == 0xFFFFFFFF) { res = FR_DISK_ERR; break; }
            if (stat == 1) { res = FR_INT_ERR; break; }
            if (stat == 0) n++;
        } while (++clst < (*fatfs)->n_fatent);
    } else {
        clst = (*fatfs)->n_fatent;
        sect = (*fatfs)->fatbase;
        i = 0; p = 0;
        do {
            if (!i) {
                res = move_window(*fatfs, sect++);
                if (res != FR_OK) break;
                p = (*fatfs)->win;
                i = SS(*fatfs);
            }
            if (fat == FS_FAT16) {
                if (LD_WORD(p) == 0) n++;
                p += 2; i -= 2;
            } else {
                if ((LD_DWORD(p) & 0xFFFFFFFF) == 0) n++;
                p += 4; i -= 4;
            }
        } while (--clst);
    }
    (*fatfs)->free_clust = n;
    if (fat == FS_FAT32) (*fatfs)->fsi_flag = 1;
    *nclst = n;
}
LEAVE_FF(*fatfs, res);
}

/*
-----*/
/* Truncate File */
```

```
/*-----*/
FRESULT f_truncate (
    FIL *fp           /* Pointer to the file object */
)
{
    FRESULT res;
    DWORD ncl;

    res = validate(fp->fs, fp->id);                      /* Check validity of the object */
    if (res == FR_OK) {
        if (fp->flag & FA__ERROR) {                         /* Check abort flag */
            res = FR_INT_ERR;
        } else {
            if (!(fp->flag & FA_WRITE)) {                  /* Check access mode */
                res = FR_DENIED;
            }
        }
    }
    if (res == FR_OK) {
        if (fp->fsize > fp->fptr) {
            fp->fsize = fp->fptr; /* Set file size to current R/W point */
            fp->flag |= FA_WRITTEN;
            if (fp->fptr == 0) { /* When set file size to zero, remove entire
cluster chain */
                res = remove_chain(fp->fs, fp->org_clust);
                fp->org_clust = 0;
            } else {                                         /* When truncate a part of
the file, remove remaining clusters */
                ncl = get_fat(fp->fs, fp->curr_clust);
                res = FR_OK;
                if (ncl == 0xFFFFFFFF) res = FR_DISK_ERR;
                if (ncl == 1) res = FR_INT_ERR;
                if (res == FR_OK && ncl < fp->fs->n_fatent) {
                    res = put_fat(fp->fs, fp->curr_clust, 0xFFFFFFFF);
                    if (res == FR_OK) res = remove_chain(fp->fs, ncl);
                }
            }
        }
        if (res != FR_OK) fp->flag |= FA__ERROR;
    }
}
```

```
}

    LEAVE_FF(fp->fs, res);

}

/*-----
 * Delete a File or Directory
 *-----
 */

FRESULT f_unlink (
    const TCHAR *path           /* Pointer to the file or directory path */
)
{
    FRESULT res;
    DIR dj, sdj;
    BYTE *dir;
    DWORD dclst;
    DEF_NAMEBUF;

    res = chk_mounted(&path, &dj.fs, 1);
    if (res == FR_OK) {
        INIT_BUF(dj);
        res = follow_path(&dj, path);           /* Follow the file path */
        if (_FS_RPATH && res == FR_OK && (dj.fn[NS] & NS_DOT))
            res = FR_INVALID_NAME;             /* Cannot remove dot entry */
#if _FS_SHARE
        if (res == FR_OK) res = chk_lock(&dj, 2); /* Cannot remove open file */
#endif
        if (res == FR_OK) {                   /* The object is
accessible */
            dir = dj.dir;
            if (!dir) {
                res = FR_INVALID_NAME;         /* Cannot remove the start
directory */
            } else {
                if (dir[DIR_Attr] & AM_RDO)
```

```

        res = FR_DENIED;           /* Cannot remove R/O
object */

    }

dclst = LD_CLUST(dir);

if (res == FR_OK && (dir[DIR_Attr] & AM_DIR)) {      /* Is it a
sub-dir? */

    if (dclst < 2) {

        res = FR_INT_ERR;

    } else {

        mem_cpy(&sdj, &dj, sizeof(DIR));      /* Check if
the sub-dir is empty or not */

        sdj.sclust = dclst;

        res = dir_sdi(&sdj, 2);           /* Exclude dot
entries */

        if (res == FR_OK) {

            res = dir_read(&sdj);

            if (res == FR_OK           /* Not
empty dir */

#ifndef _FS_RPATH

                || dclst == sdj.fs->cdir      /* Current dir
*/
#endif

#endif

        ) res = FR_DENIED;

        if (res == FR_NO_FILE) res = FR_OK; /*

Empty */

    }

}

if (res == FR_OK) {

    res = dir_remove(&dj);           /* Remove the directory entry
*/
}

if (res == FR_OK) {

    if (dclst)                      /* Remove the
cluster chain if exist */

        res = remove_chain(dj.fs, dclst);

    if (res == FR_OK) res = sync(dj.fs);

}

}

FREE_BUF();

}

LEAVE_FF(dj.fs, res);
}

```

```
/*-----*/
/* Create a Directory */
/*-----*/

FRESULT f_mkdir (
    const TCHAR *path           /* Pointer to the directory path */
)
{
    FRESULT res;
    DIR dj;
    BYTE *dir, n;
    DWORD dsc, dcl, pcl, tim = get_fattime();
    DEF_NAMEBUF;

    res = chk_mounted(&path, &dj.fs, 1);
    if (res == FR_OK) {
        INIT_BUF(dj);
        res = follow_path(&dj, path);                      /* Follow the file path */
        if (res == FR_OK) res = FR_EXIST;                  /* Any object with same name
is already existing */
        if (_FS_RPATH && res == FR_NO_FILE && (dj.fn[NS] & NS_DOT))
            res = FR_INVALID_NAME;
        if (res == FR_NO_FILE) {                            /* Can create a new
directory */
            dcl = create_chain(dj.fs, 0);                 /* Allocate a cluster for the
new directory table */
            res = FR_OK;
            if (dcl == 0) res = FR_DENIED;                /* No space to
allocate a new cluster */
            if (dcl == 1) res = FR_INT_ERR;
            if (dcl == 0xFFFFFFFF) res = FR_DISK_ERR;
            if (res == FR_OK)                             /* Flush FAT
*/
            res = move_window(dj.fs, 0);
            if (res == FR_OK) {                          /* Initialize
the new directory table */
                dsc = clust2sect(dj.fs, dcl);

```

```

        dir = dj.fs->win;
        mem_set(dir, 0, SS(dj.fs));
        mem_set(dir+DIR_Name, ' ', 8+3);      /* Create ".." entry */
        dir[DIR_Name] = '.';
        dir[DIR_Attr] = AM_DIR;
        ST_DWORD(dir+DIR_WrtTime, tim);
        ST_CLUST(dir, dcl);
        mem_cpy(dir+32, dir, 32);           /* Create ".." */

entry */
        dir[33] = '.'; pcl = dj.sclust;
        if (dj.fs->fs_type == FS_FAT32 && pcl == dj.fs->dirbase)
            pcl = 0;
        ST_CLUST(dir+32, pcl);
        for (n = dj.fs->csizer; n; n--) {      /* Write dot entries
and clear following sectors */
            dj.fs->winsect = dsc++;
            dj.fs->wflag = 1;
            res = move_window(dj.fs, 0);
            if (res != FR_OK) break;
            mem_set(dir, 0, SS(dj.fs));
        }
    }

    if (res == FR_OK) res = dir_register(&dj); /* Register the object
to the directory */
    if (res != FR_OK) {
        remove_chain(dj.fs, dcl);           /* Could not
register, remove cluster chain */
    } else {
        dir = dj.dir;
        dir[DIR_Attr] = AM_DIR;           /*
Attribute */
        ST_DWORD(dir+DIR_WrtTime, tim);   /* Created
time */
        ST_CLUST(dir, dcl);             /*
Table start cluster */
        dj.fs->wflag = 1;
        res = sync(dj.fs);
    }
}

FREE_BUF();

}

LEAVE_FF(dj.fs, res);

```

```
}

/*-----*/
/* Change Attribute */
/*-----*/

FRESULT f_chmod (
    const TCHAR *path,      /* Pointer to the file path */
    BYTE value,             /* Attribute bits */
    BYTE mask               /* Attribute mask to change */
)

{
    FRESULT res;
    DIR dj;
    BYTE *dir;
    DEF_NAMEBUF;

    res = chk_mounted(&path, &dj.fs, 1);
    if (res == FR_OK) {
        INIT_BUF(dj);
        res = follow_path(&dj, path);           /* Follow the file path */
        FREE_BUF();
        if (_FS_RPATH && res == FR_OK && (dj.fn[NS] & NS_DOT))
            res = FR_INVALID_NAME;
        if (res == FR_OK) {
            dir = dj.dir;
            if (!dir) {                         /* Is it a
root directory? */
                res = FR_INVALID_NAME;
            } else {                           /* File or sub
directory */
                mask &= AM_RDO|AM_HID|AM_SYS|AM_ARC; /* Valid attribute
mask */
                dir[DIR_Attr] = (value & mask) | (dir[DIR_Attr] &
(BYTE)~mask); /* Apply attribute change */
                dj.fs->wflag = 1;
                res = sync(dj.fs);
            }
        }
    }
}
```

```
        }

    }

    LEAVE_FF(dj.fs, res);

}

/*-----*/
/* Change Timestamp */
/*-----*/

FRESULT f_utime (
    const TCHAR *path,      /* Pointer to the file/directory name */
    const FILINFO *fno       /* Pointer to the time stamp to be set */
)
{
    FRESULT res;
    DIR dj;
    BYTE *dir;
    DEF_NAMEBUF;

    res = chk_mounted(&path, &dj.fs, 1);
    if (res == FR_OK) {
        INIT_BUF(dj);
        res = follow_path(&dj, path); /* Follow the file path */
        FREE_BUF();
        if (_FS_RPATH && res == FR_OK && (dj.fn[NS] & NS_DOT))
            res = FR_INVALID_NAME;
        if (res == FR_OK) {
            dir = dj.dir;
            if (!dir) {                                /* Root directory */
                res = FR_INVALID_NAME;
            } else {                                    /* File or sub-
directory */
                ST_WORD(dir+DIR_WrtTime, fno->ftime);
                ST_WORD(dir+DIR_WrtDate, fno->fdate);
            }
        }
    }
}
```

```
    dj.fs->wflag = 1;
    res = sync(dj.fs);
}
}

LEAVE_FF(dj.fs, res);
}

/*-----
/* Rename File/Directory
/*-----*/
```

FRESULT f_rename (

```
    const TCHAR *path_old, /* Pointer to the old name */
    const TCHAR *path_new /* Pointer to the new name */
)
```

{

```
    FRESULT res;
    DIR djo, djn;
    BYTE buf[21], *dir;
    DWORD dw;
    DEF_NAMEBUF;
```

res = chk_mounted(&path_old, &djo.fs, 1);
 if (res == FR_OK) {
 djn.fs = djo.fs;
 INIT_BUF(djo);
 res = follow_path(&djo, path_old); /* Check old object */
 if (_FS_RPATH && res == FR_OK && (djo.fn[NS] & NS_DOT))
 res = FR_INVALID_NAME;
 #if _FS_SHARE
 if (res == FR_OK) res = chk_lock(&djo, 2);
 #endif
 if (res == FR_OK) { /* Old object
is found */
 /* Old object
is found */

```

        if (!djo.dir) {                                /* Is
root dir? */

            res = FR_NO_FILE;

        } else {

            mem_cpy(buf, djo.dir+DIR_Attr, 21);      /* Save the
object information except for name */

            mem_cpy(&djn, &djo, sizeof(DIR));          /* Check new
object */

            res = follow_path(&djn, path_new);

            if (res == FR_OK) res = FR_EXIST;           /* The new
object name is already existing */

            if (res == FR_NO_FILE) {                     /* Is
it a valid path and no name collision? */

/* Start critical section that any interruption or error can cause cross-link */

            res = dir_register(&djn);                  /* Register the new entry */

            if (res == FR_OK) {

                dir = djn.dir;
/* Copy object information except for name */

                mem_cpy(dir+13, buf+2, 19);

                dir[DIR_Attr] = buf[0] | AM_ARC;

                djn.fs->wflag = 1;

                if (djo.sclust != djn.sclust &&
(dir[DIR_Attr] & AM_DIR)) {                   /* Update .. entry in the directory if needed */

                    dw = clust2sect(djn.fs,
LD_CLUST(dir));

                    if (!dw) {

                        res = FR_INT_ERR;

                    } else {

                        res = move_window(djn.fs,
dw);

                        dir = djn.fs->win+32; /* ..

entry */

                        if (res == FR_OK && dir[1] ==
'.')

                            dw = (djn.fs->fs_type
== FS_FAT32 && djn.sclust == djn.fs->dirbase) ? 0 : djn.sclust;

                            ST_CLUST(dir, dw);

                            djn.fs->wflag = 1;

                        }

                    }

                }

                if (res == FR_OK) {

                    res = dir_remove(&djo);

                    if (res == FR_OK)

/* Remove old entry */

```

```
        res = sync(djo.fs);
    }

}

/* End critical section */

}

}

FREE_BUF();

}

LEAVE_FF(djo.fs, res);

}

#endif /* !_FS_READONLY */

#endif /* _FS_MINIMIZE == 0 */

#endif /* _FS_MINIMIZE <= 1 */

#endif /* _FS_MINIMIZE <= 2 */

/*-----*/
/* Forward data to the stream directly (available on only tiny cfg) */
/*-----*/
#if _USE_FORWARD && _FS_TINY

FRESULT f_forward (
    FIL *fp,                                /* Pointer to the file object
*/
    UINT (*func) (const BYTE*,UINT),          /* Pointer to the streaming function */
    UINT btr,                                /* Number of bytes to forward
*/
    UINT *bf                                 /* Pointer to number of bytes
forwarded */
)
{
    FRESULT res;
    DWORD remain, clst, sect;
    UINT rcnt;
    BYTE csect;

    *bf = 0;           /* Initialize byte counter */

```

```

        res = validate(fp->fs, fp->id);                                /* Check
validity of the object */

        if (res != FR_OK) LEAVE_FF(fp->fs, res);

        if (fp->flag & FA_ERROR)                                         /* Check error
flag */

            LEAVE_FF(fp->fs, FR_INT_ERR);

        if (!(fp->flag & FA_READ))                                         /* Check
access mode */

            LEAVE_FF(fp->fs, FR_DENIED);

remain = fp->fsize - fp->fptr;

        if (btr > remain) btr = (UINT)remain;                            /* Truncate btr by
remaining bytes */

for ( ; btr && (*func)(0, 0); )                                         /* Repeat
until all data transferred or stream becomes busy */

        fp->fptr += rcnt, *bf += rcnt, btr -= rcnt) {

        csect = (BYTE)(fp->fptr / SS(fp->fs) & (fp->fs->csize - 1));      /*

Sector offset in the cluster */

        if ((fp->fptr % SS(fp->fs)) == 0) {                           /* On the sector
boundary? */

            if (!csect) {                                                 /* On
the cluster boundary? */

                clst = (fp->fptr == 0) ?                               /* On the top
of the file? */

                    fp->org_clust : get_fat(fp->fs, fp->curr_clust);

                if (clst <= 1) ABORT(fp->fs, FR_INT_ERR);

                if (clst == 0xFFFFFFFF) ABORT(fp->fs, FR_DISK_ERR);

                fp->curr_clust = clst;                                 /* Update
current cluster */

            }

            sect = clust2sect(fp->fs, fp->curr_clust);    /* Get current data sector */

            if (!sect) ABORT(fp->fs, FR_INT_ERR);

            sect += csect;

            if (move_window(fp->fs, sect))                           /* Move sector
window */

                ABORT(fp->fs, FR_DISK_ERR);

            fp->dsect = sect;

            rcnt = SS(fp->fs) - (WORD)(fp->fptr % SS(fp->fs)); /* Forward data from
sector window */

            if (rcnt > btr) rcnt = btr;

            rcnt = (*func)((&fp->fs->win[(WORD)fp->fptr % SS(fp->fs)], rcnt);

            if (!rcnt) ABORT(fp->fs, FR_INT_ERR);

```

```
}

    LEAVE_FF(fp->fs, FR_OK);

}

#endif /* _USE_FORWARD */

#endif _USE_MKFS && !_FS_READONLY

/*-----
/* Create File System on the Drive
/*-----*/
#define N_ROOTDIR      512          /* Multiple of 32 */
#define N_FATS          1           /* 1 or 2 */

FRESULT f_mkfs (
    BYTE drv,                  /* Logical drive number */
    BYTE sfd,                  /* Partitioning rule 0:FDISK, 1:SFD */
    UINT au                    /* Allocation unit size [bytes] */
)

{
    static const WORD vst[] = { 1024,    512,    256,    128,     64,      32,      16,       8,
4,      2,      0};

    static const WORD cst[] = {32768, 16384, 8192, 4096, 2048, 16384, 8192, 4096,
2048, 1024, 512};

    BYTE fmt, md, *tbl;
    DWORD n_clst, vs, n, wsect;
    UINT i;

    DWORD b_vol, b_fat, b_dir, b_data; /* Offset (LBA) */
    DWORD n_vol, n_rsv, n_fat, n_dir; /* Size */
    FATFS *fs;
    DSTATUS stat;

    /* Check mounted drive and clear work area */
    if (drv >= _VOLUMES) return FR_INVALID_DRIVE;
    fs = FatFs[drv];
    if (!fs) return FR_NOT_ENABLED;
    fs->fs_type = 0;
```

```

drv = LD2PD(drv);

/* Get disk statics */

stat = disk_initialize(drv);
if (stat & STA_NOINIT) return FR_NOT_READY;
if (stat & STA_PROTECT) return FR_WRITE_PROTECTED;

#if _MAX_SS != 512                                /* Get disk sector size */
    if (disk_ioctl(drv, GET_SECTOR_SIZE, &SS(fs)) != RES_OK)
        return FR_DISK_ERR;
#endif

if (disk_ioctl(drv, GET_SECTOR_COUNT, &n_vol) != RES_OK || n_vol < 128)
    return FR_DISK_ERR;

b_vol = (sfd) ? 0 : 63;             /* Volume start sector */
n_vol -= b_vol;
if (au & (au - 1)) au = 0;          /* Check validity of the allocation unit
size */
if (!au) {                         /* AU auto selection */
    vs = n_vol / (2000 / (SS(fs) / 512));
    for (i = 0; vs < vst[i]; i++) ;
    au = cst[i];
}
au /= SS(fs);                      /* Number of sectors per cluster */
if (au == 0) au = 1;
if (au > 128) au = 128;

/* Pre-compute number of clusters and FAT syb-type */
n_clst = n_vol / au;
fmt = FS_FAT12;
if (n_clst >= MIN_FAT16) fmt = FS_FAT16;
if (n_clst >= MIN_FAT32) fmt = FS_FAT32;

/* Determine offset and size of FAT structure */
if (fmt == FS_FAT32) {
    n_fat = ((n_clst * 4) + 8 + SS(fs) - 1) / SS(fs);
    n_rsv = 32;
    n_dir = 0;
} else {
    n_fat = (fmt == FS_FAT12) ? (n_clst * 3 + 1) / 2 + 3 : (n_clst * 2) + 4;
    n_fat = (n_fat + SS(fs) - 1) / SS(fs);
    n_rsv = 1;
}

```

```

n_dir = N_ROOTDIR * 32UL / SS(fs);
}

b_fat = b_vol + n_rsv;                                /* FAT area start sector */
b_dir = b_fat + n_fat * N_FATS;                        /* Directory area start sector */
b_data = b_dir + n_dir;                                 /* Data area start sector */
if (n_vol < b_data + au) return FR_MKFS_ABORTED;      /* Too small volume */

/* Align data start sector to erase block boundary (for flash memory media) */
if (disk_ioctl(drv, GET_BLOCK_SIZE, &n) != RES_OK || !n || n > 32768) n = 1;
n = (b_data + n - 1) & ~n;                            /* Next nearest erase block from current
data start */

n = (n - b_data) / N_FATS;

if (fmt == FS_FAT32) {                                    /* FAT32: Move FAT offset */
    n_rsv += n;
    b_fat += n;
} else {                                                 /* FAT12/16: Expand FAT size */
    n_fat += n;
}

/* Determine number of cluster and final check of validity of the FAT sub-type */
n_clst = (n_vol - n_rsv - n_fat * N_FATS - n_dir) / au;
if ((fmt == FS_FAT16 && n_clst < MIN_FAT16)
    || (fmt == FS_FAT32 && n_clst < MIN_FAT32))
    return FR_MKFS_ABORTED;

/* Create partition table if required */
if (sfd) {
    md = 0xF0;
} else {
    DWORD n_disk = b_vol + n_vol;

    mem_set(fs->win, 0, SS(fs));
    tbl = fs->win+MBR_Table;
    ST_DWORD(tbl, 0x00010180);                          /* Partition start in CHS */
    if (n_disk < 63UL * 255 * 1024) {                  /* Partition end in CHS */
        n_disk = n_disk / 63 / 255;
        tbl[7] = (BYTE)n_disk;
        tbl[6] = (BYTE)((n_disk >> 2) | 63);
    } else {
        ST_WORD(&tbl[6], 0xFFFF);
    }
}

```

```

        }

tbl[5] = 254;

if (fmt != FS_FAT32)                                /* System ID */

    tbl[4] = (n_vol < 0x10000) ? 0x04 : 0x06;

else

    tbl[4] = 0x0c;

ST_DWORD(tbl+8, 63);                                /* Partition start in LBA */

ST_DWORD(tbl+12, n_vol);                            /* Partition size in LBA */

ST_WORD(tbl+64, 0xAA55);                            /* Signature */

if (disk_write(drv, fs->win, 0, 1) != RES_OK)

    return FR_DISK_ERR;

md = 0xF8;

}

/* Create volume boot record */

tbl = fs->win;                                     /* Clear sector */

mem_set(tbl, 0, SS(fs));

mem_cpy(tbl, "\xEB\xFE\x90" "MSDOS5.0", 11); /* Boot code, OEM name */

i = SS(fs);                                         /* Sector size */

ST_WORD(tbl+BPB_BytsPerSec, i);

tbl[BPB_SecPerClus] = (BYTE)au;                     /* Sectors per cluster */

ST_WORD(tbl+BPB_RsvdSecCnt, n_rsv);                /* Reserved sectors */

tbl[BPB_NumFATs] = N_FATS;                          /* Number of FATs */

i = (fmt == FS_FAT32) ? 0 : N_ROOTDIR;             /* Number of rootdir entries */

ST_WORD(tbl+BPB_RootEntCnt, i);

if (n_vol < 0x10000) {                             /* Number of total sectors */

    ST_WORD(tbl+BPB_TotSec16, n_vol);

} else {

    ST_DWORD(tbl+BPB_TotSec32, n_vol);

}

tbl[BPB_Media] = md;                               /* Media descriptor */

ST_WORD(tbl+BPB_SecPerTrk, 63);                   /* Number of sectors per track */

ST_WORD(tbl+BPB_NumHeads, 255);                  /* Number of heads */

ST_DWORD(tbl+BPB_HiddSec, b_vol);                 /* Hidden sectors */

n = get_fattime();                                /* Use current time as VSN */

if (fmt == FS_FAT32) {

    ST_DWORD(tbl+BS_VolID32, n);                  /* VSN */

    ST_DWORD(tbl+BPB_FATSz32, n_fat);            /* Number of sectors per FAT */

```

```

        ST_DWORD(tbl+BPB_RootClus, 2);           /* Root directory start
cluster (2) */

        ST_WORD(tbl+BPB_FSIInfo, 1);             /* FSInfo record offset
(VBR+1) */

        ST_WORD(tbl+BPB_BkBootSec, 6);           /* Backup boot record offset
(VBR+6) */

        tbl[BS_DrvNum32] = 0x80;                 /* Drive number */

        tbl[BS_Bootsig32] = 0x29;                /* Extended boot signature */

        mem_cpy(tbl+BS_VolLab32, "NO NAME      " "FAT32      ", 19); /* Volume
label, FAT signature */

    } else {

        ST_DWORD(tbl+BS_VolID, n);             /* VSN */

        ST_WORD(tbl+BPB_FATSz16, n_fat);       /* Number of sectors per FAT */

        tbl[BS_DrvNum] = 0x80;                 /* Drive number */

        tbl[BS_Bootsig] = 0x29;                /* Extended boot
signature */

        mem_cpy(tbl+BS_VolLab, "NO NAME      " "FAT      ", 19); /* Volume
label, FAT signature */

    }

    ST_WORD(tbl+BS_55AA, 0xAA55);           /* Signature (Offset is fixed here
regardless of sector size) */

if (disk_write(drv, tbl, b_vol, 1) != RES_OK)/* Write original (VBR) */

    return FR_DISK_ERR;

if (fmt == FS_FAT32)                      /* Write backup (VBR+6) */

    disk_write(drv, tbl, b_vol + 6, 1);

/* Initialize FAT area */

wsect = b_fat;

for (i = 0; i < N_FATS; i++) {

    mem_set(tbl, 0, SS(fs));              /* 1st sector of the FAT */

    n = md;                                /* Media
descriptor byte */

    if (fmt != FS_FAT32) {

        n |= (fmt == FS_FAT12) ? 0x00FFFF00 : 0xFFFFFFF00;

        ST_DWORD(tbl+0, n);                 /* Reserve cluster #0-
1 (FAT12/16) */

    } else {

        n |= 0xFFFFFFF00;

        ST_DWORD(tbl+0, n);                 /* Reserve cluster #0-
1 (FAT32) */

        ST_DWORD(tbl+4, 0xFFFFFFFF);

        ST_DWORD(tbl+8, 0xFFFFFFFF); /* Reserve cluster #2 for root dir
*/
    }
}

```

```

        if (disk_write(drv, tbl, wsect++, 1) != RES_OK)
            return FR_DISK_ERR;

        mem_set(tbl, 0, SS(fs));                                /* Fill following FAT entries
with zero */

        for (n = 1; n < n_fat; n++) {                         /* This loop may take a time on
FAT32 volume due to many single sector write */
            if (disk_write(drv, tbl, wsect++, 1) != RES_OK)
                return FR_DISK_ERR;
        }
    }

/* Initialize root directory */

i = (fmt == FS_FAT32) ? au : n_dir;
do {
    if (disk_write(drv, tbl, wsect++, 1) != RES_OK)
        return FR_DISK_ERR;
} while (--i);

#endif /* USE_ERASE /* Erase data area if needed */

{
    DWORD eb[2];

    eb[0] = wsect; eb[1] = wsect + n_clst * au - 1;
    disk_ioctl(drv, CTRL_ERASE_SECTOR, eb);
}

/* Create FSInfo if needed */

if (fmt == FS_FAT32) {
    ST_WORD(tbl+BS_55AA, 0xAA55);
    ST_DWORD(tbl+FSI_LeadSig, 0x41615252);
    ST_DWORD(tbl+FSI_StrucSig, 0x61417272);
    ST_DWORD(tbl+FSI_Free_Count, n_clst - 1);
    ST_DWORD(tbl+FSI_Nxt_Free, 0xFFFFFFFF);
    disk_write(drv, tbl, b_vol + 1, 1); /* Write original (VBR+1) */
    disk_write(drv, tbl, b_vol + 7, 1); /* Write backup (VBR+7) */
}

return (disk_ioctl(drv, CTRL_SYNC, (void*)0) == RES_OK) ? FR_OK : FR_DISK_ERR;
}

```

```

#endif /* _USE_MKFS && !_FS_READONLY */

#if _USE_STRFUNC
/*-----*/
/* Get a string from the file */
/*-----*/
TCHAR* f_gets (
    TCHAR* buff,      /* Pointer to the string buffer to read */
    int len,          /* Size of string buffer (characters) */
    FIL* fil         /* Pointer to the file object */
)
{
    int n = 0;
    TCHAR c, *p = buff;
    BYTE s[2];
    UINT rc;

    while (n < len - 1) { /* Read bytes until buffer gets filled */
        f_read(fil, s, 1, &rc);
        if (rc != 1) break; /* Break on EOF or error */
        c = s[0];
        #if _LFN_UNICODE
        /* Read a character in UTF-8
        encoding */
        if (c >= 0x80) {
            if (c < 0xC0) continue; /* Skip stray trailer */
            if (c < 0xE0) { /* Two-byte sequense */
                f_read(fil, s, 1, &rc);
                if (rc != 1) break;
                c = ((c & 0x1F) << 6) | (s[0] & 0x3F);
                if (c < 0x80) c = '?';
            } else {
                if (c < 0xF0) { /* Three-byte sequense */
                    f_read(fil, s, 2, &rc);
                    if (rc != 2) break;
                }
            }
        }
    }
}

```

```
        c = (c << 12) | ((s[0] & 0x3F) << 6) | (s[1] &
0x3F);

        if (c < 0x800) c = '?';

    } else { /* Reject four-byte sequence
*/ c = '?';
}

}

}

#endif

#if _USE_STRFUNC >= 2
    if (c == '\r') continue; /* Strip '\r' */
#endif

*p++ = c;
n++;
if (c == '\n') break; /* Break on EOL */
}

*p = 0;

return n ? buff : 0; /* When no data read (eof or error), return
with error. */
}

#endif !_FS_READONLY
#include <stdarg.h>
/*-----*/
/* Put a character to the file */
/*-----*/
int f_putc (
    TCHAR c, /* A character to be output */
    FIL* fil /* Pointer to the file object */
)
{
    UINT bw, btw;
    BYTE s[3];

#if _USE_STRFUNC >= 2
    if (c == '\n') f_putc ('\r', fil); /* LF -> CRLF conversion */
#endif
```

```
#if _LFN_UNICODE      /* Write the character in UTF-8 encoding */

    if (c < 0x80) {                      /* 7-bit */
        s[0] = (BYTE)c;
        btw = 1;
    } else {
        if (c < 0x800) {                  /* 11-bit */
            s[0] = (BYTE)(0xC0 | (c >> 6));
            s[1] = (BYTE)(0x80 | (c & 0x3F));
            btw = 2;
        } else {                         /* 16-bit */
            s[0] = (BYTE)(0xE0 | (c >> 12));
            s[1] = (BYTE)(0x80 | ((c >> 6) & 0x3F));
            s[2] = (BYTE)(0x80 | (c & 0x3F));
            btw = 3;
        }
    }
}

#else                           /* Write the character without conversion */

    s[0] = (BYTE)c;
    btw = 1;
#endif

f_write(fil, s, btw, &bw);           /* Write the char to the file */

return (bw == btw) ? 1 : EOF; /* Return the result */
}

/*-----*/
/* Put a string to the file */
/*-----*/

int f_puts (
    const TCHAR* str,      /* Pointer to the string to be output */
    FIL* fil               /* Pointer to the file object */
)
{
    int n;
```

```
for (n = 0; *str; str++, n++) {  
    if (f_putc(*str, fil) == EOF) return EOF;  
}  
return n;  
}  
  
/*-----*/  
/* Put a formatted string to the file */  
/*-----*/  
int f_printf (  
    FIL* fil, /* Pointer to the file object */  
    const TCHAR* str, /* Pointer to the format string */  
    ... /* Optional arguments... */  
)  
{  
    va_list arp;  
    BYTE f, r;  
    UINT i, w;  
    ULONG val;  
    TCHAR c, d, s[16];  
    int res, cc;  
  
    va_start(arp, str);  
  
    for (cc = res = 0; cc != EOF; res += cc) {  
        c = *str++;  
        if (c == 0) break; /* End of string */  
        if (c != '%') { /* Non escape character */  
            cc = f_putc(c, fil);  
            if (cc != EOF) cc = 1;  
            continue;  
        }  
        w = f = 0;  
        c = *str++;  
        if (c == '0') { /* Flag: '0' padding */
```

```

f = 1; c = *str++;
}

while (IsDigit(c)) { /* Precision */
    w = w * 10 + c - '0';
    c = *str++;
}

if (c == 'l' || c == 'L') { /* Prefix: Size is long int */
    f |= 2; c = *str++;
}

if (!c) break;
d = c;

if (IsLower(d)) d -= 0x20;

switch (d) { /* Type is... */
case 'S': /* String */
    cc = f_puts(va_arg(ar, TCHAR*), fil); continue;
case 'C': /* Character */
    cc = f_putc((TCHAR)va_arg(ar, int), fil); continue;
case 'B': /* Binary */
    r = 2; break;
case 'O': /* Octal */
    r = 8; break;
case 'D': /* Signed decimal */
case 'U': /* Unsigned decimal */
    r = 10; break;
case 'X': /* Hexdecimal */
    r = 16; break;
default: /* Unknown */
    cc = f_putc(c, fil); continue;
}

/* Get an argument */

val = (f & 2) ? va_arg(ar, long) : ((d == 'D') ? (long)va_arg(ar, int)
: va_arg(ar, unsigned int));

if (d == 'D' && (val & 0x80000000)) {
    val = 0 - val;
    f |= 4;
}

/* Put it in numeral string */

i = 0;
do {

```

```
d = (TCHAR)(val % r); val /= r;
if (d > 9) {
    d += 7;
    if (c == 'x') d += 0x20;
}
s[i++] = d + '0';

} while (val && i < sizeof(s) / sizeof(s[0]));

if (f & 4) s[i++] = '-';
cc = 0;
while (i < w-- && cc != EOF) {
    cc = f_putc((TCHAR)((f & 1) ? '0' : ' '), fil);
    res++;
}
do {
    cc = f_putc(s[--i], fil);
    res++;
} while (i && cc != EOF);
if (cc != EOF) cc = 0;
}

va_end(arp);

return (cc == EOF) ? cc : res;
}

#endif /* !_FS_READONLY */
#endif /* _USE_STRFUNC */

/*
*****Copyright
(c)*****
**
** http://www.powermcu.com
**
*-----File Info-----
-----
** File name: SPI_MSD_Driver.c
** Descriptions: The SPI SD Card application function
**
*----------
**
** Created by: Ya Dan
```

```
** Created date:          2011-1-4
** Version:               v1.0
** Descriptions:          The original version
**
*-----
-----
** Modified by:
** Modified date:
** Version:
** Descriptions:
**
***** */
***** */

/* Includes -----*/
#include "SPI_MSD_Driver.h"
#include <stdio.h>

/* Private define -----*/
#define PRINT_INFO 1

/* Private variables -----*/
MSD_CARDINFO CardInfo;

***** */
* Function Name : _spi_read_write
* Description   : None
* Input         : - data:
* Output        : None
* Return        : None
* Attention     : None
***** */
inline int _spi_read_write(uint8_t data)
{
    /* wait for current SSP activity complete */
    while (SSP_GetStatus(LPC_SSP0, SSP_STAT_BUSY) == SET);

    SSP_SendData(LPC_SSP0, (uint16_t) data);

    while (SSP_GetStatus(LPC_SSP0, SSP_STAT_RXFIFO_NOTEMPTY) == RESET);
```

```
    return (SSP_ReceiveData(LPC_SSP0));  
}  
  
/******  
* Function Name : MSD_SPI_Configuration  
* Description   : SD Card SPI Configuration  
* Input         : None  
* Output        : None  
* Return        : None  
* Attention     : None  
*****/  
void MSD_SPI_Configuration(void)  
{  
    PINSEL_CFG_Type PinCfg;  
    SSP_CFG_Type SSP_ConfigStruct;  
    /*  
     * Initialize SPI pin connect  
     * P1.25 - SD_CD - used as GPIO  
     * P1.20 - SCK  
     * P1.21 - SD_CS - used as GPIO  
     * P1.23 - MISO  
     * P1.24 - MOSI  
     */  
    PinCfg.Funcnum = 3;  
    PinCfg.OpenDrain = 0;  
    PinCfg.Pinmode = 0;  
    PinCfg.Portnum = 1;  
    PinCfg.Pinnum = 20;  
    PINSEL_ConfigPin(&PinCfg);  
    PinCfg.Pinnum = 23;  
    PINSEL_ConfigPin(&PinCfg);  
    PinCfg.Pinnum = 24;  
    PINSEL_ConfigPin(&PinCfg);  
    PinCfg.Pinnum = 21;  
    PinCfg.Funcnum = 0;  
    PINSEL_ConfigPin(&PinCfg);  
    PinCfg.Portnum = 1;  
    PinCfg.Pinnum = 25;
```

```
PinCfg.Funcnum = 0;
PINSEL_ConfigPin(&PinCfg);

/* P1.25 SD_CD is Input */
GPIO_SetDir(SD_CD_PORT_NUM, (1<<SD_CD_PIN_NUM), 0);
GPIO_SetValue(SD_CD_PORT_NUM, (1<<SD_CD_PIN_NUM));

/* P1.21 SD_CS is output */
GPIO_SetDir(SD_CS_PORT_NUM, (1<<SD_CS_PIN_NUM), 1);
GPIO_SetValue(SD_CS_PORT_NUM, (1<<SD_CS_PIN_NUM));

/* initialize SSP configuration structure to default */
SSP_ConfigStructInit(&SSP_ConfigStruct);
/* Initialize SSP peripheral with parameter given in structure above */
SSP_Init(LPC_SSP0, &SSP_ConfigStruct);
/* Enable SSP peripheral */

_card_disable();

MSD_SPIHighSpeed(0);

SSP_Cmd(LPC_SSP0, ENABLE);
}

/*****************/
* Function Name : MSD_SPIHighSpeed
* Description   : SD Card Speed Set
* Input         : - b_high: 1 = 4MHz, 0 = 500KHz
* Output        : None
* Return        : None
* Attention     : None
*****************/
void MSD_SPIHighSpeed(uint8_t b_high)
{
    SSP_CFG_Type SSP_ConfigStruct;

    /* initialize SSP configuration structure to default */
    SSP_ConfigStructInit(&SSP_ConfigStruct);

    /* Speed select */
}
```

```
    if(b_high == 0)
    {
        SSP_ConfigStruct.ClockRate = 500000;
    }
    else
    {
        SSP_ConfigStruct.ClockRate = 4000000;
    }

    SSP_Init(LPC_SSP0, &SSP_ConfigStruct);

    SSP_Cmd(LPC_SSP0, ENABLE);
}

/*********************************************
 * Function Name : MSD_Init
 * Description   : SD Card initialization
 * Input         : None
 * Output        : None
 * Return        : None
 * Attention     : None
*****************************************/
int MSD_Init(void)
{
    uint8_t r1;
    uint8_t buff[6] = {0};
    uint16_t retry;

    MSD_SPI_Configuration();

    /* Check , if no card insert */
    if( _card_insert() )
    {
#ifdef PRINT_INFO
        printf("There is no card detected! \r\n");
#endif
    }
    /* FATFS error flag */
    return -1;
}
```

```
/* Power on and delay some times */

for(retry=0; retry<0x100; retry++)
{
    _card_power_on();
}

/* Satrt send 74 clocks at least */

for(retry=0; retry<10; retry++)
{
    _spi_read_write(DUMMY_BYTE);
}

/* Start send CMD0 till return 0x01 means in IDLE state */

for(retry=0; retry<0xFFFF; retry++)
{
    r1 = _send_command(CMD0, 0, 0x95);

    if(r1 == 0x01)
    {
        retry = 0;
        break;
    }
}

/* Timeout return */

if(retry == 0xFFFF)
{
    #ifdef PRINT_INFO
        printf("Reset card into IDLE state failed!\r\n");
    #endif
    return 1;
}

/* Get the card type, version */

r1 = _send_command_hold(CMD8, 0x1AA, 0x87);
/* r1=0x05 -> V1.0 */
if(r1 == 0x05)
{
    CardInfo.CardType = CARDTYPE_SDV1;
```

```
/* End of CMD8, chip disable and dummy byte */
_card_disable();
_spi_read_write(DUMMY_BYTE);

/* SD1.0/MMC start initialize */

/* Send CMD55+ACMD41, No-response is a MMC card, otherwise is a SD1.0 card */
for(retry=0; retry<0xFFFF; retry++)
{
    r1 = _send_command(CMD55, 0, 0);                                /* should be return
0x01 */

    if(r1 != 0x01)
    {

#ifndef PRINT_INFO
        printf("Send CMD55 should return 0x01, response=0x%02x\r\n", r1);
#endif

    return r1;
}

r1 = _send_command(ACMD41, 0, 0);                                /* should be return
0x00 */

if(r1 == 0x00)
{
    retry = 0;
    break;
}

}

/* MMC card initialize start */

if(retry == 0xFFFF)
{
    for(retry=0; retry<0xFFFF; retry++)
    {
        r1 = _send_command(CMD1, 0, 0);                                /* should be return
0x00 */

        if(r1 == 0x00)
        {
            retry = 0;
            break;
        }
    }
}
```

```
    /* Timeout return */

    if(retry == 0xFFFF)
    {

#define PRINT_INFO
        printf("Send CMD1 should return 0x00, response=0x%02x\r\n", r1);

#endif
        return 2;
    }

CardInfo.CardType = CARDTYPE_MMC;

#define PRINT_INFO
    printf("Card Type : MMC\r\n");
#endif
}

/* SD1.0 card detected, print information */

#define PRINT_INFO
else
{
    printf("Card Type : SD V1\r\n");
}

#endif

/* Set spi speed high */
MSD_SPIHighSpeed(1);

/* CRC disable */
r1 = _send_command(CMD59, 0, 0x01);
if(r1 != 0x00)
{
#define PRINT_INFO
    printf("Send CMD59 should return 0x00, response=0x%02x\r\n", r1);

#endif
    return r1;           /* response error, return r1 */
}

/* Set the block size */
r1 = _send_command(CMD16, MSD_BLOCKSIZE, 0xFF);
if(r1 != 0x00)
{
```

```
#ifdef PRINT_INFO
    printf("Send CMD16 should return 0x00, response=0x%02x\r\n", r1);
#endif
    return r1;           /* response error, return r1 */
}

}

/* r1=0x01 -> V2.x, read OCR register, check version */
else if(r1 == 0x01)
{
    /* 4Bytes returned after CMD8 sent */
    buff[0] = _spi_read_write(DUMMY_BYTE);          /* should be
0x00 */
    buff[1] = _spi_read_write(DUMMY_BYTE);          /* should be
0x00 */
    buff[2] = _spi_read_write(DUMMY_BYTE);          /* should be
0x01 */
    buff[3] = _spi_read_write(DUMMY_BYTE);          /* should be
0xAA */

    /* End of CMD8, chip disable and dummy byte */
    _card_disable();
    _spi_read_write(DUMMY_BYTE);

    /* Check voltage range be 2.7-3.6V */
    if(buff[2]==0x01 && buff[3]==0xAA)
    {
        for(retry=0; retry<0xFFFF; retry++)
        {
            r1 = _send_command(CMD55, 0, 0);          /* should be
return 0x01 */
            if(r1!=0x01)
            {

#ifdef PRINT_INFO
                printf("Send CMD55 should return 0x01,
response=0x%02x\r\n", r1);
#endif
                return r1;
            }

            r1 = _send_command(ACMD41, 0x40000000, 0); /* should be return
0x00 */
            if(r1 == 0x00)
```

```
    {

        retry = 0;

        break;

    }

}

/* Timeout return */

if(retry == 0xFFFF)

{

#endif PRINT_INFO

printf("Send ACMD41 should return 0x00, response=0x%02x\r\n", r1);

#endif

return 3;

}

/* Read OCR by CMD58 */

r1 = _send_command_hold(CMD58, 0, 0);

if(r1!=0x00)

{

#endif PRINT_INFO

printf("Send CMD58 should return 0x00, response=0x%02x\r\n", r1);

#endif

return r1; /* response error, return r1 */

}

buff[0] = _spi_read_write(DUMMY_BYTE);

buff[1] = _spi_read_write(DUMMY_BYTE);

buff[2] = _spi_read_write(DUMMY_BYTE);

buff[3] = _spi_read_write(DUMMY_BYTE);

/* End of CMD58, chip disable and dummy byte */

_card_disable();

_spi_read_write(DUMMY_BYTE);

/* OCR -> CCS(bit30) 1: SDV2HC 0: SDV2 */

if(buff[0] & 0x40)

{

CardInfo.CardType = CARDTYPE_SDV2HC;

#endif PRINT_INFO
```

```
        printf("Card Type : SD V2HC\r\n");
#endif
    }
else
{
    CardInfo.CardType = CARDTYPE_SDV2;
#endifdef PRINT_INFO
    printf("Card Type : SD V2\r\n");
#endif
}

/* Set spi speed high */
MSD_SPIHighSpeed(1);
}

return 0;
}

/****************************************
* Function Name : MSD_GetCardInfo
* Description   : Get SD Card Information
* Input         : None
* Output        : None
* Return        : 0£°NO_ERR; TRUE: Error
* Attention     : None
****************************************/
int MSD_GetCardInfo(PMSD_CARDINFO cardinfo)
{
    uint8_t r1;
    uint8_t CSD_Tab[16];
    uint8_t CID_Tab[16];

    /* Send CMD9, Read CSD */
    r1 = _send_command(CMD9, 0, 0xFF);
    if(r1 != 0x00)
    {
        return r1;
    }
}
```

```
if(_read_buffer(CSD_Tab, 16, RELEASE))
{
    return 1;
}

/* Send CMD10, Read CID */
r1 = _send_command(CMD10, 0, 0xFF);
if(r1 != 0x00)
{
    return r1;
}

if(_read_buffer(CID_Tab, 16, RELEASE))
{
    return 2;
}

/* Byte 0 */
cardinfo->CSD.CSDStruct = (CSD_Tab[0] & 0xC0) >> 6;
cardinfo->CSD.SysSpecVersion = (CSD_Tab[0] & 0x3C) >> 2;
cardinfo->CSD.Reserved1 = CSD_Tab[0] & 0x03;

/* Byte 1 */
cardinfo->CSD.TAAC = CSD_Tab[1] ;
/* Byte 2 */
cardinfo->CSD.NSAC = CSD_Tab[2];
/* Byte 3 */
cardinfo->CSD.MaxBusClkFrec = CSD_Tab[3];
/* Byte 4 */
cardinfo->CSD.CardComdClasses = CSD_Tab[4] << 4;
/* Byte 5 */
cardinfo->CSD.CardComdClasses |= (CSD_Tab[5] & 0xF0) >> 4;
cardinfo->CSD.RdBlockLen = CSD_Tab[5] & 0x0F;
/* Byte 6 */
cardinfo->CSD.PartBlockRead = (CSD_Tab[6] & 0x80) >> 7;
cardinfo->CSD.WrBlockMisalign = (CSD_Tab[6] & 0x40) >> 6;
cardinfo->CSD.RdBlockMisalign = (CSD_Tab[6] & 0x20) >> 5;
cardinfo->CSD.DSRImpl = (CSD_Tab[6] & 0x10) >> 4;
cardinfo->CSD.Reserved2 = 0; /* Reserved */
cardinfo->CSD.DeviceSize = (CSD_Tab[6] & 0x03) << 10;
```

```
/* Byte 7 */
cardinfo->CSD.DeviceSize |= (CSD_Tab[7]) << 2;

/* Byte 8 */
cardinfo->CSD.DeviceSize |= (CSD_Tab[8] & 0xC0) >> 6;
cardinfo->CSD.MaxRdCurrentVDDMin = (CSD_Tab[8] & 0x38) >> 3;
cardinfo->CSD.MaxRdCurrentVDDMax = (CSD_Tab[8] & 0x07);

/* Byte 9 */
cardinfo->CSD.MaxWrCurrentVDDMin = (CSD_Tab[9] & 0xE0) >> 5;
cardinfo->CSD.MaxWrCurrentVDDMax = (CSD_Tab[9] & 0x1C) >> 2;
cardinfo->CSD.DeviceSizeMul = (CSD_Tab[9] & 0x03) << 1;

/* Byte 10 */
cardinfo->CSD.DeviceSizeMul |= (CSD_Tab[10] & 0x80) >> 7;
cardinfo->CSD.EraseGrSize = (CSD_Tab[10] & 0x7C) >> 2;
cardinfo->CSD.EraseGrMul = (CSD_Tab[10] & 0x03) << 3;

/* Byte 11 */
cardinfo->CSD.EraseGrMul |= (CSD_Tab[11] & 0xE0) >> 5;
cardinfo->CSD.WrProtectGrSize = (CSD_Tab[11] & 0x1F);

/* Byte 12 */
cardinfo->CSD.WrProtectGrEnable = (CSD_Tab[12] & 0x80) >> 7;
cardinfo->CSD.ManDeflECC = (CSD_Tab[12] & 0x60) >> 5;
cardinfo->CSD.WrSpeedFact = (CSD_Tab[12] & 0x1C) >> 2;
cardinfo->CSD.MaxWrBlockLen = (CSD_Tab[12] & 0x03) << 2;

/* Byte 13 */
cardinfo->CSD.MaxWrBlockLen |= (CSD_Tab[13] & 0xc0) >> 6;
cardinfo->CSD.WriteBlockPaPartial = (CSD_Tab[13] & 0x20) >> 5;
cardinfo->CSD.Reserved3 = 0;
cardinfo->CSD.ContentProtectAppli = (CSD_Tab[13] & 0x01);

/* Byte 14 */
cardinfo->CSD.FileFormatGrouop = (CSD_Tab[14] & 0x80) >> 7;
cardinfo->CSD.CopyFlag = (CSD_Tab[14] & 0x40) >> 6;
cardinfo->CSD.PermWrProtect = (CSD_Tab[14] & 0x20) >> 5;
cardinfo->CSD.TempWrProtect = (CSD_Tab[14] & 0x10) >> 4;
cardinfo->CSD.FileFormat = (CSD_Tab[14] & 0x0C) >> 2;
cardinfo->CSD.ECC = (CSD_Tab[14] & 0x03);

/* Byte 15 */
cardinfo->CSD.CSD_CRC = (CSD_Tab[15] & 0xFE) >> 1;
cardinfo->CSD.Reserved4 = 1;

if(cardinfo->CardType == CARDTYPE_SDV2HC)
```

```
{  
    /* Byte 7 */  
    cardinfo->CSD.DeviceSize = (uint16_t)(CSD_Tab[8]) * 256;  
    /* Byte 8 */  
    cardinfo->CSD.DeviceSize += CSD_Tab[9] ;  
}  
  
cardinfo->Capacity = cardinfo->CSD.DeviceSize * MSD_BLOCKSIZE * 1024;  
cardinfo->BlockSize = MSD_BLOCKSIZE;  
  
/* Byte 0 */  
cardinfo->CID.ManufacturerID = CID_Tab[0];  
/* Byte 1 */  
cardinfo->CID.OEM_AppliID = CID_Tab[1] << 8;  
/* Byte 2 */  
cardinfo->CID.OEM_AppliID |= CID_Tab[2];  
/* Byte 3 */  
cardinfo->CID.ProdName1 = CID_Tab[3] << 24;  
/* Byte 4 */  
cardinfo->CID.ProdName1 |= CID_Tab[4] << 16;  
/* Byte 5 */  
cardinfo->CID.ProdName1 |= CID_Tab[5] << 8;  
/* Byte 6 */  
cardinfo->CID.ProdName1 |= CID_Tab[6];  
/* Byte 7 */  
cardinfo->CID.ProdName2 = CID_Tab[7];  
/* Byte 8 */  
cardinfo->CID.ProdRev = CID_Tab[8];  
/* Byte 9 */  
cardinfo->CID.ProdSN = CID_Tab[9] << 24;  
/* Byte 10 */  
cardinfo->CID.ProdSN |= CID_Tab[10] << 16;  
/* Byte 11 */  
cardinfo->CID.ProdSN |= CID_Tab[11] << 8;  
/* Byte 12 */  
cardinfo->CID.ProdSN |= CID_Tab[12];  
/* Byte 13 */  
cardinfo->CID.Reserved1 |= (CID_Tab[13] & 0xF0) >> 4;  
/* Byte 14 */
```

```
cardinfo->CID.ManufactDate = (CID_Tab[13] & 0x0F) << 8;  
/* Byte 15 */  
cardinfo->CID.ManufactDate |= CID_Tab[14];  
/* Byte 16 */  
cardinfo->CID.CID_CRC = (CID_Tab[15] & 0xFE) >> 1;  
cardinfo->CID.Reserved2 = 1;  
  
return 0;  
}  
  
/********************************************/  
* Function Name : _read_buffer  
* Description   : None  
* Input         : - *buff:  
*                  - len:  
*                  - release:  
* Output        : None  
* Return        : 0f°NO_ERR; TRUE: Error  
* Attention     : None  
******************************************/  
int _read_buffer(uint8_t *buff, uint16_t len, uint8_t release)  
{  
    uint8_t r1;  
    uint16_t retry;  
  
    /* Card enable, Prepare to read */  
    _card_enable();  
  
    /* Wait start-token 0xFE */  
    for(retry=0; retry<2000; retry++)  
    {  
        r1 = _spi_read_write(DUMMY_BYTE);  
        if(r1 == 0xFE)  
        {  
            retry = 0;  
            break;  
        }  
    }  
}
```

```
/* Timeout return */
if(retry == 2000)
{
    _card_disable();
    return 1;
}

/* Start reading */
for(retry=0; retry<len; retry++)
{
    *(buff+retry) = _spi_read_write(DUMMY_BYTE);
}

/* 2bytes dummy CRC */
_spi_read_write(DUMMY_BYTE);
_spi_read_write(DUMMY_BYTE);

/* chip disable and dummy byte */
if(release)
{
    _card_disable();
    _spi_read_write(DUMMY_BYTE);
}

return 0;
}

/*********************************************
* Function Name  : MSD_ReadSingleBlock
* Description    : None
* Input          : - sector:
*                   - buffer:
* Output         : None
* Return         : None
* Attention      : None
*****************************************/
int MSD_ReadSingleBlock(uint32_t sector, uint8_t *buffer)
{
    uint8_t r1;
```

```
/* if ver = SD2.0 HC, sector need <<9 */
if(CardInfo.CardType != CARDTYPE_SDV2HC)
{
    sector = sector<<9;
}

/* Send CMD17 : Read single block command */
r1 = _send_command(CMD17, sector, 0);

if(r1 != 0x00)
{
    return 1;
}

/* Start read and return the result */
r1 = _read_buffer(buffer, MSD_BLOCKSIZE, RELEASE);

/* Send stop data transmit command - CMD12 */
_send_command(CMD12, 0, 0);

return r1;
}

*****
* Function Name  : MSD_ReadMultiBlock
* Description    : None
* Input          : - sector:
*                   - buffer:
*                   - NbrOfSector:
* Output         : None
* Return         : None
* Attention      : None
*****
int MSD_ReadMultiBlock(uint32_t sector, uint8_t *buffer, uint32_t NbrOfSector)
{
    uint8_t r1;
    uint32_t i;
```

```
/* if ver = SD2.0 HC, sector need <<9 */
if(CardInfo.CardType != CARDTYPE_SDV2HC)
{
    sector = sector<<9;
}

/* Send CMD18 : Read multi block command */
r1 = _send_command(CMD18, sector, 0);
if(r1 != 0x00)
{
    return 1;
}

/* Start read      */
for(i=0; i<NbrOfSector; i++)
{
    if(_read_buffer(buffer+i*MSD_BLOCKSIZE, MSD_BLOCKSIZE, HOLD))
    {
        /* Send stop data transmit command - CMD12 */
        _send_command(CMD12, 0, 0);
        /* chip disable and dummy byte */
        _card_disable();
        return 2;
    }
}

/* Send stop data transmit command - CMD12 */
_send_command(CMD12, 0, 0);

/* chip disable and dummy byte */
_card_disable();
_spi_read_write(DUMMY_BYTE);

return 0;
}

*****
* Function Name  : MSD_WriteSingleBlock
* Description    : None
```

```
* Input           : - sector:  
*                           - buffer:  
  
* Output          : None  
  
* Return          : None  
  
* Attention       : None  
*****  
  
int MSD_WriteSingleBlock(uint32_t sector, const uint8_t *buffer)  
{  
    uint8_t r1;  
    uint16_t i;  
    uint32_t retry;  
  
    /* if ver = SD2.0 HC, sector need <<9 */  
    if(CardInfo.CardType != CARDTYPE_SDV2HC)  
    {  
        sector = sector<<9;  
    }  
  
    /* Send CMD24 : Write single block command */  
    r1 = _send_command(CMD24, sector, 0);  
  
    if(r1 != 0x00)  
    {  
        return 1;  
    }  
  
    /* Card enable, Prepare to write */  
    _card_enable();  
    _spi_read_write(DUMMY_BYTE);  
    _spi_read_write(DUMMY_BYTE);  
    _spi_read_write(DUMMY_BYTE);  
    /* Start data write token: 0xFE */  
    _spi_read_write(0xFE);  
  
    /* Start single block write the data buffer */  
    for(i=0; i<MSD_BLOCKSIZE; i++)  
    {  
        _spi_read_write(*buffer++);  
    }
```

```
/* 2Bytes dummy CRC */
_spi_read_write(DUMMY_BYTE);
_spi_read_write(DUMMY_BYTE);

/* MSD card accept the data */
r1 = _spi_read_write(DUMMY_BYTE);
if((r1&0x1F) != 0x05)
{
    _card_disable();
    return 2;
}

/* Wait all the data programm finished */
retry = 0;
while(_spi_read_write(DUMMY_BYTE) == 0x00)
{
    /* Timeout return */
    if(retry++ == 0x40000)
    {
        _card_disable();
        return 3;
    }
}

/* chip disable and dummy byte */
_card_disable();
_spi_read_write(DUMMY_BYTE);

return 0;
}

*****
* Function Name : MSD_WriteMultiBlock
* Description   : None
* Input         : - sector:
*                  - buffer:
*                  - NbrOfSector:
* Output        : None
```

```
* Return      : None
* Attention   : None
*****
int MSD_WriteMultiBlock(uint32_t sector, const uint8_t *buffer, uint32_t NbrOfSector)
{
    uint8_t r1;
    uint16_t i;
    uint32_t n;
    uint32_t retry;

    /* if ver = SD2.0 HC, sector need <<9 */
    if(CardInfo.CardType != CARDTYPE_SDV2HC)
    {
        sector = sector<<9;
    }

    /* Send command ACMD23 berfore multi write if is not a MMC card */
    if(CardInfo.CardType != CARDTYPE_MMC)
    {
        _send_command(ACMD23, NbrOfSector, 0x00);
    }

    /* Send CMD25 : Write nulti block command */
    r1 = _send_command(CMD25, sector, 0);

    if(r1 != 0x00)
    {
        return 1;
    }

    /* Card enable, Prepare to write */
    _card_enable();
    _spi_read_write(DUMMY_BYTE);
    _spi_read_write(DUMMY_BYTE);
    _spi_read_write(DUMMY_BYTE);

    for(n=0; n<NbrOfSector; n++)
    {
        /* Start multi block write token: 0xFC */
    }
}
```

```
_spi_read_write(0xFC);

for(i=0; i<MSD_BLOCKSIZE; i++)
{
    _spi_read_write(*buffer++);
}

/* 2Bytes dummy CRC */
_spi_read_write(DUMMY_BYTE);
_spi_read_write(DUMMY_BYTE);

/* MSD card accept the data */
r1 = _spi_read_write(DUMMY_BYTE);
if((r1&0x1F) != 0x05)
{
    _card_disable();
    return 2;
}

/* Wait all the data programm finished      */
retry = 0;
while(_spi_read_write(DUMMY_BYTE) != 0xFF)
{
    /* Timeout return */
    if(retry++ == 0x40000)
    {
        _card_disable();
        return 3;
    }
}

/* Send end of transmit token: 0xFD */
r1 = _spi_read_write(0xFD);
if(r1 == 0x00)
{
    return 4;
}
```

```
/* Wait all the data programm finished */

retry = 0;

while(_spi_read_write(DUMMY_BYTE) != 0xFF)
{

    /* Timeout return */
    if(retry++ == 0x40000)
    {

        _card_disable();

        return 5;
    }
}

/* chip disable and dummy byte */

_card_disable();
_spi_read_write(DUMMY_BYTE);

return 0;
}

/*****
 * Function Name   : _send_command
 * Description     : None
 * Input          : - cmd:
 *                   - arg:
 *                   - crc:
 * Output         : None
 * Return         : R1 value, response from card
 * Attention      : None
 *****/
int _send_command(uint8_t cmd, uint32_t arg, uint8_t crc)
{
    uint8_t r1;
    uint8_t retry;

    /* Dummy byte and chip enable */
    _spi_read_write(DUMMY_BYTE);
    _card_enable();
```

```
/* Command, argument and crc */

_spi_read_write(cmd | 0x40);

_spi_read_write(arg >> 24);

_spi_read_write(arg >> 16);

_spi_read_write(arg >> 8);

_spi_read_write(arg);

_spi_read_write(crc);

/* Wait response, quit till timeout */

for(retry=0; retry<200; retry++)

{

    r1 = _spi_read_write(DUMMY_BYTE);

    if(r1 != 0xFF)

    {

        break;

    }

}

/* Chip disable and dummy byte */

_card_disable();

_spi_read_write(DUMMY_BYTE);

return r1;

}

*****  
* Function Name : _send_command_hold  
* Description   : None  
* Input         : - cmd:  
*                   - arg:  
*                   - crc:  
* Output        : None  
* Return        : R1 value, response from card  
* Attention     : None  
*****  
int _send_command_hold(uint8_t cmd, uint32_t arg, uint8_t crc)
{
    uint8_t r1;
    uint8_t retry;
```

```
/* Dummy byte and chip enable */
_spi_read_write(DUMMY_BYTE);
_card_enable();

/* Command, argument and crc */
_spi_read_write(cmd | 0x40);
_spi_read_write(arg >> 24);
_spi_read_write(arg >> 16);
_spi_read_write(arg >> 8);
_spi_read_write(arg);
_spi_read_write(crc);

/* Wait response, quit till timeout */
for(retry=0; retry<200; retry++)
{
    r1 = _spi_read_write(DUMMY_BYTE);
    if(r1 != 0xFF)
    {
        break;
    }
}

return r1;
}

/*********************************************
 * $Id$      lpc17xx_ssp.c          2010-06-18
 *//**
 * @file      lpc17xx_ssp.c
 * @brief     Contains all functions support for SSP firmware library on LPC17xx
 * @version   3.0
 * @date      18. June. 2010
 * @author    NXP MCU SW Application Team
 *
 * Copyright (C) 2010, NXP Semiconductor
 * All rights reserved.
 *
```

```
*****
* Software that is described herein is for illustrative purposes only
* which provides customers with programming information regarding the
* products. This software is supplied "AS IS" without any warranties.
* NXP Semiconductors assumes no responsibility or liability for the
* use of the software, conveys no license or title under any patent,
* copyright, or mask work right to the product. NXP Semiconductors
* reserves the right to make changes in the software without
* notification. NXP Semiconductors also make no representation or
* warranty that such application will be suitable for the specified
* use without further testing or modification.
*****/
```

```
/* Peripheral group ----- */
/** @addtogroup SSP
 * @{
 *
 */
```

```
/* Includes ----- */
#include "lpc17xx_ssp.h"
#include "lpc17xx_clkpwr.h"
```

```
/* If this source file built with example, the LPC17xx FW library configuration
 * file in each example directory ("lpc17xx_libcfg.h") must be included,
 * otherwise the default FW library configuration file must be included instead
 */
#ifndef __BUILD_WITH_EXAMPLE__
#include "lpc17xx_libcfg.h"
#else
#include "lpc17xx_libcfg_default.h"
#endif /* __BUILD_WITH_EXAMPLE__ */
```

```
#ifdef _SSP

/* Public Functions ----- */
/** @addtogroup SSP_Public_Functions
 * @{
 *
```

```
*/  
  
static void setSSPclock (LPC_SSP_TypeDef *SSPx, uint32_t target_clock);  
  
/**********************************************************//**  
  
 * @brief           Setup clock rate for SSP device  
 * @param[in]   SSPx   SSP peripheral definition, should be:  
 *                   - LPC_SSP0: SSP0 peripheral  
 *                   - LPC_SSP1: SSP1 peripheral  
 * @param[in]   target_clock : clock of SSP (Hz)  
 * @return          None  
*****  
  
static void setSSPclock (LPC_SSP_TypeDef *SSPx, uint32_t target_clock)  
{  
    uint32_t prescale, cr0_div, cmp_clk, ssp_clk;  
  
    CHECK_PARAM(PARAM_SSPx(SSPx));  
  
    /* The SSP clock is derived from the (main system oscillator / 2),  
       so compute the best divider from that clock */  
    if (SSPx == LPC_SSP0){  
        ssp_clk = CLKPWR_GetPCLK (CLKPWR_PCLKSEL_SSP0);  
    } else if (SSPx == LPC_SSP1) {  
        ssp_clk = CLKPWR_GetPCLK (CLKPWR_PCLKSEL_SSP1);  
    } else {  
        return;  
    }  
  
    /* Find closest divider to get at or under the target frequency.  
       Use smallest prescale possible and rely on the divider to get  
       the closest target frequency */  
    cr0_div = 0;  
    cmp_clk = 0xFFFFFFFF;  
    prescale = 2;  
    while (cmp_clk > target_clock)  
    {  
        cmp_clk = ssp_clk / ((cr0_div + 1) * prescale);  
        if (cmp_clk > target_clock)  
        {  
            cr0_div++;  
        }  
    }  
}
```

```
        if (cr0_div > 0xFF)
    {
        cr0_div = 0;
        prescale += 2;
    }
}

/* Write computed prescaler and divider back to register */
SSPx->CRO &= (~SSP_CRO_SCR(0xFF)) & SSP_CRO_BITMASK;
SSPx->CRO |= (SSP_CRO_SCR(cr0_div)) & SSP_CRO_BITMASK;
SSPx->CPSR = prescale & SSP_CPSR_BITMASK;
}

/***
 * @}
 */

/* Public Functions ----- */
/** @addtogroup SSP_Public_Functions
 * @{
 */

/***** *****/
* @brief           Initializes the SSPx peripheral according to the specified
*                  parameters in the SSP_ConfigStruct.
* @param[in] SSPx   SSP peripheral selected, should be:
*                  - LPC_SSP0: SSP0 peripheral
*                  - LPC_SSP1: SSP1 peripheral
* @param[in] SSP_ConfigStruct Pointer to a SSP_CFG_Type structure
*                  that contains the configuration information for the
*                  specified SSP peripheral.
* @return          None
*****
void SSP_Init(LPC_SSP_TypeDef *SSPx, SSP_CFG_Type *SSP_ConfigStruct)
{
    uint32_t tmp;

    CHECK_PARAM(PARAM_SSPx(SSPx));
}
```

```
if(SSPx == LPC_SSP0) {  
    /* Set up clock and power for SSP0 module */  
    CLKPWR_ConfigPPWR (CLKPWR_PCONP_PCSSP0, ENABLE);  
}  
else if(SSPx == LPC_SSP1) {  
    /* Set up clock and power for SSP1 module */  
    CLKPWR_ConfigPPWR (CLKPWR_PCONP_PCSSP1, ENABLE);  
}  
else {  
    return;  
}  
  
/* Configure SSP, interrupt is disable, LoopBack mode is disable,  
 * SSP is disable, Slave output is disable as default  
 */  
tmp = ((SSP_ConfigStruct->CPHA) | (SSP_ConfigStruct->CPOL) \  
| (SSP_ConfigStruct->FrameFormat) | (SSP_ConfigStruct->Databit))  
& SSP_CRO_BITMASK;  
// write back to SSP control register  
SSPx->CRO = tmp;  
  
tmp = SSP_ConfigStruct->Mode & SSP_CR1_BITMASK;  
// Write back to CR1  
SSPx->CR1 = tmp;  
  
// Set clock rate for SSP peripheral  
setSSPclock(SSPx, SSP_ConfigStruct->ClockRate);  
}  
  
/********************************************/  
* @brief De-initializes the SSPx peripheral registers to their  
*        default reset values.  
* @param[in] SSPx SSP peripheral selected, should be:  
*                  - LPC_SSP0: SSP0 peripheral  
*                  - LPC_SSP1: SSP1 peripheral  
* @return None  
*****
```

void SSP_DeInit(LPC_SSP_TypeDef* SSPx)
{
 CHECK_PARAM(PARAM_SSPx(SSPx));

```
if (SSPx == LPC_SSP0) {
    /* Set up clock and power for SSP0 module */
    CLKPWR_ConfigPPWR (CLKPWR_PCONP_PCSSP0, DISABLE);
} else if (SSPx == LPC_SSP1) {
    /* Set up clock and power for SSP1 module */
    CLKPWR_ConfigPPWR (CLKPWR_PCONP_PCSSP1, DISABLE);
}

/***** */

/* @brief      Get data size bit selected
 * @param[in]   SSPx pointer to LPC_SSP_TypeDef structure, should be:
 *
 *              - LPC_SSP0: SSP0 peripheral
 *
 *              - LPC_SSP1: SSP1 peripheral
 *
 * @return      Data size, could be:
 *
 *              - SSP_DATABIT_4: 4 bit transfer
 *
 *              - SSP_DATABIT_5: 5 bit transfer
 *
 *              ...
 *
 *              - SSP_DATABIT_16: 16 bit transfer
 */
uint8_t SSP_GetDataSize(LPC_SSP_TypeDef* SSPx)
{
    CHECK_PARAM(PARAM_SSPx(SSPx));
    return (SSPx->CRO & (0xF));
}

/***** */

/* @brief      Fills each SSP_InitStruct member with its default value:
 *
 *              - CPHA = SSP_CPHA_FIRST
 *
 *              - CPOL = SSP_CPOL_HI
 *
 *              - ClockRate = 1000000
 *
 *              - Databit = SSP_DATABIT_8
 *
 *              - Mode = SSP_MASTER_MODE
 *
 *              - FrameFormat = SSP_FRAME_SSP
 *
 * @param[in]   SSP_InitStruct Pointer to a SSP_CFG_Type structure
 *
 *              which will be initialized.
 *
 * @return      None
 */

```

```
void SSP_ConfigStructInit(SSP_CFG_Type *SSP_InitStruct)
{
    SSP_InitStruct->CPHA = SSP_CPHA_FIRST;
    SSP_InitStruct->CPOL = SSP_CPOL_HI;
    SSP_InitStruct->ClockRate = 1000000;
    SSP_InitStruct->Databit = SSP_DATABIT_8;
    SSP_InitStruct->Mode = SSP_MASTER_MODE;
    SSP_InitStruct->FrameFormat = SSP_FRAME_SPI;
}

/*****brief*****/  

* @brief          Enable or disable SSP peripheral's operation  

* @param[in]  SSPx  SSP peripheral, should be:  

*              - LPC_SSP0: SSP0 peripheral  

*              - LPC_SSP1: SSP1 peripheral  

* @param[in]  NewState New State of SSPx peripheral's operation  

* @return         none  

*****/  

void SSP_Cmd(LPC_SSP_TypeDef* SSPx, FunctionalState NewState)
{
    CHECK_PARAM(PARAM_SSPx(SSPx));
    CHECK_PARAM(PARAM_FUNCTIONALSTATE(NewState));

    if (NewState == ENABLE)
    {
        SSPx->CR1 |= SSP_CR1_SSP_EN;
    }
    else
    {
        SSPx->CR1 &= (~SSP_CR1_SSP_EN) & SSP_CR1_BITMASK;
    }
}

/*****brief*****/  

* @brief          Enable or disable Loop Back mode function in SSP peripheral  

* @param[in]  SSPx  SSP peripheral selected, should be:  

*              - LPC_SSP0: SSP0 peripheral  

*              - LPC_SSP1: SSP1 peripheral
```

```
* @param[in] NewState      New State of Loop Back mode, should be:  
*                                     - ENABLE: Enable this function  
*                                     - DISABLE: Disable this function  
* @return       None  
*****  
void SSP_LoopBackCmd(LPC_SSP_TypeDef* SSPx, FunctionalState NewState)  
{  
    CHECK_PARAM(PARAM_SSPx(SSPx));  
    CHECK_PARAM(PARAM_FUNCTIONALSTATE(NewState));  
  
    if (NewState == ENABLE)  
    {  
        SSPx->CR1 |= SSP_CR1_LBM_EN;  
    }  
    else  
    {  
        SSPx->CR1 &= (~SSP_CR1_LBM_EN) & SSP_CR1_BITMASK;  
    }  
}  
  
*****/**/  
* @brief          Enable or disable Slave Output function in SSP peripheral  
* @param[in] SSPx  SSP peripheral selected, should be:  
*                   - LPC_SSP0: SSP0 peripheral  
*                   - LPC_SSP1: SSP1 peripheral  
* @param[in] NewState      New State of Slave Output function, should be:  
*                                     - ENABLE: Slave Output in normal  
* operation  
*                                     - DISABLE: Slave Output is disabled.  
* This blocks  
*                                     SSP controller from driving the  
* transmit data  
*                                     line (MISO)  
* Note:           This function is available when SSP peripheral in Slave mode  
* @return       None  
*****  
void SSP_SlaveOutputCmd(LPC_SSP_TypeDef* SSPx, FunctionalState NewState)  
{  
    CHECK_PARAM(PARAM_SSPx(SSPx));  
    CHECK_PARAM(PARAM_FUNCTIONALSTATE(NewState));
```

```
    if (NewState == ENABLE)
    {
        SSPx->CR1 &= (~SSP_CR1_SO_DISABLE) & SSP_CR1_BITMASK;
    }
    else
    {
        SSPx->CR1 |= SSP_CR1_SO_DISABLE;
    }
}

/******************/


* @brief          Transmit a single data through SSPx peripheral
* @param[in] SSPx  SSP peripheral selected, should be:
*                  - LPC_SSP0: SSP0 peripheral
*                  - LPC_SSP1: SSP1 peripheral
* @param[in] Data  Data to transmit (must be 16 or 8-bit long,
*                   this depend on SSP data bit number
* configured)
* @return         none
******************/

void SSP_SendData(LPC_SSP_TypeDef* SSPx, uint16_t Data)
{
    CHECK_PARAM(PARAM_SSPx(SSPx));

    SSPx->DR = SSP_DR_BITMASK(Data);
}

/******************/


* @brief          Receive a single data from SSPx peripheral
* @param[in] SSPx  SSP peripheral selected, should be
*                  - LPC_SSP0: SSP0 peripheral
*                  - LPC_SSP1: SSP1 peripheral
* @return         Data received (16-bit long)
******************/

uint16_t SSP_ReceiveData(LPC_SSP_TypeDef* SSPx)
{
```

```
CHECK_PARAM(PARAM_SSPx(SSPx));\n\n    return ((uint16_t) (SSP_DR_BITMASK(SSPx->DR)));\n}\n\n/*********************\n *\n * @brief          SSP Read write data function\n *\n * @param[in]  SSPx  Pointer to SSP peripheral, should be\n *\n *                  - LPC_SSP0: SSP0 peripheral\n *\n *                  - LPC_SSP1: SSP1 peripheral\n *\n * @param[in]  dataCfg Pointer to a SSP_DATA_SETUP_Type structure that\n *\n *                  contains specified information about\n transmit\n *\n *                  data configuration.\n *\n * @param[in]  xfType Transfer type, should be:\n *\n *                  - SSP_TRANSFER_POLLING: Polling mode\n *\n *                  - SSP_TRANSFER_INTERRUPT: Interrupt mode\n *\n * @return         Actual Data length has been transferred in polling mode.\n *\n *                  In interrupt mode, always return (0)\n *\n *                  Return (-1) if error.\n *\n * Note: This function can be used in both master and slave mode.\n */\n\nint32_t SSP_ReadWrite (LPC_SSP_TypeDef *SSPx, SSP_DATA_SETUP_Type *dataCfg, \n                      SSP_TRANSFER_Type xfType)\n{\n    uint8_t *rdata8;\n    uint8_t *wdata8;\n    uint16_t *rdata16;\n    uint16_t *wdata16;\n    uint32_t stat;\n    uint32_t tmp;\n    int32_t dataword;\n\n    dataCfg->rx_cnt = 0;\n    dataCfg->tx_cnt = 0;\n    dataCfg->status = 0;\n\n    /* Clear all remaining data in RX FIFO */\n    while (SSPx->SR & SSP_SR_RNE) {
```

```
    tmp = (uint32_t) SSP_ReceiveData(SSPx);

}

// Clear status
SSPx->ICR = SSP_ICR_BITMASK;
if(SSP_GetDataSize(SSPx)>8)
    dataword = 1;
else dataword = 0;

// Polling mode -----
-----
if (xfType == SSP_TRANSFER_POLLING){
    if (dataword == 0){
        rdata8 = (uint8_t *)dataCfg->rx_data;
        wdata8 = (uint8_t *)dataCfg->tx_data;
    } else {
        rdata16 = (uint16_t *)dataCfg->rx_data;
        wdata16 = (uint16_t *)dataCfg->tx_data;
    }
    while ((dataCfg->tx_cnt != dataCfg->length) || (dataCfg->rx_cnt != dataCfg->length)){
        if ((SSPx->SR & SSP_SR_TNF) && (dataCfg->tx_cnt != dataCfg->length)){
            // Write data to buffer
            if(dataCfg->tx_data == NULL){
                if (dataword == 0){
                    SSP_SendData(SSPx, 0xFF);
                    dataCfg->tx_cnt++;
                } else {
                    SSP_SendData(SSPx, 0xFFFF);
                    dataCfg->tx_cnt += 2;
                }
            } else {
                if (dataword == 0){
                    SSP_SendData(SSPx, *wdata8);
                    wdata8++;
                    dataCfg->tx_cnt++;
                } else {
                    SSP_SendData(SSPx, *wdata16);
                    wdata16++;
                    dataCfg->tx_cnt += 2;
                }
            }
        }
    }
}
```

```
        }

    }

}

// Check overrun error

if ((stat = SSPx->RIS) & SSP_RIS_ROR){

    // save status and return

    dataCfg->status = stat | SSP_STAT_ERROR;

    return (-1);

}

// Check for any data available in RX FIFO

while ((SSPx->SR & SSP_SR_RNE) && (dataCfg->rx_cnt != dataCfg->length)) {

    // Read data from SSP data

    tmp = SSP_ReceiveData(SSPx);

    // Store data to destination

    if (dataCfg->rx_data != NULL)

    {

        if (dataword == 0){

            *(rdata8) = (uint8_t) tmp;

            rdata8++;

        } else {

            *(rdata16) = (uint16_t) tmp;

            rdata16++;

        }

    }

    // Increase counter

    if (dataword == 0){

        dataCfg->rx_cnt++;

    } else {

        dataCfg->rx_cnt += 2;

    }

}

// save status

dataCfg->status = SSP_STAT_DONE;
```

```
        if (dataCfg->tx_data != NULL) {
            return dataCfg->tx_cnt;
        } else if (dataCfg->rx_data != NULL) {
            return dataCfg->rx_cnt;
        } else {
            return (0);
        }

    }

// Interrupt mode -----
-----
else if (xfType == SSP_TRANSFER_INTERRUPT) {

    while ((SSPx->SR & SSP_SR_TNF) && (dataCfg->tx_cnt != dataCfg->length)) {
        // Write data to buffer
        if(dataCfg->tx_data == NULL) {
            if (dataword == 0) {
                SSP_SendData(SSPx, 0xFF);
                dataCfg->tx_cnt++;
            } else {
                SSP_SendData(SSPx, 0xFFFF);
                dataCfg->tx_cnt += 2;
            }
        } else {
            if (dataword == 0) {
                SSP_SendData(SSPx, (*(uint8_t *)((uint32_t)dataCfg-
>tx_data + dataCfg->tx_cnt)));
                dataCfg->tx_cnt++;
            } else {
                SSP_SendData(SSPx, (*(uint16_t *)
((uint32_t)dataCfg->tx_data + dataCfg->tx_cnt)));
                dataCfg->tx_cnt += 2;
            }
        }
    }

    // Check error
    if ((stat = SSPx->RIS) & SSP_RIS_ROR) {
        // save status and return
        dataCfg->status = stat | SSP_STAT_ERROR;
        return (-1);
    }
}
```

```
// Check for any data available in RX FIFO
while ((SSPx->SR & SSP_SR_RNE) && (dataCfg->rx_cnt != dataCfg->length)) {
    // Read data from SSP data
    tmp = SSP_ReceiveData(SSPx);

    // Store data to destination
    if (dataCfg->rx_data != NULL)
    {
        if (dataword == 0) {
            *(uint8_t *)((uint32_t)dataCfg->rx_data +
dataCfg->rx_cnt) = (uint8_t) tmp;
        } else {
            *(uint16_t *)((uint32_t)dataCfg->rx_data +
dataCfg->rx_cnt) = (uint16_t) tmp;
        }
    }

    // Increase counter
    if (dataword == 0) {
        dataCfg->rx_cnt++;
    } else {
        dataCfg->rx_cnt += 2;
    }
}

// If there more data to sent or receive
if ((dataCfg->rx_cnt != dataCfg->length) || (dataCfg->tx_cnt != dataCfg->length)) {
    // Enable all interrupt
    SSPx->IMSC = SSP_IMSC_BITMASK;
} else {
    // Save status
    dataCfg->status = SSP_STAT_DONE;
}

return (0);
}

return (-1);
}
```

```

/*****//**

* @brief Checks whether the specified SSP status flag is set or not
* @param[in] SSPx SSP peripheral selected, should be:
*
* - LPC_SSP0: SSP0 peripheral
*
* - LPC_SSP1: SSP1 peripheral

* @param[in] FlagType Type of flag to check status, should be one
* of following:
*
* - SSP_STAT_TXFIFO_EMPTY: TX FIFO is
empty

* - SSP_STAT_TXFIFO_NOTFULL: TX FIFO
is not full

* - SSP_STAT_RXFIFO_NOTEEMPTY: RX FIFO
is not empty

* - SSP_STAT_RXFIFO_FULL: RX FIFO is
full

* - SSP_STAT_BUSY: SSP peripheral is
busy

* @return New State of specified SSP status flag
*****/

FlagStatus SSP_GetStatus(LPC_SSP_TypeDef* SSPx, uint32_t FlagType)
{
    CHECK_PARAM(PARAM_SSP(SSPx));
    CHECK_PARAM(PARAM_SSP_STAT(FlagType));

    return ((SSPx->SR & FlagType) ? SET : RESET);
}

/*****//**

* @brief Enable or disable specified interrupt type in SSP peripheral
* @param[in] SSPx SSP peripheral selected, should be:
*
* - LPC_SSP0: SSP0 peripheral
*
* - LPC_SSP1: SSP1 peripheral

* @param[in] IntType Interrupt type in SSP peripheral, should be:
*
* - SSP_INTCFG_ROR: Receive Overrun interrupt
*
* - SSP_INTCFG_RT: Receive Time out interrupt
*
* - SSP_INTCFG_RX: RX FIFO is at least half full interrupt
*
* - SSP_INTCFG_TX: TX FIFO is at least half empty interrupt

* @param[in] NewState New State of specified interrupt type, should be:
*
* - ENABLE: Enable this interrupt type
*
* - DISABLE: Disable this interrupt type

* @return None

```

```

* Note: We can enable/disable multi-interrupt type by OR multi value
***** */

void SSP_IntConfig(LPC_SSP_TypeDef *SSPx, uint32_t IntType, FunctionalState NewState)
{
    CHECK_PARAM(PARAM_SSPx(SSPx));

    if (NewState == ENABLE)
    {
        SSPx->IMSC |= IntType;
    }
    else
    {
        SSPx->IMSC &= (~IntType) & SSP_IMSC_BITMASK;
    }
}

/** */

* @brief      Check whether the specified Raw interrupt status flag is
*             set or not
* @param[in]  SSPx      SSP peripheral selected, should be:
*                   - LPC_SSP0: SSP0 peripheral
*                   - LPC_SSP1: SSP1 peripheral
* @param[in]  RawIntType   Raw Interrupt Type, should be:
*                   - SSP_INTSTAT_RAW_ROR: Receive Overrun interrupt
*                   - SSP_INTSTAT_RAW_RT: Receive Time out interrupt
*                   - SSP_INTSTAT_RAW_RX: RX FIFO is at least half full
* interrupt
*                   - SSP_INTSTAT_RAW_TX: TX FIFO is at least half empty
* interrupt
* @return     New State of specified Raw interrupt status flag in SSP peripheral
* Note: Enabling/Disabling specified interrupt in SSP peripheral does not
*       effect to Raw Interrupt Status flag.
***** */

IntStatus SSP_GetRawIntStatus(LPC_SSP_TypeDef *SSPx, uint32_t RawIntType)
{
    CHECK_PARAM(PARAM_SSPx(SSPx));
    CHECK_PARAM(PARAM_SSP_INTSTAT_RAW(RawIntType));

    return ((SSPx->RIS & RawIntType) ? SET : RESET);
}

```

```
/**************************************************************************/ **

* @brief          Get Raw Interrupt Status register
* @param[in]    SSPx    SSP peripheral selected, should be:
*                 - LPC_SSP0: SSP0 peripheral
*                 - LPC_SSP1: SSP1 peripheral
* @return         Raw Interrupt Status (RIS) register value
/**************************************************************************/

uint32_t SSP_GetRawIntStatusReg(LPC_SSP_TypeDef *SSPx)
{
    CHECK_PARAM(PARAM_SSPx(SSPx));
    return (SSPx->RIS);
}

/**************************************************************************/ **

* @brief          Check whether the specified interrupt status flag is
*                 set or not
* @param[in]    SSPx    SSP peripheral selected, should be:
*                 - LPC_SSP0: SSP0 peripheral
*                 - LPC_SSP1: SSP1 peripheral
* @param[in]    IntType Raw Interrupt Type, should be:
*                 - SSP_INTSTAT_ROR: Receive Overrun interrupt
*                 - SSP_INTSTAT_RT: Receive Time out interrupt
*                 - SSP_INTSTAT_RX: RX FIFO is at least half full interrupt
*                 - SSP_INTSTAT_TX: TX FIFO is at least half empty interrupt
* @return         New State of specified interrupt status flag in SSP peripheral
* Note: Enabling/Disabling specified interrupt in SSP peripheral effects
*           to Interrupt Status flag.
/**************************************************************************/

IntStatus SSP.GetIntStatus (LPC_SSP_TypeDef *SSPx, uint32_t IntType)
{
    CHECK_PARAM(PARAM_SSPx(SSPx));
    CHECK_PARAM(PARAM_SSP_INTSTAT(IntType));

    return ((SSPx->MIS & IntType) ? SET : RESET);
}

/**************************************************************************/ **

* @brief          Clear specified interrupt pending in SSP peripheral
```

```

* @param[in] SSPx    SSP peripheral selected, should be:
*
*                         - LPC_SSPO: SSP0 peripheral
*
*                         - LPC_SSPI: SSP1 peripheral
*
* @param[in] IntType Interrupt pending to clear, should be:
*
*                         - SSP_INTCLR_ROR: clears the "frame was
received when
*
*                         RxFIFO was full" interrupt.
*
*                         - SSP_INTCLR_RT: clears the "Rx FIFO was not
empty and
*
*                         has not been read for a timeout period"
interrupt.

* @return             None
*****
void SSP_ClearIntPending(LPC_SSP_TypeDef *SSPx, uint32_t IntType)
{
    CHECK_PARAM(PARAM_SSPx(SSPx));
    CHECK_PARAM(PARAM_SSP_INTCLR(IntType));

    SSPx->ICR = IntType;
}

/*****
* @brief             Enable/Disable DMA function for SSP peripheral
* @param[in] SSPx    SSP peripheral selected, should be:
*
*                         - LPC_SSPO: SSP0 peripheral
*
*                         - LPC_SSPI: SSP1 peripheral
*
* @param[in] DMAMode Type of DMA, should be:
*
*                         - SSP_DMA_TX: DMA for the transmit FIFO
*
*                         - SSP_DMA_RX: DMA for the Receive FIFO
*
* @param[in] NewState New State of DMA function on SSP peripheral,
*
*                         should be:
*
*                         - ENALBE: Enable this function
*
*                         - DISABLE: Disable this function
*
* @return             None
*****
void SSP_DMACmd(LPC_SSP_TypeDef *SSPx, uint32_t DMAMode, FunctionalState NewState)
{
    CHECK_PARAM(PARAM_SSPx(SSPx));
    CHECK_PARAM(PARAM_SSP_DMA(DMAMode));
    CHECK_PARAM(PARAM_FUNCTIONALSTATE(NewState));
}

```

```
    if (NewState == ENABLE)
    {
        SSPx->DMACR |= DMAMode;
    }
    else
    {
        SSPx->DMACR &= (~DMAMode) & SSP_DMA_BITMASK;
    }
}

/***
 * @}
 */

#endif /* _SSP */

/***
 * @}
 */

/* ----- End Of File ----- */

/*
* $Id$          lpc17xx_pinsel.c          2010-05-21
* //**
* @file         lpc17xx_pinsel.c
* @brief        Contains all functions support for Pin connect block firmware
*               library on LPC17xx
* @version      2.0
* @date         21. May. 2010
* @author       NXP MCU SW Application Team
*
* Copyright(C) 2010, NXP Semiconductor
* All rights reserved.
*
* ****
* Software that is described herein is for illustrative purposes only
*
```

```
* which provides customers with programming information regarding the
* products. This software is supplied "AS IS" without any warranties.
* NXP Semiconductors assumes no responsibility or liability for the
* use of the software, conveys no license or title under any patent,
* copyright, or mask work right to the product. NXP Semiconductors
* reserves the right to make changes in the software without
* notification. NXP Semiconductors also make no representation or
* warranty that such application will be suitable for the specified
* use without further testing or modification.

*****
```

```
/* Peripheral group ----- */
/** @defgroup PINSEL
 * @{
 */

/* Includes ----- */
#include "lpc17xx_pinsel.h"

/* Public Functions ----- */
static void set_PinFunc ( uint8_t portnum, uint8_t pinnum, uint8_t funcnum);
static void set_ResistorMode ( uint8_t portnum, uint8_t pinnum, uint8_t modenum);
static void set_OpenDrainMode( uint8_t portnum, uint8_t pinnum, uint8_t modenum);

*****/**/

* @brief           Setup the pin selection function
* @param[in]      portnum PORT number,
*                 should be one of the following:
*                 - PINSEL_PORT_0      : Port 0
*                 - PINSEL_PORT_1      : Port 1
*                 - PINSEL_PORT_2      : Port 2
*                 - PINSEL_PORT_3      : Port 3
*
* @param[in]      pinnum Pin number,
*                 should be one of the following:
*                 - PINSEL_PIN_0 : Pin 0
*                 - PINSEL_PIN_1 : Pin 1
*                 - PINSEL_PIN_2 : Pin 2
```

```
- PINSEL_PIN_3 : Pin 3
- PINSEL_PIN_4 : Pin 4
- PINSEL_PIN_5 : Pin 5
- PINSEL_PIN_6 : Pin 6
- PINSEL_PIN_7 : Pin 7
- PINSEL_PIN_8 : Pin 8
- PINSEL_PIN_9 : Pin 9
- PINSEL_PIN_10 : Pin 10
- PINSEL_PIN_11 : Pin 11
- PINSEL_PIN_12 : Pin 12
- PINSEL_PIN_13 : Pin 13
- PINSEL_PIN_14 : Pin 14
- PINSEL_PIN_15 : Pin 15
- PINSEL_PIN_16 : Pin 16
- PINSEL_PIN_17 : Pin 17
- PINSEL_PIN_18 : Pin 18
- PINSEL_PIN_19 : Pin 19
- PINSEL_PIN_20 : Pin 20
- PINSEL_PIN_21 : Pin 21
- PINSEL_PIN_22 : Pin 22
- PINSEL_PIN_23 : Pin 23
- PINSEL_PIN_24 : Pin 24
- PINSEL_PIN_25 : Pin 25
- PINSEL_PIN_26 : Pin 26
- PINSEL_PIN_27 : Pin 27
- PINSEL_PIN_28 : Pin 28
- PINSEL_PIN_29 : Pin 29
- PINSEL_PIN_30 : Pin 30
- PINSEL_PIN_31 : Pin 31

* @param[in] funcnum Function number,
*
*                      should be one of the following:
*
*                      - PINSEL_FUNC_0 : default function
*
*                      - PINSEL_FUNC_1 : first alternate function
*
*                      - PINSEL_FUNC_2 : second alternate function
*
*                      - PINSEL_FUNC_3 : third alternate function
*
*
* @return             None
*****

```

```
static void set_PinFunc ( uint8_t portnum, uint8_t pinnum, uint8_t funcnum)
{
    uint32_t pinnum_t = pinnum;
    uint32_t pinselreg_idx = 2 * portnum;
    uint32_t *pPinCon = (uint32_t *)&LPC_PINCON->PINSEL0;

    if (pinnum_t >= 16) {
        pinnum_t -= 16;
        pinselreg_idx++;
    }

    *(uint32_t *) (pPinCon + pinselreg_idx) &= ~(0x03UL << (pinnum_t * 2));
    *(uint32_t *) (pPinCon + pinselreg_idx) |= ((uint32_t)funcnum) << (pinnum_t * 2);
}

/*********************//**/

* @brief           Setup resistor mode for each pin
* @param[in]      portnum PORT number,
*                 should be one of the following:
*                 - PINSEL_PORT_0      : Port 0
*                 - PINSEL_PORT_1      : Port 1
*                 - PINSEL_PORT_2      : Port 2
*                 - PINSEL_PORT_3      : Port 3
* @param[in]      pinnum Pin number,
*                 should be one of the following:
*                 - PINSEL_PIN_0 : Pin 0
*                 - PINSEL_PIN_1 : Pin 1
*                 - PINSEL_PIN_2 : Pin 2
*                 - PINSEL_PIN_3 : Pin 3
*                 - PINSEL_PIN_4 : Pin 4
*                 - PINSEL_PIN_5 : Pin 5
*                 - PINSEL_PIN_6 : Pin 6
*                 - PINSEL_PIN_7 : Pin 7
*                 - PINSEL_PIN_8 : Pin 8
*                 - PINSEL_PIN_9 : Pin 9
*                 - PINSEL_PIN_10 : Pin 10
*                 - PINSEL_PIN_11 : Pin 11
*                 - PINSEL_PIN_12 : Pin 12
*                 - PINSEL_PIN_13 : Pin 13
*                 - PINSEL_PIN_14 : Pin 14
```

```
        - PINSEL_PIN_15 : Pin 15
        - PINSEL_PIN_16 : Pin 16
        - PINSEL_PIN_17 : Pin 17
        - PINSEL_PIN_18 : Pin 18
        - PINSEL_PIN_19 : Pin 19
        - PINSEL_PIN_20 : Pin 20
        - PINSEL_PIN_21 : Pin 21
        - PINSEL_PIN_22 : Pin 22
        - PINSEL_PIN_23 : Pin 23
        - PINSEL_PIN_24 : Pin 24
        - PINSEL_PIN_25 : Pin 25
        - PINSEL_PIN_26 : Pin 26
        - PINSEL_PIN_27 : Pin 27
        - PINSEL_PIN_28 : Pin 28
        - PINSEL_PIN_29 : Pin 29
        - PINSEL_PIN_30 : Pin 30
        - PINSEL_PIN_31 : Pin 31

* @param[in]  modenum: Mode number,
*
*                      should be one of the following:
*
*          - PINSEL_PINMODE_PULLUP      : Internal pull-up resistor
*          - PINSEL_PINMODE_TRISTATE : Tri-state
*          - PINSEL_PINMODE_PULLDOWN : Internal pull-down resistor

* @return           None
*****
void set_ResistorMode ( uint8_t portnum, uint8_t pinnum, uint8_t modenum)
{
    uint32_t pinnum_t = pinnum;
    uint32_t pinmodereg_idx = 2 * portnum;
    uint32_t *pPinCon = (uint32_t *)&LPC_PINCON->PINMODE0;

    if (pinnum_t >= 16) {
        pinnum_t -= 16;
        pinmodereg_idx++ ;
    }

    *(uint32_t *) (pPinCon + pinmodereg_idx) &= ~ (0x03UL << (pinnum_t * 2));
    *(uint32_t *) (pPinCon + pinmodereg_idx) |= ((uint32_t)modenum) << (pinnum_t * 2);
```

}

***** */

```
* @brief           Setup Open drain mode for each pin
* @param[in]      portnum PORT number,
*                 should be one of the following:
*                 - PINSEL_PORT_0      : Port 0
*                 - PINSEL_PORT_1      : Port 1
*                 - PINSEL_PORT_2      : Port 2
*                 - PINSEL_PORT_3      : Port 3
*
* @param[in]      pinnum Pin number,
*                 should be one of the following:
*                 - PINSEL_PIN_0 : Pin 0
*                 - PINSEL_PIN_1 : Pin 1
*                 - PINSEL_PIN_2 : Pin 2
*                 - PINSEL_PIN_3 : Pin 3
*                 - PINSEL_PIN_4 : Pin 4
*                 - PINSEL_PIN_5 : Pin 5
*                 - PINSEL_PIN_6 : Pin 6
*                 - PINSEL_PIN_7 : Pin 7
*                 - PINSEL_PIN_8 : Pin 8
*                 - PINSEL_PIN_9 : Pin 9
*                 - PINSEL_PIN_10 : Pin 10
*                 - PINSEL_PIN_11 : Pin 11
*                 - PINSEL_PIN_12 : Pin 12
*                 - PINSEL_PIN_13 : Pin 13
*                 - PINSEL_PIN_14 : Pin 14
*                 - PINSEL_PIN_15 : Pin 15
*                 - PINSEL_PIN_16 : Pin 16
*                 - PINSEL_PIN_17 : Pin 17
*                 - PINSEL_PIN_18 : Pin 18
*                 - PINSEL_PIN_19 : Pin 19
*                 - PINSEL_PIN_20 : Pin 20
*                 - PINSEL_PIN_21 : Pin 21
*                 - PINSEL_PIN_22 : Pin 22
*                 - PINSEL_PIN_23 : Pin 23
*                 - PINSEL_PIN_24 : Pin 24
*                 - PINSEL_PIN_25 : Pin 25
```

```

        - PINSEL_PIN_26 : Pin 26
        - PINSEL_PIN_27 : Pin 27
        - PINSEL_PIN_28 : Pin 28
        - PINSEL_PIN_29 : Pin 29
        - PINSEL_PIN_30 : Pin 30
        - PINSEL_PIN_31 : Pin 31

* @param[in]  modenum  Open drain mode number,
*
*                      should be one of the following:
*
*                      - PINSEL_PINMODE_NORMAL : Pin is in the normal (not open
drain) mode
*
*                      - PINSEL_PINMODE_OPENDRAIN : Pin is in the open drain mode
*
* @return           None
*****
void set_OpenDrainMode( uint8_t portnum, uint8_t pinnum, uint8_t modenum)
{
    uint32_t *pPinCon = (uint32_t *)&LPC_PINCON->PINMODE_ODO;

    if (modenum == PINSEL_PINMODE_OPENDRAIN){
        *(uint32_t *) (pPinCon + portnum) |= (0x01UL << pinnum);
    } else {
        *(uint32_t *) (pPinCon + portnum) &= ~(0x01UL << pinnum);
    }
}

/* End of Public Functions ----- */

/* Public Functions ----- */
/** @addtogroup PINSEL_Public_Functions
 *
* @{
*
***** //**/

* @brief          Configure trace function
* @param[in]  NewState State of the Trace function configuration,
*
*                      should be one of the following:
*
*                      - ENABLE : Enable Trace Function
*
*                      - DISABLE : Disable Trace Function
*
* @return         None

```

```
******/  
void PINSEL_ConfigTraceFunc(FunctionalState NewState)  
{  
    if (NewState == ENABLE) {  
        LPC_PINCON->PINSEL10 |= (0x01UL << 3);  
    } else if (NewState == DISABLE) {  
        LPC_PINCON->PINSEL10 &= ~(0x01UL << 3);  
    }  
}  
  
*****/**/  
 * @brief          Setup I2C0 pins  
 * @param[in]      i2cPinMode I2C pin mode,  
 *                 should be one of the following:  
 *                 - PINSEL_I2C_Normal_Mode : The standard drive mode  
 *                 - PINSEL_I2C_Fast_Mode : Fast Mode Plus drive mode  
 *  
 * @param[in]      filterSlewRateEnable should be:  
 *                 - ENABLE: Enable filter and slew rate.  
 *                 - DISABLE: Disable filter and slew rate.  
 *  
 * @return         None  
*****/  
void PINSEL_SetI2C0Pins(uint8_t i2cPinMode, FunctionalState filterSlewRateEnable)  
{  
    uint32_t regVal;  
  
    if (i2cPinMode == PINSEL_I2C_Fast_Mode){  
        regVal = PINSEL_I2CPADCFG_SCLDRV0 | PINSEL_I2CPADCFG_SDADDRV0;  
    }  
  
    if (filterSlewRateEnable == DISABLE){  
        regVal = PINSEL_I2CPADCFG_SCLI2C0 | PINSEL_I2CPADCFG_SDAI2C0;  
    }  
    LPC_PINCON->I2CPADCFG = regVal;  
}  
  
*****/**/
```

```
* @brief           Configure Pin corresponding to specified parameters passed
*                   in the PinCfg
* @param[in]      PinCfg  Pointer to a PINSEL_CFG_Type structure
*                   that contains the configuration information for the
*                   specified pin.
* @return          None
*****void PINSEL_ConfigPin(PINSEL_CFG_Type *PinCfg)
{
    set_PinFunc(PinCfg->Portnum, PinCfg->Pinnum, PinCfg->Funcnum);
    set_ResistorMode(PinCfg->Portnum, PinCfg->Pinnum, PinCfg->Pinmode);
    set_OpenDrainMode(PinCfg->Portnum, PinCfg->Pinnum, PinCfg->OpenDrain);
}

/**
 * @}
 */

/***
 * @}
 */

/* ----- End Of File ----- */
*****
```

* \$Id\$ lpc17xx_gpio.c 2010-05-21
*/
* @file lpc17xx_gpio.c
* @brief Contains all functions support for GPIO firmware
* library on LPC17xx
* @version 2.0
* @date 21. May. 2010
* @author NXP MCU SW Application Team
*
* Copyright (C) 2010, NXP Semiconductor
* All rights reserved.

```
*  
*****  
* Software that is described herein is for illustrative purposes only  
* which provides customers with programming information regarding the  
* products. This software is supplied "AS IS" without any warranties.  
* NXP Semiconductors assumes no responsibility or liability for the  
* use of the software, conveys no license or title under any patent,  
* copyright, or mask work right to the product. NXP Semiconductors  
* reserves the right to make changes in the software without  
* notification. NXP Semiconductors also make no representation or  
* warranty that such application will be suitable for the specified  
* use without further testing or modification.  
*****/  
  
/* Peripheral group ----- */  
/** @addtogroup GPIO  
 * @ {  
 * /  
  
/* Includes ----- */  
#include "lpc17xx_gpio.h"  
  
/* If this source file built with example, the LPC17xx FW library configuration  
* file in each example directory ("lpc17xx_libcfg.h") must be included,  
* otherwise the default FW library configuration file must be included instead  
*/  
#ifdef __BUILD_WITH_EXAMPLE__  
#include "lpc17xx_libcfg.h"  
#else  
#include "lpc17xx_libcfg_default.h"  
#endif /* __BUILD_WITH_EXAMPLE__ */  
  
#ifdef _GPIO  
  
/* Private Functions ----- */  
  
static LPC_GPIO_TypeDef *GPIO_GetPointer(uint8_t portNum);  
static GPIO_HalfWord_TypeDef *FIO_HalfWordGetPointer(uint8_t portNum);
```

```
static GPIO_Bit_TypeDef *FIO_BitGetPointer(uint8_t portNum);

/*****brief Get pointer to FIO peripheral due to FIO port
 * @param[in] portNum Port Number value, should be in range from 0 to 4.
 * @return Pointer to FIO peripheral
 *****/
static LPC_GPIO_TypeDef *GPIO_GetPointer(uint8_t portNum)
{
    LPC_GPIO_TypeDef *pGPIO = NULL;

    switch (portNum) {
        case 0:
            pGPIO = LPC_GPIO0;
            break;
        case 1:
            pGPIO = LPC_GPIO1;
            break;
        case 2:
            pGPIO = LPC_GPIO2;
            break;
        case 3:
            pGPIO = LPC_GPIO3;
            break;
        case 4:
            pGPIO = LPC_GPIO4;
            break;
        default:
            break;
    }

    return pGPIO;
}

/*****brief Get pointer to FIO peripheral in halfword accessible style
 * due to FIO port
 * @param[in] portNum Port Number value, should be in range from 0 to 4.
 * @return Pointer to FIO peripheral
 *****/
```

```
*****
static GPIO_HalfWord_TypeDef *FIO_HalfWordGetPointer(uint8_t portNum)
{
    GPIO_HalfWord_TypeDef *pFIO = NULL;

    switch (portNum) {
        case 0:
            pFIO = GPIO0_HalfWord;
            break;
        case 1:
            pFIO = GPIO1_HalfWord;
            break;
        case 2:
            pFIO = GPIO2_HalfWord;
            break;
        case 3:
            pFIO = GPIO3_HalfWord;
            break;
        case 4:
            pFIO = GPIO4_HalfWord;
            break;
        default:
            break;
    }

    return pFIO;
}

/** @brief          Get pointer to FIO peripheral in byte accessible style
 * @param[in]      portNum      Port Number value, should be in range from 0 to 4.
 * @return         Pointer to FIO peripheral
 */
static GPIO_Byte_TypeDef *FIO_ByteGetPointer(uint8_t portNum)
{
    GPIO_Byte_TypeDef *pFIO = NULL;

    switch (portNum) {
```

```
    case 0:
        pFIO = GPIO0_Byte;
        break;

    case 1:
        pFIO = GPIO1_Byte;
        break;

    case 2:
        pFIO = GPIO2_Byte;
        break;

    case 3:
        pFIO = GPIO3_Byte;
        break;

    case 4:
        pFIO = GPIO4_Byte;
        break;

    default:
        break;
    }

    return pFIO;
}

/* End of Private Functions ----- */
```

```
/* Public Functions ----- */
/** @addtogroup GPIO_Public_Functions
 * @{
 */


```

```
/* GPIO -----
 */
```

```
*****//**
 * @brief          Set Direction for GPIO port.
 * @param[in]      portNum      Port Number value, should be in range from 0 to 4
 * @param[in]      bitValue     Value that contains all bits to set direction,
 *                             in range from 0 to 0xFFFFFFFF.
 *
```

```
*                                     example: value 0x5 to set direction
for bit 0 and bit 1.

* @param[in]  dir                  Direction value, should be:
*
*                                         - 0: Input.
*
*                                         - 1: Output.

* @return                 None

*

* Note: All remaining bits that are not activated in bitValue (value '0')
* will not be effected by this function.

*****void GPIO_SetDir(uint8_t portNum, uint32_t bitValue, uint8_t dir)
{

    LPC_GPIO_TypeDef *pGPIO = GPIO_GetPointer(portNum);

    if (pGPIO != NULL) {

        // Enable Output

        if (dir) {

            pGPIO->FIODIR |= bitValue;

        }

        // Enable Input

        else {

            pGPIO->FIODIR &= ~bitValue;

        }

    }

}

*****/* @brief          Set Value for bits that have output direction on GPIO port.

* @param[in]  portNum      Port number value, should be in range from 0 to 4
* @param[in]  bitValue     Value that contains all bits on GPIO to set,
*                         in range from 0 to 0xFFFFFFFF.

*                                         example: value 0x5 to set bit 0 and
bit 1.

* @return                 None

*

* Note:

* - For all bits that has been set as input direction, this function will
not effect.

* - For all remaining bits that are not activated in bitValue (value '0')
```

```
* will not be effected by this function.  
*****  
void GPIO_SetValue(uint8_t portNum, uint32_t bitValue)  
{  
    LPC_GPIO_TypeDef *pGPIO = GPIO_GetPointer(portNum);  
  
    if (pGPIO != NULL) {  
        pGPIO->FIOSET = bitValue;  
    }  
}  
  
*****//**  
 * @brief           Clear Value for bits that have output direction on GPIO port.  
 * @param[in]      portNum       Port number value, should be in range from 0 to 4  
 * @param[in]      bitValue     Value that contains all bits on GPIO to clear,  
 *                             in range from 0 to 0xFFFFFFFF.  
 *                             example: value 0x5 to clear bit 0  
 and bit 1.  
 * @return          None  
 *  
 * Note:  
 * - For all bits that has been set as input direction, this function will  
 * not effect.  
 * - For all remaining bits that are not activated in bitValue (value '0')  
 * will not be effected by this function.  
*****/  
void GPIO_ClearValue(uint8_t portNum, uint32_t bitValue)  
{  
    LPC_GPIO_TypeDef *pGPIO = GPIO_GetPointer(portNum);  
  
    if (pGPIO != NULL) {  
        pGPIO->FIOCLR = bitValue;  
    }  
}  
  
*****//**  
 * @brief           Read Current state on port pin that have input direction of GPIO  
 * @param[in]      portNum       Port number to read value, in range from 0 to 4  
 * @return          Current value of GPIO port.  
 *
```

```
* Note: Return value contain state of each port pin (bit) on that GPIO regardless
* its direction is input or output.

*****
uint32_t GPIO_ReadValue(uint8_t portNum)
{
    LPC_GPIO_TypeDef *pGPIO = GPIO_GetPointer(portNum);

    if (pGPIO != NULL) {
        return pGPIO->FIOPIN;
    }

    return (0);
}

****

* @brief          Enable GPIO interrupt (just used for P0.0-P0.30, P2.0-P2.13)
* @param[in]      portNum      Port number to read value, should be: 0 or 2
* @param[in]      bitValue     Value that contains all bits on GPIO to enable,
*                             in range from 0 to 0xFFFFFFFF.
* @param[in]      edgeState    state of edge, should be:
*                             - 0: Rising edge
*                             - 1: Falling edge
* @return         None
****

void GPIO_IntCmd(uint8_t portNum, uint32_t bitValue, uint8_t edgeState)
{
    if((portNum == 0)&&(edgeState == 0))
        LPC_GPIOINT->IO0IntEnR = bitValue;
    else if ((portNum == 2)&&(edgeState == 0))
        LPC_GPIOINT->IO2IntEnR = bitValue;
    else if ((portNum == 0)&&(edgeState == 1))
        LPC_GPIOINT->IO0IntEnF = bitValue;
    else if ((portNum == 2)&&(edgeState == 1))
        LPC_GPIOINT->IO2IntEnF = bitValue;
    else
        //Error
        while(1);
}
```

```

/*********************//**
 * @brief           Get GPIO Interrupt Status (just used for P0.0-P0.30, P2.0-P2.13)
 * @param[in]      portNum       Port number to read value, should be: 0 or 2
 * @param[in]      pinNum        Pin number, should be: 0..30 (with port 0) and 0..13
 *
 *                               (with port 2)
 * @param[in]      edgeState     state of edge, should be:
 *
 *                               - 0: Rising edge
 *
 *                               - 1: Falling edge
 * @return          Bool         could be:
 *
 *                               - ENABLE: Interrupt has been generated due
 * to a rising
 *
 *                               - edge on P0.0
 *
 *                               - DISABLE: A rising edge has not been
 * detected on P0.0
 *****/
FunctionalState GPIO_GetIntStatus(uint8_t portNum, uint32_t pinNum, uint8_t edgeState)
{
    if((portNum == 0) && (edgeState == 0))//Rising Edge
        return ((FunctionalState)((LPC_GPIOINT->IO0IntStatR)>>pinNum)& 0x1));
    else if ((portNum == 2) && (edgeState == 0))
        return ((FunctionalState)((LPC_GPIOINT->IO2IntStatR)>>pinNum)& 0x1));
    else if ((portNum == 0) && (edgeState == 1))//Falling Edge
        return ((FunctionalState)((LPC_GPIOINT->IO0IntStatF)>>pinNum)& 0x1));
    else if ((portNum == 2) && (edgeState == 1))
        return ((FunctionalState)((LPC_GPIOINT->IO2IntStatF)>>pinNum)& 0x1));
    else
        //Error
        while(1);
}
/*********************//**
 * @brief           Clear GPIO interrupt (just used for P0.0-P0.30, P2.0-P2.13)
 * @param[in]      portNum       Port number to read value, should be: 0 or 2
 * @param[in]      bitValue      Value that contains all bits on GPIO to enable,
 *
 *                               in range from 0 to 0xFFFFFFFF.
 * @return          None
 *****/
void GPIO_ClearInt(uint8_t portNum, uint32_t bitValue)
{
    if(portNum == 0)
        LPC_GPIOINT->IO0IntClr = bitValue;

```

```
    else if (portNum == 2)
        LPC_GPIOINT->IO2IntClr = bitValue;
    else
        //Invalid portNum
        while(1);
}

/* FIO word accessible -----
 */
/* Stub function for FIO (word-accessible) style */

/**
 * @brief The same with GPIO_SetDir()
 */
void FIO_SetDir(uint8_t portNum, uint32_t bitValue, uint8_t dir)
{
    GPIO_SetDir(portNum, bitValue, dir);
}

/**
 * @brief The same with GPIO_SetValue()
 */
void FIO_SetValue(uint8_t portNum, uint32_t bitValue)
{
    GPIO_SetValue(portNum, bitValue);
}

/**
 * @brief The same with GPIO_ClearValue()
 */
void FIO_ClearValue(uint8_t portNum, uint32_t bitValue)
{
    GPIO_ClearValue(portNum, bitValue);
}

/**
 * @brief The same with GPIO_ReadValue()
 */
uint32_t FIO_ReadValue(uint8_t portNum)
{
```

```
        return (GPIO_ReadValue(portNum)) ;

    }

/***
 * @brief The same with GPIO_IntCmd()
 */
void FIO_IntCmd(uint8_t portNum, uint32_t bitValue, uint8_t edgeState)
{
    GPIO_IntCmd(portNum, bitValue, edgeState);
}

/***
 * @brief The same with GPIO.GetIntStatus()
 */
FunctionalState FIO.GetIntStatus(uint8_t portNum, uint32_t pinNum, uint8_t edgeState)
{
    return (GPIO.GetIntStatus(portNum, pinNum, edgeState));
}

/***
 * @brief The same with GPIO_ClearInt()
 */
void FIO_ClearInt(uint8_t portNum, uint32_t bitValue)
{
    GPIO_ClearInt(portNum, bitValue);
}

/********************************************/***/

* @brief Set mask value for bits in FIO port
* @param[in] portNum Port number, in range from 0 to 4
* @param[in] bitValue Value that contains all bits in to set,
*                     in range from 0 to 0xFFFFFFFF.
* @param[in] maskValue Mask value contains state value for each bit:
*                     - 0: not mask.
*                     - 1: mask.
* @return None
*
* Note:
* - All remaining bits that are not activated in bitValue (value '0')
* will not be effected by this function.
```

```
* - After executing this function, in mask register, value '0' on each bit
* enables an access to the corresponding physical pin via a read or write access,
* while value '1' on bit (masked) that corresponding pin will not be changed
* with write access and if read, will not be reflected in the updated pin.

*****
void FIO_SetMask(uint8_t portNum, uint32_t bitValue, uint8_t maskValue)
{
    LPC_GPIO_TypeDef *pFIO = GPIO_GetPointer(portNum);

    if(pFIO != NULL) {
        // Mask
        if (maskValue) {
            pFIO->FIOMASK |= bitValue;
        }
        // Un-mask
        else {
            pFIO->FIOMASK &= ~bitValue;
        }
    }
}

/*
 * FIO halfword accessible -----
 */

*****/**/

* @brief          Set direction for FIO port in halfword accessible style
* @param[in]      portNum      Port number, in range from 0 to 4
* @param[in]      halfwordNum  HalfWord part number, should be 0 (lower) or 1(upper)
* @param[in]      bitValue     Value that contains all bits in to set direction,
*                             in range from 0 to 0xFFFF.
* @param[in]      dir          Direction value, should be:
*                             - 0: Input.
*                             - 1: Output.
* @return         None
*
* Note: All remaining bits that are not activated in bitValue (value '0')
* will not be effected by this function.
*****
void FIO_HalfWordSetDir(uint8_t portNum, uint8_t halfwordNum, uint16_t bitValue, uint8_t
dir)
```

```
{  
    GPIO_HalfWord_TypeDef *pFIO = FIO_HalfWordGetPointer(portNum);  
  
    if(pFIO != NULL) {  
        // Output direction  
  
        if (dir) {  
            // Upper  
  
            if(halfwordNum) {  
                pFIO->FIODIRU |= bitValue;  
            }  
  
            // lower  
  
            else {  
                pFIO->FIODIRL |= bitValue;  
            }  
        }  
  
        // Input direction  
  
        else {  
            // Upper  
  
            if(halfwordNum) {  
                pFIO->FIODIRU &= ~bitValue;  
            }  
  
            // lower  
  
            else {  
                pFIO->FIODIRL &= ~bitValue;  
            }  
        }  
    }  
}  
  
/******//**  
  
 * @brief Set mask value for bits in FIO port in halfword accessible style  
 * @param[in] portNum Port number, in range from 0 to 4  
 * @param[in] halfwordNum HalfWord part number, should be 0 (lower) or 1(upper)  
 * @param[in] bitValue Value that contains all bits in to set,  
 *                     in range from 0 to 0xFFFF.  
 * @param[in] maskValue Mask value contains state value for each bit:  
 *                     - 0: not mask.  
 *                     - 1: mask.  
 * @return None
```

```
*  
* Note:  
* - All remaining bits that are not activated in bitValue (value '0')  
* will not be effected by this function.  
* - After executing this function, in mask register, value '0' on each bit  
* enables an access to the corresponding physical pin via a read or write access,  
* while value '1' on bit (masked) that corresponding pin will not be changed  
* with write access and if read, will not be reflected in the updated pin.  
*****  
void FIO_HalfWordSetMask(uint8_t portNum, uint8_t halfwordNum, uint16_t bitValue,  
uint8_t maskValue)  
{  
    GPIO_HalfWord_TypeDef *pFIO = FIO_HalfWordGetPointer(portNum);  
    if(pFIO != NULL) {  
        // Mask  
        if (maskValue) {  
            // Upper  
            if(halfwordNum) {  
                pFIO->FIOMASKU |= bitValue;  
            }  
            // lower  
            else {  
                pFIO->FIOMASKL |= bitValue;  
            }  
        }  
        // Un-mask  
        else {  
            // Upper  
            if(halfwordNum) {  
                pFIO->FIOMASKU &= ~bitValue;  
            }  
            // lower  
            else {  
                pFIO->FIOMASKL &= ~bitValue;  
            }  
        }  
    }  
}
```

```
*****/**/  
 * @brief Set bits for FIO port in halfword accessible style  
 * @param[in] portNum Port number, in range from 0 to 4  
 * @param[in] halfwordNum HalfWord part number, should be 0 (lower) or 1(upper)  
 * @param[in] bitValue Value that contains all bits in to set,  
 *                     in range from 0 to 0xFFFF.  
 * @return None  
 *  
 * Note:  
 * - For all bits that has been set as input direction, this function will  
 * not effect.  
 * - For all remaining bits that are not activated in bitValue (value '0')  
 * will not be effected by this function.  
*****/**/  
void FIO_HalfWordSetValue(uint8_t portNum, uint8_t halfwordNum, uint16_t bitValue)  
{  
    GPIO_HalfWord_TypeDef *pFIO = FIO_HalfWordGetPointer(portNum);  
    if(pFIO != NULL) {  
        // Upper  
        if(halfwordNum) {  
            pFIO->FIOSETU = bitValue;  
        }  
        // lower  
        else {  
            pFIO->FIOSETL = bitValue;  
        }  
    }  
}  
  
*****/**/  
 * @brief Clear bits for FIO port in halfword accessible style  
 * @param[in] portNum Port number, in range from 0 to 4  
 * @param[in] halfwordNum HalfWord part number, should be 0 (lower) or 1(upper)  
 * @param[in] bitValue Value that contains all bits in to clear,  
 *                     in range from 0 to 0xFFFF.  
 * @return None  
 *  
 * Note:
```

```
* - For all bits that has been set as input direction, this function will
* not effect.

* - For all remaining bits that are not activated in bitValue (value '0')
* will not be effected by this function.

*****
void FIO_HalfWordClearValue(uint8_t portNum, uint8_t halfwordNum, uint16_t bitValue)
{
    GPIO_HalfWord_TypeDef *pFIO = FIO_HalfWordGetPointer(portNum);

    if(pFIO != NULL) {
        // Upper
        if(halfwordNum) {
            pFIO->FIOCLRU = bitValue;
        }
        // lower
        else {
            pFIO->FIOCLRL = bitValue;
        }
    }
}

/*****
 * @brief           Read Current state on port pin that have input direction of GPIO
 *                 in halfword accessible style.
 * @param[in]      portNum       Port number, in range from 0 to 4
 * @param[in]      halfwordNum   HalfWord part number, should be 0 (lower) or 1(upper)
 * @return          Current value of FIO port pin of specified halfword.
 * Note: Return value contain state of each port pin (bit) on that FIO regardless
 * its direction is input or output.
 ****/
uint16_t FIO_HalfWordReadValue(uint8_t portNum, uint8_t halfwordNum)
{
    GPIO_HalfWord_TypeDef *pFIO = FIO_HalfWordGetPointer(portNum);

    if(pFIO != NULL) {
        // Upper
        if(halfwordNum) {
            return (pFIO->FIOPINU);
        }
        // lower
```

```
        else {
            return (pFIO->FIOPINL);
        }
    }

    return (0);
}

/* FIO Byte accessible ----- */

/********************* brief *************************/
* @brief           Set direction for FIO port in byte accessible style
* @param[in]      portNum       Port number, in range from 0 to 4
* @param[in]      byteNum       Byte part number, should be in range from 0 to 3
* @param[in]      bitValue      Value that contains all bits in to set direction,
*                               in range from 0 to 0xFF.
* @param[in]      dir          Direction value, should be:
*                               - 0: Input.
*                               - 1: Output.
* @return         None
*
* Note: All remaining bits that are not activated in bitValue (value '0')
* will not be effected by this function.
********************* end brief ******************/


void FIO_ByteSetDir(uint8_t portNum, uint8_t byteNum, uint8_t bitValue, uint8_t dir)
{
    GPIO_Byte_TypeDef *pFIO = FIO_ByteGetPointer(portNum);

    if (pFIO != NULL) {
        // Output direction
        if (dir) {
            if (byteNum <= 3) {
                pFIO->FIODIR[byteNum] |= bitValue;
            }
        }
        // Input direction
        else {
            if (byteNum <= 3) {
                pFIO->FIODIR[byteNum] &= ~bitValue;
            }
        }
    }
}
```

```

        }

    }

}

/*****



* @brief          Set mask value for bits in FIO port in byte accessible style
* @param[in]      portNum       Port number, in range from 0 to 4
* @param[in]      byteNum       Byte part number, should be in range from 0 to 3
* @param[in]      bitValue      Value that contains all bits in to set mask,
*                               in range from 0 to 0xFF.
* @param[in]      maskValue     Mask value contains state value for each bit:
*                               - 0: not mask.
*                               - 1: mask.

* @return         None

*

* Note:
* - All remaining bits that are not activated in bitValue (value '0')
* will not be effected by this function.
* - After executing this function, in mask register, value '0' on each bit
* enables an access to the corresponding physical pin via a read or write access,
* while value '1' on bit (masked) that corresponding pin will not be changed
* with write access and if read, will not be reflected in the updated pin.
*****



void FIO_ByteSetMask(uint8_t portNum, uint8_t byteNum, uint8_t bitValue, uint8_t
maskValue)
{
    GPIO_Byte_TypeDef *pFIO = FIO_ByteGetPointer(portNum);
    if(pFIO != NULL) {
        // Mask
        if (maskValue) {
            if (byteNum <= 3) {
                pFIO->FIOMASK[byteNum] |= bitValue;
            }
        }
        // Un-mask
        else {
            if (byteNum <= 3) {
                pFIO->FIOMASK[byteNum] &= ~bitValue;
            }
        }
    }
}

```

```
}

}

/*****



* @brief          Set bits for FIO port in byte accessible style
* @param[in]      portNum        Port number, in range from 0 to 4
* @param[in]      byteNum        Byte part number, should be in range from 0 to 3
* @param[in]      bitValue       Value that contains all bits in to set,
*                               in range from 0 to 0xFF.
* @return         None
*
* Note:
* - For all bits that has been set as input direction, this function will
* not effect.
* - For all remaining bits that are not activated in bitValue (value '0')
* will not be effected by this function.
*****


void FIO_ByteSetValue(uint8_t portNum, uint8_t byteNum, uint8_t bitValue)
{
    GPIO_Byte_TypeDef *pFIO = FIO_ByteGetPointer(portNum);
    if (pFIO != NULL) {
        if (byteNum <= 3) {
            pFIO->FIOSET[byteNum] = bitValue;
        }
    }
}

/*****



* @brief          Clear bits for FIO port in byte accessible style
* @param[in]      portNum        Port number, in range from 0 to 4
* @param[in]      byteNum        Byte part number, should be in range from 0 to 3
* @param[in]      bitValue       Value that contains all bits in to clear,
*                               in range from 0 to 0xFF.
* @return         None
*
* Note:
* - For all bits that has been set as input direction, this function will
```

```
* not effect.

* - For all remaining bits that are not activated in bitValue (value '0')
* will not be effected by this function.

*****
void FIO_ByteClearValue(uint8_t portNum, uint8_t byteNum, uint8_t bitValue)
{
    GPIO_Byte_TypeDef *pFIO = FIO_ByteGetPointer(portNum);

    if (pFIO != NULL) {
        if (byteNum <= 3) {
            pFIO->FIOCLR[byteNum] = bitValue;
        }
    }
}

/****

* @brief          Read Current state on port pin that have input direction of GPIO
*                 in byte accessible style.
* @param[in]      portNum      Port number, in range from 0 to 4
* @param[in]      byteNum      Byte part number, should be in range from 0 to 3
* @return         Current value of FIO port pin of specified byte part.
* Note: Return value contain state of each port pin (bit) on that FIO regardless
* its direction is input or output.

*****
uint8_t FIO_ByteReadValue(uint8_t portNum, uint8_t byteNum)
{
    GPIO_Byte_TypeDef *pFIO = FIO_ByteGetPointer(portNum);

    if (pFIO != NULL) {
        if (byteNum <= 3) {
            return (pFIO->FIOPIN[byteNum]);
        }
    }
    return (0);
}

/***
* @}
*/

```

```
#endif /* _GPIO */  
  
/**  
 * @}  
 */  
  
/* ----- End Of File ----- */  
  
*****  
* $Id$      lpc17xx_clkpwr.c          2010-06-18  
*/  
  
* @file      lpc17xx_clkpwr.c  
* @brief     Contains all functions support for Clock and Power Control  
*             firmware library on LPC17xx  
* @version   3.0  
* @date      18. June. 2010  
* @author    NXP MCU SW Application Team  
*  
* Copyright (C) 2010, NXP Semiconductor  
* All rights reserved.  
*  
*****  
* Software that is described herein is for illustrative purposes only  
* which provides customers with programming information regarding the  
* products. This software is supplied "AS IS" without any warranties.  
* NXP Semiconductors assumes no responsibility or liability for the  
* use of the software, conveys no license or title under any patent,  
* copyright, or mask work right to the product. NXP Semiconductors  
* reserves the right to make changes in the software without  
* notification. NXP Semiconductors also make no representation or  
* warranty that such application will be suitable for the specified  
* use without further testing or modification.  
*****/  
  
/* Peripheral group ----- */  
/** @addtogroup CLKPWR  
* @{  
*/
```

```
/* Includes ----- */
#include "lpc17xx_clkpwr.h"

/* Public Functions ----- */
/** @addtogroup CLKPWR_Public_Functions
 * @{
 */

/***** Set value of each Peripheral Clock Selection *****/
* @brief Set value of each Peripheral Clock Selection
* @param[in] ClkType Peripheral Clock Selection of each type,
*           should be one of the following:
*           - CLKPWR_PCLKSEL_WDT : WDT
*           - CLKPWR_PCLKSEL_TIMER0 : Timer 0
*           - CLKPWR_PCLKSEL_TIMER1 : Timer 1
*           - CLKPWR_PCLKSEL_UART0 : UART 0
*           - CLKPWR_PCLKSEL_UART1 : UART 1
*           - CLKPWR_PCLKSEL_PWM1 : PWM 1
*           - CLKPWR_PCLKSEL_I2C0 : I2C 0
*           - CLKPWR_PCLKSEL_SPI : SPI
*           - CLKPWR_PCLKSEL_SSP1 : SSP 1
*           - CLKPWR_PCLKSEL_DAC : DAC
*           - CLKPWR_PCLKSEL_ADC : ADC
*           - CLKPWR_PCLKSEL_CAN1 : CAN 1
*           - CLKPWR_PCLKSEL_CAN2 : CAN 2
*           - CLKPWR_PCLKSEL_ACF : ACF
*           - CLKPWR_PCLKSEL_QEI : QEI
*           - CLKPWR_PCLKSEL_PCB : PCB
*           - CLKPWR_PCLKSEL_I2C1 : I2C 1
*           - CLKPWR_PCLKSEL_SSP0 : SSP 0
*           - CLKPWR_PCLKSEL_TIMER2 : Timer 2
*           - CLKPWR_PCLKSEL_TIMER3 : Timer 3
*           - CLKPWR_PCLKSEL_UART2 : UART 2
*           - CLKPWR_PCLKSEL_UART3 : UART 3
*           - CLKPWR_PCLKSEL_I2C2 : I2C 2
*           - CLKPWR_PCLKSEL_I2S : I2S
*           - CLKPWR_PCLKSEL_RIT : RIT
```

```

        - CLKPWR_PCLKSEL_SYSCON      : SYSCON
        - CLKPWR_PCLKSEL_MC          : MC

* @param[in] DivVal Value of divider, should be:
*
*           - CLKPWR_PCLKSEL_CCLK_DIV_4 : PCLK_peripheral = CCLK/4
*
*           - CLKPWR_PCLKSEL_CCLK_DIV_1 : PCLK_peripheral = CCLK/1
*
*           - CLKPWR_PCLKSEL_CCLK_DIV_2 : PCLK_peripheral = CCLK/2
*
* @return none
*****void CLKPWR_SetPCLKDiv (uint32_t ClkType, uint32_t DivVal)
{
    uint32_t bitpos;

    bitpos = (ClkType < 32) ? (ClkType) : (ClkType - 32);

    /* PCLKSEL0 selected */
    if (ClkType < 32)
    {
        /* Clear two bit at bit position */
        LPC_SC->PCLKSEL0 &= ~(CLKPWR_PCLKSEL_BITMASK(bitpos));

        /* Set two selected bit */
        LPC_SC->PCLKSEL0 |= (CLKPWR_PCLKSEL_SET(bitpos, DivVal));
    }
    /* PCLKSEL1 selected */
    else
    {
        /* Clear two bit at bit position */
        LPC_SC->PCLKSEL1 &= ~CLKPWR_PCLKSEL_BITMASK(bitpos);

        /* Set two selected bit */
        LPC_SC->PCLKSEL1 |= (CLKPWR_PCLKSEL_SET(bitpos, DivVal));
    }
}

***** * @brief          Get current value of each Peripheral Clock Selection

```

```

* @param[in] ClkType Peripheral Clock Selection of each type,
*
*                      should be one of the following:
*
* - CLKPWR_PCLKSEL_WDT          : WDT
*
* - CLKPWR_PCLKSEL_TIMER0       : Timer 0
*
* - CLKPWR_PCLKSEL_TIMER1       : Timer 1
*
* - CLKPWR_PCLKSEL_UART0        : UART 0
*
* - CLKPWR_PCLKSEL_UART1        : UART 1
*
* - CLKPWR_PCLKSEL_PWM1         : PWM 1
*
* - CLKPWR_PCLKSEL_I2C0         : I2C 0
*
* - CLKPWR_PCLKSEL_SPI          : SPI
*
* - CLKPWR_PCLKSEL_SSP1         : SSP 1
*
* - CLKPWR_PCLKSEL_DAC          : DAC
*
* - CLKPWR_PCLKSEL_ADC          : ADC
*
* - CLKPWR_PCLKSEL_CAN1         : CAN 1
*
* - CLKPWR_PCLKSEL_CAN2         : CAN 2
*
* - CLKPWR_PCLKSEL_ACF          : ACF
*
* - CLKPWR_PCLKSEL_QEI          : QEI
*
* - CLKPWR_PCLKSEL_PCB          : PCB
*
* - CLKPWR_PCLKSEL_I2C1         : I2C 1
*
* - CLKPWR_PCLKSEL_SSP0         : SSP 0
*
* - CLKPWR_PCLKSEL_TIMER2        : Timer 2
*
* - CLKPWR_PCLKSEL_TIMER3        : Timer 3
*
* - CLKPWR_PCLKSEL_UART2         : UART 2
*
* - CLKPWR_PCLKSEL_UART3         : UART 3
*
* - CLKPWR_PCLKSEL_I2C2         : I2C 2
*
* - CLKPWR_PCLKSEL_I2S           : I2S
*
* - CLKPWR_PCLKSEL_RIT           : RIT
*
* - CLKPWR_PCLKSEL_SYSCON        : SYSCON
*
* - CLKPWR_PCLKSEL_MC            : MC

* @return          Value of Selected Peripheral Clock Selection
***** */

uint32_t CLKPWR_GetPCLKSEL (uint32_t ClkType)
{
    uint32_t bitpos, retval;

    if (ClkType < 32)
    {
        bitpos = ClkType;
    }
}

```

```
        retval = LPC_SC->PCLKSEL0;
    }
else
{
    bitpos = ClkType - 32;
    retval = LPC_SC->PCLKSEL1;
}

retval = CLKPWR_PCLKSEL_GET(bitpos, retval);
return retval;
}

/*********************//**/

* @brief           Get current value of each Peripheral Clock
* @param[in]      ClkType Peripheral Clock Selection of each type,
*                 should be one of the following:
*
* - CLKPWR_PCLKSEL_WDT          : WDT
* - CLKPWR_PCLKSEL_TIMER0       : Timer 0
* - CLKPWR_PCLKSEL_TIMER1       : Timer 1
* - CLKPWR_PCLKSEL_UART0        : UART 0
* - CLKPWR_PCLKSEL_UART1        : UART 1
* - CLKPWR_PCLKSEL_PWM1         : PWM 1
* - CLKPWR_PCLKSEL_I2C0          : I2C 0
* - CLKPWR_PCLKSEL_SPI          : SPI
* - CLKPWR_PCLKSEL_SSP1          : SSP 1
* - CLKPWR_PCLKSEL_DAC          : DAC
* - CLKPWR_PCLKSEL_ADC          : ADC
* - CLKPWR_PCLKSEL_CAN1          : CAN 1
* - CLKPWR_PCLKSEL_CAN2          : CAN 2
* - CLKPWR_PCLKSEL_ACF          : ACF
* - CLKPWR_PCLKSEL_QEI          : QEI
* - CLKPWR_PCLKSEL_PCB          : PCB
* - CLKPWR_PCLKSEL_I2C1          : I2C 1
* - CLKPWR_PCLKSEL_SSP0          : SSP 0
* - CLKPWR_PCLKSEL_TIMER2        : Timer 2
* - CLKPWR_PCLKSEL_TIMER3        : Timer 3
* - CLKPWR_PCLKSEL_UART2          : UART 2
```

```
        - CLKPWR_PCLKSEL_UART3      : UART 3
        - CLKPWR_PCLKSEL_I2C2       : I2C 2
        - CLKPWR_PCLKSEL_I2S         : I2S
        - CLKPWR_PCLKSEL_RIT         : RIT
        - CLKPWR_PCLKSEL_SYSCON     : SYSCON
        - CLKPWR_PCLKSEL_MC          : MC

* @return           Value of Selected Peripheral Clock
*****
uint32_t CLKPWR_GetPCLK (uint32_t ClkType)
{
    uint32_t retval, div;

    retval = SystemCoreClock;
    div = CLKPWR_GetPCLKSEL(ClkType);

    switch (div)
    {
        case 0:
            div = 4;
            break;

        case 1:
            div = 1;
            break;

        case 2:
            div = 2;
            break;

        case 3:
            div = 8;
            break;
    }
    retval /= div;

    return retval;
}
```

```
/**************************************************************************/<!--

* @brief           Configure power supply for each peripheral according to NewState

* @param[in] PPType Type of peripheral used to enable power,
*
*                   should be one of the following:
*
* - CLKPWR_PCONP_PCTIMO          : Timer 0
*   - CLKPWR_PCONP_PCTIM1        : Timer 1
*   - CLKPWR_PCONP_PCUART0       : UART 0
*   - CLKPWR_PCONP_PCUART1       : UART 1
*   - CLKPWR_PCONP_PCPWM1        : PWM 1
*   - CLKPWR_PCONP_PCI2CO        : I2C 0
*   - CLKPWR_PCONP_PCSPI         : SPI
*   - CLKPWR_PCONP_PCRTC         : RTC
*   - CLKPWR_PCONP_PCSSP1        : SSP 1
*   - CLKPWR_PCONP_PCAD          : ADC
*   - CLKPWR_PCONP_PCAN1         : CAN 1
*   - CLKPWR_PCONP_PCAN2         : CAN 2
*   - CLKPWR_PCONP_PCGPIO        : GPIO
*   - CLKPWR_PCONP_PCRIT         : RIT
*   - CLKPWR_PCONP_PCMC          : MC
*   - CLKPWR_PCONP_PCQEI         : QEI
*   - CLKPWR_PCONP_PCI2C1        : I2C 1
*   - CLKPWR_PCONP_PCSSP0        : SSP 0
*   - CLKPWR_PCONP_PCTIM2        : Timer 2
*   - CLKPWR_PCONP_PCTIM3        : Timer 3
*   - CLKPWR_PCONP_PCUART2       : UART 2
*   - CLKPWR_PCONP_PCUART3       : UART 3
*   - CLKPWR_PCONP_PCI2C2        : I2C 2
*   - CLKPWR_PCONP_PCI2S          : I2S
*   - CLKPWR_PCONP_PCGPDMA       : GPDMA
*   - CLKPWR_PCONP_PCENET         : Ethernet
*   - CLKPWR_PCONP_PCUSB          : USB

*
* @param[in] NewState      New state of Peripheral Power, should be:
*
*   - ENABLE      : Enable power for this peripheral
*   - DISABLE     : Disable power for this peripheral
*
* @return none</pre>
```

```
*****  
void CLKPWR_ConfigPPWR (uint32_t PPType, FunctionalState NewState)  
{  
    if (NewState == ENABLE)  
    {  
        LPC_SC->PCONP |= PPType & CLKPWR_PCONP_BITMASK;  
    }  
    else if (NewState == DISABLE)  
    {  
        LPC_SC->PCONP &= (~PPType) & CLKPWR_PCONP_BITMASK;  
    }  
}  
  
*****  
/* @brief           Enter Sleep mode with co-operated instruction by the Cortex-M3.  
 * @param[in]      None  
 * @return         None  
 */  
void CLKPWR_Sleep(void)  
{  
    LPC_SC->PCON = 0x00;  
    /* Sleep Mode*/  
    __WFI();  
}  
  
*****  
/* @brief           Enter Deep Sleep mode with co-operated instruction by the Cortex-M3.  
 * @param[in]      None  
 * @return         None  
 */  
void CLKPWR_DeepSleep(void)  
{  
    /* Deep-Sleep Mode, set SLEEPDEEP bit */  
    SCB->SCR = 0x4;  
    LPC_SC->PCON = 0x8;  
    /* Deep Sleep Mode*/  
    __WFI();
```

}

```
/**************************************************************************/<!--

 * @brief          Enter Power Down mode with co-operated instruction by the Cortex-M3.

 * @param[in]      None

 * @return         None

/**************************************************************************/void CLKPWR_PowerDown(void){/* Deep-Sleep Mode, set SLEEPDEEP bit */    SCB-&gt;SCR = 0x4;    LPC_SC-&gt;PCON = 0x09;    /* Power Down Mode*/    __WFI();}

/**************************************************************************/<!--

 * @brief          Enter Deep Power Down mode with co-operated instruction by the Cortex-M3.

 * @param[in]      None

 * @return         None

/**************************************************************************/void CLKPWR_DeepPowerDown(void){/* Deep-Sleep Mode, set SLEEPDEEP bit */    SCB-&gt;SCR = 0x4;    LPC_SC-&gt;PCON = 0x03;    /* Deep Power Down Mode*/    __WFI();}

/***
 * @}
 */

/***
 * @}
 */

</pre>
```

*/

/* ----- End Of File ----- */

* \$Id\$ lpc17xx_libcfg_default.c 2010-05-21

**/ **

* @file lpc17xx_libcfg_default.c

* @brief Library configuration source file (default), used to build
* library without examples

* @version 2.0

* @date 21. May. 2010

* @author NXP MCU SW Application Team

*

* Copyright (C) 2010, NXP Semiconductor

* All rights reserved.

*

* Software that is described herein is for illustrative purposes only

* which provides customers with programming information regarding the

* products. This software is supplied "AS IS" without any warranties.

* NXP Semiconductors assumes no responsibility or liability for the

* use of the software, conveys no license or title under any patent,

* copyright, or mask work right to the product. NXP Semiconductors

* reserves the right to make changes in the software without

* notification. NXP Semiconductors also make no representation or

* warranty that such application will be suitable for the specified

* use without further testing or modification.

/* Library group ----- */

/** @addtogroup LIBCFG_DEFAULT

* @{

*/

/* Includes ----- */

#include "lpc17xx_libcfg_default.h"

```
/* Public Functions ----- */

/** @defgroup LIBCFG_DEFAULT_Public_Functions
 * @{
 *
 */

#ifndef __BUILD_WITH_EXAMPLE__


#define DEBUG
/* ****
 * @brief Reports the name of the source file and the source line number
 * where the CHECK_PARAM error has occurred.
 * @param[in] file Pointer to the source file name
 * @param[in] line assert_param error line source number
 * @return None
 */
void check_failed(uint8_t *file, uint32_t line)
{
    /* User can add his own implementation to report the file name and line number,
     * ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */

    /* Infinite loop */
    while(1);
}

#endif /* DEBUG */


#endif /* __BUILD_WITH_EXAMPLE__ */

/** @}
 */
/* -----
 * ----- End Of File ----- */

```

```
*****Copyright
(c) *****

**
**                               http://www.powermcu.com
**

*-----File Info-----
-----

** File name:          AsciiLib.c
** Descriptions:        ASCII×Ö;â °áíòÈ;Ä£  8*16 ×"Íª²ÊEAÍÔÊ¾Éè¾Æ
**

*-----


** Created by:         AVRman
** Created date:       2010-11-2
** Version:            1.0
** Descriptions:       The original version
**

*-----


** Modified by:
** Modified date:
** Version:
** Descriptions:

*****
****/


/* Includes ----- */
#include "AsciiLib.h"


#endif ASCII_8X16_MS_Gothic

static unsigned char const AsciiLib[95][16] = {

{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00},/*"
",0*/


{0x00,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x00,0x00,0x18,0x18,0x00,0x00},/*"!",
1*/


{0x36,0x24,0x48,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00},/*"",
2*/
```

```
{0x00,0x24,0x24,0x24,0xFE,0x48,0x48,0x48,0xFC,0x48,0x48,0x48,0x48,0x00},/*="#"  
3*/  
  
{0x10,0x38,0x54,0x92,0x92,0x50,0x30,0x18,0x14,0x12,0x92,0x92,0x54,0x38,0x10,0x00},/*"$"  
4*/  
  
{0x00,0x62,0x92,0x94,0x94,0x68,0x08,0x10,0x20,0x2C,0x52,0x52,0x92,0x8C,0x00,0x00},/*%""  
5*/  
  
{0x00,0x30,0x48,0x48,0x48,0x30,0x20,0x54,0x94,0x88,0x88,0x94,0x62,0x00,0x00},/*"&"  
6*/  
  
{0x30,0x30,0x10,0x20,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00},/*"!"  
7*/  
  
{0x04,0x08,0x10,0x10,0x20,0x20,0x20,0x20,0x20,0x20,0x10,0x10,0x08,0x04,0x00},/*"(" "  
8*/  
  
{0x40,0x20,0x10,0x10,0x08,0x08,0x08,0x08,0x08,0x08,0x10,0x10,0x20,0x40,0x00},/*")"  
9*/  
  
{0x00,0x00,0x00,0x10,0x92,0x54,0x38,0x10,0x38,0x54,0x92,0x10,0x00,0x00,0x00,0x00},/*"**"  
10*/  
  
{0x00,0x00,0x00,0x00,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x00,0x00,0x00,0x00,0x00},/*"+"  
11*/  
  
{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00},/*"-"  
12*/  
  
{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00},/*".."  
14*/  
  
{0x00,0x02,0x02,0x04,0x04,0x08,0x08,0x10,0x20,0x20,0x40,0x40,0x80,0x80,0x00,0x00},/*"/"  
15*/  
  
{0x00,0x30,0x48,0x84,0x84,0x84,0x84,0x84,0x84,0x84,0x84,0x84,0x48,0x30,0x00,0x00},/*"0"  
16*/  
  
{0x00,0x10,0x70,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x00,0x00},/*"1"  
17*/  
  
{0x00,0x30,0x48,0x84,0x84,0x04,0x08,0x08,0x10,0x20,0x20,0x40,0x80,0xFC,0x00,0x00},/*"2"  
18*/
```

{0x00,0x30,0x48,0x84,0x84,0x04,0x08,0x30,0x08,0x04,0x84,0x84,0x48,0x30,0x00,0x00}, /*"3",
19*/

{0x00,0x08,0x08,0x18,0x18,0x28,0x28,0x48,0x48,0x88,0xFC,0x08,0x08,0x08,0x00,0x00}, /*"4",
20*/

{0x00,0xFC,0x80,0x80,0xB0,0xC8,0x84,0x04,0x04,0x84,0x48,0x30,0x00,0x00}, /*"5",
21*/

{0x00,0x30,0x48,0x84,0x84,0x80,0xB0,0xC8,0x84,0x84,0x84,0x84,0x48,0x30,0x00,0x00}, /*"6",
22*/

{0x00,0xFC,0x04,0x04,0x08,0x08,0x10,0x10,0x10,0x20,0x20,0x20,0x20,0x00,0x00}, /*"7",
23*/

{0x00,0x30,0x48,0x84,0x84,0x84,0x48,0x30,0x48,0x84,0x84,0x84,0x48,0x30,0x00,0x00}, /*"8",
24*/

{0x00,0x30,0x48,0x84,0x84,0x84,0x4C,0x34,0x04,0x84,0x84,0x48,0x30,0x00,0x00}, /*"9",
25*/

{0x00,0x00,0x00,0x00,0x00,0x30,0x30,0x00,0x00,0x00,0x00,0x30,0x30,0x00,0x00,0x00}, /*":",
26*/

{0x00,0x00,0x00,0x00,0x00,0x30,0x30,0x00,0x00,0x00,0x00,0x30,0x30,0x10,0x20,0x00}, /*";",
27*/

{0x00,0x00,0x04,0x08,0x10,0x20,0x40,0x80,0x40,0x20,0x10,0x08,0x04,0x00,0x00,0x00}, /*"<",
28*/

{0x00,0x00,0x00,0x00,0x00,0x7C,0x00,0x00,0x00,0x7C,0x00,0x00,0x00,0x00,0x00,0x00}, /*"=",
29*/

{0x00,0x00,0x80,0x40,0x20,0x10,0x08,0x04,0x08,0x10,0x20,0x40,0x80,0x00,0x00,0x00}, /*">",
30*/

{0x00,0x30,0x48,0x84,0x84,0x04,0x08,0x10,0x20,0x00,0x00,0x30,0x30,0x00,0x00}, /*"?",
31*/

{0x00,0x38,0x44,0x82,0x9A,0xAA,0xAA,0xAA,0x9C,0x80,0x42,0x3C,0x00,0x00}, /*"@",
32*/

{0x00,0x10,0x10,0x28,0x28,0x28,0x44,0x44,0x44,0x7C,0x82,0x82,0x82,0x00,0x00}, /*"A",
33*/

{0x00,0xF8,0x84,0x82,0x82,0x82,0x84,0xF8,0x84,0x82,0x82,0x82,0x84,0xF8,0x00,0x00}, /*"B",
34*/

{0x00,0x38,0x44,0x82,0x82,0x80,0x80,0x80,0x80,0x82,0x82,0x44,0x38,0x00,0x00}, /*"C",
35*/

{0x00,0xF8,0x84,0x82,0x82,0x82,0x82,0x82,0x82,0x82,0x84,0xF8,0x00,0x00}, /*"D",
36*/

{0x00,0xFE,0x80,0x80,0x80,0x80,0x80,0xFC,0x80,0x80,0x80,0x80,0x80,0x80,0x80,0x00,0x00}, /*"E",
37*/

{0x00,0xFE,0x80,0x80,0x80,0x80,0x80,0x80,0x80,0x80,0x80,0x80,0x80,0x80,0x80,0x00,0x00}, /*"F",
38*/

{0x00,0x38,0x44,0x82,0x82,0x80,0x80,0x80,0x8E,0x82,0x82,0x82,0x46,0x3A,0x00,0x00}, /*"G",
39*/

{0x00,0x82,0x82,0x82,0x82,0x82,0x82,0x82,0x82,0x82,0x82,0x82,0x82,0x82,0x00,0x00}, /*"H",
40*/

{0x00,0x38,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x38,0x00,0x00}, /*"I",
41*/

{0x00,0x04,0x04,0x04,0x04,0x04,0x04,0x04,0x04,0x84,0x84,0x48,0x30,0x00,0x00}, /*"J",
42*/

{0x00,0x82,0x84,0x84,0x88,0x90,0x90,0xA0,0xD0,0x88,0x88,0x84,0x82,0x82,0x00,0x00}, /*"K",
43*/

{0x00,0x80,0x80,0x80,0x80,0x80,0x80,0x80,0x80,0x80,0x80,0x80,0x80,0x80,0x00,0x00}, /*"L",
44*/

{0x00,0x82,0x82,0xC6,0xC6,0xC6,0xAA,0xAA,0xAA,0x92,0x92,0x92,0x00,0x00}, /*"M",
45*/

{0x00,0x82,0x82,0xC2,0xC2,0xA2,0xA2,0x92,0x92,0x8A,0x8A,0x86,0x86,0x82,0x00,0x00}, /*"N",
46*/

{0x00,0x38,0x44,0x82,0x82,0x82,0x82,0x82,0x82,0x82,0x82,0x44,0x38,0x00,0x00}, /*"O",
47*/

{0x00,0xF8,0x84,0x82,0x82,0x82,0x84,0xF8,0x80,0x80,0x80,0x80,0x80,0x80,0x00,0x00}, /*"P",
48*/

{0x00,0x38,0x44,0x82,0x82,0x82,0x82,0x82,0x82,0x82,0x92,0x8A,0x44,0x3A,0x00,0x00}, /*"Q",
49*/

```
{0x00,0xF8,0x84,0x82,0x82,0x82,0x84,0xF8,0x88,0x88,0x84,0x84,0x82,0x82,0x00,0x00},/*"R",  
50*/
```

```
{0x00,0x38,0x44,0x82,0x82,0x80,0x60,0x18,0x04,0x02,0x82,0x82,0x44,0x38,0x00,0x00},/*"S",  
51*/
```

```
{0x00,0xFE,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x00,0x00},/*"T",  
52*/
```

```
{0x00,0x82,0x82,0x82,0x82,0x82,0x82,0x82,0x82,0x82,0x82,0x82,0x44,0x38,0x00,0x00},/*"U",  
53*/
```

```
{0x00,0x82,0x82,0x82,0x44,0x44,0x44,0x28,0x28,0x28,0x10,0x10,0x10,0x10,0x00,0x00},/*"V",  
54*/
```

```
{0x00,0x92,0x92,0x92,0x92,0xAA,0xAA,0xAA,0x44,0x44,0x44,0x44,0x44,0x44,0x00,0x00},/*"W",  
55*/
```

```
{0x00,0x82,0x82,0x44,0x44,0x28,0x28,0x10,0x28,0x28,0x44,0x44,0x82,0x82,0x00,0x00},/*"X",  
56*/
```

```
{0x00,0x82,0x82,0x44,0x44,0x28,0x28,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x00,0x00},/*"Y",  
57*/
```

```
{0x00,0xFE,0x02,0x04,0x04,0x08,0x08,0x10,0x20,0x20,0x40,0x40,0x80,0xFE,0x00,0x00},/*"Z",  
58*/
```

```
{0x7C,0x40,0x40,0x40,0x40,0x40,0x40,0x40,0x40,0x40,0x40,0x40,0x40,0x40,0x40,0x7C,0x00},/*"[",  
59*/
```

```
{0x00,0x82,0x82,0x44,0x44,0x28,0x28,0x7C,0x10,0x10,0x7C,0x10,0x10,0x10,0x00,0x00},/*"\\",  
60*/
```

```
{0x7C,0x04,0x04,0x04,0x04,0x04,0x04,0x04,0x04,0x04,0x04,0x04,0x04,0x04,0x04,0x7C,0x00},/*"]",  
61*/
```

```
{0x10,0x28,0x44,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00},/*"^",  
62*/
```

```
{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0xFF},/*"_",  
63*/
```

```
{0x30,0x30,0x10,0x20,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00},/*"!!",  
64*/
```

```
{0x00,0x00,0x00,0x00,0x00,0x78,0x84,0x04,0x3C,0x44,0x84,0x8C,0x76,0x00,0x00},/*"a",
65*/\n\n{0x00,0x80,0x80,0x80,0x80,0xB8,0xC4,0x82,0x82,0x82,0xC4,0xB8,0x00,0x00},/*"b",
66*/\n\n{0x00,0x00,0x00,0x00,0x00,0x3C,0x42,0x80,0x80,0x80,0x80,0x42,0x3C,0x00,0x00},/*"c",
67*/\n\n{0x00,0x02,0x02,0x02,0x02,0x3A,0x46,0x82,0x82,0x82,0x46,0x3A,0x00,0x00},/*"d",
68*/\n\n{0x00,0x00,0x00,0x00,0x00,0x38,0x44,0x82,0xFE,0x80,0x80,0x42,0x3C,0x00,0x00},/*"e",
69*/\n\n{0x00,0x18,0x20,0x20,0x20,0xF8,0x20,0x20,0x20,0x20,0x20,0x20,0x20,0x00,0x00},/*"f",
70*/\n\n{0x00,0x00,0x00,0x00,0x00,0x3A,0x44,0x44,0x38,0x40,0x7C,0x82,0x82,0x7C,0x00},/*"g",
71*/\n\n{0x00,0x80,0x80,0x80,0x80,0xB8,0xC4,0x82,0x82,0x82,0x82,0x82,0x82,0x00,0x00},/*"h",
72*/\n\n{0x00,0x00,0x10,0x10,0x00,0x00,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x00,0x00},/*"i",
73*/\n\n{0x00,0x00,0x10,0x10,0x00,0x00,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x60,0x00},/*"j",
74*/\n\n{0x00,0x80,0x80,0x80,0x80,0x80,0x84,0x88,0x90,0xA0,0xD0,0x88,0x84,0x82,0x00,0x00},/*"k",
75*/\n\n{0x00,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x00,0x00},/*"l",
76*/\n\n{0x00,0x00,0x00,0x00,0x00,0xAC,0xD2,0x92,0x92,0x92,0x92,0x92,0x92,0x00,0x00},/*"m",
77*/\n\n{0x00,0x00,0x00,0x00,0x00,0xB8,0xC4,0x82,0x82,0x82,0x82,0x82,0x82,0x00,0x00},/*"n",
78*/\n\n{0x00,0x00,0x00,0x00,0x00,0x38,0x44,0x82,0x82,0x82,0x82,0x44,0x38,0x00,0x00},/*"o",
79*/\n\n{0x00,0x00,0x00,0x00,0x00,0xB8,0xC4,0x82,0x82,0x82,0xC4,0xB8,0x80,0x80,0x00},/*"p",
80*/
```

```
{0x00,0x00,0x00,0x00,0x00,0x00,0x3A,0x46,0x82,0x82,0x82,0x46,0x3A,0x02,0x02,0x00},/*"q",
81*/
{0x00,0x00,0x00,0x00,0x00,0x00,0x2E,0x30,0x20,0x20,0x20,0x20,0x20,0x20,0x00,0x00},/*"r",
82*/
{0x00,0x00,0x00,0x00,0x00,0x00,0x7C,0x82,0x80,0x60,0x1C,0x02,0x82,0x7C,0x00,0x00},/*"s",
83*/
{0x00,0x00,0x20,0x20,0x20,0xF8,0x20,0x20,0x20,0x20,0x20,0x18,0x00,0x00},/*"t",
84*/
{0x00,0x00,0x00,0x00,0x00,0x00,0x82,0x82,0x82,0x82,0x82,0x46,0x3A,0x00,0x00},/*"u",
85*/
{0x00,0x00,0x00,0x00,0x00,0x00,0x82,0x82,0x44,0x44,0x28,0x28,0x10,0x10,0x00,0x00},/*"v",
86*/
{0x00,0x00,0x00,0x00,0x00,0x00,0x92,0x92,0x92,0xAA,0xAA,0x44,0x44,0x44,0x00,0x00},/*"w",
87*/
{0x00,0x00,0x00,0x00,0x00,0x00,0x82,0x44,0x28,0x10,0x10,0x28,0x44,0x82,0x00,0x00},/*"x",
88*/
{0x00,0x00,0x00,0x00,0x00,0x00,0x82,0x82,0x44,0x44,0x28,0x10,0x10,0x20,0xC0,0x00},/*"y",
89*/
{0x00,0x00,0x00,0x00,0x00,0x00,0xFE,0x04,0x08,0x10,0x20,0x40,0x80,0xFE,0x00,0x00},/*"z",
90*/
{0x1C,0x10,0x10,0x10,0x10,0x10,0x10,0x20,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x1C,0x00},/*"(",
91*/
{0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x10},/*")",
92*/
{0x70,0x10,0x10,0x10,0x10,0x10,0x08,0x10,0x10,0x10,0x10,0x10,0x10,0x10,0x70,0x00},/*"}",
93*/
};

#endif
```

```
#ifdef ASCII_8X16_System

static unsigned char const AsciiLib[95][16] = {

{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00},/*",
",0*/


{0x00,0x00,0x00,0x18,0x3C,0x3C,0x18,0x18,0x00,0x18,0x18,0x00,0x00,0x00,0x00,0x00},/*!"|,
1*/


{0x00,0x00,0x00,0x66,0x66,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00},/*""|,
2*/


{0x00,0x00,0x00,0x36,0x36,0x7F,0x36,0x36,0x7F,0x36,0x36,0x00,0x00,0x00,0x00,0x00},/*#"|,
3*/


{0x00,0x18,0x18,0x3C,0x66,0x60,0x30,0x18,0x0C,0x06,0x66,0x3C,0x18,0x18,0x00,0x00},/*"$|,
4*/


{0x00,0x00,0x70,0xD8,0xDA,0x76,0x0C,0x18,0x30,0x6E,0x5B,0x1B,0x0E,0x00,0x00,0x00},/*%"|,
5*/


{0x00,0x00,0x00,0x38,0x6C,0x6C,0x38,0x60,0x6F,0x66,0x66,0x3B,0x00,0x00,0x00,0x00},/*%"|,
6*/


{0x00,0x00,0x00,0x18,0x18,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00},/*!"|,
7*/


{0x00,0x00,0x00,0x0C,0x18,0x18,0x30,0x30,0x30,0x30,0x18,0x18,0x0C,0x00,0x00},/*"("|,
8*/


{0x00,0x00,0x00,0x30,0x18,0x18,0x0C,0x0C,0x0C,0x0C,0x18,0x18,0x30,0x00,0x00},/*")"|,
9*/


{0x00,0x00,0x00,0x00,0x36,0x1C,0x7F,0x1C,0x36,0x00,0x00,0x00,0x00,0x00},/*""|,
10*/


{0x00,0x00,0x00,0x00,0x18,0x18,0x7E,0x18,0x18,0x00,0x00,0x00,0x00,0x00,0x00},/*"+|,
11*/


{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x1C,0x1C,0x0C,0x18,0x00,0x00},/*",|,
12*/


{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00},/*"-|,
13*/
```

```
{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x1C,0x1C,0x00,0x00,0x00,0x00},/*".",  
14*/
```

```
{0x00,0x00,0x00,0x06,0x06,0x0C,0x0C,0x18,0x18,0x30,0x30,0x60,0x60,0x00,0x00,0x00},/*"/",  
15*/
```

```
{0x00,0x00,0x00,0x1E,0x33,0x37,0x37,0x33,0x3B,0x3B,0x33,0x1E,0x00,0x00,0x00,0x00},/*"0",  
16*/
```

```
{0x00,0x00,0x00,0x0C,0x1C,0x7C,0x0C,0x0C,0x0C,0x0C,0x00,0x00,0x00,0x00},/*"1",  
17*/
```

```
{0x00,0x00,0x00,0x3C,0x66,0x66,0x06,0x0C,0x18,0x30,0x60,0x7E,0x00,0x00,0x00,0x00},/*"2",  
18*/
```

```
{0x00,0x00,0x00,0x3C,0x66,0x66,0x06,0x1C,0x06,0x66,0x66,0x3C,0x00,0x00,0x00,0x00},/*"3",  
19*/
```

```
{0x00,0x00,0x00,0x30,0x36,0x36,0x66,0x7F,0x06,0x06,0x00,0x00,0x00,0x00},/*"4",  
20*/
```

```
{0x00,0x00,0x00,0x7E,0x60,0x60,0x7C,0x06,0x06,0x0C,0x78,0x00,0x00,0x00,0x00},/*"5",  
21*/
```

```
{0x00,0x00,0x00,0x1C,0x18,0x30,0x7C,0x66,0x66,0x66,0x66,0x3C,0x00,0x00,0x00,0x00},/*"6",  
22*/
```

```
{0x00,0x00,0x00,0x7E,0x06,0x0C,0x0C,0x18,0x18,0x30,0x30,0x00,0x00,0x00,0x00},/*"7",  
23*/
```

```
{0x00,0x00,0x00,0x3C,0x66,0x66,0x76,0x3C,0x6E,0x66,0x66,0x3C,0x00,0x00,0x00,0x00},/*"8",  
24*/
```

```
{0x00,0x00,0x00,0x3C,0x66,0x66,0x66,0x3E,0x0C,0x18,0x38,0x00,0x00,0x00,0x00},/*"9",  
25*/
```

```
{0x00,0x00,0x00,0x00,0x00,0x1C,0x1C,0x00,0x00,0x00,0x1C,0x1C,0x00,0x00,0x00,0x00},/*":",  
26*/
```

```
{0x00,0x00,0x00,0x00,0x00,0x1C,0x1C,0x00,0x00,0x00,0x1C,0x1C,0x0C,0x18,0x00,0x00},/*";",  
27*/
```

```
{0x00,0x00,0x00,0x06,0x0C,0x18,0x30,0x60,0x30,0x18,0x0C,0x06,0x00,0x00,0x00,0x00},/*"<",  
28*/
```

```
{0x00,0x00,0x00,0x00,0x00,0x00,0x7E,0x00,0x7E,0x00,0x00,0x00,0x00,0x00,0x00,0x00},/*"=",  
29*/
```

{0x00,0x00,0x00,0x60,0x30,0x18,0x0C,0x06,0x0C,0x18,0x30,0x60,0x00,0x00,0x00,0x00},/*">",
30*/

{0x00,0x00,0x00,0x3C,0x66,0x66,0x0C,0x18,0x18,0x00,0x18,0x00,0x00,0x00,0x00},/*"?",
31*/

{0x00,0x00,0x00,0x7E,0xC3,0xC3,0xCF,0xDB,0xDB,0xCF,0xC0,0x7F,0x00,0x00,0x00,0x00},/*"@",
32*/

{0x00,0x00,0x00,0x18,0x3C,0x66,0x66,0x66,0x7E,0x66,0x66,0x66,0x00,0x00,0x00,0x00},/*"A",
33*/

{0x00,0x00,0x00,0x7C,0x66,0x66,0x66,0x7C,0x66,0x66,0x66,0x7C,0x00,0x00,0x00,0x00},/*"B",
34*/

{0x00,0x00,0x00,0x3C,0x66,0x66,0x60,0x60,0x60,0x66,0x66,0x3C,0x00,0x00,0x00,0x00},/*"C",
35*/

{0x00,0x00,0x00,0x78,0x6C,0x66,0x66,0x66,0x66,0x6C,0x78,0x00,0x00,0x00,0x00},/*"D",
36*/

{0x00,0x00,0x00,0x7E,0x60,0x60,0x60,0x7C,0x60,0x60,0x60,0x7E,0x00,0x00,0x00,0x00},/*"E",
37*/

{0x00,0x00,0x00,0x7E,0x60,0x60,0x60,0x7C,0x60,0x60,0x60,0x60,0x00,0x00,0x00,0x00},/*"F",
38*/

{0x00,0x00,0x00,0x3C,0x66,0x66,0x60,0x60,0x6E,0x66,0x66,0x3E,0x00,0x00,0x00,0x00},/*"G",
39*/

{0x00,0x00,0x00,0x66,0x66,0x66,0x7E,0x66,0x66,0x66,0x66,0x00,0x00,0x00,0x00},/*"H",
40*/

{0x00,0x00,0x00,0x3C,0x18,0x18,0x18,0x18,0x18,0x18,0x3C,0x00,0x00,0x00,0x00},/*"I",
41*/

{0x00,0x00,0x00,0x06,0x06,0x06,0x06,0x06,0x66,0x66,0x3C,0x00,0x00,0x00,0x00},/*"J",
42*/

{0x00,0x00,0x00,0x66,0x66,0x6C,0x6C,0x78,0x6C,0x6C,0x66,0x66,0x00,0x00,0x00,0x00},/*"K",
43*/

{0x00,0x00,0x00,0x60,0x60,0x60,0x60,0x60,0x60,0x60,0x7E,0x00,0x00,0x00,0x00},/*"L",
44*/

{0x00,0x00,0x00,0x63,0x63,0x77,0x6B,0x6B,0x6B,0x63,0x63,0x63,0x00,0x00,0x00,0x00}, /*"M",
45*/

{0x00,0x00,0x00,0x63,0x63,0x73,0x7B,0x6F,0x67,0x63,0x63,0x63,0x00,0x00,0x00,0x00}, /*"N",
46*/

{0x00,0x00,0x00,0x3C,0x66,0x66,0x66,0x66,0x66,0x66,0x66,0x3C,0x00,0x00,0x00,0x00}, /*"O",
47*/

{0x00,0x00,0x00,0x7C,0x66,0x66,0x66,0x7C,0x60,0x60,0x60,0x60,0x00,0x00,0x00,0x00}, /*"P",
48*/

{0x00,0x00,0x00,0x3C,0x66,0x66,0x66,0x66,0x66,0x66,0x66,0x3C,0x0C,0x06,0x00,0x00}, /*"Q",
49*/

{0x00,0x00,0x00,0x7C,0x66,0x66,0x66,0x7C,0x6C,0x66,0x66,0x66,0x00,0x00,0x00,0x00}, /*"R",
50*/

{0x00,0x00,0x00,0x3C,0x66,0x60,0x30,0x18,0x0C,0x06,0x66,0x3C,0x00,0x00,0x00,0x00}, /*"S",
51*/

{0x00,0x00,0x00,0x7E,0x18,0x18,0x18,0x18,0x18,0x18,0x18,0x00,0x00,0x00,0x00}, /*"T",
52*/

{0x00,0x00,0x00,0x66,0x66,0x66,0x66,0x66,0x66,0x66,0x3C,0x00,0x00,0x00,0x00}, /*"U",
53*/

{0x00,0x00,0x00,0x66,0x66,0x66,0x66,0x66,0x66,0x3C,0x18,0x00,0x00,0x00,0x00}, /*"V",
54*/

{0x00,0x00,0x00,0x63,0x63,0x63,0x6B,0x6B,0x6B,0x36,0x36,0x00,0x00,0x00,0x00}, /*"W",
55*/

{0x00,0x00,0x00,0x66,0x34,0x18,0x18,0x2C,0x66,0x66,0x66,0x00,0x00,0x00,0x00}, /*"X",
56*/

{0x00,0x00,0x00,0x66,0x66,0x66,0x3C,0x18,0x18,0x18,0x18,0x00,0x00,0x00,0x00}, /*"Y",
57*/

{0x00,0x00,0x00,0x7E,0x06,0x06,0x0C,0x18,0x30,0x60,0x60,0x7E,0x00,0x00,0x00,0x00}, /*"Z",
58*/

{0x00,0x00,0x00,0x3C,0x30,0x30,0x30,0x30,0x30,0x30,0x30,0x30,0x30,0x30,0x30,0x00}, /*"[",
59*/

{0x00,0x00,0x00,0x60,0x60,0x30,0x18,0x18,0x0C,0x0C,0x06,0x06,0x00,0x00,0x00,0x00}, /*"\\",
60*/

```
{0x00,0x00,0x00,0x3C,0x0C,0x0C,0x0C,0x0C,0x0C,0x0C,0x0C,0x0C,0x0C,0x0C,0x3C,0x00},/*"]",  
61*/
```

```
{0x00,0x18,0x3C,0x66,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00},/*"^",  
62*/
```

```
{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0xFF,0x00},/*"_",  
63*/
```

```
{0x00,0x00,0x00,0x18,0x18,0x18,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00},/*""",  
64*/
```

```
{0x00,0x00,0x00,0x00,0x00,0x3C,0x06,0x06,0x3E,0x66,0x66,0x3E,0x00,0x00,0x00,0x00},/*"a",  
65*/
```

```
{0x00,0x00,0x00,0x60,0x60,0x7C,0x66,0x66,0x66,0x66,0x66,0x7C,0x00,0x00,0x00,0x00},/*"b",  
66*/
```

```
{0x00,0x00,0x00,0x00,0x00,0x3C,0x66,0x60,0x60,0x60,0x66,0x3C,0x00,0x00,0x00,0x00},/*"c",  
67*/
```

```
{0x00,0x00,0x00,0x06,0x06,0x3E,0x66,0x66,0x66,0x66,0x3E,0x00,0x00,0x00,0x00},/*"d",  
68*/
```

```
{0x00,0x00,0x00,0x00,0x00,0x3C,0x66,0x66,0x7E,0x60,0x60,0x3C,0x00,0x00,0x00,0x00},/*"e",  
69*/
```

```
{0x00,0x00,0x00,0x1E,0x30,0x30,0x30,0x7E,0x30,0x30,0x30,0x30,0x00,0x00,0x00,0x00},/*"f",  
70*/
```

```
{0x00,0x00,0x00,0x00,0x00,0x3E,0x66,0x66,0x66,0x66,0x3E,0x06,0x06,0x7C,0x00},/*"g",  
71*/
```

```
{0x00,0x00,0x00,0x60,0x60,0x7C,0x66,0x66,0x66,0x66,0x66,0x66,0x00,0x00,0x00,0x00},/*"h",  
72*/
```

```
{0x00,0x00,0x18,0x18,0x00,0x78,0x18,0x18,0x18,0x18,0x7E,0x00,0x00,0x00,0x00},/*"i",  
73*/
```

```
{0x00,0x00,0x0C,0x0C,0x00,0x3C,0x0C,0x0C,0x0C,0x0C,0x0C,0x0C,0x0C,0x0C,0x78,0x00},/*"j",  
74*/
```

```
{0x00,0x00,0x00,0x60,0x60,0x66,0x66,0x6C,0x78,0x6C,0x66,0x66,0x00,0x00,0x00,0x00},/*"k",  
75*/
```

```
{0x00,0x00,0x00,0x78,0x18,0x18,0x18,0x18,0x18,0x18,0x18,0x7E,0x00,0x00,0x00,0x00},/*"l",  
76*/  
  
{0x00,0x00,0x00,0x00,0x00,0x7E,0x6B,0x6B,0x6B,0x6B,0x6B,0x63,0x00,0x00,0x00,0x00},/*"m",  
77*/  
  
{0x00,0x00,0x00,0x00,0x00,0x7C,0x66,0x66,0x66,0x66,0x66,0x66,0x00,0x00,0x00,0x00},/*"n",  
78*/  
  
{0x00,0x00,0x00,0x00,0x00,0x3C,0x66,0x66,0x66,0x66,0x66,0x3C,0x00,0x00,0x00,0x00},/*"o",  
79*/  
  
{0x00,0x00,0x00,0x00,0x00,0x7C,0x66,0x66,0x66,0x66,0x66,0x7C,0x60,0x60,0x60,0x00},/*"p",  
80*/  
  
{0x00,0x00,0x00,0x00,0x00,0x3E,0x66,0x66,0x66,0x66,0x3E,0x06,0x06,0x06,0x00},/*"q",  
81*/  
  
{0x00,0x00,0x00,0x00,0x00,0x66,0x6E,0x70,0x60,0x60,0x60,0x00,0x00,0x00,0x00},/*"r",  
82*/  
  
{0x00,0x00,0x00,0x00,0x00,0x3E,0x60,0x60,0x06,0x06,0x7C,0x00,0x00,0x00,0x00},/*"s",  
83*/  
  
{0x00,0x00,0x00,0x30,0x30,0x7E,0x30,0x30,0x30,0x30,0x30,0x1E,0x00,0x00,0x00,0x00},/*"t",  
84*/  
  
{0x00,0x00,0x00,0x00,0x66,0x66,0x66,0x66,0x66,0x66,0x3E,0x00,0x00,0x00,0x00},/*"u",  
85*/  
  
{0x00,0x00,0x00,0x00,0x00,0x66,0x66,0x66,0x66,0x66,0x3C,0x18,0x00,0x00,0x00,0x00},/*"v",  
86*/  
  
{0x00,0x00,0x00,0x00,0x00,0x63,0x6B,0x6B,0x6B,0x36,0x36,0x00,0x00,0x00,0x00},/*"w",  
87*/  
  
{0x00,0x00,0x00,0x00,0x00,0x66,0x66,0x3C,0x18,0x3C,0x66,0x66,0x00,0x00,0x00,0x00},/*"x",  
88*/  
  
{0x00,0x00,0x00,0x00,0x00,0x66,0x66,0x66,0x66,0x66,0x3C,0x0C,0x18,0xF0,0x00},/*"y",  
89*/  
  
{0x00,0x00,0x00,0x00,0x00,0x7E,0x06,0x0C,0x18,0x30,0x60,0x7E,0x00,0x00,0x00,0x00},/*"z",  
90*/  
  
{0x00,0x00,0x00,0x0C,0x18,0x18,0x18,0x30,0x60,0x30,0x18,0x18,0x18,0x0C,0x00,0x00},/*{"  
91*/
```

```
{0x00,0x00,0x00,0x18,0x18,0x18,0x18,0x18,0x18,0x18,0x18,0x18,0x18,0x18,0x18,0x18,0x00},/*"|",
92*/
```

```
{0x00,0x00,0x00,0x30,0x18,0x18,0x0C,0x06,0x0C,0x18,0x18,0x18,0x30,0x00,0x00},/*"}|,
93*/
```

```
{0x00,0x00,0x00,0x71,0xDB,0x8E,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00},/*"~",
94*/
};

#endif
```

```
*****
* Function Name : GetASCIICode
* Description   : È;ASCIIÃœ Èý¾Ý
* Input         : - ASCII: ÈæÈéuÃSCIIÃœè¬Èç'A',BaseAddr »ùÖ·
  ¸ÃSCIIïÖ¾' úÃéÖÙFLASHÖþµÃ‘¢Í»ÖÄ
* Output        : - *pBuffer: 'æ·Ãßý¾ÝµÃÖ,Öé
* Return        : None
* Attention     : None
*****
```

```
void GetASCIICode(unsigned char* pBuffer,unsigned char ASCII)
{
    memcpy(pBuffer,AsciiLib[(ASCII - 32)] ,16);
}
```

```
*****
END FILE
*****
```

```
*****
*****Copyright
(c) ****
**
```

```
**          http://www.powermcu.com
```

Alberto Palomo Alonso.

```
**  
-----File Info-----  
-----  
** File name: GLCD.h  
** Descriptions: TFT (IO)  
**  
**-----  
-----  
** Created by: AVRman  
** Created date: 2015-1-26  
** Version: 2.0  
** Descriptions: The original version  
**  
**-----  
-----  
** Modified by:  
** Modified date:  
** Version:  
** Descriptions:  
******/  
  
/* Includes ----- */  
#include <LPC17xx.H>  
#include "GLCD.h"  
#include "AsciiLib.h"  
  
/* La orientación se modifica en GLCD.h */  
  
/* Private variables ----- */  
static uint8_t LCD_Code;  
  
/* Private define ----- */  
#define ILI9320 0 /* 0x9320 */  
#define ILI9325 1 /* 0x9325 */  
#define ILI9325A 2 /* 0x9325A */  
#define ILI9325C 3 /* 0x9325C */  
#define ILI9325D 4 /* 0x9325D */  
#define ST7781 5 /* 0x7783 */  
#define LGDP4531 6 /* 0x4531 */
```

Alberto Palomo Alonso.

```

* Return      : None
* Attention   : None
*****static __attribute__((always_inline)) void LCD_Send (uint16_t byte)
{
    uint32_t temp;
    temp = byte;

    LPC_GPIO2->FIOPIN =  (LPC_GPIO2->FIOPIN & (~(0x000000FF))) | (temp & 0x00FF);
    /* Write D0..D7 */

    LPC_GPIO0->FIOPIN =  (LPC_GPIO0->FIOPIN & (~(0x007F8000))) | ((temp << 7) &
0x007F8000);      /* Write D8..D15 */
}

*****/* Function Name  : LCD_Read
* Description   : LCD读取
* Input        : - byte: byte to be read
* Output       : None
* Return       : 读取的字节
* Attention    : None
*****static __attribute__((always_inline)) uint16_t LCD_Read (void)
{
    uint32_t low,high;

    LPC_GPIO2->FIODIR &= ~(0x000000FF);           /* P2.0...P2.7  Input DB[0..7]
*/
    LPC_GPIO0->FIODIR &= ~(0x007F8000);           /* P0.15...P0.22 Input DB[8..15]*/
    low  = LPC_GPIO2->FIOPIN & 0x000000ff;          /* Read D0..D7 */
    high = LPC_GPIO0->FIOPIN & 0x007f8000;          /* Read D8..D15 */
    low |= (high >> 7);

    LPC_GPIO2->FIODIR |= 0x000000FF;             /* P2.0...P2.7  Output DB[0..7]
*/
    LPC_GPIO0->FIODIR |= 0x007F8000;
    /* P0.15...P0.22 Output DB[8..15]*/

    return  low;
}

```

```
*****
* Function Name  : LCD_WriteIndex
* Description    : LCDÐ'¾À`æÈ÷µØÖ·
* Input          : - index: ¾À`æÈ÷µØÖ·
* Output         : None
* Return         : None
* Attention      : None
*****
static __attribute__((always_inline)) void LCD_WriteIndex(uint16_t index)
{
    LCD_CS(0);
    LCD_RS(0);
    LCD_RD(1);
    LCD_Send(index);
    LCD_WR(0);
    __nop(); /* delay */
    LCD_WR(1);
    LCD_CS(1);
}

*****
* Function Name  : LCD_WriteData
* Description    : LCDÐ'¾À`æÈ÷ÊÝ¾Ý
* Input          : - index: ¾À`æÈ÷ÊÝ¾Ý
* Output         : None
* Return         : None
* Attention      : None
*****
static __attribute__((always_inline)) void LCD_WriteData(uint16_t data)
{
    LCD_CS(0);
    LCD_RS(1);
    LCD_Send(data);
    LCD_WR(0);
    __nop(); /* delay */
    LCD_WR(1);
    LCD_CS(1);
}
```

```
*****
* Function Name  : LCD_ReadData
* Description    : ȐÅÈ; ÆØEE÷ÊÝ³Ý
* Input          : None
* Output         : None
* Return         :  »ØȐÅÈ; µ½µÅÊÝ³Ý
* Attention      : None
*****
static __attribute__((always_inline)) uint16_t LCD_ReadData(void)
{
    uint16_t value;

    LCD_CS(0);
    LCD_RS(1);
    LCD_WR(1);
    LCD_RD(0);
    value = LCD_Read();

    LCD_RD(1);
    LCD_CS(1);

    return value;
}

*****
* Function Name  : LCD_WriteReg
* Description    : Writes to the selected LCD register.
* Input          : - LCD_Reg: address of the selected register.
*                  - LCD_RegValue: value to write to the selected register.
* Output         : None
* Return         : None
* Attention      : None
*****
static __attribute__((always_inline)) void LCD_WriteReg(uint16_t LCD_Reg,uint16_t
LCD_RegValue)
{
    /* Write 16-bit Index, then Write Reg */
    LCD_WriteIndex(LCD_Reg);
    /* Write 16-bit Reg */
    LCD_WriteData(LCD_RegValue);
```

```
}

/*****  
* Function Name : LCD_WriteReg  
* Description    : Reads the selected LCD Register.  
* Input          : None  
* Output         : None  
* Return         : LCD Register Value.  
* Attention      : None  
*****/  
  
static __attribute__((always_inline)) uint16_t LCD_ReadReg(uint16_t LCD_Reg)  
{  
    uint16_t LCD_RAM;  
  
    /* Write 16-bit Index (then Read Reg) */  
    LCD_WriteIndex(LCD_Reg);  
    /* Read 16-bit Reg */  
    LCD_RAM = LCD_ReadData();  
    return LCD_RAM;  
}  
  
/*****  
* Function Name : LCD_SetCursor  
* Description    : Sets the cursor position.  
* Input          : - Xpos: specifies the X position.  
*                  - Ypos: specifies the Y position.  
* Output         : None  
* Return         : None  
* Attention      : None  
*****/  
  
static __attribute__((always_inline)) void LCD_SetCursor(uint16_t Xpos,uint16_t Ypos)  
{  
    #if ( DISP_ORIENTATION == 90 ) || ( DISP_ORIENTATION == 270 )  
  
        uint16_t temp = Xpos;  
  
        Xpos = Ypos;  
        Ypos = ( MAX_X - 1 ) - temp;  
    #endif  
}
```

```
#elif  ( DISP_ORIENTATION == 0 ) || ( DISP_ORIENTATION == 180 )

#endif

switch( LCD_Code )
{
    default:          /* 0x9320 0x9325 0x5408 0x1505 0x0505 0x7783 0x4531 0x4535 */
        LCD_WriteReg(0x0020, Xpos );
        LCD_WriteReg(0x0021, Ypos );
        break;

    case HX8347A:   /* 0x0047 */
        LCD_WriteReg(0x02, Xpos>>8 );
        LCD_WriteReg(0x03, Xpos );

        LCD_WriteReg(0x06, Ypos>>8 );
        LCD_WriteReg(0x07, Ypos );

        break;
}

}

/*****
 * Function Name : delay_ms
 * Description   : Delay Time
 * Input         : - nCount: Delay Time
 * Output        : None
 * Return        : None
 * Return        : None
 * Attention     : None
 ****/
static void delay_ms(uint16_t ms)
{
    uint16_t i,j;
    for( i = 0; i < ms; i++ )
    {
        for( j = 0; j < 1141; j++ );
    }
}
```

```
*****
* Function Name  : LCD_Initialization
* Description    : Initialize TFT Controller.
* Input          : None
* Output         : None
* Return         : None
* Attention      : None
*****
void LCD_Initialization(void)
{
    uint16_t DeviceCode;

    LCD_Configuration();
    delay_ms(100);

    DeviceCode = LCD_ReadReg(0x0000);           /* read LCD ID */
    /* recognition different screenf- use different Initialization*/
    if( DeviceCode == 0x9325 || DeviceCode == 0x9328 )
    {
        LCD_Code = ILI9325;

        LCD_WriteReg(0xB1, 0x00A5);           /* only
                                                 ILI9325C have oxB1 Register */
        DeviceCode = LCD_ReadReg(0x00B1);       /* only ILI9325C have oxB1
                                                 Register */
        if ( DeviceCode == 0x00A5)
        {
            LCD_Code = ILI9325C;
            LCD_WriteReg(0xB1, 0x0000);
        }
        else
        {
            LCD_Code = ILI9325A;
        }
        LCD_WriteReg(0xE5, 0x78F0); /* set SRAM internal timing */
        LCD_WriteReg(0x01, 0x0100); /* set Driver Output Control */
        LCD_WriteReg(0x02, 0x0700); /* set 1 line inversion */
        LCD_WriteReg(0x03, 0x1030); /* set GRAM write direction and BGR=1 */
        LCD_WriteReg(0x04, 0x0000); /* Resize register */
        LCD_WriteReg(0x08, 0x0207); /* set the back porch and front porch */
    }
}
```

```
LCD_WriteReg(0x09, 0x0000); /* set non-display area refresh cycle
ISC[3:0] */

LCD_WriteReg(0x0A, 0x0000); /* FMARK function */

LCD_WriteReg(0x0C, 0x0000); /* RGB interface setting */

LCD_WriteReg(0x0D, 0x0000); /* Frame marker Position */

LCD_WriteReg(0x0F, 0x0000); /* RGB interface polarity */

/******************Power On sequence ******************/

LCD_WriteReg(0x10, 0x0000); /* SAP, BT[3:0], AP, DSTB, SLP, STB */

LCD_WriteReg(0x11, 0x0007); /* DC1[2:0], DC0[2:0], VC[2:0] */

LCD_WriteReg(0x12, 0x0000); /* VREG1OUT voltage */

LCD_WriteReg(0x13, 0x0000); /* VDV[4:0] for VCOM amplitude */

LCD_WriteReg(0x07, 0x0001);

delay_ms(200);

/* Dis-charge capacitor power voltage */

LCD_WriteReg(0x10, 0x1090); /* SAP, BT[3:0], AP, DSTB, SLP, STB */

LCD_WriteReg(0x11, 0x0227); /* Set DC1[2:0], DC0[2:0], VC[2:0] */

delay_ms(50); /* Delay 50ms

*/
LCD_WriteReg(0x12, 0x001F);

delay_ms(50); /* Delay 50ms

*/
LCD_WriteReg(0x13, 0x1500); /* VDV[4:0] for VCOM amplitude */

LCD_WriteReg(0x29, 0x0027); /* 04 VCM[5:0] for VCOMH */

LCD_WriteReg(0x2B, 0x000D); /* Set Frame Rate */

delay_ms(50); /* Delay 50ms

*/
LCD_WriteReg(0x20, 0x0000); /* GRAM horizontal Address */

LCD_WriteReg(0x21, 0x0000); /* GRAM Vertical Address */

/* ----- Adjust the Gamma Curve ----- */

LCD_WriteReg(0x30, 0x0000);

LCD_WriteReg(0x31, 0x0707);

LCD_WriteReg(0x32, 0x0307);

LCD_WriteReg(0x35, 0x0200);

LCD_WriteReg(0x36, 0x0008);

LCD_WriteReg(0x37, 0x0004);

LCD_WriteReg(0x38, 0x0000);

LCD_WriteReg(0x39, 0x0707);

LCD_WriteReg(0x3C, 0x0002);

LCD_WriteReg(0x3D, 0x1D04);

/* ----- Set GRAM area ----- */

LCD_WriteReg(0x50, 0x0000); /* Horizontal GRAM Start Address */
```

```
        LCD_WriteReg(0x51, 0x00EF); /* Horizontal GRAM End Address */
        LCD_WriteReg(0x52, 0x0000); /* Vertical GRAM Start Address */
        LCD_WriteReg(0x53, 0x013F); /* Vertical GRAM Start Address */
        LCD_WriteReg(0x60, 0xA700); /* Gate Scan Line */
        LCD_WriteReg(0x61, 0x0001); /* NDL,VLE, REV */
        LCD_WriteReg(0x6A, 0x0000); /* set scrolling line */
        /* ----- Partial Display Control ----- */
        LCD_WriteReg(0x80, 0x0000);
        LCD_WriteReg(0x81, 0x0000);
        LCD_WriteReg(0x82, 0x0000);
        LCD_WriteReg(0x83, 0x0000);
        LCD_WriteReg(0x84, 0x0000);
        LCD_WriteReg(0x85, 0x0000);
        /* ----- Panel Control ----- */
        LCD_WriteReg(0x90, 0x0010);
        LCD_WriteReg(0x92, 0x0600);
        LCD_WriteReg(0x07, 0x0133); /* 262K color and display ON */
    }
else if( DeviceCode == 0x9320 || DeviceCode == 0x9300 )
{
    LCD_Code = ILI9320;
    LCD_WriteReg(0x00,0x0000);
    LCD_WriteReg(0x01,0x0100); /* Driver Output Contral */
    LCD_WriteReg(0x02,0x0700); /* LCD Driver Waveform Contral */
    LCD_WriteReg(0x03,0x1018); /* Entry Mode Set */

    LCD_WriteReg(0x04,0x0000); /* Scalling Contral */
    LCD_WriteReg(0x08,0x0202); /* Display Contral */
    LCD_WriteReg(0x09,0x0000); /* Display Contral 3.(0x0000) */
    LCD_WriteReg(0x0a,0x0000); /* Frame Cycle Contral.(0x0000) */
    LCD_WriteReg(0x0c,(1<<0)); /* Extern Display Interface Contral */
    LCD_WriteReg(0x0d,0x0000); /* Frame Maker Position */
    LCD_WriteReg(0x0f,0x0000); /* Extern Display Interface Contral 2. */

    delay_ms(100); /* delay 100 ms */

    LCD_WriteReg(0x07,0x0101); /* Display Contral */
    delay_ms(100); /* delay 100 ms */
}
```

```
LCD_WriteReg(0x10, (1<<12) | (0<<8) | (1<<7) | (1<<6) | (0<<4));      /* Power
Control 1.(0x16b0)      */
LCD_WriteReg(0x11,0x0007);                                              /* Power Control 2 */
LCD_WriteReg(0x12,(1<<8) | (1<<4) | (0<<0));
/* Power Control 3.(0x0138)    */
LCD_WriteReg(0x13,0xb000);                                              /* Power Control 4 */
LCD_WriteReg(0x29,0x0000);                                              /* Power Control 7 */

LCD_WriteReg(0x2b,(1<<14) | (1<<4));

LCD_WriteReg(0x50,0);          /* Set X Start */
LCD_WriteReg(0x51,239);       /* Set X End */
LCD_WriteReg(0x52,0);          /* Set Y Start */
LCD_WriteReg(0x53,319);       /* Set Y End */

LCD_WriteReg(0x60,0x2700);     /* Driver Output Control */
LCD_WriteReg(0x61,0x0001);     /* Driver Output Control */
LCD_WriteReg(0x6a,0x0000);     /* Vertical Srcoll Control */

LCD_WriteReg(0x80,0x0000);     /* Display Position? Partial Display 1 */
LCD_WriteReg(0x81,0x0000);     /* RAM Address Start? Partial Display 1 */
LCD_WriteReg(0x82,0x0000);     /* RAM Address End-Partial Display 1 */
LCD_WriteReg(0x83,0x0000);     /* Dispsty Position? Partial Display 2 */
LCD_WriteReg(0x84,0x0000);     /* RAM Address Start? Partial Display 2 */
LCD_WriteReg(0x85,0x0000);     /* RAM Address End? Partial Display 2 */

LCD_WriteReg(0x90,(0<<7) | (16<<0));      /* Frame Cycle Contral.(0x0013)
*/
LCD_WriteReg(0x92,0x0000);     /* Panel Interface Contral 2.(0x0000) */
LCD_WriteReg(0x93,0x0001);     /* Panel Interface Contral 3. */
LCD_WriteReg(0x95,0x0110);     /* Frame Cycle Contral.(0x0110)      */
LCD_WriteReg(0x97,(0<<8));
LCD_WriteReg(0x98,0x0000);     /* Frame Cycle Contral */

LCD_WriteReg(0x07,0x0173);

}

else if( DeviceCode == 0x1505 || DeviceCode == 0x0505 )
{
    LCD_Code = R61505U;
```

```
/* initializing funciton */

LCD_WriteReg(0xe5,0x8000); /* Set the internal vcore voltage */

LCD_WriteReg(0x00,0x0001); /* start OSC */

LCD_WriteReg(0x2b,0x0010); /* Set the frame rate as 80 when the internal
resistor is used for oscillator circuit */

LCD_WriteReg(0x01,0x0100); /* s720 to s1 ; G1 to G320 */

LCD_WriteReg(0x02,0x0700); /* set the line inversion */

LCD_WriteReg(0x03,0x1018); /* 65536 colors */

LCD_WriteReg(0x04,0x0000);

LCD_WriteReg(0x08,0x0202); /* specify the line number of front and back
porch periods respectively */

LCD_WriteReg(0x09,0x0000);

LCD_WriteReg(0x0a,0x0000);

LCD_WriteReg(0x0c,0x0000); /* select internal system clock */

LCD_WriteReg(0x0d,0x0000);

LCD_WriteReg(0x0f,0x0000);

LCD_WriteReg(0x50,0x0000); /* set windows adress */

LCD_WriteReg(0x51,0x00ef);

LCD_WriteReg(0x52,0x0000);

LCD_WriteReg(0x53,0x013f);

LCD_WriteReg(0x60,0x2700);

LCD_WriteReg(0x61,0x0001);

LCD_WriteReg(0x6a,0x0000);

LCD_WriteReg(0x80,0x0000);

LCD_WriteReg(0x81,0x0000);

LCD_WriteReg(0x82,0x0000);

LCD_WriteReg(0x83,0x0000);

LCD_WriteReg(0x84,0x0000);

LCD_WriteReg(0x85,0x0000);

LCD_WriteReg(0x90,0x0010);

LCD_WriteReg(0x92,0x0000);

LCD_WriteReg(0x93,0x0003);

LCD_WriteReg(0x95,0x0110);

LCD_WriteReg(0x97,0x0000);

LCD_WriteReg(0x98,0x0000);

/* power setting function */

LCD_WriteReg(0x10,0x0000);

LCD_WriteReg(0x11,0x0000);

LCD_WriteReg(0x12,0x0000);

LCD_WriteReg(0x13,0x0000);
```

```
delay_ms(100);

LCD_WriteReg(0x10,0x17b0);

LCD_WriteReg(0x11,0x0004);

delay_ms(50);

LCD_WriteReg(0x12,0x013e);

delay_ms(50);

LCD_WriteReg(0x13,0x1f00);

LCD_WriteReg(0x29,0x000f);

delay_ms(50);

LCD_WriteReg(0x20,0x0000);

LCD_WriteReg(0x21,0x0000);

/* initializing function */

LCD_WriteReg(0x30,0x0204);

LCD_WriteReg(0x31,0x0001);

LCD_WriteReg(0x32,0x0000);

LCD_WriteReg(0x35,0x0206);

LCD_WriteReg(0x36,0x0600);

LCD_WriteReg(0x37,0x0500);

LCD_WriteReg(0x38,0x0505);

LCD_WriteReg(0x39,0x0407);

LCD_WriteReg(0x3c,0x0500);

LCD_WriteReg(0x3d,0x0503);

/* display on */

LCD_WriteReg(0x07,0x0173);

}

else if( DeviceCode == 0x5408 )

{

LCD_Code = SPFD5408B;

LCD_WriteReg(0x0001,0x0100); /* Driver Output Contral Register */
LCD_WriteReg(0x0002,0x0700); /* LCD Driving Waveform Contral */
LCD_WriteReg(0x0003,0x1030); /* Entry Mode */

LCD_WriteReg(0x0004,0x0000); /* Scalling Control register */
LCD_WriteReg(0x0008,0x0207); /* Display Control 2 */
LCD_WriteReg(0x0009,0x0000); /* Display Control 3 */
LCD_WriteReg(0x000A,0x0000); /* Frame Cycle Control */
```

```

        LCD_WriteReg(0x000C,0x0000);      /* External Display Interface Control 1 */
        LCD_WriteReg(0x000D,0x0000);      /* Frame Maker Position */
        LCD_WriteReg(0x000F,0x0000);      /* External Display Interface Control 2 */
        delay_ms(50);

        LCD_WriteReg(0x0007,0x0101);      /* Display Control */
        delay_ms(50);

        LCD_WriteReg(0x0010,0x16B0);      /* Power Control 1 */
        LCD_WriteReg(0x0011,0x0001);      /* Power Control 2 */
        LCD_WriteReg(0x0017,0x0001);      /* Power Control 3 */
        LCD_WriteReg(0x0012,0x0138);      /* Power Control 4 */
        LCD_WriteReg(0x0013,0x0800);      /* Power Control 5 */
        LCD_WriteReg(0x0029,0x0009);      /* NVM read data 2 */
        LCD_WriteReg(0x002a,0x0009);      /* NVM read data 3 */
        LCD_WriteReg(0x00a4,0x0000);
        LCD_WriteReg(0x0050,0x0000);      /* ÉèÖÃ²Ù×÷’°¿ÚµÄXÖá¿ªÊ¾ÁÐ */
        LCD_WriteReg(0x0051,0x00EF);      /* ÉèÖÃ²Ù×÷’°¿ÚµÄXÖá½áÊøÁÐ */
        LCD_WriteReg(0x0052,0x0000);      /* ÉèÖÃ²Ù×÷’°¿ÚµÄYÖá¿ªÊ¾DD */
        LCD_WriteReg(0x0053,0x013F);      /* ÉèÖÃ²Ù×÷’°¿ÚµÄYÖá½áÊøDD */

        LCD_WriteReg(0x0060,0x2700);      /* Driver Output Control */
        /*

ÉèÖÃÆÁÃ»µÄµåÊýÖÔ¾°É··ÅèµÄÆØÊ¾DD */

        LCD_WriteReg(0x0061,0x0003);      /* Driver Output Control */
        LCD_WriteReg(0x006A,0x0000);      /* Vertical Scroll Control */

        LCD_WriteReg(0x0080,0x0000);      /* Display Position "C Partial Display 1 */
*/
        LCD_WriteReg(0x0081,0x0000);      /* RAM Address Start "C Partial Display 1 */
*/
        LCD_WriteReg(0x0082,0x0000);      /* RAM address End - Partial Display 1 */
        LCD_WriteReg(0x0083,0x0000);      /* Display Position "C Partial Display 2 */
*/
        LCD_WriteReg(0x0084,0x0000);      /* RAM Address Start "C Partial Display 2 */
*/
        LCD_WriteReg(0x0085,0x0000);      /* RAM address End "C Partail Display2 */
        LCD_WriteReg(0x0090,0x0013);      /* Frame Cycle Control */
        LCD_WriteReg(0x0092,0x0000);      /* Panel Interface Control 2 */
        LCD_WriteReg(0x0093,0x0003);      /* Panel Interface control 3 */
        LCD_WriteReg(0x0095,0x0110);      /* Frame Cycle Control */
        LCD_WriteReg(0x0007,0x0173);

}
else if( DeviceCode == 0x4531 )

```

```
{  
    LCD_Code = LGDP4531;  
    /* Setup display */  
    LCD_WriteReg(0x00,0x0001);  
    LCD_WriteReg(0x10,0x0628);  
    LCD_WriteReg(0x12,0x0006);  
    LCD_WriteReg(0x13,0x0A32);  
    LCD_WriteReg(0x11,0x0040);  
    LCD_WriteReg(0x15,0x0050);  
    LCD_WriteReg(0x12,0x0016);  
    delay_ms(50);  
    LCD_WriteReg(0x10,0x5660);  
    delay_ms(50);  
    LCD_WriteReg(0x13,0x2A4E);  
    LCD_WriteReg(0x01,0x0100);  
    LCD_WriteReg(0x02,0x0300);  
    LCD_WriteReg(0x03,0x1030);  
    LCD_WriteReg(0x08,0x0202);  
    LCD_WriteReg(0x0A,0x0000);  
    LCD_WriteReg(0x30,0x0000);  
    LCD_WriteReg(0x31,0x0402);  
    LCD_WriteReg(0x32,0x0106);  
    LCD_WriteReg(0x33,0x0700);  
    LCD_WriteReg(0x34,0x0104);  
    LCD_WriteReg(0x35,0x0301);  
    LCD_WriteReg(0x36,0x0707);  
    LCD_WriteReg(0x37,0x0305);  
    LCD_WriteReg(0x38,0x0208);  
    LCD_WriteReg(0x39,0x0F0B);  
    delay_ms(50);  
    LCD_WriteReg(0x41,0x0002);  
    LCD_WriteReg(0x60,0x2700);  
    LCD_WriteReg(0x61,0x0001);  
    LCD_WriteReg(0x90,0x0119);  
    LCD_WriteReg(0x92,0x010A);  
    LCD_WriteReg(0x93,0x0004);  
    LCD_WriteReg(0xA0,0x0100);  
    delay_ms(50);  
    LCD_WriteReg(0x07,0x0133);
```

```

        delay_ms(50);

        LCD_WriteReg(0xA0,0x0000);

    }

    else if( DeviceCode == 0x4535 )

    {

        LCD_Code = LGDP4535;

        LCD_WriteReg(0x15, 0x0030); /* Set the internal vcore voltage */

        LCD_WriteReg(0x9A, 0x0010); /* Start internal OSC */

        LCD_WriteReg(0x11, 0x0020); /* set SS and SM bit */

        LCD_WriteReg(0x10, 0x3428); /* set 1 line inversion */

        LCD_WriteReg(0x12, 0x0002); /* set GRAM write direction and BGR=1

*/
        LCD_WriteReg(0x13, 0x1038); /* Resize register */

        delay_ms(40);

        LCD_WriteReg(0x12, 0x0012); /* set the back porch and front porch

*/
        delay_ms(40);

        LCD_WriteReg(0x10, 0x3420); /* set non-display area refresh cycle

ISC[3:0] */

        LCD_WriteReg(0x13, 0x3045); /* FMARK function */

        delay_ms(70);

        LCD_WriteReg(0x30, 0x0000); /* RGB interface setting */

        LCD_WriteReg(0x31, 0x0402); /* Frame marker Position */

        LCD_WriteReg(0x32, 0x0307); /* RGB interface polarity */

        LCD_WriteReg(0x33, 0x0304); /* SAP, BT[3:0], AP, DSTB, SLP, STB */

        LCD_WriteReg(0x34, 0x0004); /* DC1[2:0], DC0[2:0], VC[2:0] */

        LCD_WriteReg(0x35, 0x0401); /* VREG1OUT voltage */

        LCD_WriteReg(0x36, 0x0707); /* VDV[4:0] for VCOM amplitude */

        LCD_WriteReg(0x37, 0x0305); /* SAP, BT[3:0], AP, DSTB, SLP, STB */

        LCD_WriteReg(0x38, 0x0610); /* DC1[2:0], DC0[2:0], VC[2:0] */

        LCD_WriteReg(0x39, 0x0610); /* VREG1OUT voltage */

        LCD_WriteReg(0x01, 0x0100); /* VDV[4:0] for VCOM amplitude */

        LCD_WriteReg(0x02, 0x0300); /* VCM[4:0] for VCOMH */

        LCD_WriteReg(0x03, 0x1030); /* GRAM horizontal Address */

        LCD_WriteReg(0x08, 0x0808); /* GRAM Vertical Address */

        LCD_WriteReg(0x0A, 0x0008);

        LCD_WriteReg(0x60, 0x2700); /* Gate Scan Line */

        LCD_WriteReg(0x61, 0x0001); /* NDL,VLE, REV */

        LCD_WriteReg(0x90, 0x013E);

        LCD_WriteReg(0x92, 0x0100);

        LCD_WriteReg(0x93, 0x0100);

```

```
    LCD_WriteReg(0xA0, 0x3000);

    LCD_WriteReg(0xA3, 0x0010);

    LCD_WriteReg(0x07, 0x0001);

    LCD_WriteReg(0x07, 0x0021);

    LCD_WriteReg(0x07, 0x0023);

    LCD_WriteReg(0x07, 0x0033);

    LCD_WriteReg(0x07, 0x0133);

}

else if( DeviceCode == 0x7783 )

{

    LCD_Code = ST7781;

    /* Start Initial Sequence */

    LCD_WriteReg(0x00FF,0x0001);

    LCD_WriteReg(0x00F3,0x0008);

    LCD_WriteReg(0x0001,0x0100);

    LCD_WriteReg(0x0002,0x0700);

    LCD_WriteReg(0x0003,0x1030);

    LCD_WriteReg(0x0008,0x0302);

    LCD_WriteReg(0x0008,0x0207);

    LCD_WriteReg(0x0009,0x0000);

    LCD_WriteReg(0x000A,0x0000);

    LCD_WriteReg(0x0010,0x0000);

    LCD_WriteReg(0x0011,0x0005);

    LCD_WriteReg(0x0012,0x0000);

    LCD_WriteReg(0x0013,0x0000);

    delay_ms(50);

    LCD_WriteReg(0x0010,0x12B0);

    delay_ms(50);

    LCD_WriteReg(0x0011,0x0007);

    delay_ms(50);

    LCD_WriteReg(0x0012,0x008B);

    delay_ms(50);

    LCD_WriteReg(0x0013,0x1700);

    delay_ms(50);

    LCD_WriteReg(0x0029,0x0022);

    LCD_WriteReg(0x0030,0x0000);

    LCD_WriteReg(0x0031,0x0707);

    LCD_WriteReg(0x0032,0x0505);

    LCD_WriteReg(0x0035,0x0107);
```

```
    LCD_WriteReg(0x0036,0x0008);

    LCD_WriteReg(0x0037,0x0000);

    LCD_WriteReg(0x0038,0x0202);

    LCD_WriteReg(0x0039,0x0106);

    LCD_WriteReg(0x003C,0x0202);

    LCD_WriteReg(0x003D,0x0408);

    delay_ms(50);

    LCD_WriteReg(0x0050,0x0000);

    LCD_WriteReg(0x0051,0x00EF);

    LCD_WriteReg(0x0052,0x0000);

    LCD_WriteReg(0x0053,0x013F);

    LCD_WriteReg(0x0060,0xA700);

    LCD_WriteReg(0x0061,0x0001);

    LCD_WriteReg(0x0090,0x0033);

    LCD_WriteReg(0x002B,0x000B);

    LCD_WriteReg(0x0007,0x0133);

}

else /* special ID */

{

    DeviceCode = LCD_ReadReg(0x67);

    if( DeviceCode == 0x0047 )

    {

        LCD_Code = HX8347A;

        LCD_WriteReg(0x0042,0x0008);

        /* Gamma setting */

        LCD_WriteReg(0x0046,0x00B4);

        LCD_WriteReg(0x0047,0x0043);

        LCD_WriteReg(0x0048,0x0013);

        LCD_WriteReg(0x0049,0x0047);

        LCD_WriteReg(0x004A,0x0014);

        LCD_WriteReg(0x004B,0x0036);

        LCD_WriteReg(0x004C,0x0003);

        LCD_WriteReg(0x004D,0x0046);

        LCD_WriteReg(0x004E,0x0005);

        LCD_WriteReg(0x004F,0x0010);

        LCD_WriteReg(0x0050,0x0008);

        LCD_WriteReg(0x0051,0x000a);

        /* Window Setting */
    }
}
```

```
LCD_WriteReg(0x0002,0x0000);
LCD_WriteReg(0x0003,0x0000);
LCD_WriteReg(0x0004,0x0000);
LCD_WriteReg(0x0005,0x00EF);
LCD_WriteReg(0x0006,0x0000);
LCD_WriteReg(0x0007,0x0000);
LCD_WriteReg(0x0008,0x0001);
LCD_WriteReg(0x0009,0x003F);
delay_ms(10);

LCD_WriteReg(0x0001,0x0006);
LCD_WriteReg(0x0016,0x00C8);
LCD_WriteReg(0x0023,0x0095);
LCD_WriteReg(0x0024,0x0095);
LCD_WriteReg(0x0025,0x00FF);
LCD_WriteReg(0x0027,0x0002);
LCD_WriteReg(0x0028,0x0002);
LCD_WriteReg(0x0029,0x0002);
LCD_WriteReg(0x002A,0x0002);
LCD_WriteReg(0x002C,0x0002);
LCD_WriteReg(0x002D,0x0002);
LCD_WriteReg(0x003A,0x0001);
LCD_WriteReg(0x003B,0x0001);
LCD_WriteReg(0x003C,0x00F0);
LCD_WriteReg(0x003D,0x0000);
delay_ms(20);

LCD_WriteReg(0x0035,0x0038);
LCD_WriteReg(0x0036,0x0078);
LCD_WriteReg(0x003E,0x0038);
LCD_WriteReg(0x0040,0x000F);
LCD_WriteReg(0x0041,0x00F0);
LCD_WriteReg(0x0038,0x0000);
/* Power Setting */
LCD_WriteReg(0x0019,0x0049);
LCD_WriteReg(0x0093,0x000A);
delay_ms(10);

LCD_WriteReg(0x0020,0x0020);
LCD_WriteReg(0x001D,0x0003);
LCD_WriteReg(0x001E,0x0000);
LCD_WriteReg(0x001F,0x0009);
```



```
if( LCD_Code == HX8347A )
{
    LCD_WriteReg(0x02,0x00);
    LCD_WriteReg(0x03,0x00);

    LCD_WriteReg(0x04,0x00);
    LCD_WriteReg(0x05,0xEF);

    LCD_WriteReg(0x06,0x00);
    LCD_WriteReg(0x07,0x00);

    LCD_WriteReg(0x08,0x01);
    LCD_WriteReg(0x09,0x3F);
}

else
{
    LCD_SetCursor(0,0);
}

LCD_WriteIndex(0x0022);
for( index = 0; index < MAX_X * MAX_Y; index++ )
{
    LCD_WriteData(Color);
}
}

/*****************************************
* Function Name : LCD_BGR2RGB
* Description   : RRRRGGGGGGBBBBB ,Ãa BBBBGGGGRRRRR ,nE%
* Input         : - color: BRG ÑOÉ«Öµ
* Output        : None
* Return        : RGB ÑOÉ«Öµ
* Attention     : ÅUz°-Eýu-ÖÅ
*****************************************/
static uint16_t LCD_BGR2RGB(uint16_t color)
{
    uint16_t r, g, b, rgb;
```

```
b = ( color>>0 ) & 0x1f;
g = ( color>>5 ) & 0x3f;
r = ( color>>11 ) & 0x1f;

rgb = (b<<11) + (g<<5) + (r<<0);

return( rgb );
}

/*****************/
* Function Name : LCD_GetPoint
* Description   : »ñÈ¡Ö, ¶"×ù±êµÃÑØÉ«Öµ
* Input         : - Xpos: Row Coordinate
*                 - Xpos: Line Coordinate
* Output        : None
* Return        : Screen Color
* Attention     : None
/*****************/
uint16_t LCD_GetPoint(uint16_t Xpos,uint16_t Ypos)
{
    uint16_t dummy;

    LCD_SetCursor(Xpos,Ypos);
    LCD_WriteIndex(0x0022);

    switch( LCD_Code )
    {
        case ST7781:
        case LGDP4531:
        case LGDP4535:
        case ILI9325C:
        case ILI9325D:
            dummy = LCD_ReadData(); /* Empty read */
            dummy = LCD_ReadData();
            return dummy;
        case HX8347A:
        {
            uint8_t red,green,blue;
```

```

        dummy = LCD_ReadData(); /* Empty read */

        red = LCD_ReadData() >> 3;
        green = LCD_ReadData() >> 2;
        blue = LCD_ReadData() >> 3;
        dummy = (uint16_t) ( ( red<<11 ) | ( green << 5 ) | blue );
    }

    return dummy;

default: /* 0x9320 0x9325 0x5408 0x1505 0x0505 */
    dummy = LCD_ReadData(); /* Empty read */
    dummy = LCD_ReadData();
    return LCD_BGR2RGB( dummy );
}

}

/****************************************
* Function Name : LCD_SetPoint
* Description   : ÓÚÖ,¶``ù±ê»-µã
* Input         : - Xpos: Row Coordinate
*                  - Ypos: Line Coordinate
* Output        : None
* Return        : None
* Attention     : None
****************************************/
void LCD_SetPoint(uint16_t Xpos,uint16_t Ypos,uint16_t point)
{
    if( Xpos >= MAX_X || Ypos >= MAX_Y )
    {
        return;
    }
    LCD_SetCursor(Xpos,Ypos);
    LCD_WriteReg(0x0022,point);
}

/****************************************
* Function Name : LCD_DrawLine
* Description   : Bresenham's line algorithm
* Input         : - x1: AµãÐÐù±ê

```

```

*
*           - y1: AµãÁÐ×ù±ê
*
*           - x2: BµãÐÐ×ù±ê
*
*           - y2: BµãÁÐ×ù±ê
*
*           - color: ÍßÑÑÉ«
*
* Output      : None
*
* Return     : None
*
* Attention   : None
*****
void LCD_DrawLine( uint16_t x0, uint16_t y0, uint16_t x1, uint16_t y1 , uint16_t color )
{
    short dx,dy;      /* ¶ºÓåX YÖáÉÍÊÖö¹ÓµÃ±äÁçÖµ */
    short temp;       /* Eðµã ÖÖµã` óÐ;±È½Í »»»» ÊÝ³4Ý±µÄÖÐ¾ä±äÁç */

    if( x0 > x1 )      /* XÖáÉÍEðµã` óÓÚÖÖµã »»»» ÊÝ³4Ý */
    {
        temp = x1;
        x1 = x0;
        x0 = temp;
    }

    if( y0 > y1 )      /* YÖáÉÍEðµã` óÓÚÖÖµã »»»» ÊÝ³4Ý */
    {
        temp = y1;
        y1 = y0;
        y0 = temp;
    }

    dx = x1-x0;        /* XÖá·½ÍðÉÍµÃÖöÁç */
    dy = y1-y0;        /* YÖá·½ÍðÉÍµÃÖöÁç */

    if( dx == 0 )       /* XÖáÉíÃ»ÓÐÔöÁç »-`¹Ö±Íß */
    {
        do
        {
            LCD_SetPoint(x0, y0, color); /* ÖðµãÍÖÊ¾ Áè`¹Ö±Íß */
            y0++;
        }
        while( y1 >= y0 );
        return;
    }
}

```

```

if( dy == 0 )      /* YÖáÉÍÃ»ÓÐÓöÁç »-Ë@ËzÖ±Íß */
{
    do
    {
        LCD_SetPoint(x0, y0, color); /* ÖðµãÍÖÊ¾ ÅèË@ËzÍß */
        x0++;
    }
    while( x1 >= x0 );
    return;
}

/* ²¼À¾È-°°Å · (Bresenham) Ëã ..»-Íß */
if( dx > dy )           /* ¿¿züXÖá */
{
    temp = 2 * dy - dx;          /* ¼EEäíÅ, öµãµÄ»ÖÅ */
    while( x0 != x1 )
    {
        LCD_SetPoint(x0,y0,color); /* »-Ëðµã */
        x0++;                      /* XÖáÉÍ¾Ó1 */
        if( temp > 0 )            /* ÅÐ¶ÍÍÄíÅ, öµãµÄ»ÖÅ */
        {
            y0++;                  /* ÎºÖðÉÍÍàÁÚµã£-¾'£ ``x0+1,y0+1£© */
            temp += 2 * dy - 2 * dx;
        }
    }
    else
    {
        temp += 2 * dy;          /* ÅÐ¶ÍÍÄíÅ, öµãµÄ»ÖÅ */
    }
    LCD_SetPoint(x0,y0,color);
}
else
{
    temp = 2 * dx - dy;          /* ¿¿züYÖá */
    while( y0 != y1 )
    {
        LCD_SetPoint(x0,y0,color);
        y0++;
        if( temp > 0 )
        {

```

```

        x0++;
        temp+=2*dy-2*dx;
    }
    else
    {
        temp += 2 * dy;
    }
}

LCD_SetPoint(x0,y0,color);
}

}

/*********************************************
* Function Name : PutChar
* Description   : LcdEAEIÈiâî»ÖÄiÔE%Ö», Ö×Ö·û
* Input         : - Xpos: È@E½×ø±ê
*                  - Ypos: '¹Ö±×ø±ê
*                  - ASCII: ÍÔE%µÃ×Ö·û
*                  - charColor: ×Ö·ûÑÖÉ«
*                  - bkColor: ±³¾ºÑÖÉ«
* Output        : None
* Return        : None
* Attention     : None
*****
void PutChar( uint16_t Xpos, uint16_t Ypos, uint8_t ASCII, uint16_t charColor, uint16_t
bkColor )
{
    uint16_t i, j;
    uint8_t buffer[16], tmp_char;
    GetASCIICode(buffer,ASCII); /* È;×ÖÄfÊY%Y */
    for( i=0; i<16; i++ )
    {
        tmp_char = buffer[i];
        for( j=0; j<8; j++ )
        {
            if( (tmp_char >> 7 - j) & 0x01 == 0x01 )
            {
                LCD_SetPoint( Xpos + j, Ypos + i, charColor ); /* ×Ö·ûÑÖÉ« */
            }
        }
    }
}

```

```
{  
    LCD_SetPoint( Xpos + j, Ypos + i, bkColor ); /* ±³¾°ÑÖÉ« */  
}  
}  
}  
}  
  
*****  
* Function Name : GUI_Text  
* Description   : ÔÚÖ, ¶''xù±êïÖÊ¾xÖ·û'®  
* Input         : - Xpos: ÐÐxù±ê  
*                 - Ypos: ÁÐxù±ê  
*                 - str: xÖ·û'®  
*                 - charColor: xÖ·ûÑÖÉ«  
*                 - bkColor: ±³¾°ÑÖÉ«  
* Output        : None  
* Return        : None  
* Attention     : None  
*****  
void GUI_Text(uint16_t Xpos, uint16_t Ypos, uint8_t *str,uint16_t Color, uint16_t  
bkColor)  
{  
    uint8_t TempChar;  
    do  
    {  
        TempChar = *str++;  
        PutChar( Xpos, Ypos, TempChar, Color, bkColor );  
        if( Xpos < MAX_X - 8 )  
        {  
            Xpos += 8;  
        }  
        else if ( Ypos < MAX_Y - 16 )  
        {  
            Xpos = 0;  
            Ypos += 16;  
        }  
        else  
        {  
            Xpos = 0;  
            Ypos = 0;  
    }
```

```
    }  
}  
while ( *str != 0 );  
}  
  
*****  
***** END FILE *****  
***** /*****
```

Archivos extensión H.

```
#define DEFAULT

#ifndef DEFAULT
#define __IP1B 192
#define __IP2B 168
#define __IP3B 1
#define __IP4B 120

#define __GW1B 192
#define __GW2B 168
#define __GW3B 1
#define __GW4B 20

#define __MASK1B      255
#define __MASK2B      255
#define __MASK3B      255
#define __MASK4B      0
#endif

/**-----//-----*/
//          @filename      header.h
//
//          @version       0.00
//
//          @author        Alberto Palomo Alonso
//          @brief         Cabezera del código fuente, agrupa todos los archivos.
//          @category      Principal.
//
```

```
//          @map           @include
//
//          @end
//
//
//          //
//          //
//          //
//          //
//          //
//          //
//          //-----//-----//
//          //
//          //
//          //-----//-----//-----**/
//      Librería de registros.

#ifndef LPC17XX
#define LPC17XX
#include "LPC17XX.H"
#endif

//      Símbolos del sistema.

#ifndef SYSTEMSYMBOLS
#define SYSTEMSYMBOLS
#include "Systemsymbols.h"
#endif

//      Pwm.

#ifndef PWM
#define PWM
#include "PWM.h"
#endif

//      Configura

#ifndef CONFIGURA
#define CONFIGURA
#include "configura.h"
#endif
```

Alberto Palomo Alonso.

Alberto Palomo Alonso.

```
// @version          0.00
//
// @author           Alberto Palomo Alonso
// @@
// @brief            Cabecera del código fuente de Statechart.c
// @@
// @category         Principal.
// @@
// @map              @include
// @@
// @private
// @@
// @types
// @@
// @funcdef
// @@
// @end
// @@
// -----
// -----
// @include          Estos son los archivos utilizados con el
// statechart.
// @@
// -----
// -----
// **/
// LCD
#ifndef GLCD
```

```
#define GLCD
#include      "GLCD.h"
#include      "TouchPanel.h"
#include      "menu.h"
#endif
#ifndef SYSTEMSYMBOLS
#define SYSTEMSYMBOLS
#include      "Systemsymbols.h"
#endif
#ifndef STRING
#define STRING
#include      <string.h>
#endif
#ifndef WTD
#define WTD
#include      "WDT.h"
#endif
#ifndef PWM
#define PWM
#include      "PWM.h"
#endif
#ifndef STDIO
#define STDIO
#include      <stdio.h>
#endif
#ifndef MIGLOBAL
#define MIGLOBAL
#include      "miGlobal.h"
#endif
#ifndef TIMERS
#define TIMERS
#include      "Timers.h"
#endif
#ifndef RTL
#define RTL
#include      "RTL.h"
#endif
#ifndef HTTPSOURCE
#define HTTPSOURCE
```

```
#include      "HTTP_SOURCE.h"
#endif
#ifndef LUT
#define LUT
#include "LUT.h"
#endif
#ifndef DAC
#define DAC
#include "DAC.h"
#endif
#ifndef UFONO
#define UFONO
#include "uFono.h"
#endif
#ifndef ONEWIRE
#define ONEWIRE
#include "OneWire.h"
#endif
/*-----*/
//          @private           Estos son los simbolos correspondientes al
statechart.                                     //
//                                              //**
//-----*/
#define PANTALLA_INICIO      0
#define PANTALLA_MEDIDAS1    1
#define PANTALLA_MEDIDAS2    2
#define PANTALLA_AJUSTES     3
#define PANTALLA_LOADING     4
#define PANTALLA_VALORES     5

#define MAXIMOX              240
#define MAXIMOY              320
#define CLEAR_BUFFER          ""
```

```
/**-----//  
//  
//  
//  
//          @types      Tipos utilizados en el statechart.  COPIADOS DE menu.c  
//  
//  
//  
//-----**/  
typedef struct {  
    uint16_t x;  
    uint16_t y;  
    uint16_t size_x;  
    uint16_t size_y;  
    uint8_t  pressed;  
}screenZone_t;  
/**-----//  
//  
//  
//-----**/  
//          @funcdef      Estas son las funciones correspondientes al  
statechart. //  
//  
//  
//-----**/  
void __mainLoop__        ( void );  
void __configuraLCD__   ( void );  
void __pintaInicio__   ( void );  
void __pintaAjustes__  ( void );  
void __pintaMedidas1__ ( void );  
void __pintaMedidas2__ ( void );  
void squareButton        ( screenZone_t * zone , char *  
                           text , uint16_t textColor , uint16_t lineColor);  
void checkTouchPanel    ( void );  
int8_t zoneNewPressed  ( screenZone_t * zone );  
void squareBox           ( screenZone_t * zone , uint16_t  
                           color);
```

Alberto Palomo Alonso.

```
void __pintaCargandoSeno__ ( void );
void __pintaCargandoConexion__ ( void );
void __pintaCargandoDone__ ( void );
void __pintaCargandoInicio__ ( void );
void __pintaCargandoIniciando__ ( void );
void __pintaValores__ ( void );

/**-----//  
//  
//  
// @end ENDFILE.  
//  
//  
//-----**/
```

```
*****Copyright
(c)*****
**
**                               http://www.powermcu.com
**
**-----File Info-----
-----
** File name:          GLCD.h
** Descriptions:       TFT (IO)
**
**-----File Info-----
-----
** Created by:          AVRman
** Created date:        2015-1-26
** Version:             2.0
** Descriptions:        The original version
**
**-----File Info-----
-----
** Modified by:
** Modified date:
```

```
** Version:  
** Descriptions:  
*****  
***** /  
  
#ifndef __GLCD_H  
#define __GLCD_H  
  
#include <LPC17xx.H>  
  
/* Private define -----*/  
  
/* LCD Interface */  
#define PIN_CS      (1 << 8)  
#define PIN_RS      (1 << 27)  
#define PIN_WR      (1 << 28)  
#define PIN_RD      (1 << 29)  
  
#define LCD_CS(x)   ((x) ? (LPC_GPIO2->FIOSET = PIN_CS) : (LPC_GPIO2->FIOCLR = PIN_CS));  
#define LCD_RS(x)   ((x) ? (LPC_GPIO1->FIOSET = PIN_RS) : (LPC_GPIO1->FIOCLR = PIN_RS));  
#define LCD_WR(x)   ((x) ? (LPC_GPIO1->FIOSET = PIN_WR) : (LPC_GPIO1->FIOCLR = PIN_WR));  
#define LCD_RD(x)   ((x) ? (LPC_GPIO1->FIOSET = PIN_RD) : (LPC_GPIO1->FIOCLR = PIN_RD));  
  
/* Private define -----*/  
#define DISP_ORIENTATION 0 /* angle 0 90 */  
  
#if (DISP_ORIENTATION == 90) || (DISP_ORIENTATION == 270)  
  
#define MAX_X 320  
#define MAX_Y 240  
  
#elif (DISP_ORIENTATION == 0) || (DISP_ORIENTATION == 180)  
  
#define MAX_X 240  
#define MAX_Y 320  
  
#endif
```

```
/* LCD color */

#define White          0xFFFF
#define Black          0x0000
#define Grey           0xF7DE
#define Blue           0x001F
#define Blue2          0x051F
#define Red            0xF800
#define Magenta        0xF81F
#define Green          0x07E0
#define Cyan           0x7FFF
#define Yellow         0xFFE0

/********************* Function prototypes -----*/
void LCD_Initializtion(void);
void LCD_Clear(uint16_t Color);
uint16_t LCD_GetPoint(uint16_t Xpos,uint16_t Ypos);
void LCD_SetPoint(uint16_t Xpos,uint16_t Ypos,uint16_t point);
void LCD_DrawLine( uint16_t x0, uint16_t y0, uint16_t x1, uint16_t y1 , uint16_t color );
void PutChar( uint16_t Xpos, uint16_t Ypos, uint8_t ASCII, uint16_t charColor, uint16_t bkColor );
void GUI_Text(uint16_t Xpos, uint16_t Ypos, uint8_t *str,uint16_t Color, uint16_t bkColor );

#endif
```

```
*****  
*****  
END FILE  
*****  
*****  
  
/*-----//  
// @filename configura.h //  
// @version 0.00 //  
//  
// @author Alberto Palomo Alonso //  
//  
//  
// @brief Cabecera para el código de configuración.c //  
//  
//  
// @category Principal. //  
//  
// @map @include //  
// @private //  
// @funcdef //  
// @end //  
//  
//  
//  
//-----//  
//  
//
```

```
// @include Estos son los archivos utilizados con el código de
configuración. //  
//  
//-----*/  
// PWM  
#ifndef PWM  
#define PWM  
#include "PWM.h"  
#endif  
#ifndef GLCD  
#define GLDC  
#include "GLCD.h"  
#include "TouchPanel.h"  
#include "menu.h"  
#endif  
#ifndef STATECHART  
#define STATECHART  
#include "Statechart.h"  
#endif  
#ifndef RTC  
#define RTC  
#include "RTC.h"  
#endif  
#ifndef SYSTEMSYMBOLS  
#define SYSTEMSYMBOLS  
#include "Systemsymbols.h"  
#endif  
#ifndef WDT  
#define WDT  
#include "WDT.h"  
#endif  
#ifndef HTTP_SOURCE  
#define HTTP_SOURCE  
#include "HTTP_SOURCE.h"  
#endif  
#ifndef TIMERS  
#define TIMERS
```

```
#include      "Timers.h"
#endif
#ifndef ANEMOMETRO
#define ANEMOMETRO
#include      "Anemometro.h"
#endif
#ifndef DAC
#define DAC
#include      "DAC.h"
#endif
#ifndef UVA30A
#define UVA30A
#include      "UVA30A.h"
#endif
#endif
#ifndef UFONO
#define UFONO
#include      "uFono.h"
#endif
#ifndef LUT
#define LUT
#include      "LUT.h"
#endif
#ifndef DMA
#define DMA
#include      "DMA.h"
#endif
#ifndef I2C
#define I2C
#include      "I2C.h"
#endif
#ifndef UART0
#define UART0
#include      "UART0.h"
#endif
#ifndef UART3
#define UART3
#include      "UART3.h"
#endif
/**-----//
```

```
//  
//  
//  
//          @private           Estos son los símbolos correspondientes a la  
configuración.                                //  
  
//  
//  
//-----**/  
#define ACTIVOS_TODOS2      0x003F  
#define ACTIVOS_1_2          0x0001  
#define ACTIVOS_2_2          0x0002  
#define ACTIVOS_1_1          0x0100  
#define ACTIVOS_2_1          0x0200  
#define ACTIVOS_6_2          0x0020  
#define ACTIVOS_6_1          0x2000  
  
#define MAXIMO_PRESION      1500  
#define MINIMO_PRESION      500  
#define MAXIMO_TEMPERATURA   50  
#define MINIMO_TEMPERATURA  -10  
/*-----*/  
//  
//  
//-----**/  
//          @funcdef           Estas son las funciones correspondientes a la  
configuración.                                //  
  
//  
//  
//-----**/  
void __configuraPrograma__    (      void     );  
void __iniciaVariables__     (      void     );  
/*-----*/  
//  
//  
//-----**/  
//  
//  
//-----**/
```

Alberto Palomo Alonso.

```
//          @end      ENDFILE.  
//  
//  
//-----**/  
  
/*-----*/  
//          @filename    configura.h  
//  
//          @version     0.00  
//  
//          @author      Alberto Palomo Alonso  
//  
//  
//          @brief       Cabecera para el código de configuración.c  
//  
//  
//          @category   Principal.  
//  
//  
//          @map        @include  
//  
//          @private  
//  
//          @funcdef  
//  
//          @end  
//  
//  
//-----*/
```

```
//  
//  
  
//          @include      Estos son los archivos utilizados con el código de  
configuración.                                //  
  
//  
//  
//-----*/  
  
// PWM  
  
#ifndef PWM  
  
#define PWM  
  
#include      "PWM.h"  
  
#endif  
  
#ifndef GLCD  
  
#define GLDC  
  
#include      "GLCD.h"  
  
#include      "TouchPanel.h"  
  
#include      "menu.h"  
  
#endif  
  
#ifndef STATECHART  
  
#define STATECHART  
  
#include      "Statechart.h"  
  
#endif  
  
#ifndef RTC  
  
#define RTC  
  
#include      "RTC.h"  
  
#endif  
  
#ifndef SYSTEMSYMBOLS  
  
#define SYSTEMSYMBOLS  
  
#include      "Systemsymbols.h"  
  
#endif  
  
#ifndef WDT  
  
#define WDT  
  
#include      "WDT.h"  
  
#endif  
  
#ifndef HTTP_SOURCE  
  
#define HTTP_SOURCE  
  
#include "HTTP_SOURCE.h"
```

```
#endif

#ifndef TIMERS
#define TIMERS
#include "Timers.h"
#endif

#ifndef ANEMOMETRO
#define ANEMOMETRO
#include "Anemometro.h"
#endif

#ifndef DAC
#define DAC
#include "DAC.h"
#endif

#ifndef UVA30A
#define UVA30A
#include "UVA30A.h"
#endif

#ifndef UFONO
#define UFONO
#include "uFono.h"
#endif

#ifndef LUT
#define LUT
#include "LUT.h"
#endif

#ifndef DMA
#define DMA
#include "DMA.h"
#endif

#ifndef I2C
#define I2C
#include "I2C.h"
#endif

#ifndef UART0
#define UART0
#include "UART0.h"
#endif

#ifndef UART3
#define UART3

```

```
#include      "UART3.h"
#endif

/**-----*/
//          @private
//configuración.          Estos son los símbolos correspondientes a la
//                         //configuration.          //symbols corresponding to the configuration.

//          //
//          //
//-----*/          //

#define ACTIVOS_TODOS2      0x003F
#define ACTIVOS_1_2           0x0001
#define ACTIVOS_2_2           0x0002
#define ACTIVOS_1_1           0x0100
#define ACTIVOS_2_1           0x0200
#define ACTIVOS_6_2           0x0020
#define ACTIVOS_6_1           0x2000

#define MAXIMO_PRESION       1500
#define MINIMO_PRESION        500
#define MAXIMO_TEMPERATURA    50
#define MINIMO_TEMPERATURA   -10

/**-----*/
//          //
//          //
//-----*/          //

//          @funcdef
//configuración.          Estas son las funciones correspondientes a la
//                         //configuration.          //functions corresponding to the configuration.

//          //
//          //
//-----*/          //

void __configuraPrograma__  ( void );
void __iniciaVariables__   ( void );

/**-----*/
//          //
```

```
//  
//  
  
// @end ENDFILE.  
//  
//  
//  
//-----**/  
  
/*-----*/  
// @filename Anemometro.h //  
// @version 0.00 //  
// @author Alberto Palomo Alonso //  
//  
//  
//  
// @brief Esta es la cabecera donde se declara todo lo utilizado en  
el anemómetro. //  
//  
//  
// @category Opcional.  
//  
//  
// @map @include  
//  
// @private  
//  
// @funcdef  
//  
// @end  
//  
//
```

```
-----//-----  
//  
//  
//  
//  
// @include Includes pertenecientes al módulo del PWM.  
//  
//  
//  
//-----**/  
#ifndef LPC17XX  
#define LPC17XX  
#include "LPC17XX.H"  
#endif  
#ifndef SYSTEMSYMBOLS  
#define SYSTEMSYMBOLS  
#include "Systemsymbols.h"  
#endif  
#ifndef TIMERS  
#define TIMERS  
#include "Timers.h"  
#endif  
/*-----**/  
//  
//  
//  
// @private Estos son los símbolos correspondientes al  
// anemómetro.  
//  
//  
//-----**/  
#define PULSOS_VUELTA 2  
#define DIAMETRO_ANEMOMETRO 14 // En centímetros.  
#define PULSOS_CENTIMETRO (float)PULSOS_VUELTA/((float)PI*(float)DIAMETRO_ANEMOMETRO)  
#define CTCR_MASCARA 0x0 // Dejar al TC contando.  
#define CCR_MASCARA_EN 0x5 // Generar interrupción.  
#define CCR_MASCARA_DIS 0x4 // Desactivar interrupción.
```

Alberto Palomo Alonso.

Alberto Palomo Alonso.

```
#endif

#ifndef LUT
#define LUT
#include "LUT.h"
#endif

#ifndef DMA
#define DMA
#include "DMA.h"
#endif

#ifndef UFONO
#define UFONO
#include "uFono.h"
#endif

/**-----
-----//  

//  

//  

//-----**/  

//          @private           Estos son los símbolos correspondientes a la  

configuración.//  

//  

//  

//-----**/  

#define PCONP_ADC_ON    (1 << 12)
#define PINSEL_ADC01   (1 << 16)
#define PINMODE_ADC01  (3 << 16)
#define BRUST_PIN        (1 << 16)
#define SEL_CANAL1      (1 << 1)
#define SEL_CANAL_GLOBAL (1 << 8)
#define ADC_POWER        (1 << 21)
#define ADC_START        (0x6 << 24)
#define CLK_DIV_MAX     (0xFF << 8)

#define RESISTENCIA_PULL    70.00
#define LDRRESISTENCIA_MAX 100
#define LDRRESISTENCIA_MIN  1
#define BRILLO_MAX          100
#define BRILLO_MIN           1
```

Alberto Palomo Alonso.

```
#define VREF           3.3
#define VINDICE         10

/**-----*/
//          //

//          //          //

//          //          //          //

//          @funcdef      Estas son las funciones correspondientes a la
configuración.                                //

//          //

//          //          //

//          //          //          //

void __configuraLDR__ (      void      );
void ponAudioDMA      (      void      );

/**-----*/
//          //

//          //          //

//          @end          ENDFILE.

//          //

//          //          //

//          //          //          //

//          //          //          //

//          @filename      uFono.h
//          //

//          @version        0.00
//          //

//          @author         Alberto Palomo Alonso
//          //

//          //

//          //          //

//          @brief          Cabecera para configurar el audio del micrófono.
//          //
```

```
//  
//  
//      @category          Opcional.  
//  
//  
//  
//      @map               @include  
//  
//  
//      @funcdef  
//  
//      @end  
//  
//  
//  
//  
//  
//  
//-----//  
//  
//  
//  
//  
//  
//  
//  
//  
//  
//      @include          Estos son los archivos utilizados para el audio del  
micrófono.                                //  
//  
//  
//-----**/  
#ifndef SYSTEMSYMBOLS  
#define SYSTEMSYMBOLS  
#include      "Systemsymbols.h"  
#endif  
#ifndef LPC17XX  
#define LPC17XX  
#include      "LPC17XX.H"  
#endif  
#ifndef TIMERS  
#define TIMERS  
#include      "Timers.h"  
#endif
```

Alberto Palomo Alonso.

```
/*-----//  
// @filename UVA30A.h  
//  
// @version 0.00  
//  
// @author Alberto Palomo Alonso  
//  
//  
// @brief Cabecera para el código de UVA30A.c  
//  
//  
// @category Periférico.  
//  
//  
// @map @include  
//  
// @private  
//  
// @funcdef  
//  
// @end  
//  
//  
//  
//  
//-----//  
//  
// @include Estos son los archivos utilizados con el código de  
configuración. //
```

```
//  
//  
//-----**/  
#ifndef LPC17XX  
#define LPC17XX  
#include      "LPC17XX.H"  
#endif  
#ifndef LDR  
#define LDR  
#include      "LDR.h"  
#endif  
/*-----**/  
//  
//  
//-----**/  
//  
//  
//-----**/  
//  
//  
//-----**/  
#define LDR_primero    1  
#define PINSEL_ADC02   (1 << 18)  
#define PINMODE_ADC02  (3 << 18)  
#define SEL_CANAL2     (1 << 2)  
/*-----**/  
//  
//  
//-----**/  
//  
//-----**/  
//  
//-----**/  
//  
//-----**/  
void  __configuraUVA30A__( void );
```

```
/**-----//  
//  
//  
//  
//  
//  
//  
//  
//-----**/  
  
/**-----//  
// @filename OneWire.h  
//  
// @version 0.00  
//  
// @author Alberto Palomo Alonso  
//  
//  
//  
// @brief Cabecera para configurar el protocolo OneWire.  
//  
//  
//  
// @category Opcional.  
//  
//  
//  
// @map @include  
//  
// @funcdef  
//  
// @end  
//  
//  
//
```

```
//  
//  
//-----//  
//  
//  
  
//          @include      Estos son los archivos utilizados para el protocolo  
OneWire.                                //  
  
//  
//  
//-----*/  
#ifndef SYSTEMSYMBOLS  
#define SYSTEMSYMBOLS  
  
#include      "Systemsymbols.h"  
  
#endif  
  
/*-----//  
//  
//  
//  
//-----*/  
  
//          @private      Estos son los símbolos correspondientes al  
protocolo OneWire.                                //  
  
//  
//  
//-----*/  
#define SIZEOF_TRAMA           40          //      Tamaño del array como buffer  
de 8 bits.  
  
#define SIZEOF_SLOT             8           //      Los últimos 8 bits son el  
checksum.  
  
#define OW_RESET_BAJO           0.018       //      Tiempo a nivel bajo de la señal de  
start. (MCU -> Sensor)  
                                         /** @CHANGE: 0.04 */  
#define OW_RESET_ALTO           0.02        //      Tiempo a nivel alto de la  
señal de start. (MCU -> Sensor)  
#define OW_RESPUESTA_BAJO        0.08        //      Señal respuesta del sensor a  
nivel bajo.  
#define OW_RESPUESTA_ALTO        0.08        //      Señal respuesta del sensor a  
nivel alto.  
                                         /** @CHANGE: 0.078 */
```

```

#define OW_MANDADO0          0.076 // Duración del tiempo de un bit = 0.
#define OW_MANDADO1          0.120 // Duración del tiempo de un bit = 1.

#define PERMISO_DIFERENCIAL -0.005 // Error que le permito tener al sensor.
#define PERMISO_PROPORCIONAL 0.005 // Error que le permito tener al sensor.

#define OW_PULL_UP           0x0      // Función de pull up.
#define OW_PULL_DOWN          0x3      // Función de pull down.
#define OW_CAPTURE_FUNC        0x3      // Función capture.
#define OW_CTCR_MASCARA       0x0      // Dejar TC contando.
#define OW_CCR_MASCARA_EN     0x30     // Activo por flanco de bajada.
#define OW_CCR_MASCARA_DIS    0x20     // Desactivo por flanco.
#define CAP11_IR               0x00000020// El IR de CAP1.1
#define MRO_IR                 0x00000001// El IR de MRO.

#define OWDEEPSLEEP            0
#define OWINICIO                1
#define OWESPERANDO             2
#define OWESPERANDO_BIT         3
#define OWCHECKSUM              4
#define OWGENERA                 5
#define OWESPERANDO_SEQ          6

#define LIMITE_FALLOS           5
#define BITOW                     19

#define US_AHORA                  (LPC_TIM3->TC)
#define PIN_OWp                   19
#define PIN_OW                    (1      <<      19)
#define CONFIG_OUT                (LPC_GPIO1->FIODIR |= PIN_OW)
#define CONFIG_IN                 (LPC_GPIO1->FIODIR &= ~ (PIN_OW))
#define CLEAR_PIN                 (LPC_GPIO1->FIOCLR = PIN_OW)
#define SET_PIN                   (LPC_GPIO1->FIOSET = PIN_OW)
#define ENTRADA                   ((LPC_GPIO1->FIOPIN >> PIN_OWp) & 1)

/*-----*/
-----//



//



//
```

```
//          @funcdef      Estas son las funciones correspondientes al
protocalo OneWire.                      //
//                                         //
//-----**/
void __configuraOW__( void ) ;           // Configuración del protocolo OneWire.
void mideTemperatura( void ) ;           // Código de medición de temperatura.
void activaMedidaOW ( void ) ;           // Lanza el activador del one wire.

void StateChartOneWire ( uint32_t DeltaCap );
void OWSetPin ( uint8_t Nivel ) ;
void OWConfiguraEntrada ( void ) ;
void OWConfiguraSalida( void ) ;
void ErrorRx ( void ) ;
void ErrorTx ( void ) ;
void InvalidChecksum ( void ) ;

//-----**//
//                                         //
//-----**/                                //

//          @end      ENDFILE.
//                                         //
//                                         //
//-----**/                                //

//-----**//
//          @filename      PWM.h           //
//          @version        0.00
//                                         //
//          @author         Alberto Palomo Alonso
//                                         //
//                                         //
```



```
//          @include           Estos son los archivos utilizados con el código
PWM.                                         //  

//  

//  

//-----*/  

#ifndef LPC17XX  

#define LPC17XX  

#include "LPC17XX.H"  

#endif  

#ifndef SYSTEMSYMBOLS  

#define SYSTEMSYMBOLS  

#include "Systemsymbols.h"  

#endif  

/*-----*/  

//  

//  

//-----*/  

//          @private           Estos son los símbolos correspondientes al módulo
PWM.                                         //  

//  

//  

//-----*/  

#define F pwm      50          // Velocidad de 20Hz.  

#define T pwm     0.02         // Periodo de 20ms.  

#define TCR_MASK    0x9        // Activo el contador
y el PWM.  

#define MCR_MASK     0x4        // Para el set del
PWM.  

#define MCR_MASK_RESET 0x5      // Para resetear el MCR.  

#define PCONP_MASK   0x1 << 6    // Para encender el PWM.  

#define MODO_CICLO    1          // Para la función
modificaPulso().  

#define MODO_SERVO     0          // Para la función
modificaPulso().  

#define PWM1          1
```

```

#define PWM2 2
#define PWM3 3
#define PWM4 4
#define PWM5 5
#define PWM6 6

#define OPEN_DRAIN 0x2
#define PULL_UP 0x0
#define PULL_DOWN 0x3
#define UNUSED 0x1

#define KMX 1.3 // Constante de corrección máxima.
#define KMN 0.6 // Constante de corrección mínima.
#define MINIMO_SERVO 0.001
#define MAXIMO_SERVO 0.002

#define TEMP_MAX 32 // Visionado de temperatura.
#define TEMP_MIN 17 // Visionado de temperatura.

/**-----// */

// @funcdef Estas son las funciones correspondientes al módulo PWM.
// */

//-----**/
void __configuraPWM__(CualesPWM); // float FrecuenciaPWM , uint16_t Default = 0x3F3F -> Todo activado.
void modificaPulso(uint8_t Modo , uint8_t Ciclo , float Minimo , float Maximo );
void softMod(GradosActuales, GradosObjetivo , uint8_t Minimo , float Maximo , uint32_t PWMn);

/**-----// */
// */

```

```
//  
//  
// @end      ENDFILE.  
//  
//  
//  
//-----**/  
  
/*-----*/  
// @filename    DAC.h  
//  
// @version  
//  
// @author      Alberto Palomo Alonso  
//  
//  
// @brief       Cabecera para el código de DAC.c  
//  
//  
// @category    Periférico.  
//  
//  
// @map         @include  
//  
// @private  
//  
// @funcdef  
//  
// @end
```


Alberto Palomo Alonso.

Alberto Palomo Alonso.

```
//      @author      Alberto Palomo Alonso
//      @brief      Este es el programa donde se encuentra la cabecera de LUT.c
//      @category    Opcional.
//      @map        @include
//      @private
//      @funcdef
//      @end
//      @include      Includes pertenecientes al módulo del anemómetro.
//      @endif
//      @include      "Systemsymbols.h"
//      @endif
//      @ifndef LDR
//      @define LDR
//      @include      "LDR.h"
```

```
#endif

#ifndef DMA
#define DMA
#include "DMA.h"
#endif

#ifndef MATH
#define MATH
#include <math.h>
#endif

#define MUESTRAS_SENO      32
#define BRILLO_LDR          0
#define BRILLO2CICLO_LDR    1
#define INDICE_UVA          2
#define BRILLO_LDR_NOLUT    3

void goto_LUT( float dato , uint8_t LUTn , float * ret_f , uint8_t * ret_8 , uint16_t *
ret_16 , uint32_t * ret_32);
void crearSeno( void );
void ponTonoDMA( void );
/*-----
-----*/
//          //
//          @end      ENDFILE.
//          //
//          //
//          //-----*/
//-----**/


/*-----*/
-----//


//          @filename      DMA.h
//
```

Alberto Palomo Alonso.

```
// @version 0.00
//
// @author Alberto Palomo Alonso
//
// @brief Cabecera del configurado del DMA.
//
// @category Opcional.
//
// @map @include
//
// @private
//
// @funcdef
//
// @end
//
// -----
// -----
// @include Estos son los archivos utilizados con el código del
// DMA.
//
// -----
// -----
// **/
#ifndef SYSTEMSYMBOLS
#define SYSTEMSYMBOLS
#include "Systemsymbols.h"
```

Alberto Palomo Alonso.

```

#endif

#ifndef LDR
#define LDR
#include "LDR.h"
#endif

#ifndef LUT
#define LUT
#include "LUT.h"
#endif

#include <LPC17xx.H>
#include <math.h>

/**-----
-----*/
// 
// 
// 

//          @private           Estos son los simbolos correspondientes al DMA.
// 
// 
//          // 
//-----*
//-----*/ 

#define      N_samples_wave      32    // N° de muestras por ciclo
#define Ftono        400
/**-----
-----*/
// 
// 
// 

//          @funcdef           Estas son las funciones que se usan en el DMA.
// 
// 
//-----*
//-----*/ 

void __configuraDMA__ (      void      );      //      Configurador del DMA.
void __configuraTono__(      void      );      //      Configurador del tono.
void __configuraAudio__ (      void      );      //      Configurador del audio.
/**-----
-----*/

```

```
//  
//  
  
//           @end          ENDFILE.  
//  
//  
//  
//-----**/  
  
/*-----//  
-----//  
//           @filename      WDT.h  
//  
//           @version       0.00  
//  
//           @author        Alberto Palomo Alonso  
//  
//  
//           @brief         Cabecera del configurado del WDT.  
//  
//  
//           @category     Opcional.  
//  
//  
//           @map          @include  
//  
//           @private  
//  
//           @funcdef
```



```
//  
//  
//  
// @funcdef      Estas son las funciones que se usan en el WDT.  
//  
//  
//  
//-----**/  
void __configuraWDT__      ( void );  
void alimentaWDT          ( void );  
/**-----//  
//  
//-----**/  
// @end      ENDFILE.  
//  
//  
//-----**/  
//-----**/  
// @filename      UART0.h  
//  
// @version      0.00  
//  
// @author      Alberto Palomo Alonso  
//  
//  
//-----**/  
// @brief      Este es el programa que recoge la transmisión por UART0.  
//  
//-----**/  
//
```

```
//           @category          Opcional.  
//  
//  
//  
//           @map             @include  
//  
//           @function  
//  
//           @end  
//  
//  
//  
//-----//  
//  
//  
//-----//  
//           @include          Includes pertenecientes a e la transmisión  
asíncrona.          //  
//  
//  
//-----//  
//-----**/  
#ifndef LPC17XX  
#define LPC17XX  
#include      "LPC17XX.H"  
#endif  
#ifndef SYSTEMSYMBOLS  
#define SYSTEMSYMBOLS  
#include      "Systemsymbols.h"  
#endif  
#ifndef STDIO  
#define STDIO  
#include      <stdio.h>  
#endif  
#ifndef STRING  
#define STRING  
#include      <string.h>  
#endif
```

```
#ifndef PWM
#define PWM
#include "PWM.h"
#endif

#ifndef UART
#define UART
#include "uart.h"
#endif

#ifndef MIGLOBAL
#define MIGLOBAL
#include "miGlobal.h"
#endif

/**-----
-----//  

//  

//  

//  

//  

//  

//  

//  

//-----**/  

#define DLP 7
#define UART0_MTX (1 << 1)
#define UART0_MRX (1 << 2)
#define RetornoDeCarro 13
#define CADMAX 120
  

#define COM10 "GIVE SUGAR\r"
#define COM11 "GIVE IP\r"
#define COM12 "GIVE TEMPERATURA\r"
#define COM13 "GIVE PRESION\r"
#define COM14 "GIVE VIENTO\r"
#define COM15 "GIVE LUGAR\r"
#define COM16 "GIVE INDICEUV\r"
#define COM17 "GIVE HORA\r"
#define COM18 "GIVE HUMEDAD\r"
#define COM19 "GIVE BRILLO\r"
```

```
#define COM20           "SET BRILLO\r"
#define COM21           "SET HORA\r"
#define COM22           "SET MIN TEMP\r"
#define COM23           "SET MAX TEMP\r"
#define COM24           "SET MIN PRES\r"
#define COM25           "SET MAX PRES\r"
#define COM26           "SET TEMPERATURA\r"
#define COM27           "SET PRESION\r"

#define COM3            "KILL\r"

#define COM0            "ABOUT\r"
#define COM4            "HELP\r"
#define COM41           "HELP GIVE\r"
#define COM42           "HELP SET\r"

#define UART_TX         0
#define UART_RX_BRILLO 1
#define UART_RX_MINT   2
#define UART_RX_MAXT   3
#define UART_RX_MINP   4
#define UART_RX_MAXP   5
#define UART_RX_HORA   6
#define UART_RX_VARM   7

/**-----*/
//          @funcdef      Estas son las funciones correspondientes al
protocolo UART0.                                //



//          @funcdef      Estas son las funciones correspondientes al
protocolo UART0.                                //



void    __configuraUART0__    (      void     );
uint8_t procesarComando(      char *  );

/**-----*/
//          @funcdef      Estas son las funciones correspondientes al
protocolo UART0.                                //



//
```

```
//  
//  
// @end      ENDFILE.  
//  
//  
//-----**/v  
  
/*  
 * uart_.h  
 *  
 * Created on: 1-Oct-2011  
 * Author: J.M.V.C.  
 */  
  
#ifndef UART_H_  
#define UART_H_  
  
// Accepted Error baud rate value (in percent unit)  
#define UART_ACCEPTED_BAUDRATE_ERROR    3  
  
#define CHAR_8_BIT                      (3 << 0)  
#define STOP_1_BIT                      (0 << 2)  
#define PARITY_NONE                     (0 << 3)  
#define DLAB_ENABLE                     (1 << 7)  
#define FIFO_ENABLE                     (1 << 0)  
#define RBR_IRQ_ENABLE                 (1 << 0)  
#define THRE_IRQ_ENABLE                (1 << 1)  
#define UART_LSR_THRE                  (1 << 5)  
#define RDA_INTERRUPT                 (2 << 1)  
#define CTI_INTERRUPT                  (6 << 1)  
void uart0_init(int baudrate);  
void tx_cadena_UART0(char *ptr);  
unsigned char procesarComando (char* buff);  
#endif /* UART_H_ */
```

```
/*
-----//  
// @filename HTTP_SOURCE.h //  
// @version 0.00  
//  
// @author Alberto Palomo Alonso //  
//  
// @brief Cabecera que configura la página WEB. //  
//  
// @category Opcional.  
//  
// @map @include  
//  
// @funcdef  
//  
// @end  
//  
//  
//  
//  
//-----//  
//  
// @include Estos son los archivos utilizados en el código de //  
// configuración.  
//  
//
```

Alberto Palomo Alonso.

```
/**-----**/  
#ifndef NETCONFIG  
#define NETCONFIG  
#include <Net_Config.h>  
#endif  
  
#ifndef STDIO  
#define STDIO  
#include <stdio.h>  
#endif  
  
#ifndef LPC17XX  
#define LPC17XX  
#include <LPC17XX.H>  
#endif  
  
#ifndef RTL  
#define RTL  
#include <RTL.h>  
#endif  
  
void __configuraWEB__( void );  
void __mantenerTCP__( void );  
  
/**-----**/  
//  
//  
//  
// @end  
//  
//  
//  
//-----**/  
  
/**-----**/
```

```
//          @filename      HTTP_CGI.h
//          @version       0.00
//
//          @author        Alberto Palomo Alonso
//          @brief         Cabecera que configura los callback la página WEB.
//          @category      Opcional.
//
//          @map           @include
//
//          @private
//
//          @funcdef
//
//          @end
//
//          @include
//          -----
//          -----
//          //          @include      Estos son los archivos utilizados en el código de
//          configuración.                                //
//          //
//          //          @include
//          -----
//          -----**/
#ifndef NETCONFIG
```



```
#define ALTITUD      'a'  
#define LONGITUD      'x'  
#define LATITUD      'y'  
#define ANYO          'A'  
#define MES           'M'  
#define DIA           'D'  
#define HORAS         'H'  
#define MINUTOS        'T'  
#define SEGUNDOS       'S'  
  
/*-----  
-----*/  
  
//  
  
//  
  
//-----  
-----**/  
  
//          @funcdef      Estas son las funciones correspondientes al cgi.  
//  
//  
//  
//-----  
-----**/  
  
void cgi_process_var  (      U8*      qs);      //      NO SE USA.  
void cgi_process_data (      U8      tipo ,      U8      *      qs ,  
      U16      longitud);      //      NO SE USA.  
U16 cgi_func          (      U8      *      env ,      U8      *      buff  
      ,      U16      bufflen ,      U32      *      pcgi);  
  
/*-----  
-----*/  
  
//  
  
//  
  
//-----  
-----**/  
  
//          @end      ENDFILE.  
//  
//  
//  
//-----  
-----**/  
  
/*-----  
-----*/
```

Alberto Palomo Alonso.

```
//           @filename          RTC.h
//                                     //
//           @version           0.00
//
//           @author            Alberto Palomo Alonso
//                                     //
//           @brief             Cabecera del código RTC.
//                                     //
//           @category          Opcional.
//
//           @map               @include
//
//           @private
//
//           @funcdef
//
//           @end
//
//           @include
//                                     //
//           @
//           //-----//fuente.          Estos son los archivos utilizados con el código
//                                     //fuente.
//           //
//           @
//           //-----//**/
//           #ifndef SYSTEMSYMBOLS
```

Alberto Palomo Alonso.

Alberto Palomo Alonso.

```
/**-----**/
```

```
/**-----**/
```

```
//           @filename      Timers.h
```

```
//
```

```
//           @version       0.00
```

```
//
```

```
//           @author        Alberto Palomo Alonso
```

```
//
```

```
//
```

```
//           @brief         Cabecera para configurar los timers.
```

```
//
```

```
//
```

```
//           @category     Principal.
```

```
//
```

```
//
```

```
//           @map          @include
```

```
//
```

```
//           @funcdef
```

```
//
```

```
//           @end
```

```
//
```

```
//
```

```
//           @include
```

```
//-----**/
```

```
//
```

```
//           @include
```

```
Estos son los archivos utilizados para los timers.
```

```
//
```

```
//  
//  
//-----**/  
  
#ifndef RTL  
  
#define RTL  
  
#include "RTL.h"  
  
#endif  
  
#ifndef LPC17XX  
  
#define LPC17XX  
  
#include "LPC17XX.H"  
  
#endif  
  
#ifndef SYSTEMSYMBOLS  
  
#define SYSTEMSYMBOLS  
  
#include "Systemsymbols.h"  
  
#endif  
  
#ifndef ANEMOMETRO  
  
#define ANEMOMETRO  
  
#include "Anemometro.h"  
  
#endif  
  
#ifndef LDR  
  
#define LDR  
  
#include "LDR.h"  
  
#endif  
  
#ifndef DAC  
  
#define DAC  
  
#include "DAC.h"  
  
#endif  
  
#ifndef PWM  
  
#define PWM  
  
#include "PWM.h"  
  
#endif  
  
#ifndef UFONO  
  
#define UFONO  
  
#include "uFono.h"  
  
#endif  
  
#ifndef ONEWIRE  
  
#define ONEWIRE  
  
#include "OneWire.h"  
  
#endif
```

Alberto Palomo Alonso.

```
//  
// @funcdef Estas son las funciones correspondientes a los  
timers. //  
  
//  
//  
//-----**/  
void __configuraSysTick__ ( void ); // TCP.  
void __configuraTimer0__ ( void ); // Muestreo.  
void __configuraTimer1__ ( void ); // ...  
void __configuraTimer2__ ( void ); // Audio.  
void __configuraTimer3__ ( void ); // ...  
/*-----**/  
//  
//  
//-----**/  
//  
// @end ENDFILE.  
  
//  
//  
//-----**/  
  
/*****  
* @file core_cm3.h  
* @brief CMSIS Cortex-M3 Core Peripheral Access Layer Header File  
* @version V2.01  
* @date 06. December 2010  
*  
* @note  
* Copyright (C) 2009-2010 ARM Limited. All rights reserved.  
*  
* @par  
* ARM Limited (ARM) is supplying this software for use with Cortex-M  
* processor based microcontrollers. This file can be freely distributed  
* within development tools that are supporting such ARM based processors.
```

```
*  
* @par  
* THIS SOFTWARE IS PROVIDED "AS IS". NO WARRANTIES, WHETHER EXPRESS, IMPLIED  
* OR STATUTORY, INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF  
* MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE APPLY TO THIS SOFTWARE.  
* ARM SHALL NOT, IN ANY CIRCUMSTANCES, BE LIABLE FOR SPECIAL, INCIDENTAL, OR  
* CONSEQUENTIAL DAMAGES, FOR ANY REASON WHATSOEVER.  
*  
*****  
#if defined ( __ICCARM__ )  
#pragma system_include /* treat file as system include file for MISRA check */  
#endif  
  
#ifdef __cplusplus  
extern "C" {  
#endif  
  
#ifndef __CORE_CM3_H_GENERIC  
#define __CORE_CM3_H_GENERIC  
  
*****  
* CMSIS definitions  
*****  
/** @ingroup CMSIS  
* @addtogroup CMSIS_core_definitions CMSIS Core Definitions  
This file defines all structures and symbols for CMSIS core:  
- CMSIS version number  
- Cortex-M core  
- Cortex-M core Revision Number  
@{  
*/  
  
/* CMSIS CM3 definitions */  
  
#define __CM3_CMSIS_VERSION_MAIN (0x02)  
/*!< [31:16] CMSIS HAL main version */  
  
#define __CM3_CMSIS_VERSION_SUB (0x00)  
/*!< [15:0] CMSIS HAL sub version */  
  
#define __CM3_CMSIS_VERSION ((__CM3_CMSIS_VERSION_MAIN << 16) |  
__CM3_CMSIS_VERSION_SUB) /*!< CMSIS HAL version number */
```

```

#define __CORTEX_M          (0x03)
/*!< Cortex core          */

#if defined ( __CC_ARM    )
    #define __ASM           __asm
    for ARM Compiler        /*!< asm keyword
                                */
    #define __INLINE         __inline
    keyword for ARM Compiler /*!< inline

#endif

#elif defined ( __ICCARM__ )
    #define __ASM           __asm
    for IAR Compiler        /*!< asm keyword
                                */
    #define __INLINE         inline
    keyword for IAR Compiler. Only available in High optimization mode! /*!< inline

#endif

#elif defined ( __GNUC__   )
    #define __ASM           __asm
    for GNU Compiler         /*!< asm keyword
                                */
    #define __INLINE         inline
    keyword for GNU Compiler /*!< inline

#endif

#elif defined ( __TASKING__ )
    #define __ASM           __asm
    for TASKING Compiler    /*!< asm keyword
                                */
    #define __INLINE         inline
    keyword for TASKING Compiler /*!< inline

#endif

#endif

#include <stdint.h>           /*!< standard types definitions
                                */
#include "core_cmInstr.h"       /*!< Core Instruction Access
                                */
#include "core_cmFunc.h"        /*!< Core Function Access
                                */

#endif /* __CORE_CM3_H_GENERIC */

#ifndef __CMSIS_GENERIC

#ifndef __CORE_CM3_H_DEPENDANT
#define __CORE_CM3_H_DEPENDANT

```

```
/* IO definitions (access restrictions to peripheral registers) */

#ifndef __cplusplus

#define __I volatile /*!< defines 'read only' permissions
*/
#define __I volatile const /*!< defines 'read only' permissions
*/
#define __O volatile /*!< defines 'write only' permissions
*/
#define __IO volatile /*!< defines 'read / write' permissions
*/

/*}@ end of group CMSIS_core_definitions */

/*****************
 * Register Abstraction
 *****************/
/** @ingroup CMSIS
 * @defgroup CMSIS_core_register CMSIS Core Register
Core Register contain:
- Core Register
- Core NVIC Register
- Core SCB Register
- Core SysTick Register
- Core Debug Register
- Core MPU Register
*/
/** \ingroup CMSIS_core_register
\defgroup CMSIS_CORE CMSIS Core
Type definitions for the Cortex-M Core Registers
@{
*/
/** \brief Union type to access the Application Program Status Register (APSR).
*/
typedef union
```

```

{
    struct
    {
#if (__CORTEX_M != 0x04)
        uint32_t _reserved0:27;           /*!< bit: 0..26 Reserved
*/
#else
        uint32_t _reserved0:16;           /*!< bit: 0..15 Reserved
*/
        uint32_t GE:4;                  /*!< bit: 16..19 Greater than or Equal flags
*/
        uint32_t _reserved1:7;           /*!< bit: 20..26 Reserved
*/
#endif
        uint32_t Q:1;                  /*!< bit: 27 Saturation condition flag
*/
        uint32_t V:1;                  /*!< bit: 28 Overflow condition code flag
*/
        uint32_t C:1;                  /*!< bit: 29 Carry condition code flag
*/
        uint32_t Z:1;                  /*!< bit: 30 Zero condition code flag
*/
        uint32_t N:1;                  /*!< bit: 31 Negative condition code flag
*/
    } b;
    /*!< Structure used for bit access
*/
    uint32_t w;                      /*!< Type used for word access
*/
} APSR_Type;

/** \brief Union type to access the Interrupt Program Status Register (IPSR).
 */
typedef union
{
    struct
    {
        uint32_t ISR:9;                /*!< bit: 0.. 8 Exception number
*/
        uint32_t _reserved0:23;         /*!< bit: 9..31 Reserved
*/
    } b;
    /*!< Structure used for bit access
*/
    uint32_t w;                      /*!< Type used for word access
*/
} IPSR_Type;

```

```
/** \brief Union type to access the Special-Purpose Program Status Registers (xPSR).
 */
typedef union
{
    struct
    {
        uint32_t ISR:9;                                /*!< bit: 0..8 Exception number
    */
#ifndef __CORTEX_M != 0x04
        uint32_t _reserved0:15;                         /*!< bit: 9..23 Reserved
    */
#else
        uint32_t _reserved0:7;                          /*!< bit: 9..15 Reserved
    */
        uint32_t GE:4;                                 /*!< bit: 16..19 Greater than or Equal flags
    */
        uint32_t _reserved1:4;                          /*!< bit: 20..23 Reserved
    */
#endif
        uint32_t T:1;                                 /*!< bit: 24 Thumb bit          (read 0)
    */
        uint32_t IT:2;                               /*!< bit: 25..26 saved IT state (read 0)
    */
        uint32_t Q:1;                                /*!< bit: 27 Saturation condition flag
    */
        uint32_t V:1;                                /*!< bit: 28 Overflow condition code flag
    */
        uint32_t C:1;                                /*!< bit: 29 Carry condition code flag
    */
        uint32_t Z:1;                                /*!< bit: 30 Zero condition code flag
    */
        uint32_t N:1;                                /*!< bit: 31 Negative condition code flag
    */
        } b;                                     /*!< Structure used for bit access
    */
    uint32_t w;                                  /*!< Type      used for word access
}
} xPSR_Type;

/** \brief Union type to access the Control Registers (CONTROL).
 */
typedef union
{
```

```

struct
{
    uint32_t nPRIV:1;                                /*!< bit:      0 Execution privilege in Thread
mode */
    uint32_t SPSEL:1;                                /*!< bit:      1 Stack to be used
*/
    uint32_t FPCA:1;                                /*!< bit:      2 FP extension active flag
*/
    uint32_t _reserved0:29;                            /*!< bit:  3..31 Reserved
*/
} b;                                                 /*!< Structure used for bit access
*/
uint32_t w;                                         /*!< Type      used for word access
*/
} CONTROL_Type;

/*@} end of group CMSIS_CORE */

/** \ingroup CMSIS_core_register
\defgroup CMSIS_NVIC CMSIS NVIC
Type definitions for the Cortex-M NVIC Registers
@{
*/

/** \brief Structure type to access the Nested Vectored Interrupt Controller (NVIC).
*/
typedef struct
{
    __IO uint32_t ISER[8];                           /*!< Offset: 0x000 (R/W)  Interrupt Set Enable
Register           */
    uint32_t RESERVED0[24];
    __IO uint32_t ICER[8];                           /*!< Offset: 0x080 (R/W)  Interrupt Clear
Enable Register   */
    uint32_t RSERVED1[24];
    __IO uint32_t ISPR[8];                           /*!< Offset: 0x100 (R/W)  Interrupt Set Pending
Register           */
    uint32_t RESERVED2[24];
    __IO uint32_t ICPR[8];                           /*!< Offset: 0x180 (R/W)  Interrupt Clear
Pending Register   */
    uint32_t RESERVED3[24];
    __IO uint32_t IABR[8];                           /*!< Offset: 0x200 (R/W)  Interrupt Active bit
Register           */
    uint32_t RESERVED4[56];
}

```

```

    __IO uint8_t  IP[240];           /*!< Offset: 0x300 (R/W)  Interrupt Priority
Register (8Bit wide) */

    uint32_t RESERVED5[644];        /*!< Offset: 0xE00 ( /W)  Software Trigger

    __O uint32_t STIR;             /*!< Offset: 0xE00 ( /W)  Software Trigger
Interrupt Register */

} NVIC_Type;

/*@} end of group CMSIS_NVIC */

/* ** \ingroup CMSIS_core_register
\defgroup CMSIS_SCB CMSIS SCB
Type definitions for the Cortex-M System Control Block Registers
@{
*/

/** \brief Structure type to access the System Control Block (SCB).
*/
typedef struct
{
    __I uint32_t CPUID;           /*!< Offset: 0x000 (R/ )  CPU ID Base Register
 */

    __IO uint32_t ICSR;           /*!< Offset: 0x004 (R/W)  Interrupt Control
State Register */

    __IO uint32_t VTOR;           /*!< Offset: 0x008 (R/W)  Vector Table Offset
Register */

    __IO uint32_t AIRCR;          /*!< Offset: 0x00C (R/W)  Application Interrupt
/ Reset Control Register */

    __IO uint32_t SCR;            /*!< Offset: 0x010 (R/W)  System Control
Register */

    __IO uint32_t CCR;            /*!< Offset: 0x014 (R/W)  Configuration Control
Register */

    __IO uint8_t  SHP[12];         /*!< Offset: 0x018 (R/W)  System Handlers
Priority Registers (4-7, 8-11, 12-15) */

    __IO uint32_t SHCSR;          /*!< Offset: 0x024 (R/W)  System Handler
Control and State Register */

    __IO uint32_t CFSR;           /*!< Offset: 0x028 (R/W)  Configurable Fault
Status Register */

    __IO uint32_t HFSR;           /*!< Offset: 0x02C (R/W)  Hard Fault Status
Register */

    __IO uint32_t DFSR;           /*!< Offset: 0x030 (R/W)  Debug Fault Status
Register */

    __IO uint32_t MMFAR;          /*!< Offset: 0x034 (R/W)  Mem Manage Address
Register */

    __IO uint32_t BFAR;           /*!< Offset: 0x038 (R/W)  Bus Fault Address
Register */
}

```

```

    __IO uint32_t AFSR;           /*!< Offset: 0x03C (R/W) Auxiliary Fault
Status Register               */
    __I  uint32_t PFR[2];        /*!< Offset: 0x040 (R/ ) Processor Feature
Register                      */
    __I  uint32_t DFR;          /*!< Offset: 0x048 (R/ ) Debug Feature
Register                      */
    __I  uint32_t ADR;          /*!< Offset: 0x04C (R/ ) Auxiliary Feature
Register                      */
    __I  uint32_t MMFR[4];       /*!< Offset: 0x050 (R/ ) Memory Model Feature
Register                      */
    __I  uint32_t ISAR[5];       /*!< Offset: 0x060 (R/ ) ISA Feature Register
*/
} SCB_Type;

/* SCB CPUID Register Definitions */

#define SCB_CPUID_IMPLEMENTER_Pos      24
/*!< SCB CPUID: IMPLEMENTER Position */
#define SCB_CPUID_IMPLEMENTER_Msk      (0xFFUL << SCB_CPUID_IMPLEMENTER_Pos)
/*!< SCB CPUID: IMPLEMENTER Mask */

#define SCB_CPUID_VARIANT_Pos         20
/*!< SCB CPUID: VARIANT Position */
#define SCB_CPUID_VARIANT_Msk         (0xFUL << SCB_CPUID_VARIANT_Pos)
/*!< SCB CPUID: VARIANT Mask */

#define SCB_CPUID_PARTNO_Pos          4
/*!< SCB CPUID: PARTNO Position */
#define SCB_CPUID_PARTNO_Msk          (0xFFFFUL << SCB_CPUID_PARTNO_Pos)
/*!< SCB CPUID: PARTNO Mask */

#define SCB_CPUID_REVISION_Pos        0
/*!< SCB CPUID: REVISION Position */
#define SCB_CPUID_REVISION_Msk        (0xFUL << SCB_CPUID_REVISION_Pos)
/*!< SCB CPUID: REVISION Mask */

/* SCB Interrupt Control State Register Definitions */

#define SCB_ICSR_NMIPENDSET_Pos      31
/*!< SCB ICSR: NMIPENDSET Position */
#define SCB_ICSR_NMIPENDSET_Msk      (1UL << SCB_ICSR_NMIPENDSET_Pos)
/*!< SCB ICSR: NMIPENDSET Mask */

#define SCB_ICSR_PENDSVSET_Pos       28
/*!< SCB ICSR: PENDSVSET Position */
#define SCB_ICSR_PENDSVSET_Msk       (1UL << SCB_ICSR_PENDSVSET_Pos)
/*!< SCB ICSR: PENDSVSET Mask */

#define SCB_ICSR_PENDSVCLR_Pos       27
/*!< SCB ICSR: PENDSVCLR Position */

```

```

#define SCB_ICSR_PENDSVCLR_Msk          27
/*!< SCB ICSR: PENDSVCLR Mask */

#define SCB_ICSR_PENDSTSET_Pos          26
/*!< SCB ICSR: PENDSTSET Position */

#define SCB_ICSR_PENDSTSET_Msk          (1UL << SCB_ICSR_PENDSTSET_Pos)
/*!< SCB ICSR: PENDSTSET Mask */

#define SCB_ICSR_PENDSTCLR_Pos          25
/*!< SCB ICSR: PENDSTCLR Position */

#define SCB_ICSR_PENDSTCLR_Msk          (1UL << SCB_ICSR_PENDSTCLR_Pos)
/*!< SCB ICSR: PENDSTCLR Mask */

#define SCB_ICSR_ISRPREEMPT_Pos          23
/*!< SCB ICSR: ISRPREEMPT Position */

#define SCB_ICSR_ISRPREEMPT_Msk          (1UL << SCB_ICSR_ISRPREEMPT_Pos)
/*!< SCB ICSR: ISRPREEMPT Mask */

#define SCB_ICSR_ISRPENDING_Pos          22
/*!< SCB ICSR: ISRPENDING Position */

#define SCB_ICSR_ISRPENDING_Msk          (1UL << SCB_ICSR_ISRPENDING_Pos)
/*!< SCB ICSR: ISRPENDING Mask */

#define SCB_ICSR_VECTPENDING_Pos          12
/*!< SCB ICSR: VECTPENDING Position */

#define SCB_ICSR_VECTPENDING_Msk          (0x1FFUL << SCB_ICSR_VECTPENDING_Pos)
/*!< SCB ICSR: VECTPENDING Mask */

#define SCB_ICSR_RETTOBASE_Pos           11
/*!< SCB ICSR: RETTOBASE Position */

#define SCB_ICSR_RETTOBASE_Msk           (1UL << SCB_ICSR_RETTOBASE_Pos)
/*!< SCB ICSR: RETTOBASE Mask */

#define SCB_ICSR_VECTACTIVE_Pos           0
/*!< SCB ICSR: VECTACTIVE Position */

#define SCB_ICSR_VECTACTIVE_Msk           (0x1FFUL << SCB_ICSR_VECTACTIVE_Pos)
/*!< SCB ICSR: VECTACTIVE Mask */

/* SCB Interrupt Control State Register Definitions */

#define SCB_VTOR_TBLBASE_Pos             29
/*!< SCB VTOR: TBLBASE Position */

#define SCB_VTOR_TBLBASE_Msk             (1UL << SCB_VTOR_TBLBASE_Pos)
/*!< SCB VTOR: TBLBASE Mask */

#define SCB_VTOR_TBLOFF_Pos              7
/*!< SCB VTOR: TBLOFF Position */

#define SCB_VTOR_TBLOFF_Msk              (0x3FFFFFFUL << SCB_VTOR_TBLOFF_Pos)
/*!< SCB VTOR: TBLOFF Mask */

```

```
/* SCB Application Interrupt and Reset Control Register Definitions */

#define SCB_AIRCR_VECTKEY_Pos           16
/*!< SCB AIRCR: VECTKEY Position */

#define SCB_AIRCR_VECTKEY_Msk          (0xFFFFUL << SCB_AIRCR_VECTKEY_Pos)
/*!< SCB AIRCR: VECTKEY Mask */

#define SCB_AIRCR_VECTKEYSTAT_Pos      16
/*!< SCB AIRCR: VECTKEYSTAT Position */

#define SCB_AIRCR_VECTKEYSTAT_Msk      (0xFFFFUL << SCB_AIRCR_VECTKEYSTAT_Pos)
/*!< SCB AIRCR: VECTKEYSTAT Mask */

#define SCB_AIRCR_ENDIANESS_Pos        15
/*!< SCB AIRCR: ENDIANESS Position */

#define SCB_AIRCR_ENDIANESS_Msk        (1UL << SCB_AIRCR_ENDIANESS_Pos)
/*!< SCB AIRCR: ENDIANESS Mask */

#define SCB_AIRCR_PRIGROUP_Pos         8
/*!< SCB AIRCR: PRIGROUP Position */

#define SCB_AIRCR_PRIGROUP_Msk         (7UL << SCB_AIRCR_PRIGROUP_Pos)
/*!< SCB AIRCR: PRIGROUP Mask */

#define SCB_AIRCR_SYSRESETREQ_Pos      2
/*!< SCB AIRCR: SYSRESETREQ Position */

#define SCB_AIRCR_SYSRESETREQ_Msk      (1UL << SCB_AIRCR_SYSRESETREQ_Pos)
/*!< SCB AIRCR: SYSRESETREQ Mask */

#define SCB_AIRCR_VECTCLRACTIVE_Pos    1
/*!< SCB AIRCR: VECTCLRACTIVE Position */

#define SCB_AIRCR_VECTCLRACTIVE_Msk    (1UL << SCB_AIRCR_VECTCLRACTIVE_Pos)
/*!< SCB AIRCR: VECTCLRACTIVE Mask */

#define SCB_AIRCR_VECTRESET_Pos        0
/*!< SCB AIRCR: VECTRESET Position */

#define SCB_AIRCR_VECTRESET_Msk        (1UL << SCB_AIRCR_VECTRESET_Pos)
/*!< SCB AIRCR: VECTRESET Mask */

/* SCB System Control Register Definitions */

#define SCB_SCR_SEVONPEND_Pos          4
/*!< SCB SCR: SEVONPEND Position */

#define SCB_SCR_SEVONPEND_Msk          (1UL << SCB_SCR_SEVONPEND_Pos)
/*!< SCB SCR: SEVONPEND Mask */

#define SCB_SCR_SLEEPDEEP_Pos          2
/*!< SCB SCR: SLEEPDEEP Position */

#define SCB_SCR_SLEEPDEEP_Msk          (1UL << SCB_SCR_SLEEPDEEP_Pos)
/*!< SCB SCR: SLEEPDEEP Mask */
```

```

#define SCB_SCR_SLEEPONEXIT_Pos           1
/*!< SCB SCR: SLEEPONEXIT Position */

#define SCB_SCR_SLEEPONEXIT_Msk          (1UL << SCB_SCR_SLEEPONEXIT_Pos)
/*!< SCB SCR: SLEEPONEXIT Mask */

/* SCB Configuration Control Register Definitions */

#define SCB_CCR_STKALIGN_Pos            9
/*!< SCB CCR: STKALIGN Position */

#define SCB_CCR_STKALIGN_Msk           (1UL << SCB_CCR_STKALIGN_Pos)
/*!< SCB CCR: STKALIGN Mask */

#define SCB_CCR_BFHFNIGN_Pos           8
/*!< SCB CCR: BFHFNMIGN Position */

#define SCB_CCR_BFHFNIGN_Msk          (1UL << SCB_CCR_BFHFNIGN_Pos)
/*!< SCB CCR: BFHFNMIGN Mask */

#define SCB_CCR_DIV_0_TRP_Pos          4
/*!< SCB CCR: DIV_0_TRP Position */

#define SCB_CCR_DIV_0_TRP_Msk         (1UL << SCB_CCR_DIV_0_TRP_Pos)
/*!< SCB CCR: DIV_0_TRP Mask */

#define SCB_CCR_UNALIGN_TRP_Pos        3
/*!< SCB CCR: UNALIGN_TRP Position */

#define SCB_CCR_UNALIGN_TRP_Msk       (1UL << SCB_CCR_UNALIGN_TRP_Pos)
/*!< SCB CCR: UNALIGN_TRP Mask */

#define SCB_CCR_USERSETPEND_Pos        1
/*!< SCB CCR: USERSETPEND Position */

#define SCB_CCR_USERSETPEND_Msk       (1UL << SCB_CCR_USERSETPEND_Pos)
/*!< SCB CCR: USERSETPEND Mask */

#define SCB_CCR_NONBASETHRDENA_Pos     0
/*!< SCB CCR: NONBASETHRDENA Position */

#define SCB_CCR_NONBASETHRDENA_Msk    (1UL << SCB_CCR_NONBASETHRDENA_Pos)
/*!< SCB CCR: NONBASETHRDENA Mask */

/* SCB System Handler Control and State Register Definitions */

#define SCB_SHCSR_USGFAULTENA_Pos      18
/*!< SCB SHCSR: USGFAULTENA Position */

#define SCB_SHCSR_USGFAULTENA_Msk     (1UL << SCB_SHCSR_USGFAULTENA_Pos)
/*!< SCB SHCSR: USGFAULTENA Mask */

#define SCB_SHCSR_BUSFAULTENA_Pos      17
/*!< SCB SHCSR: BUSFAULTENA Position */

#define SCB_SHCSR_BUSFAULTENA_Msk     (1UL << SCB_SHCSR_BUSFAULTENA_Pos)
/*!< SCB SHCSR: BUSFAULTENA Mask */

```

```

#define SCB_SHCSR_MEMFAULTENA_Pos           16
/*!< SCB SHCSR: MEMFAULTENA Position */

#define SCB_SHCSR_MEMFAULTENA_Msk          (1UL << SCB_SHCSR_MEMFAULTENA_Pos)
/*!< SCB SHCSR: MEMFAULTENA Mask */

#define SCB_SHCSR_SVCALLPENDED_Pos         15
/*!< SCB SHCSR: SVCALLPENDED Position */

#define SCB_SHCSR_SVCALLPENDED_Msk         (1UL << SCB_SHCSR_SVCALLPENDED_Pos)
/*!< SCB SHCSR: SVCALLPENDED Mask */

#define SCB_SHCSR_BUSFAULTPENDED_Pos       14
/*!< SCB SHCSR: BUSFAULTPENDED Position */

#define SCB_SHCSR_BUSFAULTPENDED_Msk       (1UL << SCB_SHCSR_BUSFAULTPENDED_Pos)
/*!< SCB SHCSR: BUSFAULTPENDED Mask */

#define SCB_SHCSR_MEMFAULTPENDED_Pos       13
/*!< SCB SHCSR: MEMFAULTPENDED Position */

#define SCB_SHCSR_MEMFAULTPENDED_Msk       (1UL << SCB_SHCSR_MEMFAULTPENDED_Pos)
/*!< SCB SHCSR: MEMFAULTPENDED Mask */

#define SCB_SHCSR_USGFAULTPENDED_Pos       12
/*!< SCB SHCSR: USGFAULTPENDED Position */

#define SCB_SHCSR_USGFAULTPENDED_Msk       (1UL << SCB_SHCSR_USGFAULTPENDED_Pos)
/*!< SCB SHCSR: USGFAULTPENDED Mask */

#define SCB_SHCSR_SYSTICKACT_Pos          11
/*!< SCB SHCSR: SYSTICKACT Position */

#define SCB_SHCSR_SYSTICKACT_Msk          (1UL << SCB_SHCSR_SYSTICKACT_Pos)
/*!< SCB SHCSR: SYSTICKACT Mask */

#define SCB_SHCSR_PENDSVACT_Pos           10
/*!< SCB SHCSR: PENDSVACT Position */

#define SCB_SHCSR_PENDSVACT_Msk          (1UL << SCB_SHCSR_PENDSVACT_Pos)
/*!< SCB SHCSR: PENDSVACT Mask */

#define SCB_SHCSR_MONITORACT_Pos          8
/*!< SCB SHCSR: MONITORACT Position */

#define SCB_SHCSR_MONITORACT_Msk          (1UL << SCB_SHCSR_MONITORACT_Pos)
/*!< SCB SHCSR: MONITORACT Mask */

#define SCB_SHCSR_SVCALLACT_Pos           7
/*!< SCB SHCSR: SVCALLACT Position */

#define SCB_SHCSR_SVCALLACT_Msk          (1UL << SCB_SHCSR_SVCALLACT_Pos)
/*!< SCB SHCSR: SVCALLACT Mask */

#define SCB_SHCSR_USGFAULTACT_Pos         3
/*!< SCB SHCSR: USGFAULTACT Position */

```

```
#define SCB_SHCSR_USGFAULTACT_Msk          (1UL << SCB_SHCSR_USGFAULTACT_Pos)
/*!< SCB SHCSR: USGFAULTACT Mask */

#define SCB_SHCSR_BUSFAULTACT_Pos           1
/*!< SCB SHCSR: BUSFAULTACT Position */

#define SCB_SHCSR_BUSFAULTACT_Msk          (1UL << SCB_SHCSR_BUSFAULTACT_Pos)
/*!< SCB SHCSR: BUSFAULTACT Mask */

#define SCB_SHCSR_MEMFAULTACT_Pos          0
/*!< SCB SHCSR: MEMFAULTACT Position */

#define SCB_SHCSR_MEMFAULTACT_Msk          (1UL << SCB_SHCSR_MEMFAULTACT_Pos)
/*!< SCB SHCSR: MEMFAULTACT Mask */

/* SCB Configurable Fault Status Registers Definitions */

#define SCB_CFSR_USGFAULTSR_Pos            16
/*!< SCB CFSR: Usage Fault Status Register Position */

#define SCB_CFSR_USGFAULTSR_Msk           (0xFFFFUL << SCB_CFSR_USGFAULTSR_Pos)
/*!< SCB CFSR: Usage Fault Status Register Mask */

#define SCB_CFSR_BUSFAULTSR_Pos           8
/*!< SCB CFSR: Bus Fault Status Register Position */

#define SCB_CFSR_BUSFAULTSR_Msk           (0xFFUL << SCB_CFSR_BUSFAULTSR_Pos)
/*!< SCB CFSR: Bus Fault Status Register Mask */

#define SCB_CFSR_MEMFAULTSR_Pos          0
/*!< SCB CFSR: Memory Manage Fault Status Register Position */

#define SCB_CFSR_MEMFAULTSR_Msk           (0xFFUL << SCB_CFSR_MEMFAULTSR_Pos)
/*!< SCB CFSR: Memory Manage Fault Status Register Mask */

/* SCB Hard Fault Status Registers Definitions */

#define SCB_HFSR_DEBUGEVT_Pos             31
/*!< SCB HFSR: DEBUGEVT Position */

#define SCB_HFSR_DEBUGEVT_Msk            (1UL << SCB_HFSR_DEBUGEVT_Pos)
/*!< SCB HFSR: DEBUGEVT Mask */

#define SCB_HFSR_FORCED_Pos              30
/*!< SCB HFSR: FORCED Position */

#define SCB_HFSR_FORCED_Msk              (1UL << SCB_HFSR_FORCED_Pos)
/*!< SCB HFSR: FORCED Mask */

#define SCB_HFSR_VECTTBL_Pos              1
/*!< SCB HFSR: VECTTBL Position */

#define SCB_HFSR_VECTTBL_Msk             (1UL << SCB_HFSR_VECTTBL_Pos)
/*!< SCB HFSR: VECTTBL Mask */

/* SCB Debug Fault Status Register Definitions */
```

```

#define SCB_DFSR_EXTERNAL_Pos           4
/*!< SCB DFSR: EXTERNAL Position */

#define SCB_DFSR_EXTERNAL_Msk          (1UL << SCB_DFSR_EXTERNAL_Pos)
/*!< SCB DFSR: EXTERNAL Mask */

#define SCB_DFSR_VCATCH_Pos            3
/*!< SCB DFSR: VCATCH Position */

#define SCB_DFSR_VCATCH_Msk            (1UL << SCB_DFSR_VCATCH_Pos)
/*!< SCB DFSR: VCATCH Mask */

#define SCB_DFSR_DWTTRAP_Pos           2
/*!< SCB DFSR: DWTTRAP Position */

#define SCB_DFSR_DWTTRAP_Msk          (1UL << SCB_DFSR_DWTTRAP_Pos)
/*!< SCB DFSR: DWTTRAP Mask */

#define SCB_DFSR_BKPT_Pos              1
/*!< SCB DFSR: BKPT Position */

#define SCB_DFSR_BKPT_Msk              (1UL << SCB_DFSR_BKPT_Pos)
/*!< SCB DFSR: BKPT Mask */

#define SCB_DFSR_HALTED_Pos            0
/*!< SCB DFSR: HALTED Position */

#define SCB_DFSR_HALTED_Msk            (1UL << SCB_DFSR_HALTED_Pos)
/*!< SCB DFSR: HALTED Mask */

/*@} end of group CMSIS_SCB */

/** \ingroup CMSIS_core_register
 \defgroup CMSIS_SysTick CMSIS SysTick
 Type definitions for the Cortex-M System Timer Registers
 @{
 */

/** \brief Structure type to access the System Timer (SysTick).
 */
typedef struct
{
    __IO uint32_t CTRL;                      /*!< Offset: 0x000 (R/W) SysTick Control and
                                                Status Register */
    __IO uint32_t LOAD;                       /*!< Offset: 0x004 (R/W) SysTick Reload Value
                                                Register */
    __IO uint32_t VAL;                        /*!< Offset: 0x008 (R/W) SysTick Current Value
                                                Register */
    __I  uint32_t CALIB;                     /*!< Offset: 0x00C (R/ ) SysTick Calibration
                                                Register */
}

```

```
} SysTick_Type;

/* SysTick Control / Status Register Definitions */

#define SysTick_CTRL_COUNTFLAG_Pos      16
/*!< SysTick CTRL: COUNTFLAG Position */

#define SysTick_CTRL_COUNTFLAG_Msk      (1UL << SysTick_CTRL_COUNTFLAG_Pos)
/*!< SysTick CTRL: COUNTFLAG Mask */

#define SysTick_CTRL_CLKSOURCE_Pos      2
/*!< SysTick CTRL: CLKSOURCE Position */

#define SysTick_CTRL_CLKSOURCE_Msk      (1UL << SysTick_CTRL_CLKSOURCE_Pos)
/*!< SysTick CTRL: CLKSOURCE Mask */

#define SysTick_CTRL_TICKINT_Pos        1
/*!< SysTick CTRL: TICKINT Position */

#define SysTick_CTRL_TICKINT_Msk        (1UL << SysTick_CTRL_TICKINT_Pos)
/*!< SysTick CTRL: TICKINT Mask */

#define SysTick_CTRL_ENABLE_Pos         0
/*!< SysTick CTRL: ENABLE Position */

#define SysTick_CTRL_ENABLE_Msk         (1UL << SysTick_CTRL_ENABLE_Pos)
/*!< SysTick CTRL: ENABLE Mask */

/* SysTick Reload Register Definitions */

#define SysTick_LOAD_RELOAD_Pos         0
/*!< SysTick LOAD: RELOAD Position */

#define SysTick_LOAD_RELOAD_Msk         (0xFFFFFFFFUL << SysTick_LOAD_RELOAD_Pos)
/*!< SysTick LOAD: RELOAD Mask */

/* SysTick Current Register Definitions */

#define SysTick_VAL_CURRENT_Pos         0
/*!< SysTick VAL: CURRENT Position */

#define SysTick_VAL_CURRENT_Msk         (0xFFFFFFFFUL << SysTick_VAL_CURRENT_Pos)
/*!< SysTick VAL: CURRENT Mask */

/* SysTick Calibration Register Definitions */

#define SysTick_CALIB_NOREF_Pos         31
/*!< SysTick CALIB: NOREF Position */

#define SysTick_CALIB_NOREF_Msk         (1UL << SysTick_CALIB_NOREF_Pos)
/*!< SysTick CALIB: NOREF Mask */

#define SysTick_CALIB_SKEW_Pos          30
/*!< SysTick CALIB: SKEW Position */

#define SysTick_CALIB_SKEW_Msk          (1UL << SysTick_CALIB_SKEW_Pos)
/*!< SysTick CALIB: SKEW Mask */
```

```

#define SysTick_CALIB_TENMS_Pos          0
/*!< SysTick CALIB: TENMS Position */

#define SysTick_CALIB_TENMS_Msk          (0xFFFFFFFFUL << SysTick_VAL_CURRENT_Pos)
/*!< SysTick CALIB: TENMS Mask */

/*}@} end of group CMSIS_SysTick */

/** \ingroup CMSIS_core_register
 \defgroup CMSIS_ITM CMSIS ITM
 Type definitions for the Cortex-M Instrumentation Trace Macrocell (ITM)

 @{

 */

/** \brief Structure type to access the Instrumentation Trace Macrocell Register (ITM).
 */
typedef struct
{
    __O union
    {
        __O uint8_t      u8;           /*!< Offset: 0x000 ( /W) ITM Stimulus Port 8-bit */
        __O uint16_t     u16;          /*!< Offset: 0x000 ( /W) ITM Stimulus Port 16-bit */
        __O uint32_t     u32;          /*!< Offset: 0x000 ( /W) ITM Stimulus Port 32-bit */
    } PORT [32];                  /*!< Offset: 0x000 ( /W) ITM Stimulus Port Registers */
    uint32_t RESERVED0[864];

    __IO uint32_t TER;           /*!< Offset:             (R/W) ITM Trace Enable Register */
    uint32_t RESERVED1[15];

    __IO uint32_t TPR;           /*!< Offset:             (R/W) ITM Trace Privilege Register */
    uint32_t RESERVED2[15];

    __IO uint32_t TCR;           /*!< Offset:             (R/W) ITM Trace Control Register */
    uint32_t RESERVED3[29];

    __IO uint32_t IWR;           /*!< Offset:             (R/W) ITM Integration Write Register */
    uint32_t RESERVED4[43];
}

```

```

    __IO uint32_t LAR;                      /*!< Offset:      (R/W)  ITM Lock Access
Register                         */
    __IO uint32_t LSR;                      /*!< Offset:      (R/W)  ITM Lock Status
Register                         */
    uint32_t RESERVED5[6];
    __I  uint32_t PID4;                    /*!< Offset:      (R/ )  ITM Peripheral
Identification Register #4 */
    __I  uint32_t PID5;                    /*!< Offset:      (R/ )  ITM Peripheral
Identification Register #5 */
    __I  uint32_t PID6;                    /*!< Offset:      (R/ )  ITM Peripheral
Identification Register #6 */
    __I  uint32_t PID7;                    /*!< Offset:      (R/ )  ITM Peripheral
Identification Register #7 */
    __I  uint32_t PID0;                    /*!< Offset:      (R/ )  ITM Peripheral
Identification Register #0 */
    __I  uint32_t PID1;                    /*!< Offset:      (R/ )  ITM Peripheral
Identification Register #1 */
    __I  uint32_t PID2;                    /*!< Offset:      (R/ )  ITM Peripheral
Identification Register #2 */
    __I  uint32_t PID3;                    /*!< Offset:      (R/ )  ITM Peripheral
Identification Register #3 */
    __I  uint32_t CID0;                    /*!< Offset:      (R/ )  ITM Component
Identification Register #0 */
    __I  uint32_t CID1;                    /*!< Offset:      (R/ )  ITM Component
Identification Register #1 */
    __I  uint32_t CID2;                    /*!< Offset:      (R/ )  ITM Component
Identification Register #2 */
    __I  uint32_t CID3;                    /*!< Offset:      (R/ )  ITM Component
Identification Register #3 */
} ITM_Type;

/* ITM Trace Privilege Register Definitions */
#define ITM_TPR_PRIVMASK_Pos           0
/*!< ITM TPR: PRIVMASK Position */

#define ITM_TPR_PRIVMASK_Msk          (0xFUL << ITM_TPR_PRIVMASK_Pos)
/*!< ITM TPR: PRIVMASK Mask */

/* ITM Trace Control Register Definitions */
#define ITM_TCR_BUSY_Pos              23
/*!< ITM TCR: BUSY Position */

#define ITM_TCR_BUSY_Msk              (1UL << ITM_TCR_BUSY_Pos)
/*!< ITM TCR: BUSY Mask */

#define ITM_TCR_ATBID_Pos             16
/*!< ITM TCR: ATBID Position */

#define ITM_TCR_ATBID_Msk            (0x7FUL << ITM_TCR_ATBID_Pos)
/*!< ITM TCR: ATBID Mask */

```

```

#define ITM_TCR_TSPrescale_Pos          8
/*!< ITM TCR: TSPrescale Position */

#define ITM_TCR_TSPrescale_Msk          (3UL << ITM_TCR_TSPrescale_Pos)
/*!< ITM TCR: TSPrescale Mask */

#define ITM_TCR_SWOENA_Pos             4
/*!< ITM TCR: SWOENA Position */

#define ITM_TCR_SWOENA_Msk             (1UL << ITM_TCR_SWOENA_Pos)
/*!< ITM TCR: SWOENA Mask */

#define ITM_TCR_DWTENA_Pos             3
/*!< ITM TCR: DWTENA Position */

#define ITM_TCR_DWTENA_Msk             (1UL << ITM_TCR_DWTENA_Pos)
/*!< ITM TCR: DWTENA Mask */

#define ITM_TCR_SYNCENA_Pos            2
/*!< ITM TCR: SYNCENA Position */

#define ITM_TCR_SYNCENA_Msk             (1UL << ITM_TCR_SYNCENA_Pos)
/*!< ITM TCR: SYNCENA Mask */

#define ITM_TCR_TSENA_Pos              1
/*!< ITM TCR: TSENA Position */

#define ITM_TCR_TSENA_Msk              (1UL << ITM_TCR_TSENA_Pos)
/*!< ITM TCR: TSENA Mask */

#define ITM_TCR_ITMENA_Pos             0
/*!< ITM TCR: ITM Enable bit Position */

#define ITM_TCR_ITMENA_Msk             (1UL << ITM_TCR_ITMENA_Pos)
/*!< ITM TCR: ITM Enable bit Mask */

/* ITM Integration Write Register Definitions */

#define ITM_IWR_ATVALIDDM_Pos          0
/*!< ITM IWR: ATVALIDDM Position */

#define ITM_IWR_ATVALIDDM_Msk          (1UL << ITM_IWR_ATVALIDDM_Pos)
/*!< ITM IWR: ATVALIDDM Mask */

/* ITM Integration Read Register Definitions */

#define ITM_IRR_ATREADYM_Pos           0
/*!< ITM IRR: ATREADYM Position */

#define ITM_IRR_ATREADYM_Msk           (1UL << ITM_IRR_ATREADYM_Pos)
/*!< ITM IRR: ATREADYM Mask */

/* ITM Integration Mode Control Register Definitions */

#define ITM_IMCR_INTEGRATION_Pos       0
/*!< ITM IMCR: INTEGRATION Position */

#define ITM_IMCR_INTEGRATION_Msk       (1UL << ITM_IMCR_INTEGRATION_Pos)
/*!< ITM IMCR: INTEGRATION Mask */

```

```
/* ITM Lock Status Register Definitions */

#define ITM_LSR_ByteAcc_Pos           2
/*!< ITM LSR: ByteAcc Position */

#define ITM_LSR_ByteAcc_Msk          (1UL << ITM_LSR_ByteAcc_Pos)
/*!< ITM LSR: ByteAcc Mask */

#define ITM_LSR_Access_Pos           1
/*!< ITM LSR: Access Position */

#define ITM_LSR_Access_Msk          (1UL << ITM_LSR_Access_Pos)
/*!< ITM LSR: Access Mask */

#define ITM_LSR_Present_Pos          0
/*!< ITM LSR: Present Position */

#define ITM_LSR_Present_Msk         (1UL << ITM_LSR_Present_Pos)
/*!< ITM LSR: Present Mask */

/*@} */ /* end of group CMSIS_ITM */

/** \ingroup CMSIS_core_register
 \defgroup CMSIS_InterruptType CMSIS Interrupt Type
 Type definitions for the Cortex-M Interrupt Type Register
 @{
 */

/** \brief Structure type to access the Interrupt Type Register.
 */
typedef struct
{
    uint32_t RESERVED0;

    __I uint32_t ICTR;                      /*!< Offset: 0x004 (R/ )  Interrupt Control
                                             Type Register */

    #if ((defined __CM3_REV) && (__CM3_REV >= 0x200))
        __IO uint32_t ACTLR;                  /*!< Offset: 0x008 (R/W)  Auxiliary Control
                                             Register */
    #else
        uint32_t RESERVED1;
    #endif
} InterruptType_Type;

/* Interrupt Controller Type Register Definitions */
```

```
#define IntType_ICTR_INTLINESNUM_Pos 0
/*!< InterruptType ICTR: INTLINESNUM Position */

#define IntType_ICTR_INTLINESNUM_Msk (0x1FUL << IntType_ICTR_INTLINESNUM_Pos)
/*!< InterruptType ICTR: INTLINESNUM Mask */

/* Auxiliary Control Register Definitions */

#define IntType_ACTLR_DISFOLD_Pos 2
/*!< InterruptType ACTLR: DISFOLD Position */

#define IntType_ACTLR_DISFOLD_Msk (1UL << IntType_ACTLR_DISFOLD_Pos)
/*!< InterruptType ACTLR: DISFOLD Mask */

#define IntType_ACTLR_DISDEFWBUF_Pos 1
/*!< InterruptType ACTLR: DISDEFWBUF Position */

#define IntType_ACTLR_DISDEFWBUF_Msk (1UL << IntType_ACTLR_DISDEFWBUF_Pos)
/*!< InterruptType ACTLR: DISDEFWBUF Mask */

#define IntType_ACTLR_DISMCYCINT_Pos 0
/*!< InterruptType ACTLR: DISMCYCINT Position */

#define IntType_ACTLR_DISMCYCINT_Msk (1UL << IntType_ACTLR_DISMCYCINT_Pos)
/*!< InterruptType ACTLR: DISMCYCINT Mask */

/*@} */ /* end of group CMSIS_InterruptType */

#if (_MPU_PRESENT == 1)

/** \ingroup CMSIS_core_register
\defgroup CMSIS_MPUMPU CMSIS MPU

Type definitions for the Cortex-M Memory Protection Unit (MPU)

@{

*/

/** \brief Structure type to access the Memory Protection Unit (MPU).

*/
typedef struct

{
    __I uint32_t TYPE;                                /*!< Offset: 0x000 (R/W) MPU Type Register
*/
    __IO uint32_t CTRL;                               /*!< Offset: 0x004 (R/W) MPU Control Register
*/
    __IO uint32_t RNR;                                /*!< Offset: 0x008 (R/W) MPU Region RNRber
Register */
    __IO uint32_t RBAR;                               /*!< Offset: 0x00C (R/W) MPU Region Base
Address Register */
    __IO uint32_t RASR;                               /*!< Offset: 0x010 (R/W) MPU Region Attribute
and Size Register */
}
```

```

    __IO uint32_t RBAR_A1;           /*!< Offset: 0x014 (R/W) MPU Alias 1 Region
Base Address Register          */

    __IO uint32_t RASR_A1;          /*!< Offset: 0x018 (R/W) MPU Alias 1 Region
Attribute and Size Register */

    __IO uint32_t RBAR_A2;          /*!< Offset: 0x01C (R/W) MPU Alias 2 Region
Base Address Register          */

    __IO uint32_t RASR_A2;          /*!< Offset: 0x020 (R/W) MPU Alias 2 Region
Attribute and Size Register */

    __IO uint32_t RBAR_A3;          /*!< Offset: 0x024 (R/W) MPU Alias 3 Region
Base Address Register          */

    __IO uint32_t RASR_A3;          /*!< Offset: 0x028 (R/W) MPU Alias 3 Region
Attribute and Size Register */

} MPU_Type;

/* MPU Type Register */

#define MPU_TYPE_IREGION_Pos      16
/*!< MPU TYPE: IREGION Position */

#define MPU_TYPE_IREGION_Msk      (0xFFUL << MPU_TYPE_IREGION_Pos)
/*!< MPU TYPE: IREGION Mask */

#define MPU_TYPE_DREGION_Pos      8
/*!< MPU TYPE: DREGION Position */

#define MPU_TYPE_DREGION_Msk      (0xFFUL << MPU_TYPE_DREGION_Pos)
/*!< MPU TYPE: DREGION Mask */

#define MPU_TYPE_SEPARATE_Pos     0
/*!< MPU TYPE: SEPARATE Position */

#define MPU_TYPE_SEPARATE_Msk     (1UL << MPU_TYPE_SEPARATE_Pos)
/*!< MPU TYPE: SEPARATE Mask */

/* MPU Control Register */

#define MPU_CTRL_PRIVDEFENA_Pos   2
/*!< MPU CTRL: PRIVDEFENA Position */

#define MPU_CTRL_PRIVDEFENA_Msk   (1UL << MPU_CTRL_PRIVDEFENA_Pos)
/*!< MPU CTRL: PRIVDEFENA Mask */

#define MPU_CTRL_HFNMIENA_Pos    1
/*!< MPU CTRL: HFNMIENA Position */

#define MPU_CTRL_HFNMIENA_Msk    (1UL << MPU_CTRL_HFNMIENA_Pos)
/*!< MPU CTRL: HFNMIENA Mask */

#define MPU_CTRL_ENABLE_Pos       0
/*!< MPU CTRL: ENABLE Position */

#define MPU_CTRL_ENABLE_Msk       (1UL << MPU_CTRL_ENABLE_Pos)
/*!< MPU CTRL: ENABLE Mask */

/* MPU Region Number Register */

```

```

#define MPU_RNR_REGION_Pos          0
/*!< MPU RNR: REGION Position */

#define MPU_RNR_REGION_Msk          (0xFFUL << MPU_RNR_REGION_Pos)
/*!< MPU RNR: REGION Mask */

/* MPU Region Base Address Register */

#define MPU_RBAR_ADDR_Pos           5
/*!< MPU RBAR: ADDR Position */

#define MPU_RBAR_ADDR_Msk           (0x7FFFFFFFUL << MPU_RBAR_ADDR_Pos)
/*!< MPU RBAR: ADDR Mask */

#define MPU_RBAR_VALID_Pos          4
/*!< MPU RBAR: VALID Position */

#define MPU_RBAR_VALID_Msk           (1UL << MPU_RBAR_VALID_Pos)
/*!< MPU RBAR: VALID Mask */

#define MPU_RBAR_REGION_Pos          0
/*!< MPU RBAR: REGION Position */

#define MPU_RBAR_REGION_Msk          (0xFUL << MPU_RBAR_REGION_Pos)
/*!< MPU RBAR: REGION Mask */

/* MPU Region Attribute and Size Register */

#define MPU_RASR_XN_Pos              28
/*!< MPU RASR: XN Position */

#define MPU_RASR_XN_Msk              (1UL << MPU_RASR_XN_Pos)
/*!< MPU RASR: XN Mask */

#define MPU_RASR_AP_Pos              24
/*!< MPU RASR: AP Position */

#define MPU_RASR_AP_Msk              (7UL << MPU_RASR_AP_Pos)
/*!< MPU RASR: AP Mask */

#define MPU_RASR_TEX_Pos              19
/*!< MPU RASR: TEX Position */

#define MPU_RASR_TEX_Msk              (7UL << MPU_RASR_TEX_Pos)
/*!< MPU RASR: TEX Mask */

#define MPU_RASR_S_Pos                18
/*!< MPU RASR: Shareable bit Position */

#define MPU_RASR_S_Msk                (1UL << MPU_RASR_S_Pos)
/*!< MPU RASR: Shareable bit Mask */

#define MPU_RASR_C_Pos                17
/*!< MPU RASR: Cacheable bit Position */

#define MPU_RASR_C_Msk                (1UL << MPU_RASR_C_Pos)
/*!< MPU RASR: Cacheable bit Mask */

```

```

#define MPU_RASR_B_Pos           16
/*!< MPU RASR: Bufferable bit Position */

#define MPU_RASR_B_Msk          (1UL << MPU_RASR_B_Pos)
/*!< MPU RASR: Bufferable bit Mask */

#define MPU_RASR_SRD_Pos         8
/*!< MPU RASR: Sub-Region Disable Position */

#define MPU_RASR_SRD_Msk        (0xFFUL << MPU_RASR_SRD_Pos)
/*!< MPU RASR: Sub-Region Disable Mask */

#define MPU_RASR_SIZE_Pos        1
/*!< MPU RASR: Region Size Field Position */

#define MPU_RASR_SIZE_Msk       (0x1FUL << MPU_RASR_SIZE_Pos)
/*!< MPU RASR: Region Size Field Mask */

#define MPU_RASR_ENA_Pos         0
/*!< MPU RASR: Region enable bit Position */

#define MPU_RASR_ENA_Msk        (0x1UL << MPU_RASR_ENA_Pos)
/*!< MPU RASR: Region enable bit Disable Mask */

/*@} end of group CMSIS_MPU */

#endif

/** \ingroup CMSIS_core_register
 \defgroup CMSIS_CoreDebug CMSIS Core Debug
 Type definitions for the Cortex-M Core Debug Registers
 @{
 */

/** \brief Structure type to access the Core Debug Register (CoreDebug).
 */
typedef struct
{
    __IO uint32_t DHCSR;           /*!< Offset: 0x000 (R/W)  Debug Halting Control
and Status Register */
    __IO uint32_t DCRSR;           /*!< Offset: 0x004 ( /W)  Debug Core Register
Selector Register */
    __IO uint32_t DCRDR;           /*!< Offset: 0x008 (R/W)  Debug Core Register
Data Register */
    __IO uint32_t DEMCR;           /*!< Offset: 0x00C (R/W)  Debug Exception and
Monitor Control Register */
} CoreDebug_Type;

/* Debug Halting Control and Status Register */

```

```
#define CoreDebug_DHCSR_DBGKEY_Pos      16
/*!< CoreDebug DHCSR: DBGKEY Position */

#define CoreDebug_DHCSR_DBGKEY_Msk        (0xFFFFUL << CoreDebug_DHCSR_DBGKEY_Pos)
/*!< CoreDebug DHCSR: DBGKEY Mask */

#define CoreDebug_DHCSR_S_RESET_ST_Pos    25
/*!< CoreDebug DHCSR: S_RESET_ST Position */

#define CoreDebug_DHCSR_S_RESET_ST_Msk    (1UL << CoreDebug_DHCSR_S_RESET_ST_Pos)
/*!< CoreDebug DHCSR: S_RESET_ST Mask */

#define CoreDebug_DHCSR_S_RETIRE_ST_Pos   24
/*!< CoreDebug DHCSR: S_RETIRE_ST Position */

#define CoreDebug_DHCSR_S_RETIRE_ST_Msk   (1UL << CoreDebug_DHCSR_S_RETIRE_ST_Pos)
/*!< CoreDebug DHCSR: S_RETIRE_ST Mask */

#define CoreDebug_DHCSR_S_LOCKUP_Pos      19
/*!< CoreDebug DHCSR: S_LOCKUP Position */

#define CoreDebug_DHCSR_S_LOCKUP_Msk      (1UL << CoreDebug_DHCSR_S_LOCKUP_Pos)
/*!< CoreDebug DHCSR: S_LOCKUP Mask */

#define CoreDebug_DHCSR_S_SLEEP_Pos       18
/*!< CoreDebug DHCSR: S_SLEEP Position */

#define CoreDebug_DHCSR_S_SLEEP_Msk       (1UL << CoreDebug_DHCSR_S_SLEEP_Pos)
/*!< CoreDebug DHCSR: S_SLEEP Mask */

#define CoreDebug_DHCSR_S_HALT_Pos        17
/*!< CoreDebug DHCSR: S_HALT Position */

#define CoreDebug_DHCSR_S_HALT_Msk        (1UL << CoreDebug_DHCSR_S_HALT_Pos)
/*!< CoreDebug DHCSR: S_HALT Mask */

#define CoreDebug_DHCSR_S_REGRDY_Pos      16
/*!< CoreDebug DHCSR: S_REGRDY Position */

#define CoreDebug_DHCSR_S_REGRDY_Msk      (1UL << CoreDebug_DHCSR_S_REGRDY_Pos)
/*!< CoreDebug DHCSR: S_REGRDY Mask */

#define CoreDebug_DHCSR_C_SNAPSTALL_Pos   5
/*!< CoreDebug DHCSR: C_SNAPSTALL Position */

#define CoreDebug_DHCSR_C_SNAPSTALL_Msk   (1UL << CoreDebug_DHCSR_C_SNAPSTALL_Pos)
/*!< CoreDebug DHCSR: C_SNAPSTALL Mask */

#define CoreDebug_DHCSR_C_MASKINTS_Pos    3
/*!< CoreDebug DHCSR: C_MASKINTS Position */

#define CoreDebug_DHCSR_C_MASKINTS_Msk    (1UL << CoreDebug_DHCSR_C_MASKINTS_Pos)
/*!< CoreDebug DHCSR: C_MASKINTS Mask */

#define CoreDebug_DHCSR_C_STEP_Pos        2
/*!< CoreDebug DHCSR: C_STEP Position */
```

```

#define CoreDebug_DHCSR_C_STEP_Msk          (1UL << CoreDebug_DHCSR_C_STEP_Pos)
/*!< CoreDebug DHCSR: C_STEP Mask */

#define CoreDebug_DHCSR_C_HALT_Pos          1
/*!< CoreDebug DHCSR: C_HALT Position */

#define CoreDebug_DHCSR_C_HALT_Msk          (1UL << CoreDebug_DHCSR_C_HALT_Pos)
/*!< CoreDebug DHCSR: C_HALT Mask */

#define CoreDebug_DHCSR_C_DEBUGEN_Pos        0
/*!< CoreDebug DHCSR: C_DEBUGEN Position */

#define CoreDebug_DHCSR_C_DEBUGEN_Msk        (1UL << CoreDebug_DHCSR_C_DEBUGEN_Pos)
/*!< CoreDebug DHCSR: C_DEBUGEN Mask */

/* Debug Core Register Selector Register */

#define CoreDebug_DCRSR_REGWnR_Pos          16
/*!< CoreDebug DCRSR: REGWnR Position */

#define CoreDebug_DCRSR_REGWnR_Msk          (1UL << CoreDebug_DCRSR_REGWnR_Pos)
/*!< CoreDebug DCRSR: REGWnR Mask */

#define CoreDebug_DCRSR_REGSEL_Pos          0
/*!< CoreDebug DCRSR: REGSEL Position */

#define CoreDebug_DCRSR_REGSEL_Msk          (0x1FUL << CoreDebug_DCRSR_REGSEL_Pos)
/*!< CoreDebug DCRSR: REGSEL Mask */

/* Debug Exception and Monitor Control Register */

#define CoreDebug_DEMCR_TRCENA_Pos          24
/*!< CoreDebug DEMCR: TRCENA Position */

#define CoreDebug_DEMCR_TRCENA_Msk          (1UL << CoreDebug_DEMCR_TRCENA_Pos)
/*!< CoreDebug DEMCR: TRCENA Mask */

#define CoreDebug_DEMCR_MON_REQ_Pos         19
/*!< CoreDebug DEMCR: MON_REQ Position */

#define CoreDebug_DEMCR_MON_REQ_Msk         (1UL << CoreDebug_DEMCR_MON_REQ_Pos)
/*!< CoreDebug DEMCR: MON_REQ Mask */

#define CoreDebug_DEMCR_MON_STEP_Pos        18
/*!< CoreDebug DEMCR: MON_STEP Position */

#define CoreDebug_DEMCR_MON_STEP_Msk        (1UL << CoreDebug_DEMCR_MON_STEP_Pos)
/*!< CoreDebug DEMCR: MON_STEP Mask */

#define CoreDebug_DEMCR_MON_PEND_Pos        17
/*!< CoreDebug DEMCR: MON_PEND Position */

#define CoreDebug_DEMCR_MON_PEND_Msk        (1UL << CoreDebug_DEMCR_MON_PEND_Pos)
/*!< CoreDebug DEMCR: MON_PEND Mask */

#define CoreDebug_DEMCR_MON_EN_Pos          16
/*!< CoreDebug DEMCR: MON_EN Position */

```

```

#define CoreDebug_DEMCR_MON_EN_Msk          (1UL << CoreDebug_DEMCR_MON_EN_Pos)
/*!< CoreDebug DEMCR: MON_EN Mask */

#define CoreDebug_DEMCR_VC_HARDERR_Pos      10
/*!< CoreDebug DEMCR: VC_HARDERR Position */

#define CoreDebug_DEMCR_VC_HARDERR_Msk      (1UL << CoreDebug_DEMCR_VC_HARDERR_Pos)
/*!< CoreDebug DEMCR: VC_HARDERR Mask */

#define CoreDebug_DEMCR_VC_INTERR_Pos       9
/*!< CoreDebug DEMCR: VC_INTERR Position */

#define CoreDebug_DEMCR_VC_INTERR_Msk       (1UL << CoreDebug_DEMCR_VC_INTERR_Pos)
/*!< CoreDebug DEMCR: VC_INTERR Mask */

#define CoreDebug_DEMCR_VC_BUSERR_Pos       8
/*!< CoreDebug DEMCR: VC_BUSERR Position */

#define CoreDebug_DEMCR_VC_BUSERR_Msk       (1UL << CoreDebug_DEMCR_VC_BUSERR_Pos)
/*!< CoreDebug DEMCR: VC_BUSERR Mask */

#define CoreDebug_DEMCR_VC_STATERR_Pos      7
/*!< CoreDebug DEMCR: VC_STATERR Position */

#define CoreDebug_DEMCR_VC_STATERR_Msk      (1UL << CoreDebug_DEMCR_VC_STATERR_Pos)
/*!< CoreDebug DEMCR: VC_STATERR Mask */

#define CoreDebug_DEMCR_VC_CHKERR_Pos       6
/*!< CoreDebug DEMCR: VC_CHKERR Position */

#define CoreDebug_DEMCR_VC_CHKERR_Msk       (1UL << CoreDebug_DEMCR_VC_CHKERR_Pos)
/*!< CoreDebug DEMCR: VC_CHKERR Mask */

#define CoreDebug_DEMCR_VC_NOCPERR_Pos      5
/*!< CoreDebug DEMCR: VC_NOCPERR Position */

#define CoreDebug_DEMCR_VC_NOCPERR_Msk      (1UL << CoreDebug_DEMCR_VC_NOCPERR_Pos)
/*!< CoreDebug DEMCR: VC_NOCPERR Mask */

#define CoreDebug_DEMCR_VC_MMERR_Pos        4
/*!< CoreDebug DEMCR: VC_MMERR Position */

#define CoreDebug_DEMCR_VC_MMERR_Msk        (1UL << CoreDebug_DEMCR_VC_MMERR_Pos)
/*!< CoreDebug DEMCR: VC_MMERR Mask */

#define CoreDebug_DEMCR_VC_CORERESET_Pos    0
/*!< CoreDebug DEMCR: VC_CORERESET Position */

#define CoreDebug_DEMCR_VC_CORERESET_Msk    (1UL << CoreDebug_DEMCR_VC_CORERESET_Pos)
/*!< CoreDebug DEMCR: VC_CORERESET Mask */

/*@} end of group CMSIS_CoreDebug */

/** \ingroup CMSIS_core_register

```

```

{@{
 */

/* Memory mapping of Cortex-M3 Hardware */

#define SCS_BASE          (0xE000E000UL)           /*!< System
Control Space Base Address */

#define ITM_BASE          (0xE0000000UL)           /*!< ITM Base
Address

#define CoreDebug_BASE    (0xE000EDFOUL)           /*!< Core Debug
Base Address

#define SysTick_BASE      (SCS_BASE + 0x0010UL)        /*!< SysTick Base
Address

#define NVIC_BASE         (SCS_BASE + 0x0100UL)        /*!< NVIC Base
Address

#define SCB_BASE          (SCS_BASE + 0x0D00UL)        /*!< System
Control Block Base Address

#define InterruptType_Type ((InterruptType_Type *) SCS_BASE) /*!< Interrupt
Register

#define SCB_configuration_struct ((SCB_Type *) SCB_BASE)    /*!< SCB
configuration struct

#define SysTick_configuration_struct ((SysTick_Type *) SysTick_BASE) /*!< SysTick
configuration struct

#define NVIC_configuration_struct ((NVIC_Type *) NVIC_BASE)    /*!< NVIC
configuration struct

#define ITM_configuration_struct ((ITM_Type *) ITM_BASE)      /*!< ITM
configuration struct

#define CoreDebug_configuration_struct ((CoreDebug_Type *) CoreDebug_BASE) /*!< Core Debug
configuration struct

#endif

#if (__MPU_PRESENT == 1)

#define MPU_BASE          (SCS_BASE + 0x0D90UL)        /*!< Memory
Protection Unit

#define MPU_Protection_Unit ((MPU_Type *) MPU_BASE)      /*!< Memory
Protection Unit

#endif

/*@} */

*****  

*           Hardware Abstraction Layer  

*****  

/** \ingroup CMSIS  

\addtogroup CMSIS_Core_FunctionInterface CMSIS Core Function Interface

```

```

Core Function Interface contains:

- Core NVIC Functions
- Core SysTick Functions
- Core Debug Functions
- Core Register Access Functions

*/
/* ##### CMSIS_Core_NVICFunctions CMSIS Core NVIC Functions
/** \ingroup CMSIS_Core_FunctionInterface
\defgroup CMSIS_Core_NVICFunctions CMSIS Core NVIC Functions
@{
*/
/** @addtogroup CMSIS_Core_NVICFunctions
* @{
*/
/** \brief Set Priority Grouping

This function sets the priority grouping field using the required unlock sequence.
The parameter PriorityGroup is assigned to the field SCB->AIRCR [10:8] PRIGROUP field.
Only values from 0..7 are used.
In case of a conflict between priority grouping and available
priority bits (__NVIC_PRIO_BITS) the smallest possible priority group is set.

\param [in] PriorityGroup Priority grouping field
*/
static __INLINE void NVIC_SetPriorityGrouping(uint32_t PriorityGroup)
{
    uint32_t reg_value;
    uint32_t PriorityGroupTmp = (PriorityGroup & 0x07); /* only
values 0..7 are used */
    reg_value = SCB->AIRCR; /* read
old register configuration */
    reg_value &= ~(SCB_AIRCR_VECTKEY_Msk | SCB_AIRCR_PRIGROUP_Msk); /* clear
bits to change */
    reg_value = (reg_value |
        (0x5FA << SCB_AIRCR_VECTKEY_Pos) |
        (PriorityGroupTmp << 8)); /* Insert
write key and priority group */
}

```

Alberto Palomo Alonso.

```
SCB->AIRCR = reg_value;
}

/** \brief Get Priority Grouping

This function gets the priority grouping from NVIC Interrupt Controller.
Priority grouping is SCB->AIRCR [10:8] PRIGROUP field.

\return Priority grouping field
*/
static __INLINE uint32_t NVIC_GetPriorityGrouping(void)
{
    return ((SCB->AIRCR & SCB_AIRCR_PRIGROUP_Msk) >> SCB_AIRCR_PRIGROUP_Pos); /* read
priority grouping field */
}

/** \brief Enable External Interrupt

This function enables a device specific interrupt in the NVIC interrupt controller.
The interrupt number cannot be a negative value.

\param [in] IRQn Number of the external interrupt to enable
*/
static __INLINE void NVIC_EnableIRQ(IRQn_Type IRQn)
{
    NVIC->ISER[((uint32_t)(IRQn) >> 5)] = (1 << ((uint32_t)(IRQn) & 0x1F)); /* enable
interrupt */
}

/** \brief Disable External Interrupt

This function disables a device specific interrupt in the NVIC interrupt controller.
The interrupt number cannot be a negative value.

\param [in] IRQn Number of the external interrupt to disable
*/
static __INLINE void NVIC_DisableIRQ(IRQn_Type IRQn)
```

```
{  
    NVIC->ICER[((uint32_t)(IRQn) >> 5)] = (1 << ((uint32_t)(IRQn) & 0x1F)); /* disable  
interrupt */  
}
```

```
/** \brief Get Pending Interrupt
```

This function reads the pending register in the NVIC and returns the pending bit for the specified interrupt.

```
\param [in]      IRQn  Number of the interrupt for get pending  
\return         0  Interrupt status is not pending  
\return         1  Interrupt status is pending  
*/  
static __INLINE uint32_t NVIC_GetPendingIRQ(IRQn_Type IRQn)  
{  
    return((uint32_t) ((NVIC->ISPR[(uint32_t)(IRQn) >> 5] & (1 << ((uint32_t)(IRQn) &  
0x1F))))?1:0); /* Return 1 if pending else 0 */  
}
```

```
/** \brief Set Pending Interrupt
```

This function sets the pending bit for the specified interrupt.

The interrupt number cannot be a negative value.

```
\param [in]      IRQn  Number of the interrupt for set pending  
*/  
static __INLINE void NVIC_SetPendingIRQ(IRQn_Type IRQn)  
{  
    NVIC->ISPR[((uint32_t)(IRQn) >> 5)] = (1 << ((uint32_t)(IRQn) & 0x1F)); /* set  
interrupt pending */  
}
```

```
/** \brief Clear Pending Interrupt
```

This function clears the pending bit for the specified interrupt.

The interrupt number cannot be a negative value.

```
\param [in]      IRQn  Number of the interrupt for clear pending
*/
static __INLINE void NVIC_ClearPendingIRQ(IRQn_Type IRQn)
{
    NVIC->ICPR[((uint32_t)(IRQn) >> 5)] = (1 << ((uint32_t)(IRQn) & 0x1F)); /* Clear
pending interrupt */
}

/** \brief Get Active Interrupt

This function reads the active register in NVIC and returns the active bit.

\param [in]      IRQn  Number of the interrupt for get active
\return          0  Interrupt status is not active
\return          1  Interrupt status is active
*/
static __INLINE uint32_t NVIC_GetActive(IRQn_Type IRQn)
{
    return((uint32_t)((NVIC->IABR[(uint32_t)(IRQn) >> 5] & (1 << ((uint32_t)(IRQn) &
0x1F)))?1:0)); /* Return 1 if active else 0 */
}

/** \brief Set Interrupt Priority

This function sets the priority for the specified interrupt. The interrupt
number can be positive to specify an external (device specific)
interrupt, or negative to specify an internal (core) interrupt.

Note: The priority cannot be set for every core interrupt.

\param [in]      IRQn  Number of the interrupt for set priority
\param [in]      priority  Priority to set
*/
static __INLINE void NVIC_SetPriority(IRQn_Type IRQn, uint32_t priority)
{
    if(IRQn < 0) {
        SCB->SHP[((uint32_t)(IRQn) & 0xF)-4] = ((priority << (8 - __NVIC_PRIO_BITS)) &
0xff); } /* set Priority for Cortex-M System Interrupts */
    else {

```

```
NVIC->IP[(uint32_t)(IRQn)] = ((priority << (8 - __NVIC_PRIO_BITS)) & 0xff);      }
/* set Priority for device specific Interrupts */
}
```

```
/** \brief Get Interrupt Priority
```

This function reads the priority for the specified interrupt. The interrupt number can be positive to specify an external (device specific) interrupt, or negative to specify an internal (core) interrupt.

The returned priority value is automatically aligned to the implemented priority bits of the microcontroller.

```
\param [in]    IRQn  Number of the interrupt for get priority
\return         Interrupt Priority
*/
static __INLINE uint32_t NVIC_GetPriority(IRQn_Type IRQn)
{
    if(IRQn < 0) {
        return((uint32_t)(SCB->SHP[((uint32_t)(IRQn) & 0xF)-4] >> (8 - __NVIC_PRIO_BITS)));
    /* get priority for Cortex-M system interrupts */
    } else {
        return((uint32_t)(NVIC->IP[(uint32_t)(IRQn)]           >> (8 - __NVIC_PRIO_BITS)));
    /* get priority for device specific interrupts */
    }
}
```

```
/** \brief Encode Priority
```

This function encodes the priority for an interrupt with the given priority group, preemptive priority value and sub priority value.
In case of a conflict between priority grouping and available priority bits (__NVIC_PRIO_BITS) the samllest possible priority group is set.

The returned priority value can be used for `NVIC_SetPriority(...)` function

```
\param [in]      PriorityGroup  Used priority group
\param [in]      PreemptPriority  Preemptive priority value (starting from 0)
\param [in]      SubPriority    Sub priority value (starting from 0)
```

```
\return          Encoded priority for the interrupt
*/
static __INLINE uint32_t NVIC_EncodePriority (uint32_t PriorityGroup, uint32_t
PreemptPriority, uint32_t SubPriority)
{
    uint32_t PriorityGroupTmp = (PriorityGroup & 0x07);           /* only values 0..7 are
used           */
    uint32_t PreemptPriorityBits;
    uint32_t SubPriorityBits;

    PreemptPriorityBits = ((7 - PriorityGroupTmp) > __NVIC_PRIO_BITS) ? __NVIC_PRIO_BITS :
7 - PriorityGroupTmp;
    SubPriorityBits      = ((PriorityGroupTmp + __NVIC_PRIO_BITS) < 7) ? 0 :
PriorityGroupTmp - 7 + __NVIC_PRIO_BITS;

    return (
        ((PreemptPriority & ((1 << (PreemptPriorityBits)) - 1)) << SubPriorityBits) |
        ((SubPriority      & ((1 << (SubPriorityBits      )) - 1)))
    );
}
```

/** \brief Decode Priority

This function decodes an interrupt priority value with the given priority group to preemptive priority value and sub priority value.
In case of a conflict between priority grouping and available priority bits (__NVIC_PRIO_BITS) the samllest possible priority group is set.

The priority value can be retrieved with NVIC_GetPriority(...) function

```
\param [in]      Priority      Priority value
\param [in]      PriorityGroup  Used priority group
\param [out]     pPreemptPriority  Preemptive priority value (starting from 0)
\param [out]     pSubPriority   Sub priority value (starting from 0)

*/
static __INLINE void NVIC_DecodePriority (uint32_t Priority, uint32_t PriorityGroup,
uint32_t* pPreemptPriority, uint32_t* pSubPriority)
{
    uint32_t PriorityGroupTmp = (PriorityGroup & 0x07);           /* only values 0..7 are
used           */
    uint32_t PreemptPriorityBits;
```

```

        uint32_t SubPriorityBits;

        PreemptPriorityBits = ((7 - PriorityGroupTmp) > __NVIC_PRIO_BITS) ? __NVIC_PRIO_BITS :
7 - PriorityGroupTmp;

        SubPriorityBits      = ((PriorityGroupTmp + __NVIC_PRIO_BITS) < 7) ? 0 :
PriorityGroupTmp - 7 + __NVIC_PRIO_BITS;

        *pPreemptPriority = (Priority >> SubPriorityBits) & ((1 << (PreemptPriorityBits)) -
1);

        *pSubPriority      = (Priority
                                ) & ((1 << (SubPriorityBits
                                )) - 1);

    }

/** \brief System Reset

    This function initiate a system reset request to reset the MCU.

*/
static __INLINE void NVIC_SystemReset(void)
{
    __DSB();                                     /* Ensure all outstanding
memory accesses included
                                         buffered write are
completed before reset */

    SCB->AIRCR = ((0x5FA << SCB_AIRCR_VECTKEY_Pos) |
                    (SCB->AIRCR & SCB_AIRCR_PRIGROUP_Msk) |
                    SCB_AIRCR_SYSRESETREQ_Msk);           /* Keep priority group
unchanged */

    __DSB();                                     /* Ensure completion of
memory access */

    while(1);                                    /* wait until reset */
}

/*@} end of CMSIS_Core_NVICFunctions */

/**
 * @}
 */

/* ##### SysTick function
#####
*/
/** \ingroup CMSIS_Core_FunctionInterface
\defgroup CMSIS_Core_SysTickFunctions CMSIS Core SysTick Functions

```

```

{@{
*/
/** @addtogroup CMSIS_Core_SysTickFunctions
* @{
*/
#if (_Vendor_SysTickConfig == 0)

/** \brief System Tick Configuration

This function initialises the system tick timer and its interrupt and start the
system tick timer.

Counter is in free running mode to generate periodical interrupts.

\param [in] ticks Number of ticks between two interrupts
\return 0 Function succeeded
\return 1 Function failed
*/
static __INLINE uint32_t SysTick_Config(uint32_t ticks)
{
    if (ticks > SysTick_LOAD_RELOAD_Msk)  return (1);           /* Reload value
impossible */

    SysTick->LOAD  = (ticks & SysTick_LOAD_RELOAD_Msk) - 1;      /* set reload register */
    NVIC_SetPriority (SysTick_IRQn, (1<<__NVIC_PRIO_BITS) - 1); /* set Priority for
Cortex-M0 System Interrupts */

    SysTick->VAL   = 0;                                         /* Load the SysTick
Counter Value */

    SysTick->CTRL  = SysTick_CTRL_CLKSOURCE_Msk |
                      SysTick_CTRL_TICKINT_Msk   |
                      SysTick_CTRL_ENABLE_Msk; /* Enable SysTick IRQ and
SysTick Timer */

    return (0);                                                 /* Function successful */
}

#endif

/** @}
*/

```

```
/* ##### Debug In/Output function
#####
/** \ingroup CMSIS_Core_FunctionInterface
 \defgroup CMSIS_core_DebugFunctions CMSIS Core Debug Functions
 @{
 */
```

```
extern volatile int32_t ITM_RxBuffer; /*!< external variable to
receive characters */
#define ITM_RXBUFFER_EMPTY 0x5AA55AA5 /*!< value identifying
ITM_RxBuffer is ready for next character */
```

```
/** \brief ITM Send Character
```

This function transmits a character via the ITM channel 0.
It just returns when no debugger is connected that has booked the output.
It is blocking when a debugger is connected, but the previous character send is not transmitted.

```
\param [in]      ch  Character to transmit
\return         Character to transmit
*/
static __INLINE uint32_t ITM_SendChar (uint32_t ch)
{
    if ((CoreDebug->DEMCR & CoreDebug_DEMCR_TRCENA_Msk) && /* Trace enabled */
        (ITM->TCR & ITM_TCR_ITMENA_Msk) && /* ITM enabled */
        (ITM->TER & (1UL << 0))) /* ITM Port #0 enabled */
    {
        while (ITM->PORT[0].u32 == 0);
        ITM->PORT[0].u8 = (uint8_t) ch;
    }
    return (ch);
}
```

```
/** \brief ITM Receive Character
```

This function inputs a character via external variable ITM_RxBuffer.
It just returns when no debugger is connected that has booked the output.

It is blocking when a debugger is connected, but the previous character send is not transmitted.

```
\return      Received character
\return      -1  No character received
*/
static __INLINE int32_t ITM_ReceiveChar (void) {
    int32_t ch = -1;                      /* no character available */

    if (ITM_RxBuffer != ITM_RXBUFFER_EMPTY) {
        ch = ITM_RxBuffer;
        ITM_RxBuffer = ITM_RXBUFFER_EMPTY; /* ready for next character */
    }

    return (ch);
}
```

/** \brief ITM Check Character

This function checks external variable ITM_RxBuffer whether a character is available or not.

It returns '1' if a character is available and '0' if no character is available.

```
\return      0  No character available
\return      1  Character available
*/
static __INLINE int32_t ITM_CheckChar (void) {

    if (ITM_RxBuffer == ITM_RXBUFFER_EMPTY) {
        return (0);                      /* no character available */
    } else {
        return (1);                      /* character available */
    }
}

/*@} end of CMSIS_core_DebugFunctions */
/***
 * @}
 */

```

```
#endif /* __CORE_CM3_H_DEPENDANT */  
  
#endif /* __CMSIS_GENERIC */  
  
#ifdef __cplusplus  
}  
#endif  
  
/*lint -restore */  
  
/****** */  
 * @file      LPC17xx.h  
 * @brief     CMSIS Cortex-M3 Core Peripheral Access Layer Header File for  
 *             NXP LPC17xx Device Series  
 * @version:  V1.09  
 * @date:     25. July. 2011  
 *  
 * @note  
 * Copyright (C) 2009 ARM Limited. All rights reserved.  
 *  
 * @par  
 * ARM Limited (ARM) is supplying this software for use with Cortex-M  
 * processor based microcontrollers. This file can be freely distributed  
 * within development tools that are supporting such ARM based processors.  
 *  
 * @par  
 * THIS SOFTWARE IS PROVIDED "AS IS". NO WARRANTIES, WHETHER EXPRESS, IMPLIED  
 * OR STATUTORY, INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF  
 * MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE APPLY TO THIS SOFTWARE.  
 * ARM SHALL NOT, IN ANY CIRCUMSTANCES, BE LIABLE FOR SPECIAL, INCIDENTAL, OR  
 * CONSEQUENTIAL DAMAGES, FOR ANY REASON WHATSOEVER.  
 *  
***** /  
  
#ifndef __LPC17xx_H__  
#define __LPC17xx_H__
```

```

/*
 * ===== Interrupt Number Definition =====
 *
 */

/** @addtogroup LPC17xx_System
 * @{
 */

/** @brief IRQ interrupt source definition */
typedef enum IRQn
{
    /* Cortex-M3 Processor Exceptions Numbers
     ****
    NonMaskableInt_IRQn      = -14,      /*!< 2 Non Maskable Interrupt
    */
    MemoryManagement_IRQn     = -12,      /*!< 4 Cortex-M3 Memory Management Interrupt
    */
    BusFault_IRQn             = -11,      /*!< 5 Cortex-M3 Bus Fault Interrupt
    */
    UsageFault_IRQn           = -10,      /*!< 6 Cortex-M3 Usage Fault Interrupt
    */
    SVCall_IRQn               = -5,       /*!< 11 Cortex-M3 SV Call Interrupt
    */
    DebugMonitor_IRQn          = -4,       /*!< 12 Cortex-M3 Debug Monitor Interrupt
    */
    PendSV_IRQn               = -2,       /*!< 14 Cortex-M3 Pend SV Interrupt
    */
    SysTick_IRQn               = -1,       /*!< 15 Cortex-M3 System Tick Interrupt
    */

    /* LPC17xx Specific Interrupt Numbers
     ****
    WDT_IRQn                  = 0,        /*!< Watchdog Timer Interrupt
    */
    TIMER0_IRQn                = 1,        /*!< Timer0 Interrupt
    */
    TIMER1_IRQn                = 2,        /*!< Timer1 Interrupt
    */
    TIMER2_IRQn                = 3,        /*!< Timer2 Interrupt
    */
    TIMER3_IRQn                = 4,        /*!< Timer3 Interrupt
    */
    UART0_IRQn                 = 5,        /*!< UART0 Interrupt
    */
}

```

```

    UART1_IRQn          = 6,           /*!< UART1 Interrupt
*/
    UART2_IRQn          = 7,           /*!< UART2 Interrupt
*/
    UART3_IRQn          = 8,           /*!< UART3 Interrupt
*/
    PWM1_IRQn           = 9,           /*!< PWM1 Interrupt
*/
    I2C0_IRQn           = 10,          /*!< I2C0 Interrupt
*/
    I2C1_IRQn           = 11,          /*!< I2C1 Interrupt
*/
    I2C2_IRQn           = 12,          /*!< I2C2 Interrupt
*/
    SPI_IRQn            = 13,          /*!< SPI Interrupt
*/
    SSP0_IRQn           = 14,          /*!< SSP0 Interrupt
*/
    SSP1_IRQn           = 15,          /*!< SSP1 Interrupt
*/
    PLL0_IRQn           = 16,          /*!< PLL0 Lock (Main PLL) Interrupt
*/
    RTC_IRQn             = 17,          /*!< Real Time Clock Interrupt
*/
    EINT0_IRQn           = 18,          /*!< External Interrupt 0 Interrupt
*/
    EINT1_IRQn           = 19,          /*!< External Interrupt 1 Interrupt
*/
    EINT2_IRQn           = 20,          /*!< External Interrupt 2 Interrupt
*/
    EINT3_IRQn           = 21,          /*!< External Interrupt 3 Interrupt
*/
    ADC_IRQn             = 22,          /*!< A/D Converter Interrupt
*/
    BOD_IRQn             = 23,          /*!< Brown-Out Detect Interrupt
*/
    USB_IRQn             = 24,          /*!< USB Interrupt
*/
    CAN_IRQn             = 25,          /*!< CAN Interrupt
*/
    DMA_IRQn              = 26,          /*!< General Purpose DMA Interrupt
*/
    I2S_IRQn             = 27,          /*!< I2S Interrupt
*/
    ENET_IRQn             = 28,          /*!< Ethernet Interrupt
*/
    RIT_IRQn              = 29,          /*!< Repetitive Interrupt Timer Interrupt
*/
    MCPWM_IRQn            = 30,          /*!< Motor Control PWM Interrupt
*/

```

```

        QEI_IRQn           = 31,          /*!< Quadrature Encoder Interface Interrupt
*/
        PLL1_IRQn           = 32,          /*!< PLL1 Lock (USB PLL) Interrupt
*/
        USBActivity_IRQn     = 33,          /*!< USB Activity Interrupt
*/
        CANActivity_IRQn     = 34           /*!< CAN Activity Interrupt
*/
    } IRQn_Type;
}

/*
* ===== Processor and Core Peripheral Section =====
*
*/
/* Configuration of the Cortex-M3 Processor and Core Peripherals */
#define __MPU_PRESENT      1           /*!< MPU present or not
*/
#define __NVIC_PRIO_BITS    5           /*!< Number of Bits used for Priority Levels
*/
#define __Vendor_SysTickConfig 0         /*!< Set to 1 if different SysTick Config is
used */

#include "core_cm3.h"                  /* Cortex-M3 processor and core peripherals
*/
#include "system_LPC17xx.h"             /* System Header
*/
/******
*          Device Specific Peripheral registers structures
*****
*/
#if defined ( __CC_ARM )
#pragma anon_unions
#endif

/*----- System Control (SC) -----*/
/** @brief System Control (SC) register structure definition */
typedef struct

```

```
{
    __IO uint32_t FLASHCFG;                      /* Flash Accelerator Module          */
    uint32_t RESERVED0[31];
    __IO uint32_t PLL0CON;                        /* Clocking and Power Control      */
    __IO uint32_t PLL0CFG;
    __I  uint32_t PLL0STAT;
    __O  uint32_t PLL0FEED;
    uint32_t RESERVED1[4];
    __IO uint32_t PLL1CON;
    __IO uint32_t PLL1CFG;
    __I  uint32_t PLL1STAT;
    __O  uint32_t PLL1FEED;
    uint32_t RESERVED2[4];
    __IO uint32_t PCON;
    __IO uint32_t PCONP;
    uint32_t RESERVED3[15];
    __IO uint32_t CCLKCFG;
    __IO uint32_t USBCLKCFG;
    __IO uint32_t CLKSRCSEL;
    __IO uint32_t      CANSLEEPCLR;
    __IO uint32_t      CANWAKEFLAGS;
    uint32_t RESERVED4[10];
    __IO uint32_t EXTINT;                         /* External Interrupts             */
    uint32_t RESERVED5;
    __IO uint32_t EXTMODE;
    __IO uint32_t EXTPOLAR;
    uint32_t RESERVED6[12];
    __IO uint32_t RSID;                           /* Reset                          */
    uint32_t RESERVED7[7];
    __IO uint32_t SCS;                            /* Syscon Miscellaneous Registers */
    __IO uint32_t IRCTRIM;                        /* Clock Dividers                 */
    __IO uint32_t PCLKSEL0;
    __IO uint32_t PCLKSEL1;
    uint32_t RESERVED8[4];
    __IO uint32_t USBIntSt;                       /* USB Device/OTG Interrupt Register */
    __IO uint32_t DMAREQSEL;
    __IO uint32_t CLKOUTCFG;                      /* Clock Output Configuration      */
} LPC_SC_TypeDef;
```

```
/*----- Pin Connect Block (PINCON) -----*/
/** @brief Pin Connect Block (PINCON) register structure definition */
typedef struct
{
    __IO uint32_t PINSEL0;
    __IO uint32_t PINSEL1;
    __IO uint32_t PINSEL2;
    __IO uint32_t PINSEL3;
    __IO uint32_t PINSEL4;
    __IO uint32_t PINSEL5;
    __IO uint32_t PINSEL6;
    __IO uint32_t PINSEL7;
    __IO uint32_t PINSEL8;
    __IO uint32_t PINSEL9;
    __IO uint32_t PINSEL10;
    uint32_t RESERVED0[5];

    __IO uint32_t PINMODE0;
    __IO uint32_t PINMODE1;
    __IO uint32_t PINMODE2;
    __IO uint32_t PINMODE3;
    __IO uint32_t PINMODE4;
    __IO uint32_t PINMODE5;
    __IO uint32_t PINMODE6;
    __IO uint32_t PINMODE7;
    __IO uint32_t PINMODE8;
    __IO uint32_t PINMODE9;
    __IO uint32_t PINMODE_O0;
    __IO uint32_t PINMODE_O1;
    __IO uint32_t PINMODE_O2;
    __IO uint32_t PINMODE_O3;
    __IO uint32_t PINMODE_O4;
    __IO uint32_t I2CPADCFG;
} LPC_PINCON_TypeDef;

/*----- General Purpose Input/Output (GPIO) -----*/
/** @brief General Purpose Input/Output (GPIO) register structure definition */
typedef struct
{
    union {
```

```
__IO uint32_t FIODIR;
struct {
    __IO uint16_t FIODIRL;
    __IO uint16_t FIODIRH;
};

struct {
    __IO uint8_t  FIODIRO;
    __IO uint8_t  FIODIR1;
    __IO uint8_t  FIODIR2;
    __IO uint8_t  FIODIR3;
};

uint32_t RESERVED0[3];

union {
    __IO uint32_t FIOMASK;
    struct {
        __IO uint16_t FIOMASKL;
        __IO uint16_t FIOMASKH;
    };
    struct {
        __IO uint8_t  FIOMASK0;
        __IO uint8_t  FIOMASK1;
        __IO uint8_t  FIOMASK2;
        __IO uint8_t  FIOMASK3;
    };
};

union {
    __IO uint32_t FIOPIN;
    struct {
        __IO uint16_t FIOPINL;
        __IO uint16_t FIOPINH;
    };
    struct {
        __IO uint8_t  FIOPINO;
        __IO uint8_t  FIOPIN1;
        __IO uint8_t  FIOPIN2;
        __IO uint8_t  FIOPIN3;
    };
};
```

```
union {
    __IO uint32_t FIOSET;
    struct {
        __IO uint16_t FIOSETL;
        __IO uint16_t FIOSETH;
    };
    struct {
        __IO uint8_t  FIOSET0;
        __IO uint8_t  FIOSET1;
        __IO uint8_t  FIOSET2;
        __IO uint8_t  FIOSET3;
    };
};

union {
    __O  uint32_t FIOCLR;
    struct {
        __O  uint16_t FIOCLRL;
        __O  uint16_t FIOCLRH;
    };
    struct {
        __O  uint8_t  FIOCLR0;
        __O  uint8_t  FIOCLR1;
        __O  uint8_t  FIOCLR2;
        __O  uint8_t  FIOCLR3;
    };
};

} LPC_GPIO_TypeDef;

/** @brief General Purpose Input/Output interrupt (GPIOINT) register structure
definition */
typedef struct
{
    __I  uint32_t IntStatus;
    __I  uint32_t IO0IntStatR;
    __I  uint32_t IO0IntStatF;
    __O  uint32_t IO0IntClr;
    __IO uint32_t IO0IntEnR;
    __IO uint32_t IO0IntEnF;
    uint32_t RESERVED0[3];
    __I  uint32_t IO2IntStatR;
```

```
__I  uint32_t IO2IntStatF;
__O  uint32_t IO2IntClr;
__IO uint32_t IO2IntEnR;
__IO uint32_t IO2IntEnF;
} LPC_GPIOINT_TypeDef;

/*----- Timer (TIM) -----*/
/** @brief Timer (TIM) register structure definition */
typedef struct
{
    __IO uint32_t IR;
    __IO uint32_t TCR;
    __IO uint32_t TC;
    __IO uint32_t PR;
    __IO uint32_t PC;
    __IO uint32_t MCR;
    __IO uint32_t MR0;
    __IO uint32_t MR1;
    __IO uint32_t MR2;
    __IO uint32_t MR3;
    __IO uint32_t CCR;
    __I  uint32_t CR0;
    __I  uint32_t CR1;
    uint32_t RESERVED0[2];
    __IO uint32_t EMR;
    uint32_t RESERVED1[12];
    __IO uint32_t CTCR;
} LPC_TIM_TypeDef;

/*----- Pulse-Width Modulation (PWM) -----*/
/** @brief Pulse-Width Modulation (PWM) register structure definition */
typedef struct
{
    __IO uint32_t IR;
    __IO uint32_t TCR;
    __IO uint32_t TC;
    __IO uint32_t PR;
    __IO uint32_t PC;
    __IO uint32_t MCR;
```

```
__IO uint32_t MR0;
__IO uint32_t MR1;
__IO uint32_t MR2;
__IO uint32_t MR3;
__IO uint32_t CCR;
__I  uint32_t CR0;
__I  uint32_t CR1;
__I  uint32_t CR2;
__I  uint32_t CR3;
    uint32_t RESERVED0;
__IO uint32_t MR4;
__IO uint32_t MR5;
__IO uint32_t MR6;
__IO uint32_t PCR;
__IO uint32_t LER;
    uint32_t RESERVED1[7];
__IO uint32_t CTCR;
} LPC_PWM_TypeDef;

/*----- Universal Asynchronous Receiver Transmitter (UART) -----*/
/** @brief Universal Asynchronous Receiver Transmitter (UART) register structure
definition */

typedef struct
{
    union {
        __I  uint8_t  RBR;
        __O  uint8_t  THR;
        __IO uint8_t  DLL;
        uint32_t RESERVED0;
    };
    union {
        __IO uint8_t  DLM;
        __IO uint32_t IER;
    };
    union {
        __I  uint32_t IIR;
        __O  uint8_t  FCR;
    };
    __IO uint8_t  LCR;
    uint8_t  RESERVED1[7];
}
```

```
__I  uint8_t  LSR;
      uint8_t  RESERVED2[7];
__IO uint8_t  SCR;
      uint8_t  RESERVED3[3];
__IO uint32_t ACR;
__IO uint8_t  ICR;
      uint8_t  RESERVED4[3];
__IO uint8_t  FDR;
      uint8_t  RESERVED5[7];
__IO uint8_t  TER;
      uint8_t  RESERVED6[39];
__I  uint8_t  FIFOLVL;

} LPC_UART_TypeDef;

/** @brief Universal Asynchronous Receiver Transmitter 0 (UART0) register structure
definition */

typedef struct
{
    union {
        __I  uint8_t  RBR;
        __O  uint8_t  THR;
        __IO uint8_t  DLL;
        uint32_t RESERVED0;
    };
    union {
        __IO uint8_t  DLM;
        __IO uint32_t IER;
    };
    union {
        __I  uint32_t IIR;
        __O  uint8_t  FCR;
    };
    __IO uint8_t  LCR;
        uint8_t  RESERVED1[7];
__I  uint8_t  LSR;
        uint8_t  RESERVED2[7];
__IO uint8_t  SCR;
        uint8_t  RESERVED3[3];
__IO uint32_t ACR;
__IO uint8_t  ICR;
```

```
    uint8_t RESERVED4[3];

    __IO uint8_t FDR;
    uint8_t RESERVED5[7];
    __IO uint8_t TER;
    uint8_t RESERVED6[39];
    __I  uint8_t FIFOVL;
} LPC_UART0_TypeDef;

/** @brief Universal Asynchronous Receiver Transmitter 1 (UART1) register structure
definition */

typedef struct
{
    union {
        __I  uint8_t RBR;
        __O  uint8_t THR;
        __IO uint8_t DLL;
        uint32_t RESERVED0;
    };
    union {
        __IO uint8_t DLM;
        __IO uint32_t IER;
    };
    union {
        __I  uint32_t IIR;
        __O  uint8_t FCR;
    };
    __IO uint8_t LCR;
    uint8_t RESERVED1[3];
    __IO uint8_t MCR;
    uint8_t RESERVED2[3];
    __I  uint8_t LSR;
    uint8_t RESERVED3[3];
    __I  uint8_t MSR;
    uint8_t RESERVED4[3];
    __IO uint8_t SCR;
    uint8_t RESERVED5[3];
    __IO uint32_t ACR;
    uint32_t RESERVED6;
    __IO uint32_t FDR;
    uint32_t RESERVED7;
```

```
__IO uint8_t TER;
    uint8_t RESERVED8[27];
__IO uint8_t RS485CTRL;
    uint8_t RESERVED9[3];
__IO uint8_t ADRMATCH;
    uint8_t RESERVED10[3];
__IO uint8_t RS485DLY;
    uint8_t RESERVED11[3];
__I uint8_t FIFOLVL;
} LPC_UART1_TypeDef;

/*----- Serial Peripheral Interface (SPI) -----*/
/** @brief Serial Peripheral Interface (SPI) register structure definition */
typedef struct
{
    __IO uint32_t SPCR;
    __I  uint32_t SPSR;
    __IO uint32_t SPDR;
    __IO uint32_t SPCCR;
    uint32_t RESERVED0[3];
    __IO uint32_t SPINT;
} LPC_SPI_TypeDef;

/*----- Synchronous Serial Communication (SSP) -----*/
/** @brief Synchronous Serial Communication (SSP) register structure definition */
typedef struct
{
    __IO uint32_t CR0;
    __IO uint32_t CR1;
    __IO uint32_t DR;
    __I  uint32_t SR;
    __IO uint32_t CPSR;
    __IO uint32_t IMSC;
    __IO uint32_t RIS;
    __IO uint32_t MIS;
    __IO uint32_t ICR;
    __IO uint32_t DMACR;
} LPC_SSP_TypeDef;
```

```
/*----- Inter-Integrated Circuit (I2C) -----*/
/** @brief  Inter-Integrated Circuit (I2C) register structure definition */
typedef struct
{
    __IO uint32_t I2CONSET;
    __I  uint32_t I2STAT;
    __IO uint32_t I2DAT;
    __IO uint32_t I2ADR0;
    __IO uint32_t I2SCLH;
    __IO uint32_t I2SCLL;
    __O  uint32_t I2CONCLR;
    __IO uint32_t MMCTRL;
    __IO uint32_t I2ADR1;
    __IO uint32_t I2ADR2;
    __IO uint32_t I2ADR3;
    __I  uint32_t I2DATA_BUFFER;
    __IO uint32_t I2MASK0;
    __IO uint32_t I2MASK1;
    __IO uint32_t I2MASK2;
    __IO uint32_t I2MASK3;
} LPC_I2C_TypeDef;
```

```
/*----- Inter IC Sound (I2S) -----*/
/** @brief  Inter IC Sound (I2S) register structure definition */
typedef struct
{
    __IO uint32_t I2SDAO;
    __IO uint32_t I2SDAI;
    __O  uint32_t I2STXFIFO;
    __I  uint32_t I2SRXFIFO;
    __I  uint32_t I2SSTATE;
    __IO uint32_t I2SDMA1;
    __IO uint32_t I2SDMA2;
    __IO uint32_t I2SIRQ;
    __IO uint32_t I2STXRATE;
    __IO uint32_t I2SRXRATE;
    __IO uint32_t I2STXBITRATE;
    __IO uint32_t I2SRXBITRATE;
    __IO uint32_t I2STXMODE;
```

```
__IO uint32_t I2SRXMODE;
} LPC_I2S_TypeDef;

/*----- Repetitive Interrupt Timer (RIT) -----*/
/** @brief Repetitive Interrupt Timer (RIT) register structure definition */
typedef struct
{
    __IO uint32_t RICOMPVAL;
    __IO uint32_t RIMASK;
    __IO uint8_t  RICTRL;
    uint8_t   RESERVED0[3];
    __IO uint32_t RICOUNTER;
} LPC_RIT_TypeDef;

/*----- Real-Time Clock (RTC) -----*/
/** @brief Real-Time Clock (RTC) register structure definition */
typedef struct
{
    __IO uint8_t  ILR;
    uint8_t   RESERVED0[7];
    __IO uint8_t  CCR;
    uint8_t   RESERVED1[3];
    __IO uint8_t  CIIR;
    uint8_t   RESERVED2[3];
    __IO uint8_t  AMR;
    uint8_t   RESERVED3[3];
    __I  uint32_t CTIME0;
    __I  uint32_t CTIME1;
    __I  uint32_t CTIME2;
    __IO uint8_t  SEC;
    uint8_t   RESERVED4[3];
    __IO uint8_t  MIN;
    uint8_t   RESERVED5[3];
    __IO uint8_t  HOUR;
    uint8_t   RESERVED6[3];
    __IO uint8_t  DOM;
    uint8_t   RESERVED7[3];
    __IO uint8_t  DOW;
    uint8_t   RESERVED8[3];
```

```
__IO uint16_t DOY;
    uint16_t RESERVED9;

__IO uint8_t MONTH;
    uint8_t RESERVED10[3];

__IO uint16_t YEAR;
    uint16_t RESERVED11;

__IO uint32_t CALIBRATION;

__IO uint32_t GPREG0;
__IO uint32_t GPREG1;
__IO uint32_t GPREG2;
__IO uint32_t GPREG3;
__IO uint32_t GPREG4;

__IO uint8_t RTC_AUXEN;
    uint8_t RESERVED12[3];

__IO uint8_t RTC_AUX;
    uint8_t RESERVED13[3];

__IO uint8_t ALSEC;
    uint8_t RESERVED14[3];

__IO uint8_t ALMIN;
    uint8_t RESERVED15[3];

__IO uint8_t ALHOUR;
    uint8_t RESERVED16[3];

__IO uint8_t ALDOM;
    uint8_t RESERVED17[3];

__IO uint8_t ALDOW;
    uint8_t RESERVED18[3];

__IO uint16_t ALDOY;
    uint16_t RESERVED19;

__IO uint8_t ALMON;
    uint8_t RESERVED20[3];

__IO uint16_t ALYEAR;
    uint16_t RESERVED21;

} LPC_RTC_TypeDef;

/*----- Watchdog Timer (WDT) -----*/
/** @brief Watchdog Timer (WDT) register structure definition */
typedef struct
{
    __IO uint8_t WDMOD;
```

```
    uint8_t RESERVED0[3];

    __IO uint32_t WDTC;
    __O  uint8_t WDFEED;
    uint8_t RESERVED1[3];
    __I  uint32_t WDTV;
    __IO uint32_t WDCLKSEL;
} LPC_WDT_TypeDef;

/*----- Analog-to-Digital Converter (ADC) -----*/
/** @brief Analog-to-Digital Converter (ADC) register structure definition */
typedef struct
{
    __IO uint32_t ADCR;
    __IO uint32_t ADGDR;
    uint32_t RESERVED0;
    __IO uint32_t ADINTEN;
    __I  uint32_t ADDR0;
    __I  uint32_t ADDR1;
    __I  uint32_t ADDR2;
    __I  uint32_t ADDR3;
    __I  uint32_t ADDR4;
    __I  uint32_t ADDR5;
    __I  uint32_t ADDR6;
    __I  uint32_t ADDR7;
    __I  uint32_t ADSTAT;
    __IO uint32_t ADTRM;
} LPC_ADC_TypeDef;

/*----- Digital-to-Analog Converter (DAC) -----*/
/** @brief Digital-to-Analog Converter (DAC) register structure definition */
typedef struct
{
    __IO uint32_t DACR;
    __IO uint32_t DACCTRL;
    __IO uint16_t DACCNTVAL;
} LPC_DAC_TypeDef;

/*----- Motor Control Pulse-Width Modulation (MCPWM) -----*/
/** @brief Motor Control Pulse-Width Modulation (MCPWM) register structure definition */
*/
```

```

typedef struct
{
    __I  uint32_t MCCON;
    __O  uint32_t MCCON_SET;
    __O  uint32_t MCCON_CLR;
    __I  uint32_t MCCAPCON;
    __O  uint32_t MCCAPCON_SET;
    __O  uint32_t MCCAPCON_CLR;
    __IO uint32_t MCTIM0;
    __IO uint32_t MCTIM1;
    __IO uint32_t MCTIM2;
    __IO uint32_t MCPERO;
    __IO uint32_t MCPER1;
    __IO uint32_t MCPER2;
    __IO uint32_t MCPW0;
    __IO uint32_t MCPW1;
    __IO uint32_t MCPW2;
    __IO uint32_t MCDEADTIME;
    __IO uint32_t MCCCP;
    __IO uint32_t MCCR0;
    __IO uint32_t MCCR1;
    __IO uint32_t MCCR2;
    __I  uint32_t MCINTEN;
    __O  uint32_t MCINTEN_SET;
    __O  uint32_t MCINTEN_CLR;
    __I  uint32_t MCCNTCON;
    __O  uint32_t MCCNTCON_SET;
    __O  uint32_t MCCNTCON_CLR;
    __I  uint32_t MCINTFLAG;
    __O  uint32_t MCINTFLAG_SET;
    __O  uint32_t MCINTFLAG_CLR;
    __O  uint32_t MCCAP_CLR;
} LPC_MCPWM_TypeDef;

/*----- Quadrature Encoder Interface (QEI) -----*/
/** @brief Quadrature Encoder Interface (QEI) register structure definition */
typedef struct
{
    __O  uint32_t QEICON;
}

```

```

__I  uint32_t QEISTAT;
__IO uint32_t QEICONF;
__I  uint32_t QEIPOS;
__IO uint32_t QEIMAXPOS;
__IO uint32_t CMPOS0;
__IO uint32_t CMPOS1;
__IO uint32_t CMPOS2;
__I  uint32_t INXCNT;
__IO uint32_t INXCMP;
__IO uint32_t QEILOAD;
__I  uint32_t QEITIME;
__I  uint32_t QEIVEL;
__I  uint32_t QEICAP;
__IO uint32_t VELCOMP;
__IO uint32_t FILTER;
    uint32_t RESERVED0[998];
__O  uint32_t QEIIEC;
__O  uint32_t QEIIIES;
__I  uint32_t QEIINTSTAT;
__I  uint32_t QEIIIE;
__O  uint32_t QEICLR;
__O  uint32_t QEISET;
} LPC_QEI_TypeDef;

/*----- Controller Area Network (CAN) -----*/
/** @brief Controller Area Network Acceptance Filter RAM (CANAF_RAM) structure
definition */
typedef struct
{
    __IO uint32_t mask[512];           /* ID Masks */          */
} LPC_CANAF_RAM_TypeDef;

/** @brief Controller Area Network Acceptance Filter(CANAF) register structure
definition */
typedef struct             /* Acceptance Filter Registers */ {
    __IO uint32_t AFMR;
    __IO uint32_t SFF_sa;
    __IO uint32_t SFF_GRP_sa;
    __IO uint32_t EFF_sa;
}

```

```
__IO uint32_t EFF_GRP_sa;
__IO uint32_t ENDofTable;
__I  uint32_t LUTerrAd;
__I  uint32_t LUTerr;
__IO uint32_t FCANIE;
__IO uint32_t FCANICO;
__IO uint32_t FCANIC1;
} LPC_CANAF_TypeDef;

/** @brief Controller Area Network Central (CANCR) register structure definition */
typedef struct                                /* Central Registers           */
{
    __I  uint32_t CANTxSR;
    __I  uint32_t CANRxSR;
    __I  uint32_t CANMSR;
} LPC_CANCR_TypeDef;

/** @brief Controller Area Network Controller (CAN) register structure definition */
typedef struct                                /* Controller Registers        */
{
    __IO uint32_t MOD;
    __O  uint32_t CMR;
    __IO uint32_t GSR;
    __I  uint32_t ICR;
    __IO uint32_t IER;
    __IO uint32_t BTR;
    __IO uint32_t EWL;
    __I  uint32_t SR;
    __IO uint32_t RFS;
    __IO uint32_t RID;
    __IO uint32_t RDA;
    __IO uint32_t RDB;
    __IO uint32_t TFI1;
    __IO uint32_t TID1;
    __IO uint32_t TDA1;
    __IO uint32_t TDB1;
    __IO uint32_t TFI2;
    __IO uint32_t TID2;
    __IO uint32_t TDA2;
```

```
__IO uint32_t TDB2;
__IO uint32_t TFI3;
__IO uint32_t TID3;
__IO uint32_t TDA3;
__IO uint32_t TDB3;
} LPC_CAN_TypeDef;

/*----- General Purpose Direct Memory Access (GPDMA) -----*/
/** @brief General Purpose Direct Memory Access (GPDMA) register structure definition */
typedef struct /* Common Registers */
{
    __I  uint32_t DMACIntStat;
    __I  uint32_t DMACIntTCStat;
    __O  uint32_t DMACIntTCClear;
    __I  uint32_t DMACIntErrStat;
    __O  uint32_t DMACIntErrClr;
    __I  uint32_t DMACRawIntTCStat;
    __I  uint32_t DMACRawIntErrStat;
    __I  uint32_t DMACEnbldChns;
    __IO uint32_t DMACSoftBReq;
    __IO uint32_t DMACSoftSReq;
    __IO uint32_t DMACSoftLBReq;
    __IO uint32_t DMACSoftLSReq;
    __IO uint32_t DMACConfig;
    __IO uint32_t DMACSync;
} LPC_GPDMA_TypeDef;

/** @brief General Purpose Direct Memory Access Channel (GPDMACH) register structure definition */
typedef struct /* Channel Registers */
{
    __IO uint32_t DMACCSrcAddr;
    __IO uint32_t DMACCDestAddr;
    __IO uint32_t DMACLLI;
    __IO uint32_t DMACCControl;
    __IO uint32_t DMACCConfig;
} LPC_GPDMACH_TypeDef;

/*----- Universal Serial Bus (USB) -----*/
```

```
/** @brief Universal Serial Bus (USB) register structure definition */

typedef struct

{
    __I  uint32_t HcRevision;           /* USB Host Registers          */
    __IO uint32_t HcControl;
    __IO uint32_t HcCommandStatus;
    __IO uint32_t HcInterruptStatus;
    __IO uint32_t HcInterruptEnable;
    __IO uint32_t HcInterruptDisable;
    __IO uint32_t HcHCCA;
    __I   uint32_t HcPeriodCurrentED;
    __IO uint32_t HcControlHeadED;
    __IO uint32_t HcControlCurrentED;
    __IO uint32_t HcBulkHeadED;
    __IO uint32_t HcBulkCurrentED;
    __I   uint32_t HcDoneHead;
    __IO uint32_t HcFmInterval;
    __I   uint32_t HcFmRemaining;
    __I   uint32_t HcFmNumber;
    __IO uint32_t HcPeriodicStart;
    __IO uint32_t HcLSTreshold;
    __IO uint32_t HcRhDescriptorA;
    __IO uint32_t HcRhDescriptorB;
    __IO uint32_t HcRhStatus;
    __IO uint32_t HcRhPortStatus1;
    __IO uint32_t HcRhPortStatus2;
    uint32_t RESERVED0[40];
    __I   uint32_t Module_ID;

    __I   uint32_t OTGIntSt;           /* USB On-The-Go Registers      */
    __IO uint32_t OTGIntEn;
    __O   uint32_t OTGIntSet;
    __O   uint32_t OTGIntClr;
    __IO uint32_t OTGStCtrl;
    __IO uint32_t OTGTmr;
    uint32_t RESERVED1[58];

    __I   uint32_t USBDevIntSt;        /* USB Device Interrupt Registers */
    __IO uint32_t USBDevIntEn;
```

```
__O uint32_t USBDevIntClr;
__O uint32_t USBDevIntSet;

__O uint32_t USBCmdCode;           /* USB Device SIE Command Registers */
__I  uint32_t USBCmdData;

__I  uint32_t USBRxData;          /* USB Device Transfer Registers */
__O uint32_t USBTxData;
__I  uint32_t USBRxPLen;
__O uint32_t USBTxPLen;
__IO uint32_t USBCtrl;
__O uint32_t USBDevIntPri;

__I  uint32_t USBEpIntSt;         /* USB Device Endpoint Interrupt Regs */
__IO uint32_t USBEpIntEn;
__O uint32_t USBEpIntClr;
__O uint32_t USBEpIntSet;
__O uint32_t USBEpIntPri;

__IO uint32_t USBReEp;            /* USB Device Endpoint Realization Reg*/
__O uint32_t USBEpInd;
__IO uint32_t USBMaxPSize;

__I  uint32_t USBDMARst;          /* USB Device DMA Registers */
__O uint32_t USBDMARClr;
__O uint32_t USBDMARSet;
    uint32_t RESERVED2[9];

__IO uint32_t USBUDCAH;
__I  uint32_t USBEpDMASt;
__O uint32_t USBEpDMAEn;
__O uint32_t USBEpDMADis;
__I  uint32_t USBDMAIntSt;
__IO uint32_t USBDMAIntEn;
    uint32_t RESERVED3[2];

__I  uint32_t USBEoTIntSt;
__O uint32_t USBEoTIntClr;
__O uint32_t USBEoTIntSet;
__I  uint32_t USBNDDRIntSt;
__O uint32_t USBNDDRIntClr;
```

```
__O uint32_t USBNDDRIntSet;
__I uint32_t USBSysErrIntSt;
__O uint32_t USBSysErrIntClr;
__O uint32_t USBSysErrIntSet;
    uint32_t RESERVED4[15];

union {
    __I uint32_t I2C_RX; /* USB OTG I2C Registers */
    __O uint32_t I2C_TX;
};

__I uint32_t I2C_STS;
__IO uint32_t I2C_CTL;
__IO uint32_t I2C_CLKHI;
__O uint32_t I2C_CLKLO;
    uint32_t RESERVED5[824];

union {
    __IO uint32_t USBClkCtrl; /* USB Clock Control Registers */
    __IO uint32_t OTGClkCtrl;
};

union {
    __I uint32_t USBClkSt;
    __I uint32_t OTGClkSt;
};

} LPC_USB_TypeDef;

/*----- Ethernet Media Access Controller (EMAC) -----*/
/** @brief Ethernet Media Access Controller (EMAC) register structure definition */
typedef struct
{
    __IO uint32_t MAC1; /* MAC Registers */
    __IO uint32_t MAC2;
    __IO uint32_t IPGT;
    __IO uint32_t IPGR;
    __IO uint32_t CLRT;
    __IO uint32_t MAXF;
    __IO uint32_t SUPP;
    __IO uint32_t TEST;
    __IO uint32_t MCFG;
}
```

```
__IO uint32_t MCMD;
__IO uint32_t MADR;
__O  uint32_t MWTD;
__I  uint32_t MRDD;
__I  uint32_t MIND;
    uint32_t RESERVED0[2];

__IO uint32_t SA0;
__IO uint32_t SA1;
__IO uint32_t SA2;
    uint32_t RESERVED1[45];

__IO uint32_t Command;           /* Control Registers */
__I   uint32_t Status;
__IO uint32_t RxDescriptor;
__IO uint32_t RxStatus;
__IO uint32_t RxDescriptorNumber;
__I   uint32_t RxProduceIndex;
__IO uint32_t RxConsumeIndex;
__IO uint32_t TxDescriptor;
__IO uint32_t TxStatus;
__IO uint32_t TxDescriptorNumber;
__IO uint32_t TxProduceIndex;
__I   uint32_t TxConsumeIndex;
    uint32_t RESERVED2[10];

__I   uint32_t TSV0;
__I   uint32_t TSV1;
__I   uint32_t RSV;
    uint32_t RESERVED3[3];

__IO uint32_t FlowControlCounter;
__I   uint32_t FlowControlStatus;
    uint32_t RESERVED4[34];

__IO uint32_t RxFilterCtrl;      /* Rx Filter Registers */
__IO uint32_t RxFilterWoLStatus;
__IO uint32_t RxFilterWoLClear;
    uint32_t RESERVED5;

__IO uint32_t HashFilterL;
__IO uint32_t HashFilterH;
    uint32_t RESERVED6[882];

__I   uint32_t IntStatus;        /* Module Control Registers */
__IO uint32_t IntEnable;
```

```
__O uint32_t IntClear;
__O uint32_t IntSet;
    uint32_t RESERVED7;
__IO uint32_t PowerDown;
    uint32_t RESERVED8;
__IO uint32_t Module_ID;
} LPC_EMAC_TypeDef;

#if defined ( __CC_ARM )
#pragma no_anon_unions
#endif

/*********************************************
 *                                     Peripheral memory map
 */
/*********************************************
 * Base addresses
 */
#define LPC_FLASH_BASE      (0x00000000UL)
#define LPC_RAM_BASE        (0x10000000UL)

#ifdef __LPC17XX_REV00
#define LPC_AHBRAM0_BASE   (0x20000000UL)
#define LPC_AHBRAM1_BASE   (0x20004000UL)
#else
#define LPC_AHBRAM0_BASE   (0x2007C000UL)
#define LPC_AHBRAM1_BASE   (0x20080000UL)
#endif
#define LPC_GPIO_BASE       (0x2009C000UL)
#define LPC_APB0_BASE       (0x40000000UL)
#define LPC_APB1_BASE       (0x40080000UL)
#define LPC_AHB_BASE        (0x50000000UL)
#define LPC_CM3_BASE        (0xE0000000UL)

/* APB0 peripherals
 */
#define LPC_WDT_BASE        (LPC_APB0_BASE + 0x00000)
#define LPC_TIM0_BASE        (LPC_APB0_BASE + 0x04000)
#define LPC_TIM1_BASE        (LPC_APB0_BASE + 0x08000)
#define LPC_UART0_BASE       (LPC_APB0_BASE + 0x0C000)
#define LPC_UART1_BASE       (LPC_APB0_BASE + 0x10000)
```

```

#define LPC_PWM1_BASE           (LPC_APBO_BASE + 0x18000)
#define LPC_I2C0_BASE           (LPC_APBO_BASE + 0x1C000)
#define LPC_SPI_BASE             (LPC_APBO_BASE + 0x20000)
#define LPC_RTC_BASE              (LPC_APBO_BASE + 0x24000)
#define LPC_GPIOINT_BASE         (LPC_APBO_BASE + 0x28080)
#define LPC_PINCON_BASE           (LPC_APBO_BASE + 0x2C000)
#define LPC_SSP1_BASE             (LPC_APBO_BASE + 0x30000)
#define LPC_ADC_BASE               (LPC_APBO_BASE + 0x34000)
#define LPC_CANAF_RAM_BASE        (LPC_APBO_BASE + 0x38000)
#define LPC_CANAF_BASE             (LPC_APBO_BASE + 0x3C000)
#define LPC_CANCR_BASE             (LPC_APBO_BASE + 0x40000)
#define LPC_CAN1_BASE               (LPC_APBO_BASE + 0x44000)
#define LPC_CAN2_BASE               (LPC_APBO_BASE + 0x48000)
#define LPC_I2C1_BASE               (LPC_APBO_BASE + 0x5C000)

/* APB1 peripherals                                         */
#define LPC_SSP0_BASE             (LPC_APB1_BASE + 0x08000)
#define LPC_DAC_BASE                (LPC_APB1_BASE + 0x0C000)
#define LPC_TIM2_BASE               (LPC_APB1_BASE + 0x10000)
#define LPC_TIM3_BASE               (LPC_APB1_BASE + 0x14000)
#define LPC_UART2_BASE              (LPC_APB1_BASE + 0x18000)
#define LPC_UART3_BASE              (LPC_APB1_BASE + 0x1C000)
#define LPC_I2C2_BASE               (LPC_APB1_BASE + 0x20000)
#define LPC_I2S_BASE                 (LPC_APB1_BASE + 0x28000)
#define LPC_RIT_BASE                  (LPC_APB1_BASE + 0x30000)
#define LPC_MCPWM_BASE              (LPC_APB1_BASE + 0x38000)
#define LPC_QEI_BASE                  (LPC_APB1_BASE + 0x3C000)
#define LPC_SC_BASE                   (LPC_APB1_BASE + 0x7C000)

/* AHB peripherals                                         */
#define LPC_EMAC_BASE                (LPC_AHB_BASE + 0x00000)
#define LPC_GPDMA_BASE               (LPC_AHB_BASE + 0x04000)
#define LPC_GPDMAH0_BASE              (LPC_AHB_BASE + 0x04100)
#define LPC_GPDMAH1_BASE              (LPC_AHB_BASE + 0x04120)
#define LPC_GPDMAH2_BASE              (LPC_AHB_BASE + 0x04140)
#define LPC_GPDMAH3_BASE              (LPC_AHB_BASE + 0x04160)
#define LPC_GPDMAH4_BASE              (LPC_AHB_BASE + 0x04180)
#define LPC_GPDMAH5_BASE              (LPC_AHB_BASE + 0x041A0)
#define LPC_GPDMAH6_BASE              (LPC_AHB_BASE + 0x041C0)

```

```

#define LPC_GPDMA7_BASE      (LPC_AHB_BASE + 0x041E0)
#define LPC_USB_BASE         (LPC_AHB_BASE + 0x0C000)

/* GPIOs
 */
#define LPC_GPIO0_BASE        (LPC_GPIO_BASE + 0x00000)
#define LPC_GPIO1_BASE        (LPC_GPIO_BASE + 0x00020)
#define LPC_GPIO2_BASE        (LPC_GPIO_BASE + 0x00040)
#define LPC_GPIO3_BASE        (LPC_GPIO_BASE + 0x00060)
#define LPC_GPIO4_BASE        (LPC_GPIO_BASE + 0x00080)

/*****************************************/
/* Peripheral declaration */
/*****************************************/

#define LPC_SC                ((LPC_SC_TypeDef      *) LPC_SC_BASE)
#define LPC_GPIO               ((LPC_GPIO_TypeDef    *) LPC_GPIO0_BASE)
#define LPC_GPIO1              ((LPC_GPIO_TypeDef    *) LPC_GPIO1_BASE)
#define LPC_GPIO2              ((LPC_GPIO_TypeDef    *) LPC_GPIO2_BASE)
#define LPC_GPIO3              ((LPC_GPIO_TypeDef    *) LPC_GPIO3_BASE)
#define LPC_GPIO4              ((LPC_GPIO_TypeDef    *) LPC_GPIO4_BASE)
#define LPC_WDT               ((LPC_WDT_TypeDef   *) LPC_WDT_BASE)
#define LPC_TIM0              ((LPC_TIM_TypeDef   *) LPC_TIM0_BASE)
#define LPC_TIM1              ((LPC_TIM_TypeDef   *) LPC_TIM1_BASE)
#define LPC_TIM2              ((LPC_TIM_TypeDef   *) LPC_TIM2_BASE)
#define LPC_TIM3              ((LPC_TIM_TypeDef   *) LPC_TIM3_BASE)
#define LPC_RIT               ((LPC_RIT_TypeDef  *) LPC_RIT_BASE)
#define LPC_UART0             ((LPC_UART_TypeDef *) LPC_UART0_BASE)
#define LPC_UART1             ((LPC_UART_TypeDef *) LPC_UART1_BASE)
#define LPC_UART2             ((LPC_UART_TypeDef *) LPC_UART2_BASE)
#define LPC_UART3             ((LPC_UART_TypeDef *) LPC_UART3_BASE)
#define LPC_PWM1              ((LPC_PWM_TypeDef  *) LPC_PWM1_BASE)
#define LPC_I2C0               ((LPC_I2C_TypeDef  *) LPC_I2C0_BASE)
#define LPC_I2C1               ((LPC_I2C_TypeDef  *) LPC_I2C1_BASE)
#define LPC_I2C2               ((LPC_I2C_TypeDef  *) LPC_I2C2_BASE)
#define LPC_I2S                ((LPC_I2S_TypeDef  *) LPC_I2S_BASE)
#define LPC_SPI               ((LPC_SPI_TypeDef  *) LPC_SPI_BASE)
#define LPC_RTC               ((LPC_RTC_TypeDef  *) LPC_RTC_BASE)
#define LPC_GPIOINT            ((LPC_GPIOINT_TypeDef *) LPC_GPIOINT_BASE)
#define LPC_PINCON             ((LPC_PINCON_TypeDef *) LPC_PINCON_BASE)
#define LPC_SSP                ((LPC_SSP_TypeDef  *) LPC_SSP0_BASE)

```

```

#define LPC_SSP1           ((LPC_SSP_TypeDef      *) LPC_SSP1_BASE      )
#define LPC_ADC            ((LPC_ADC_TypeDef      *) LPC_ADC_BASE      )
#define LPC_DAC            ((LPC_DAC_TypeDef      *) LPC_DAC_BASE      )
#define LPC_CANAF_RAM      ((LPC_CANAF_RAM_TypeDef *) LPC_CANAF_RAM_BASE)
#define LPC_CANAF          ((LPC_CANAF_TypeDef      *) LPC_CANAF_BASE      )
#define LPC_CANCR          ((LPC_CANCR_TypeDef      *) LPC_CANCR_BASE      )
#define LPC_CAN1            ((LPC_CAN_TypeDef      *) LPC_CAN1_BASE      )
#define LPC_CAN2            ((LPC_CAN_TypeDef      *) LPC_CAN2_BASE      )
#define LPC_MCPWM          ((LPC_MCPWM_TypeDef      *) LPC_MCPWM_BASE      )
#define LPC_QEI             ((LPC_QEI_TypeDef      *) LPC_QEI_BASE      )
#define LPC_EMAC            ((LPC_EMAC_TypeDef      *) LPC_EMAC_BASE      )
#define LPC_GPDMA           ((LPC_GPDMA_TypeDef      *) LPC_GPDMA_BASE      )
#define LPC_GPDMACH0        ((LPC_GPDMACH_TypeDef *) LPC_GPDMACH0_BASE )
#define LPC_GPDMACH1        ((LPC_GPDMACH_TypeDef *) LPC_GPDMACH1_BASE )
#define LPC_GPDMACH2        ((LPC_GPDMACH_TypeDef *) LPC_GPDMACH2_BASE )
#define LPC_GPDMACH3        ((LPC_GPDMACH_TypeDef *) LPC_GPDMACH3_BASE )
#define LPC_GPDMACH4        ((LPC_GPDMACH_TypeDef *) LPC_GPDMACH4_BASE )
#define LPC_GPDMACH5        ((LPC_GPDMACH_TypeDef *) LPC_GPDMACH5_BASE )
#define LPC_GPDMACH6        ((LPC_GPDMACH_TypeDef *) LPC_GPDMACH6_BASE )
#define LPC_GPDMACH7        ((LPC_GPDMACH_TypeDef *) LPC_GPDMACH7_BASE )
#define LPC_USB              ((LPC_USB_TypeDef      *) LPC_USB_BASE      )

/**
 * @}
 */

#endif // __LPC17xx_H__


/*-----*/
// @filename Systemsymbols.h
// @version 0.00
// @author Alberto Palomo Alonso
// @brief Este es el archivo donde son guardados los simbolos del
// sistema utilizados para los

```

```
// diferentes archivos que necesiten una llamada a
este tipo de definiciones.                                //

// @category Esencial.

// @map @none //

// @types //

// @end //

//-----**/


#ifndef LPC17XX
#define LPC17XX

#include "LPC17XX.H"

#endif

// Acopladores de código.

#ifndef null
#define null 0
#endif

#ifndef NULL
#define NULL 0
#endif

#define none 0
#define NONE 0
#define VOID void

// Símbolos prácticos de valores instantáneos.

#define TODO_1_8 0xFFFF
#define TODO_1_16 0xFFFFFFFF
#define TODO_1_32 0xFFFFFFFFFFFFFF

// Símbolos correspondientes a definiciones propias para facilitar lectura de
// código.

#define SUBIDA 1
#define BAJADA 0
#define ALTO 1
#define BAJO 0
#define FLANCO 1
```

```
#define NIVEL 0
#define INICIALMENTE_ACTIVO 1
#define INICIALMENTE_INACTIVO 0
#define FUNC0 0x0
#define FUNC1 0x1
#define FUNC2 0x2
#define FUNC3 0x3
#define NOPIN 0x00000000
#define PIN_00 0x00000001
#define PIN_01 0x00000002
#define PIN_02 0x00000004
#define PIN_03 0x00000008
#define PIN_04 0x00000010
#define PIN_05 0x00000020
#define PIN_06 0x00000040
#define PIN_07 0x00000080
#define PIN_08 0x00000100
#define PIN_09 0x00000200
#define PIN_10 0x00000400
#define PIN_11 0x00000800
#define PIN_12 0x00001000
#define PIN_13 0x00002000
#define PIN_14 0x00004000
#define PIN_15 0x00008000
#define PIN_16 0x00010000
#define PIN_17 0x00020000
#define PIN_18 0x00040000
#define PIN_19 0x00080000
#define PIN_20 0x00100000
#define PIN_21 0x00200000
#define PIN_22 0x00400000
#define PIN_23 0x00800000
#define PIN_24 0x01000000
#define PIN_25 0x02000000
#define PIN_26 0x04000000
#define PIN_27 0x08000000
#define PIN_28 0x10000000
#define PIN_29 0x20000000
#define PIN_30 0x40000000
```

```

#define PIN_31 0x80000000
// Símbolos correspondientes a relojes del sistema.

#define Fcpu 100000000
// 100MHz velocidad de la cpu.

#define APBvalue 4
// Valor del primer prescaler, 4 after-reset.

#define Prescaler 0
// Valor del prescaler, 0 after-reset.

#define Fclk (float)Fcpu/(float)APBvalue // Valor del reloj prescalado por APB.

#define Ftick Fclk/(float)(Prescaler+1) // Valor del reloj asociado a los contadores.

#define Ts0 0.5
// Tiempo de muestreo en segundos sin prescaler. (Muestras)

#define Fs0 (float)1/(float)Ts0
// Frecuencia de muestreo en Hz. (Muestras)

#define CsADC Fs0
// Frecuencia de muestreo del LDR.

#define CsCAP 10*Fs0 // Frecuencia de muestreo del UVA.

#define FsAudio 8000
// 3kHz de audio, Nyquist *= 2, Yo *= 8khz.

#define TsAudio (float)1/(float)FsAudio // Periodo de muestreo del audio.

#define DURACION_AUDIO 2
// 2 segundos de audio.

#define MUESTRAS_AUDIO DURACION_AUDIO*FsAudio // Muestras en los 2 segundos de audio.

#define MUESTRAS_SENO 32

#define LECTURA_AUDIO 25 // Pin que señala lectura de audio.

#define ESCRITURA_AUDIO 26 // Pin que señala escritura de audio.

#define MAX_PRES MODIFICABLES.Max_servo_p
#define MAX_TEMP MODIFICABLES.Max_servo_t
#define MIN_PRES MODIFICABLES.Min_servo_p
#define MIN_TEMP MODIFICABLES.Min_servo_t

// Constantes universales.

#define PI 3.141592
/*-----*/
//
```

```
//  
//  
// @types          Tipos utilizados para el programa.  
//  
//  
//-----**/  
  
typedef signed char           int8_t;  
typedef signed short int      int16_t;  
typedef signed int            int32_t;  
typedef signed long long     int64_t;  
  
typedef unsigned char         uint8_t;  
typedef unsigned short int    uint16_t;  
typedef unsigned int          uint32_t;  
typedef unsigned long long   uint64_t;  
  
typedef struct {               // Contadores de 8, 16 y 32 bits.  
    __IO uint8_t i;  
    __IO uint8_t j;  
    __IO uint16_t k;  
    __IO uint32_t Audio;  
    __IO uint32_t Segundos;  
    __IO uint32_t RITicks;  
}Counters_t;  
  
typedef struct {  
    uint8_t CHART;  
}State_t;  
  
typedef struct {  
    float Longitud;  
    float Latitud;  
    float Altura;  
}locat_t;  
  
typedef struct {  
    float Temperatura; // En grados celsius.  
    float Presion;      // En pascales.
```

```

        float  Humedad;           // En %
        float  IndiceUV;          // En UVs.

        locat_t Lugar;           // Sitio donde el GPS nos posiciona.

        float  VelViento;         // En m/s.

        float  Brillo;            // En LUX.

}misDatos_t;

typedef struct {

    __IO uint8_t  Anemometro:1;
    __IO uint8_t  AnemometroRev:1;
    __IO uint8_t  LDR:1;
    __IO  uint8_t LDRrev:1;
    __IO  uint8_t UVA:1;
    __IO  uint8_t UVAreve:1;
    __IO  uint8_t Audio:1;
    __IO  uint8_t Audiorev:1;
    __IO uint8_t  TempRev:1;

}actualizador_t;

typedef struct {

    uint32_t  source;           // Start of source area
    uint32_t  destination; // Start of destination area
    uint32_t  next;             // Address of next strLLI in chain
    uint32_t  control;           // DMACCxControl register
} DMA_t;

typedef struct {

    float  Max_servo_t;          // Done
    float  Min_servo_t;          // Done
    float  Max_servo_p;          // Done
    float  Min_servo_p;          // Done

    uint8_t Var_medida;          // Done
    uint32_t TiempoBrillo;        // Done

}modificables_t;

/**-----//



//




```

```
//          @end      ENDFILE.  
//  
//  
//  
//-----*/
```

Archivos extensión CGI.

```
t <!DOCTYPE html>
t <html>
t   <head>
t     <meta content="text/html; charset=UTF-8" http-equiv="content-type">
#
----->      OJO QUE HAY QUE PONER LA URL CORRECTA:
t     <meta http-equiv="refresh" content="20; url=http://192.168.1.120/index.cgi">
t   <title>ESTACION METEOROLOGICA</title>
t   </head>
t   <body style="background-color:rgb(200,200,200);">
t     <h1 align="center">Estacion meteorologica</h1>
t     <br />
t     <table align="center" border="1">
t       <caption>Datos medios actuales:</caption>
t       <tr>
t         <td>Temperatura:</td>
t         <td>
c t           " %f
t         </td>
t         <td>Velocidad del viento:</td>
t         <td>
c v           " %f
t         </td>
t       </tr>
t       <tr>
t         <td>Humedad:</td>
t         <td>
c h           " %f
t         </td>
t         <td>Indice UV:</td>
t         <td>
c i           " %f
t         </td>
t       </tr>
t       <tr>
```

```
t          <td>Presion:</td>
t          <td>
c p        " %f
t          </td>
t          <td>Brillo:</td>
t          <td>
c b        " %f
t          </td>
t          </tr>
t      </table>
t      <table align="center" border="1">
t      <tr>
t          <td>Altitud:</td>
t          <td>
c a        " %f
t          </td>
t          <td>Longitud:</td>
t          <td>
c x        " %f
t          </td>
t          <td>Latitud:</td>
t          <td>
c y        " %f
t          </td>
t          </tr>
t      </table>
t      <br></br>
t      <table align="center" border="1">
t      <caption>Hora de la ultima muestra:</caption>
t      <tr>
t          <td>Anyo:</td>
t          <td>
c A        " %d
t          </td>
t          <td>Mes:</td>
t          <td>
c M        " %d
t          </td>
t          <td>Dia:</td>
```

```
t          <td>
c D        " %d
t          </td>
t          <tr>
t          <td>Hora:</td>
t          <td>
c H        " %d
t          </td>
t          <td>Minuto:</td>
t          <td>
c T        " %d
t          </td>
t          <td>Segundos:</td>
t          <td>
c S        " %d
t          </td>
t          </tr>
t          </table>
t          <br></br>
t          <br></br>
t          <table style="width: 100%" border="1" align="center">
t          <tbody>
t          <tr>
t          <td>
t          <h1 style=" text-align: center;"> Magnitudes modificables: </h1>
t          <form method="GET" action="index.cgi">
t          <br> Temperatura min. :
t          <input size="10" value="-10" name="tmin" type="text">
t          <br> Temperatura max. :
t          <input size="10" value="50" name="tmax" type="text">
t          <br> Presion min. :
t          <input size="10" value="500" name="pmin" type="text">
t          <br> Presion max. :
t          <input size="10" value="1500" name="pmax" type="text">
t          <br> Segundos encendido :
t          <input size="10" value="10" name="bsec" type="text">
t          <br> <input value="si" type="radio" name="vart"> Temperatura<br>
t          <input value="si" type="radio" name="varp"> Presion<br>
t          <br> <input value="Enviar" type="submit">
```

```
t          </form>
t          </td>
t          </tr>
t          </tbody>
t          </table>
t          <br><br>
t          <br><br>
t          <br><br>
t          <br><br>
t          <table align="center">
t          <tr>
t          <td>Autor:      Alberto Palomo Alonso.</td>
t          <td>Sistemas Electronicos Digitales Avanzados.</td>
t          <td>Universidad de Alcala - Escuela politecnica superior.</td>
t          </tr>
t          </table>
t      </body>
t  </html>
.
.
```

Alberto Palomo Alonso.

Anexo III –Hojas de datos.

ARD 2

Arduino Compatibles

Controllers, Shields, Modules & Sensors

UV Detection Sensor

ARD2-2062

- Detect the intensity of UV radiation**
- Suitable for UV Index Monitoring, DIY projects, UV-A Lamp Monitoring, Gardening, Environmental monitoring**

Description

This UV Sensor is used for detecting the intensity of incident ultraviolet (UV) radiation. This form of electromagnetic radiation has shorter wavelengths than visible radiation. This module is based on the sensor UVM-30A, which has a wide spectral range of 200nm–370nm. The module outputs electrical signal which varies with the UV intensity.



Specifications

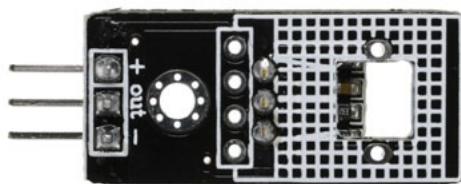
Operating Voltage	3.0–5.0VDC
Current	0.06mA (Standard)/0.1mA (Max)
Response Wavelength	200~370nm
Operating Temperature	-20~+85°C
Accuracy	±1UV Index
Colour (Board)	Black
Material	PCB

Pinout

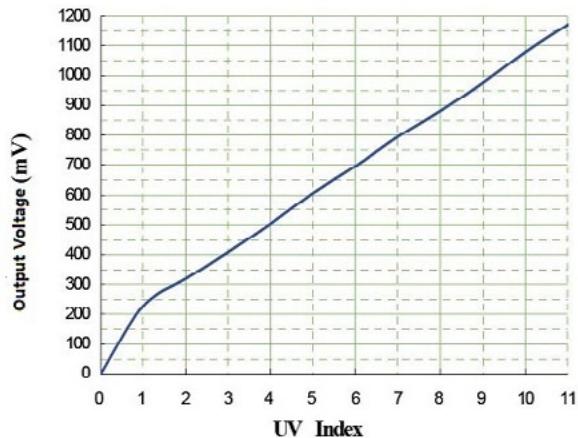
Module	Arduino	Function
+	5V	Power
-	GND	Ground Connection
Out	A0	Analog Output

Test Code

```
void setup()
{
  Serial.begin(9600); // open serial port, set the baud rate to
9600 bps
}
void loop()
{
  int sensorValue;
  sensorValue = analogRead(0); //connect UV sensors to Analog 0
  Serial.println(sensorValue); //print the value to serial
  delay(200);
}
```



ARD 2

Arduino Compatibles
Controllers, Shields, Modules & Sensors

BMP180

Digital pressure sensor

Bosch Sensortec



BOSCH
Invented for life

BMP180 Data sheet

Document revision	2.5
Document release date	5 April 2013
Document number	BST-BMP180-DS000-09
Technical reference code(s)	0 273 300 244
Notes	Data in this document are subject to change without notice. Product photos and pictures are for illustration purposes only and may differ from the real product's appearance.



BMP180

DIGITAL PRESSURE SENSOR

Key features

Pressure range:	300 ... 1100hPa (+9000m ... -500m relating to sea level)
Supply voltage:	1.8 ... 3.6V (V_{DD}) 1.62V ... 3.6V (V_{DDIO})
Package:	LGA package with metal lid Small footprint: 3.6mm x 3.8mm Super-flat: 0.93mm height
Low power:	5µA at 1 sample / sec. in standard mode
Low noise:	0.06hPa (0.5m) in ultra low power mode 0.02hPa (0.17m) advanced resolution mode
- Temperature measurement included	
- I ² C interface	
- Fully calibrated	
- Pb-free, halogen-free and RoHS compliant,	
- MSL 1	

Typical applications

- Enhancement of GPS navigation (dead-reckoning, slope detection, etc.)
- In- and out-door navigation
- Leisure and sports
- Weather forecast
- Vertical velocity indication (rise/sink speed)



BMP180 general description

The BMP180 is the function compatible successor of the BMP085, a new generation of high precision digital pressure sensors for consumer applications.

The ultra-low power, low voltage electronics of the BMP180 is optimized for use in mobile phones, PDAs, GPS navigation devices and outdoor equipment. With a low altitude noise of merely 0.25m at fast conversion time, the BMP180 offers superior performance. The I²C interface allows for easy system integration with a microcontroller.

The BMP180 is based on piezo-resistive technology for EMC robustness, high accuracy and linearity as well as long term stability.

Robert Bosch is the world market leader for pressure sensors in automotive applications. Based on the experience of over 400 million pressure sensors in the field, the BMP180 continues a new generation of micro-machined pressure sensors.



Index of Contents

1. ELECTRICAL CHARACTERISTICS	6
2. ABSOLUTE MAXIMUM RATINGS.....	8
3. OPERATION	9
3.1 GENERAL DESCRIPTION	9
3.2 GENERAL FUNCTION AND APPLICATION SCHEMATICS.....	9
3.3 MEASUREMENT OF PRESSURE AND TEMPERATURE	11
3.3.1 Hardware pressure sampling accuracy modes	12
3.3.2 Software pressure sampling accuracy modes	13
3.4 CALIBRATION COEFFICIENTS.....	13
3.5 CALCULATING PRESSURE AND TEMPERATURE	14
3.6 CALCULATING ABSOLUTE ALTITUDE.....	16
3.7 CALCULATING PRESSURE AT SEA LEVEL	17
4. GLOBAL MEMORY MAP	18
5. I²C INTERFACE	19
5.1 I ² C SPECIFICATION.....	19
5.2 DEVICE AND REGISTER ADDRESS.....	20
5.3 I ² C PROTOCOL	20
5.4 START TEMPERATURE AND PRESSURE MEASUREMENT	21
5.5 READ A/D CONVERSION RESULT OR E ² PROM DATA	22
6. PACKAGE.....	23
6.1 PIN CONFIGURATION	23
6.2 OUTLINE DIMENSIONS	24
6.2.1 Bottom view	24
6.2.2 Top view	25
6.2.3 Side view	25
6.3 MOISTURE SENSITIVITY LEVEL AND SOLDERING.....	26
6.4 RoHS COMPLIANCY.....	26
6.5 MOUNTING AND ASSEMBLY RECOMMENDATIONS	26
7. LEGAL DISCLAIMER.....	27

**BOSCH**Data sheet
BMP180

Page 5

7.1 ENGINEERING SAMPLES	27
7.2 PRODUCT USE.....	27
7.3 APPLICATION EXAMPLES AND HINTS	27
8. DOCUMENT HISTORY AND MODIFICATION.....	28



1. Electrical characteristics

If not stated otherwise, the given values are ± 3 -Sigma values over temperature/voltage range in the given operation mode. All values represent the new parts specification; additional solder drift is shown separately.

Table 1: Operating conditions, output signal and mechanical characteristics

Parameter	Symbol	Condition	Min	Typ	Max	Units
Operating temperature	T_A	operational	-40		+85	°C
		full accuracy	0		+65	
Supply voltage	V_{DD}	ripple max. 50mVpp	1.8	2.5	3.6	V
			1.62	2.5	3.6	
Supply current @ 1 sample / sec. 25°C	I_{DDLOW}	ultra low power mode		3		µA
	I_{DDSTD}	standard mode		5		µA
	I_{DDHR}	high resolution mode		7		µA
	I_{DDUHR}	Ultra high res. mode		12		µA
	I_{DDAR}	Advanced res. mode		32		µA
Peak current	I_{peak}	during conversion		650	1000	µA
Standby current	I_{DDSBM}	@ 25°C		0.1	4 ¹	µA
Relative accuracy pressure $V_{DD} = 3.3V$		950 ... 1050 hPa @ 25 °C		±0.12		hPa
				±1.0		m
		700 ... 900hPa 25 ... 40 °C		±0.12		hPa
				±1.0		m
Absolute accuracy pressure $V_{DD} = 3.3V$		300 ... 1100 hPa 0 ... +65 °C	-4.0	-1.0*	+2.0	hPa
		300 ... 1100 hPa -20 ... 0 °C	-6.0	-1.0*	+4.5	hPa
Resolution of output data		pressure		0.01		hPa
		temperature		0.1		°C
Noise in pressure		see table on page 12-13				
Absolute accuracy temperature $V_{DD} = 3.3V$		@ 25 °C	-1.5	±0.5	+1.5	°C
		0 ... +65 °C	-2.0	±1.0	+2.0	°C

¹ at 85°C



Conversion time pressure	t _{c_p_low}	ultra low power mode		3	4.5	ms
	t _{c_p_std}	standard mode		5	7.5	ms
	t _{c_p_hr}	high resolution mode		9	13.5	ms
	t _{c_p_luhr}	ultra high res. mode		17	25.5	ms
	t _{c_p_ar}	Advanced res. mode		51	76.5	ms
Conversion time temperature	t _{C_temp}	standard mode		3	4.5	ms
Serial data clock	f _{SCL}				3.4	MHz
Solder drifts		Minimum solder height 50µm	-0.5		+2	hPa
Long term stability**		12 months		±1.0		hPa

* The typical value is: -1±1

** Long term stability is specified in the full accuracy operating pressure range 0 ... 65°C



2. Absolute maximum ratings

Table 2: Absolute maximum ratings

Parameter	Condition	Min	Max	Units
Storage temperature		-40	+85	°C
Supply voltage	all pins	-0.3	+4.25	V
ESD rating	HBM, R = 1.5kΩ, C = 100pF		±2	kV
Overpressure			10,000	hPa

The BMP180 has to be handled as

Electrostatic Sensitive Device (ESD).



Figure 1: ESD



3. Operation

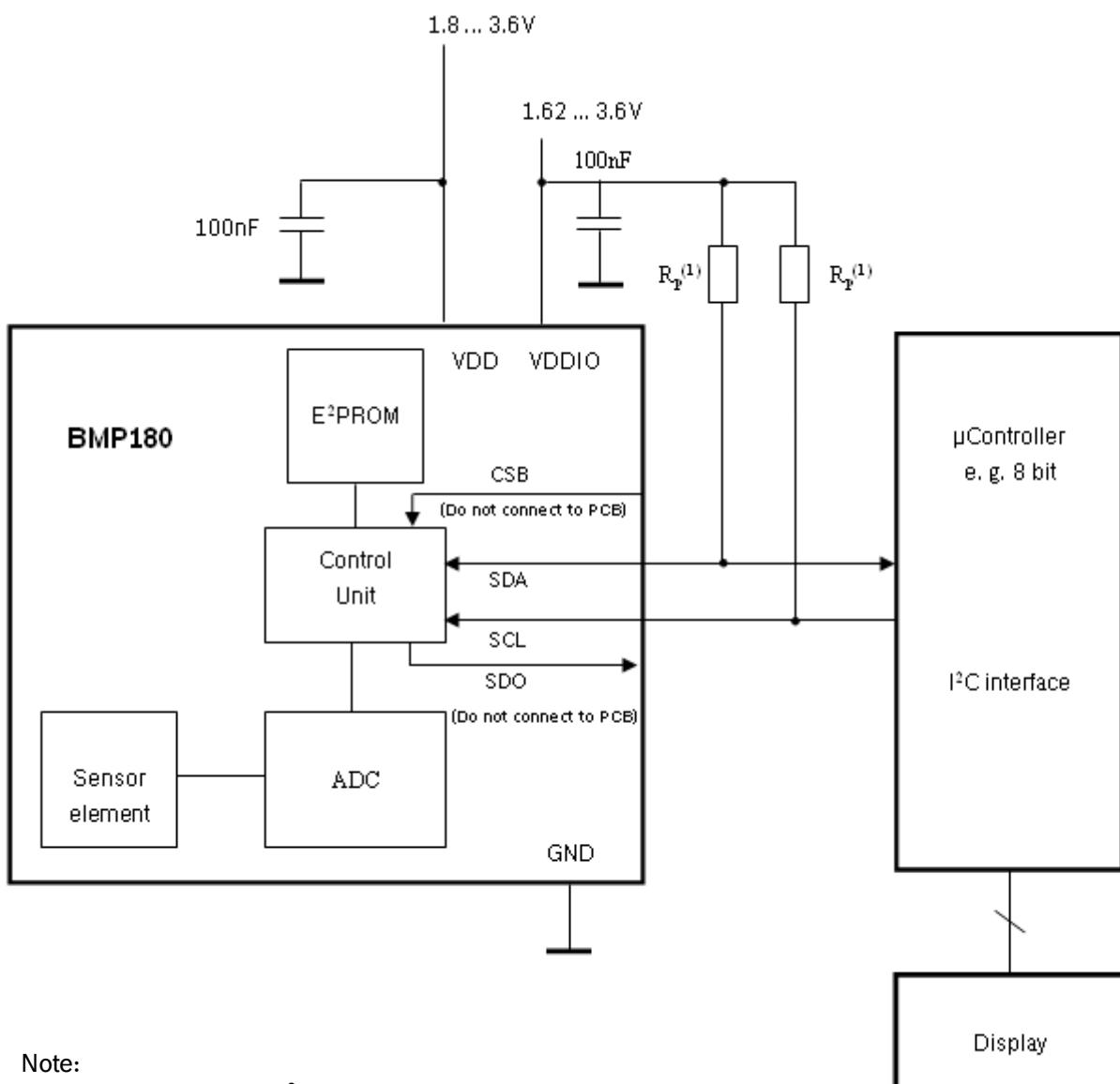
3.1 General description

The BMP180 is designed to be connected directly to a microcontroller of a mobile device via the I²C bus. The pressure and temperature data has to be compensated by the calibration data of the E²PROM of the BMP180.

3.2 General function and application schematics

The BMP180 consists of a piezo-resistive sensor, an analog to digital converter and a control unit with E²PROM and a serial I²C interface. The BMP180 delivers the uncompensated value of pressure and temperature. The E²PROM has stored 176 bit of individual calibration data. This is used to compensate offset, temperature dependence and other parameters of the sensor.

- UP = pressure data (16 to 19 bit)
- UT = temperature data (16 bit)



Note:

(1) Pull-up resistors for I²C bus, $R_p = 2.2\text{k}\Omega \dots 10\text{k}\Omega$, typ. $4.7\text{k}\Omega$

Figure 2: Typical application circuit

3.3 Measurement of pressure and temperature

For all calculations presented here an ANSI C code is available from Bosch Sensortec ("BMP180_AP").

The microcontroller sends a start sequence to start a pressure or temperature measurement. After converting time, the result value (UT or UP, respectively) can be read via the I²C interface. For calculating temperature in °C and pressure in hPa, the calibration data has to be used. These constants can be read out from the BMP180 E²PROM via the I²C interface at software initialization.

The sampling rate can be increased up to 128 samples per second (standard mode) for dynamic measurement. In this case, it is sufficient to measure the temperature only once per second and to use this value for all pressure measurements during the same period.

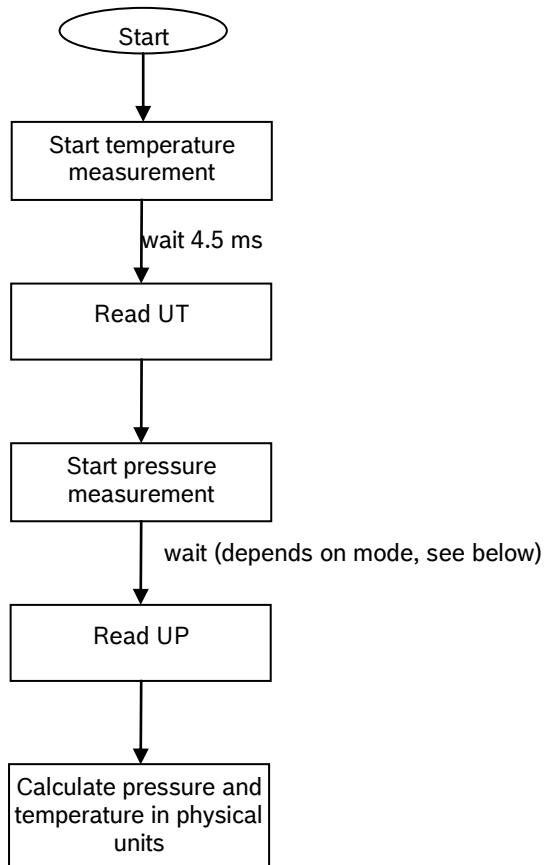


Figure 3: Measurement flow BMP180



3.3.1 Hardware pressure sampling accuracy modes

By using different modes the optimum compromise between power consumption, speed and resolution can be selected, see below table.

Table 3: Overview of BMP180 hardware accuracy modes, selected by driver software via the variable *oversampling_setting*

Mode	Parameter <i>oversampling_setting</i>	Internal number of samples	Conversion time pressure max. [ms]	Avg. current @ 1 sample/s typ. [µA]	RMS noise typ. [hPa]	RMS noise typ. [m]
ultra low power	0	1	4.5	3	0.06	0.5
standard	1	2	7.5	5	0.05	0.4
high resolution	2	4	13.5	7	0.04	0.3
ultra high resolution	3	8	25.5	12	0.03	0.25

For further information on noise characteristics see the relevant application note “Noise in pressure sensor applications”.

All modes can be performed at higher speeds, e.g. up to 128 times per second for standard mode, with the current consumption increasing proportionally to the sample rate.



3.3.2 Software pressure sampling accuracy modes

For applications where a low noise level is critical, averaging is recommended if the lower bandwidth is acceptable. Oversampling can be enabled using the software API driver (with OSR = 3).

Table 4: Overview of BMP180 software accuracy mode, selected by driver software via the variable *software_oversampling_setting*

Mode	Parameter <i>oversampling_setting</i>	software_oversampling_setting	Conversion time pressure max. [ms]	Avg. current @ 1 sample/s typ. [μ A]	RMS noise typ. [hPa]	RMS noise typ. [m]
Advanced resolution	3	1	76.5	32	0.02	0.17

3.4 Calibration coefficients

The 176 bit E²PROM is partitioned in 11 words of 16 bit each. These contain 11 calibration coefficients. Every sensor module has individual coefficients. Before the first calculation of temperature and pressure, the master reads out the E²PROM data.

The data communication can be checked by checking that none of the words has the value 0 or 0xFFFF.

Table 5: Calibration coefficients

Parameter	BMP180 reg adr	
	MSB	LSB
AC1	0xAA	0xAB
AC2	0xAC	0xAD
AC3	0xAE	0xAF
AC4	0xB0	0xB1
AC5	0xB2	0xB3
AC6	0xB4	0xB5
B1	0xB6	0xB7
B2	0xB8	0xB9
MB	0xBA	0xBB
MC	0xBC	0xBD
MD	0xBE	0xBF



3.5 Calculating pressure and temperature

The mode (ultra low power, standard, high, ultra high resolution) can be selected by the variable *oversampling_setting* (0, 1, 2, 3) in the C code.

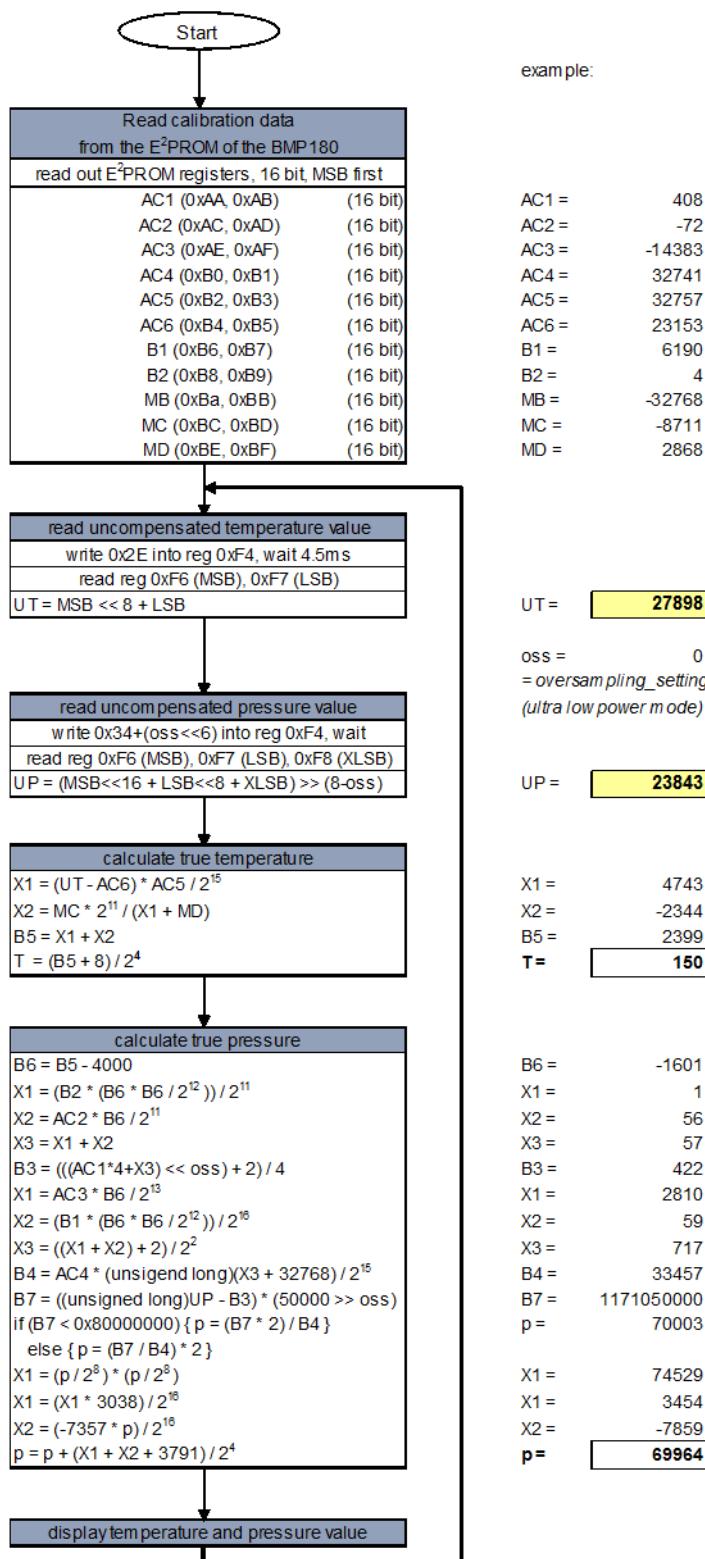
Calculation of true temperature and pressure in steps of 1Pa (= 0.01hPa = 0.01mbar) and temperature in steps of 0.1°C.

The following figure shows the detailed algorithm for pressure and temperature measurement.

This algorithm is available to customers as reference C source code ("BMP180_API") from Bosch Sensortec and via its sales and distribution partners. **Please contact your Bosch Sensortec representative for details.**



Calculation of pressure and temperature for BMP180



example:

C code function: type:

bmp180_get_cal_param

AC1 =	408	short
AC2 =	-72	short
AC3 =	-14383	short
AC4 =	32741	unsigned short
AC5 =	32757	unsigned short
AC6 =	23153	unsigned short
B1 =	6190	short
B2 =	4	short
MB =	-32768	short
MC =	-8711	short
MD =	2868	short

bmp180_get_ut

UT = 27898 long

OSS = 0 short (0 .. 3)
= oversampling_setting (ultra low power mode)

bmp180_get_up

UP = 23843 long

X1 = 4743 long
X2 = -2344 long
B5 = 2399 long
T = 150 temp in 0.1°C long

bmp180_get_temperature

B6 = -1601 long
X1 = 1 long
X2 = 56 long
X3 = 57 long
B3 = 422 long
X1 = 2810 long
X2 = 59 long
X3 = 717 long
B4 = 33457 unsigned long
B7 = 1171050000 unsigned long
P = 70003 long

X1 = 74529 long
X1 = 3454 long
X2 = -7859 long
P = 69964 press. in Pa long

Figure 4: Algorithm for pressure and temperature measurement

3.6 Calculating absolute altitude

With the measured pressure p and the pressure at sea level p_0 e.g. 1013.25hPa, the altitude in meters can be calculated with the international barometric formula:

$$\text{altitude} = 44330 * \left(1 - \left(\frac{p}{p_0} \right)^{\frac{1}{5.255}} \right)$$

Thus, a pressure change of $\Delta p = 1\text{hPa}$ corresponds to 8.43m at sea level.

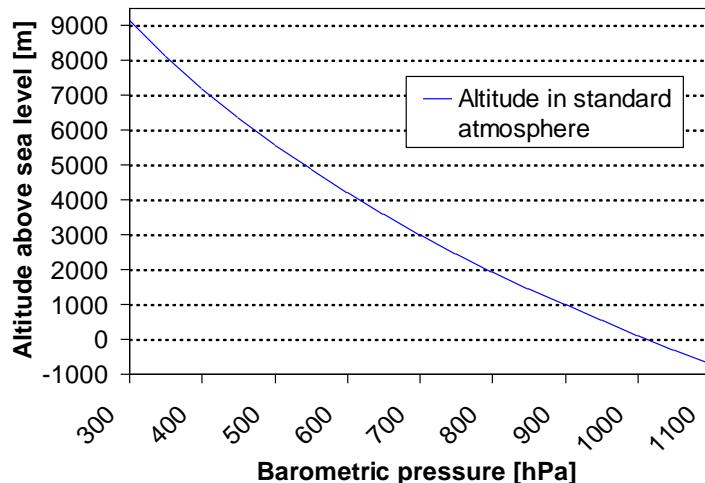


Figure 5: Transfer function: Altitude over sea level – Barometric pressure



3.7 Calculating pressure at sea level

With the measured pressure p and the absolute altitude the pressure at sea level can be calculated:

$$p_0 = \frac{p}{\left(1 - \frac{\text{altitude}}{44330}\right)^{5.255}}$$

Thus, a difference in altitude of $\Delta\text{altitude} = 10\text{m}$ corresponds to 1.2hPa pressure change at sea level.



4. Global Memory Map

The memory map below shows all externally accessible data registers which are needed to operate BMP180. The left columns show the memory addresses. The columns in the middle depict the content of each register bit. The colors of the bits indicate whether they are read-only, write-only or read- and writable. The memory is volatile so that the writable content has to be re-written after each power-on.

Not all register addresses are shown. These registers are reserved for further Bosch factory testing and trimming.

Register Name	Register Adress	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Reset state
out_xlsb	F8h	adc_out_xlsb<7:3>					0	0	0	00h
out_lsb	F7h	adc_out_lsb<7:0>								
out_msb	F6h	adc_out_msb<7:0>								
ctrl_meas	F4h	oss<1:0>		sco	measurement control					
soft reset	E0h	reset								
id	D0h	id<7:0>								
calib21 downto calib0	BFh downto AAh	calib21<7:0> downto calib0<7:0>								

Registers: Control registers | Calibration registers | Data registers | Fixed
Type: read / write | read only | read only | read only

Figure 6: Memory map

Measurement control (register F4h <4:0>): Controls measurements. Refer to Figure 6 for usage details.

Sco (register F4h <5>): Start of conversion. The value of this bit stays “1” during conversion and is reset to “0” after conversion is complete (data registers are filled).

Oss (register F4h <7:6>): controls the oversampling ratio of the pressure measurement (00b: single, 01b: 2 times, 10b: 4 times, 11b: 8 times).

Soft reset (register E0h): Write only register. If set to 0xB6, will perform the same sequence as power on reset.

Chip-id (register D0h): This value is fixed to 0x55 and can be used to check whether communication is functioning.

After conversion, data registers can be read out in any sequence (i.e. MSB first or LSB first). Using a burst read is not mandatory.



5. I²C Interface

- I²C is a digital two wire interface
- Clock frequencies up to 3.4Mbit/sec. (I²C standard, fast and high-speed mode supported)
- SCL and SDA needs a pull-up resistor, typ. 4.7kOhm to V_{DDIO}
(one resistor each for all the I²C bus)

The I²C bus is used to control the sensor, to read calibration data from the E²PROM and to read the measurement data when A/D conversion is finished. SDA (serial data) and SCL (serial clock) have open-drain outputs.

For detailed I²C-bus specification please refer to:
http://www.nxp.com/acrobat_download/literature/9398/39340011.pdf

5.1 I²C specification

Table 6: Electrical parameters for the I²C interface

Parameter	Symbol	Min.	Typ	Max.	Units
Clock input frequency	f _{SCL}			3.4	MHz
Input-low level	V _{IL}	0		0.2 * V _{DDIO}	V
Input-high level	V _{IH}	0.8 * V _{DDIO}		V _{DDIO}	V
Voltage output low level @ V _{DDIO} = 1.62V, I _{OL} = 3mA	V _{OL}			0.3	V
SDA and SCL pull-up resistor	R _{pull-up}	2.2		10	kOhm
SDA sink current @ V _{DDIO} = 1.62V, V _{OL} = 0.3V	I _{SDA_sink}		9		mA
Start-up time after power-up, before first communication	t _{Start}	10			Ms



5.2 Device and register address

The BMP180 module address is shown below. The LSB of the device address distinguishes between read (1) and write (0) operation, corresponding to address 0xEF (read) and 0xEE (write).

Table 7: BMP180 addresses

A7	A6	A5	A4	A3	A2	A1	W/R
1	1	1	0	1	1	1	0/1

5.3 I²C protocol

The I²C interface protocol has special bus signal conditions. Start (S), stop (P) and binary data conditions are shown below. At start condition, SCL is high and SDA has a falling edge. Then the slave address is sent. After the 7 address bits, the direction control bit R/W selects the read or write operation. When a slave device recognizes that it is being addressed, it should acknowledge by pulling SDA low in the ninth SCL (ACK) cycle.

At stop condition, SCL is also high, but SDA has a rising edge. Data must be held stable at SDA when SCL is high. Data can change value at SDA only when SCL is low.

Even though V_{DDIO} can be powered on before V_{DD} , there is a chance of excessive power consumption (a few mA) if this sequence is used, and the state of the output pins is undefined so that the bus can be locked. Therefore, V_{DD} *must* be powered before V_{DDIO} unless the limitations above are understood and not critical.

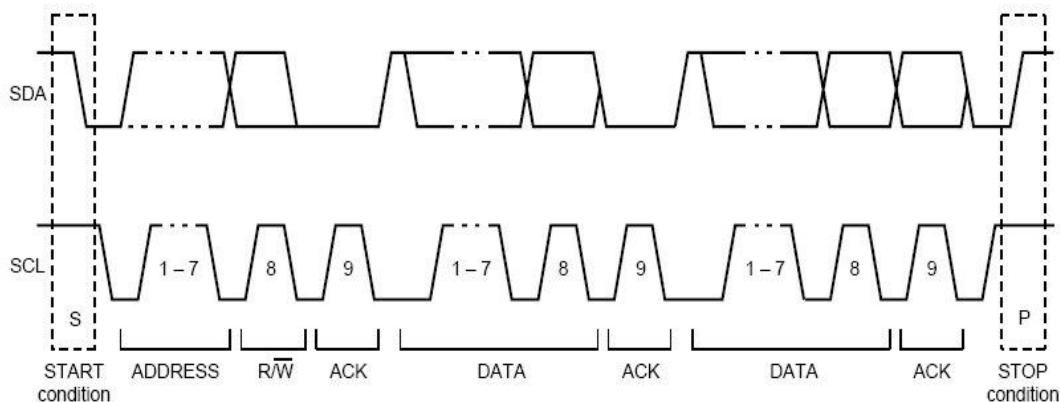


Figure 7: I²C protocol



5.4 Start temperature and pressure measurement

The timing diagrams to start the measurement of the temperature value UT and pressure value UP are shown below. After start condition the master sends the device address write, the register address and the control register data. The BMP180 sends an acknowledgement (ACKS) every 8 data bits when data is received. The master sends a stop condition after the last ACKS.

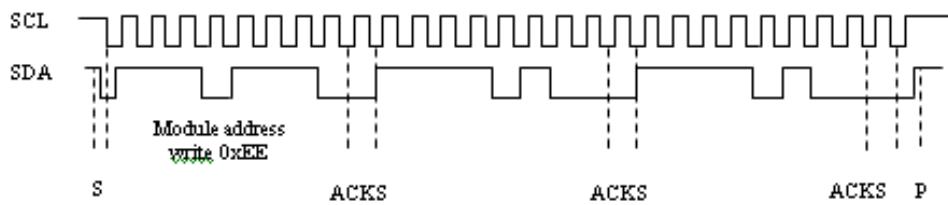


Figure 8: Timing diagram for starting pressure measurement

Abbreviations:

S	Start
P	Stop
ACKS	Acknowledge by Slave
ACKM	Acknowledge by Master
NACKM	Not Acknowledge by Master

Table 8: Control registers values for different internal oversampling_setting (oss)

Measurement	Control register value (register address 0xF4)	Max. conversion time [ms]
Temperature	0x2E	4.5
Pressure (oss = 0)	0x34	4.5
Pressure (oss = 1)	0x74	7.5
Pressure (oss = 2)	0xB4	13.5
Pressure (oss = 3)	0xF4	25.5



5.5 Read A/D conversion result or E²PROM data

To read out the temperature data word UT (16 bit), the pressure data word UP (16 to 19 bit) and the E²PROM data proceed as follows:

After the start condition the master sends the module address write command and register address. The register address selects the read register:

E²PROM data registers 0xAA to 0xBF

Temperature or pressure value UT or UP 0xF6 (MSB), 0xF7 (LSB), optionally 0xF8 (XLSB)

Then the master sends a restart condition followed by the module address read that will be acknowledged by the BMP180 (ACKS). The BMP180 sends first the 8 MSB, acknowledged by the master (ACKM), then the 8 LSB. The master sends a "not acknowledge" (NACKM) and finally a stop condition.

Optionally for ultra high resolution, the XLSB register with address 0xF8 can be read to extend the 16 bit word to up to 19 bits; refer to the application programming interface (API) software rev. 1.1 ("BMP180_API", available from Bosch Sensortec).

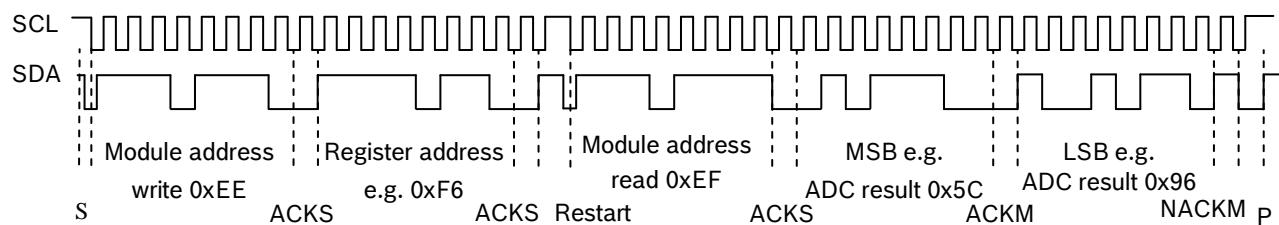


Figure 9: Timing diagram read 16 bit A/D conversion result



6. Package

6.1 Pin configuration

Picture shows the device in top view. Device pins are shown here transparently only for orientation purposes.

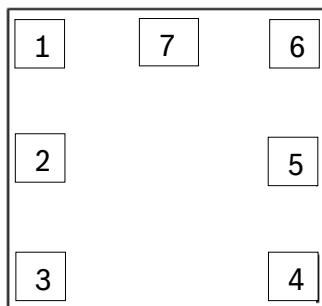


Figure 10: Layout pin configuration BMP180

Table 9: Pin configuration BMP180

in No	Name	Function
1	CSB*	Chip select
2	VDD	Power supply
3	VDDIO	Digital power supply
4	SDO*	SPI output
5	SCL	I2C serial bus clock input
6	SDA	I2C serial bus data (or SPI input)
7	GND	Ground

* A pin compatible product variant with SPI interface is possible upon customer's request. For I²C (standard case) CSB and SDO are not used, they have to be left open.

All pins have to be soldered to the PCB for symmetrical stress input even though they are not connected internally.

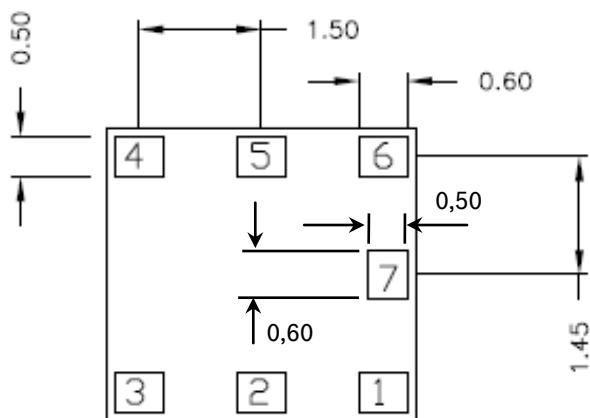


6.2 Outline dimensions

The sensor housing is a 7Pin LGA package with metal lid. Its dimensions are 3.60mm (± 0.1 mm) x 3.80mm (± 0.1 mm) x 0.93mm (± 0.07 mm).

Note: All dimensions are in mm.

6.2.1 Bottom view



BOTTOM VIEW

Figure 11: Bottom view BMP180



6.2.2 Top view

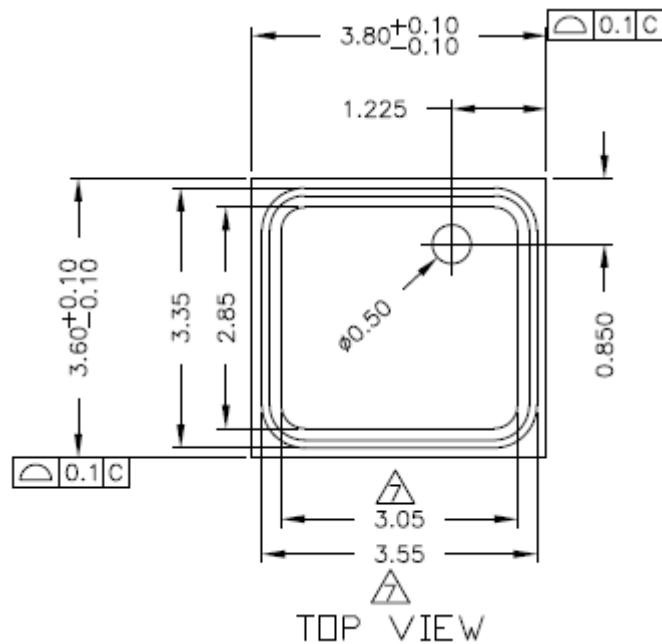


Figure 12: Top view BMP180

6.2.3 Side view

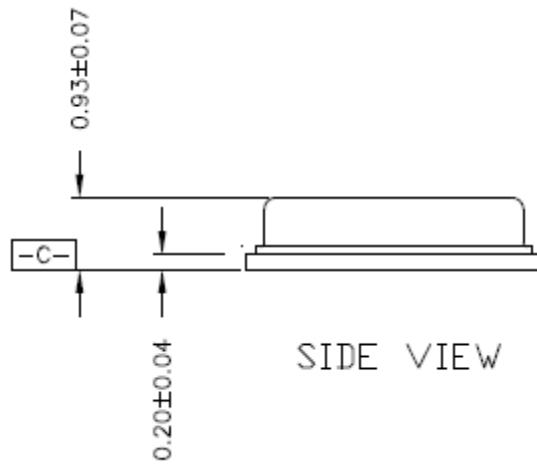


Figure 13: Side view BMP180



6.3 Moisture sensitivity level and soldering

The BMP180 is classified MSL 1 (moisture sensitivity level) according to IPC/JEDEC standards J-STD-020D and J-STD-033A.

The device can be soldered Pb-free with a peak temperature of 260°C for 20 to 40 sec. The minimum height of the solder after reflow shall be at least 50µm. This is required for good mechanical decoupling between the sensor device and the printed circuit board (PCB).

To ensure good solder-ability, the devices shall be stored at room temperature (20°C).

The soldering process can lead to an offset shift.

6.4 RoHS compliancy

The BMP180 sensor meets the requirements of the EC directive "Restriction of hazardous substances (RoHS)", please refer also to:

"Directive 2002/95/EC of the European Parliament and of the Council of 27 January 2003 on the restriction of the use of certain hazardous substances in electrical and electronic equipment".

The BMP180 sensor is also halogen-free.

6.5 Mounting and assembly recommendations

In order to achieve the specified performance for your design, the following recommendations and the "Handling, soldering & mounting instructions BMP180" should be taken into consideration when mounting a pressure sensor on a printed-circuit board (PCB):

- The clearance above the metal lid shall be 0.1mm at minimum.
- For the device housing appropriate venting needs to be provided in case the ambient pressure shall be measured.
- Liquids shall not come into direct contact with the device.
- During operation the sensor is sensitive to light, which can influence the accuracy of the measurement (photo-current of silicon).
- The BMP180 shall not be placed close to the fast heating parts. In case of gradients > 3°C/sec. it is recommended to follow Bosch Sensortec application note ANP015, "Correction of errors induced by fast temperature changes". Please contact your Bosch Sensortec representative for details.



7. Legal disclaimer

7.1 Engineering samples

Engineering Samples are marked with an asterisk (*) or (e). Samples may vary from the valid technical specifications of the product series contained in this data sheet. They are therefore not intended or fit for resale to third parties or for use in end products. Their sole purpose is internal client testing. The testing of an engineering sample may in no way replace the testing of a product series. Bosch Sensortec assumes no liability for the use of engineering samples. The Purchaser shall indemnify Bosch Sensortec from all claims arising from the use of engineering samples.

7.2 Product use

Bosch Sensortec products are developed for the consumer goods industry. They may only be used within the parameters of this product data sheet. They are not fit for use in life-sustaining or security sensitive systems. Security sensitive systems are those for which a malfunction is expected to lead to bodily harm or significant property damage. In addition, they are not fit for use in products which interact with motor vehicle systems.

The resale and/or use of products are at the purchaser's own risk and his own responsibility. The examination of fitness for the intended use is the sole responsibility of the Purchaser.

The purchaser shall indemnify Bosch Sensortec from all third party claims arising from any product use not covered by the parameters of this product data sheet or not approved by Bosch Sensortec and reimburse Bosch Sensortec for all costs in connection with such claims.

The purchaser must monitor the market for the purchased products, particularly with regard to product safety, and inform Bosch Sensortec without delay of all security relevant incidents.

7.3 Application examples and hints

With respect to any examples or hints given herein, any typical values stated herein and/or any information regarding the application of the device, Bosch Sensortec hereby disclaims any and all warranties and liabilities of any kind, including without limitation warranties of non-infringement of intellectual property rights or copyrights of any third party. The information given in this document shall in no event be regarded as a guarantee of conditions or characteristics. They are provided for illustrative purposes only and no evaluation regarding infringement of intellectual property rights or copyrights or regarding functionality, performance or error has been made.



8. Document history and modification

Rev. No	Chapter	Description of modifications/changes	Date
1.0		First edition for description of serial production material – Preliminary version	
1.1	5.1	New nomenclature of pin configuration	27 July 2010
1.2	5	Design change in package – hole in Lid and without slit	13 September 2010
1.3	3.2 5.1	- Standardizing pin naming over Bosch Sensortec products – typical application circuit - Optimizing pin description, SPI description	15 December 2010
2.0	1	- Non-preliminary version - Verifying parameter through characterization	28 January 2011
2.1	3.2 4 5.3 6.1 6.2.1	- Declaration of SDO and CSB pins in the typical application circuit - Adding global memory map and bits description - Power-up sequence - Description of used interfaces - Dimension pin7	1 April 2011
2.2	6.1	Correction of the pin configuration (editorial change)	14 April 2011
2.3	3.3	Optimizing noise performance	25 May 2011
2.4	6.3	Removed shelf-life constraints	26 January 2012
	page 2	Comparison removed	
	5.1	Voltage output low level added	
	5.3	Power on sequence of V_{DD} and V_{DDIO} defined	
2.5	1	Added max values for supply current for restricted version	15 Feb 2013
	1	Added max value for standby current for restricted version	5 Apr 2013
	Figure 4	Update of calculation of algorithm for pressure and temperature measurement	
	Page 2	Changed wording from “ultra high resolution mode” to “advanced resolution mode”	

Bosch Sensortec GmbH
Gerhard-Kindler-Strasse 8
72770 Reutlingen / Germany

contact@bosch-sensortec.com
www.bosch-sensortec.com

Modifications reserved | Printed in Germany
Specifications are subject to change without notice
Revision_2.5_042013
Document number: BST-BMP180-DS000-09



FOTORRESISTENCIA LDR 4,3mm x Ø 5,1mm

C-2795

?? Los nombres registrados y marcas que se citan son propiedad de sus respectivos titulares.

DESCRIPCION GENERAL.

Fotorresistencia o resistencia dependiente de la luz, consistente en una célula de Sulfuro de Cadmio, altamente estable, encapsulada con una resina epoxi transparente, resistente a la humedad. La respuesta espectral es similar a la del ojo humano. Su nivel de resistencia aumenta cuando el nivel de luz disminuye.

Aplicaciones: Control de contraste en televisores y monitores, control automático de la iluminación en habitaciones, juguetes y juegos electrónicos, controles industriales, interruptores crepusculares, boyas y balizas de encendido automático, auto-flash, etc...

CARACTERÍSTICAS TÉCNICAS.

Modelo	Valores máximos			Características a 25°C (nota E)					
	Tensión a 25°C (Vdc)	Potencia disipable (mW)	Temperatura ambiente (°C)	Resistencia (notaA)		? (notaC) Min.(k?) Max.(k?)	Tiempos de respuesta a 10 lx (notaD) t. subida (ms) t. bajada (ms)	Respuesta espectral (pico) (nm)	
				10 lux (2856K) Min.(k?)	0 lux (notaB) Max.(M?)				
C-2795	150	90	-25 a 75	50	140	20	0.9	60	25

Notas: A) Medido con una fuente luminosa formada por una lámpara de tungsteno, trabajando a una temperatura de color de 2856K.

B)Medición efectuada 10 segundos después de retirar una iluminación incidente de 10 lux.

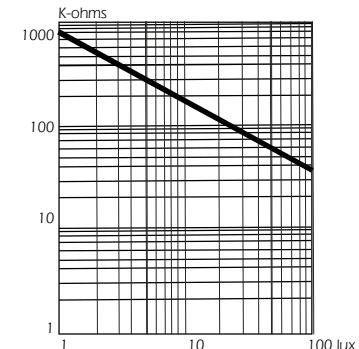
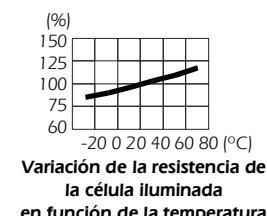
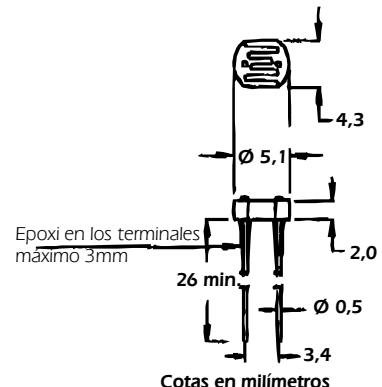
C) Sensibilidad entre 10 y 100 lux, dada por:

$$\begin{aligned} ? &= \log(R100) - \log(R10) \\ &= \log(E100) - \log(E10) \end{aligned}$$

donde R100, R10 son las resistencias a 100 y 10 lux respectivamente, y E100, E10 las iluminancias de 100 y 10 lx respectivamente.

D) Tiempo de subida es el tiempo necesario para alcanzar el 63% del nivel de saturación. Tiempo de bajada es el necesario para que la célula alcance el 37% desde el nivel saturación.

E) Todas las características están medidas con la célula LDR expuesta a la luz (100-500 lux) durante 1 o 2 horas.



CONSIDERACIONES.

Este componente está destinado para su uso por parte de profesionales, o usuarios con un nivel técnico o conocimientos suficientes, que les permita desarrollar por sí mismos los proyectos o aplicaciones deseados. Por este motivo no se facilitará asistencia técnica sobre problemas de implementación del citado componente en las aplicaciones en las que sea empleado.

Para cualquier problema relativo al funcionamiento del producto (excluidos los problemas de aplicación), póngase en contacto con nuestro **departamento técnico**. **Fax 93 432 29 95**.

Correo electrónico: sat@fadisel.com. La documentación técnica de este producto responde a una transcripción de la proporcionada por el fabricante.

Los productos de la familia "Componentes" de Cebek disponen de **1 año de garantía** a partir de la fecha de compra. Quedan excluidos el trato o manipulación incorrectos.

Disponemos de más productos que pueden interesarle, visítenos en: www.fadisel.com ó **SOLICITE GRATUITAMENTE nuestro catálogo**.

Microphone Amplifier with AGC and Low-Noise Microphone Bias

MAX9814

General Description

The MAX9814 is a low-cost, high-quality microphone amplifier with automatic gain control (AGC) and low-noise microphone bias. The device features a low-noise preamplifier, variable gain amplifier (VGA), output amplifier, microphone-bias-voltage generator and AGC control circuitry.

The low-noise preamplifier has a fixed 12dB gain, while the VGA gain automatically adjusts from 20dB to 0dB, depending on the output voltage and the AGC threshold. The output amplifier offers selectable gains of 8dB, 18dB, and 28dB. With no compression, the cascade of the amplifiers results in an overall gain of 40dB, 50dB, or 60dB. A trilevel digital input programs the output amplifier gain. An external resistive divider controls the AGC threshold and a single capacitor programs the attack/release times. A trilevel digital input programs the ratio of attack-to-release time. The hold time of the AGC is fixed at 30ms. The low-noise microphone-bias-voltage generator can bias most electret microphones.

The MAX9814 is available in the space-saving 12-bump UCSP™ (1.5mm x 2mm) and 14-pin TDFN packages. This device is specified over the -40°C to +85°C extended temperature range.

Applications

Digital Still Cameras	Two-Way Communicators
Digital Video Cameras	High-Quality Portable Recorders
PDAs	IP Phones/Telephone Conferencing
Bluetooth Headsets	
Entertainment Systems (e.g., Karaoke)	

Features

- ◆ Automatic Gain Control (AGC)
- ◆ Three Gain Settings (40dB, 50dB, 60dB)
- ◆ Programmable Attack Time
- ◆ Programmable Attack and Release Ratio
- ◆ 2.7V to 5.5V Supply Voltage Range
- ◆ Low Input-Referred Noise Density of 30nV/ $\sqrt{\text{Hz}}$
- ◆ Low THD: 0.04% (typ)
- ◆ Low-Power Shutdown Mode
- ◆ Internal Low-Noise Microphone Bias, 2V
- ◆ Available in the Space-Saving 12-Bump UCSP (1.5mm x 2mm) and 14-Pin TDFN (3mm x 3mm) Packages
- ◆ -40°C to +85°C Extended Temperature Range

Ordering Information

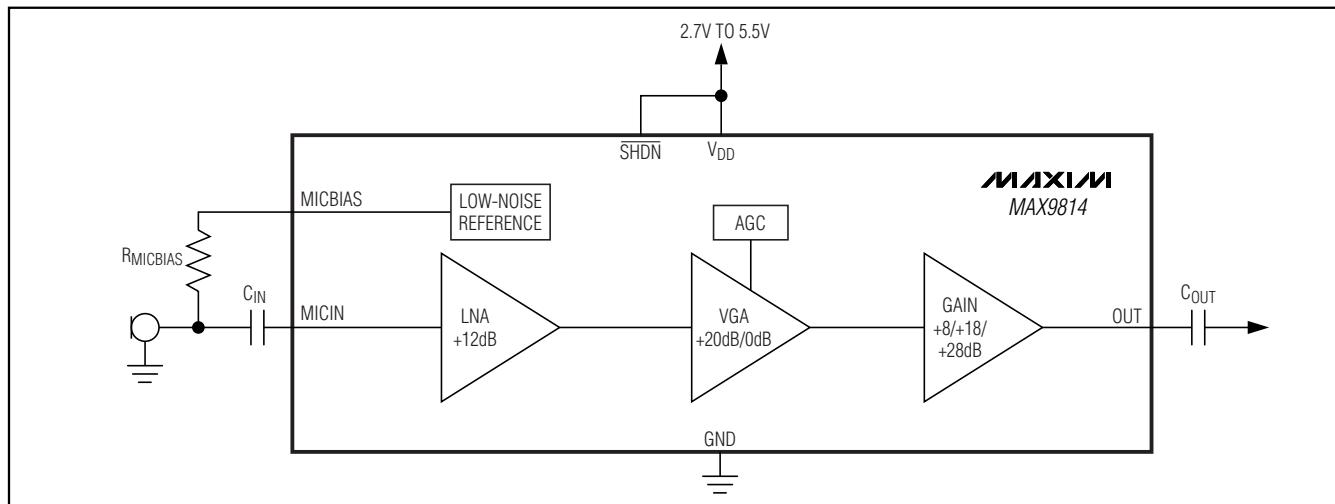
PART	TEMP RANGE	PIN-PACKAGE	PKG CODE
MAX9814EBC+T	-40°C to +85°C	12 UCSP-12	B12-3
MAX9814ETD+T	-40°C to +85°C	14 TDFN-14	T1433-2

+Denotes a lead-free package.

Pin Configurations appear at end of data sheet.

UCSP is a trademark of Maxim Integrated Products, Inc.

Simplified Block Diagram



Microphone Amplifier with AGC and Low-Noise Microphone Bias

ABSOLUTE MAXIMUM RATINGS

V _{DD} to GND	-0.3V to +6V
All Other Pins to GND	-0.3V to (V _{DD} + 0.3V)
Output Short-Circuit Duration	Continuous
Continuous Current (OUT, MICBIAS)	±100mA
All Other Pins	±20mA

Continuous Power Dissipation (T _A = +70°C)	
12-Bump UCSP (derate 6.5mW/°C above +70°C)	518mW
14-Pin TDFN (derate 16.7mW/°C above +70°C)	1481.5mW
Operating Temperature Range	-40°C to +85°C
Junction Temperature	+150°C
Lead Temperature (soldering, 10s)	+300°C
Bump Temperature (soldering) Reflow	+235°C

Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated in the operational sections of the specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

ELECTRICAL CHARACTERISTICS

(V_{DD} = 3.3V, $\overline{SHDN} = V_{DD}$, C_{CCT} = 470nF, C_{CG} = 2μF, GAIN = V_{DD}, T_A = T_{MIN} to T_{MAX}, unless otherwise specified. Typical values are at T_A = +25°C.) (Note 1)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
GENERAL						
Operating Voltage	V _{DD}	Guaranteed by PSRR test	2.7	5.5		V
Supply Current	I _{DD}		3.1	6		mA
Shutdown Supply Current	I _{SHDN}		0.01	1		μA
Input-Referred Noise Density	e _n	BW = 20kHz, all gain settings	30			nV/√Hz
Output Noise		BW = 20kHz	430			μVRMS
Signal-to-Noise Ratio	SNR	BW = 22Hz to 22kHz (500mVRMS output signal)	61			dB
		A-weighted	64			
Dynamic Range	DR	(Note 2)	60			dB
Total Harmonic Distortion Plus Noise	THD+N	f _{IN} = 1kHz, BW = 20Hz to 20kHz, R _L = 10kΩ, V _{TH} = 1V (threshold = 2V _{P-P}), V _{IN} = 0.5mVRMS, V _{CT} = 0V	0.04			%
		f _{IN} = 1kHz, BW = 20Hz to 20kHz, R _L = 10kΩ, V _{TH} = 0.1V (threshold = 200mV _{P-P}), V _{IN} = 30mVRMS, V _{CT} = 2V	0.2			
Amplifier Input BIAS	V _{IN}		1.14	1.23	1.32	V
Maximum Input Voltage	V _{IN_MAX}	1% THD	100			mV _{P-P}
Input Impedance	Z _{IN}		100			kΩ
Maximum Gain	A	GAIN = V _{DD}	39.5	40	40.5	dB
		GAIN = GND	49.5	50	50.6	
		GAIN = unconnected	59.5	60	60.5	
Minimum Gain		GAIN = V _{DD}	18.7	20	20.5	dB
		GAIN = GND	29.0	30	30.8	
		GAIN = unconnected	38.7	40	40.5	
Maximum Output Level	V _{OUT_RMS}	1% THD+N, V _{TH} = MICBIAS	0.707			VRMS
Regulated Output Level		AGC enabled, V _{TH} = 0.7V	1.26	1.40	1.54	V _{P-P}
AGC Attack Time	t _{ATTACK}	C _T = 470nF (Note 3)	1.1			ms
Attack/Release Ratio	A/R	A/R = GND	1:500			ms/ms
		A/R = V _{DD}	1:2000			
		A/R = unconnected	1:4000			

Microphone Amplifier with AGC and Low-Noise Microphone Bias

ELECTRICAL CHARACTERISTICS (continued)

($V_{DD} = 3.3V$, $\bar{SHDN} = V_{DD}$, $C_{CT} = 470nF$, $C_{CG} = 2\mu F$, GAIN = V_{DD} , $T_A = T_{MIN}$ to T_{MAX} , unless otherwise specified. Typical values are at $T_A = +25^\circ C$.) (Note 1)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
MICOUT High Output Voltage	V_{OH}	I_{OUT} sourcing 1mA		2.45		V
MICOUT Low Output Voltage	V_{OL}	I_{OUT} sinking 1mA		3		mV
MICOUT Bias		MICOUT unconnected	1.14	1.23	1.32	V
Output Impedance	Z_{OUT}			50		Ω
Minimum Resistive Load	R_{LOAD_MIN}			5		$k\Omega$
Maximum Capacitive Drive	C_{LOAD_MAX}			200		pF
Maximum Output Current	I_{OUT_MAX}	1% THD, $R_L = 500\Omega$	1	2		mA
Output Short-Circuit Current	I_{SC}		3	8		mA
Power-Supply Rejection Ratio	PSRR	AGC mode; $V_{DD} = 2.7V$ to $5.5V$ (Note 4)	35	50		dB
		$f = 217Hz$, $V_{RIPPLE} = 100mV_{P-P}$ (Note 5)		55		
		$f = 1kHz$, $V_{RIPPLE} = 100mV_{P-P}$ (Note 5)		52.5		
		$f = 10kHz$, $V_{RIPPLE} = 100mV_{P-P}$ (Note 5)		43		
MICROPHONE BIAS						
Microphone Bias Voltage	$V_{MICBIAS}$	$I_{MICBIAS} = 0.5mA$	1.84	2.0	2.18	V
Output Resistance	$R_{MICBIAS}$	$I_{MICBIAS} = 1mA$		1		Ω
Output Noise Voltage	$V_{MICBIAS_NOISE}$	$I_{MICBIAS} = 0.5mA$, BW = 22Hz to 22kHz		5.5		μV_{RMS}
Power-Supply Rejection Ratio	PSRR	DC, $V_{DD} = 2.7V$ to $5.5V$	70	80		dB
		$I_{MICBIAS} = 0.5mA$, $V_{RIPPLE} = 100mV_{P-P}$, $f_{IN} = 1kHz$		71		
TRILEVEL INPUTS (A/R, GAIN)						
Tri-Level Input Leakage Current		A/R or GAIN = V_{DD}	0.5 V_{DD} / 180k Ω	0.5 V_{DD} / 100k Ω	0.5 V_{DD} / 50k Ω	mA
		A/R or GAIN = GND	0.5 V_{DD} / 180k Ω	0.5 V_{DD} / 100k Ω	0.5 V_{DD} / 50k Ω	
Input High Voltage	V_{IH}		$V_{DD} \times 0.7$			V
Input Low Voltage	V_{IL}		$V_{DD} \times 0.3$			V
Shutdown Enable Time	t_{ON}			60		ms
Shutdown Disable Time	t_{OFF}			40		ms
DIGITAL INPUT (SHDN)						
SHDN Input Leakage Current			-1		+1	μA
Input High Voltage	V_{IH}		1.3			V
Input Low Voltage	V_{IL}			0.5		V
AGC THRESHOLD INPUT (TH)						
TH Input Leakage Current			-1		+1	μA

Note 1: Devices are production tested at $T_A = +25^\circ C$. Limits over temperature are guaranteed by design.

Note 2: Dynamic range is calculated using the EIAJ method. The input is applied at -60dBFS (0.707 μV_{RMS}), $f_{IN} = 1kHz$.

Note 3: Attack time measured as time from AGC trigger to gain reaching 90% of its final value.

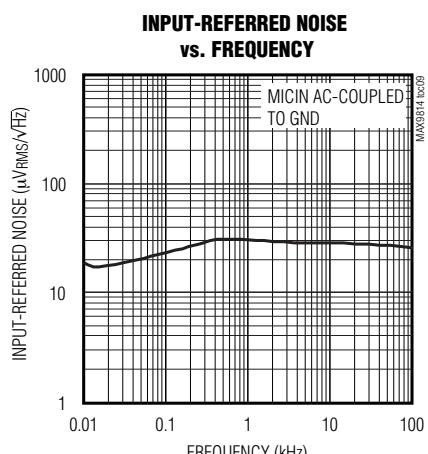
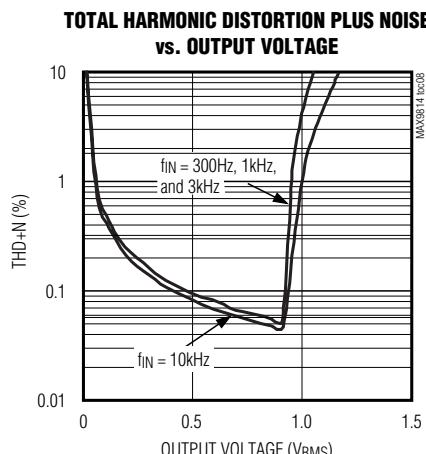
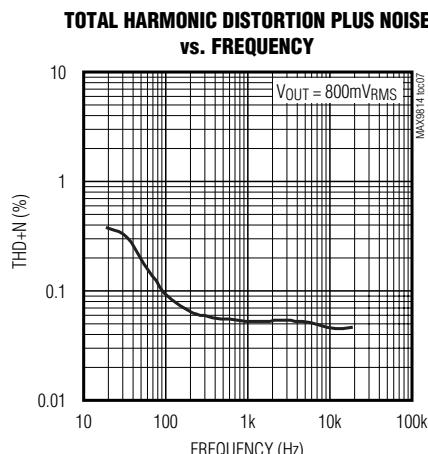
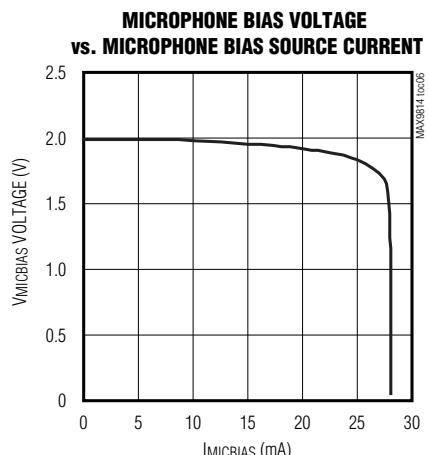
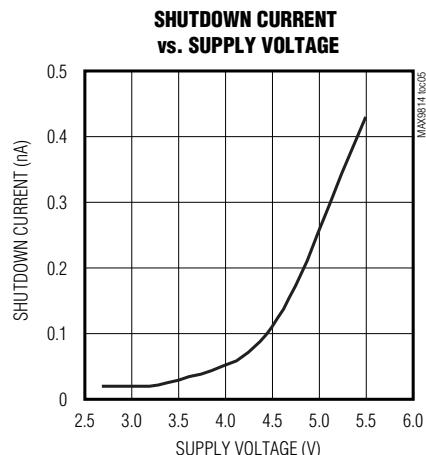
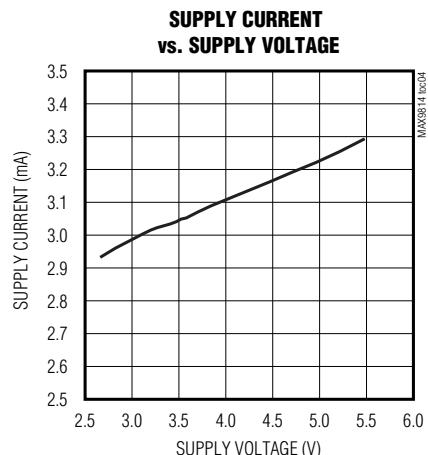
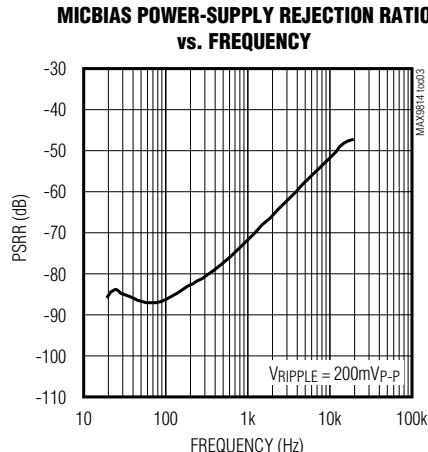
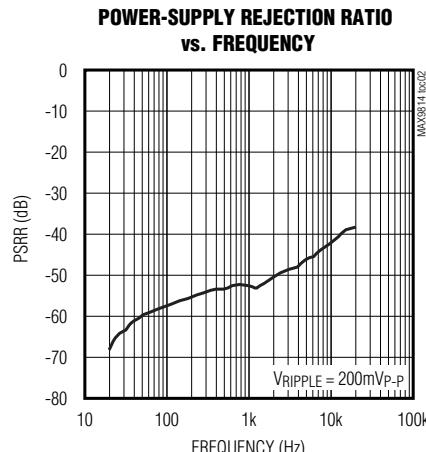
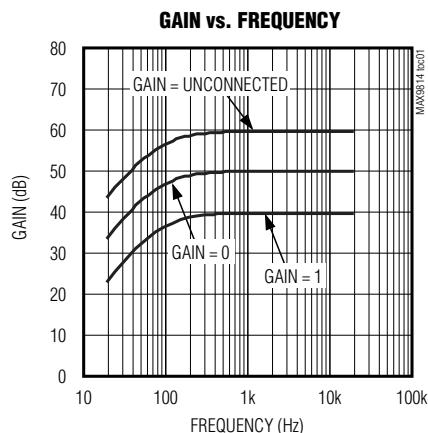
Note 4: CG is connected to an external DC voltage source, and adjusted until $V_{MICOUT} = 1.23V$.

Note 5: CG connected to GND with $2.2\mu F$.

Microphone Amplifier with AGC and Low-Noise Microphone Bias

Typical Operating Characteristics

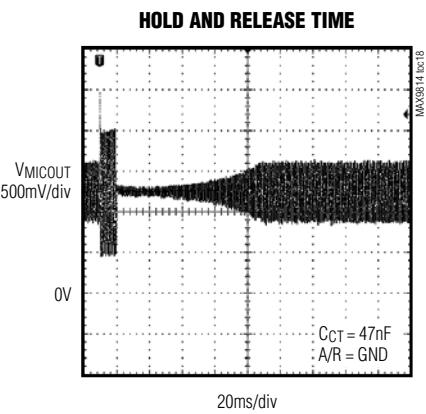
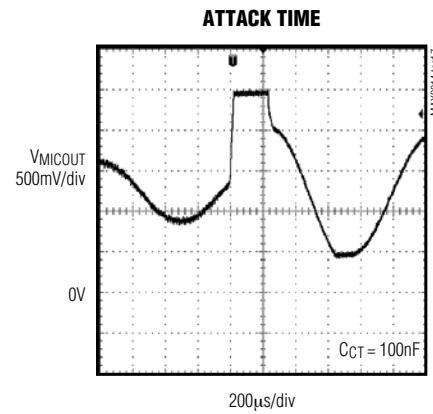
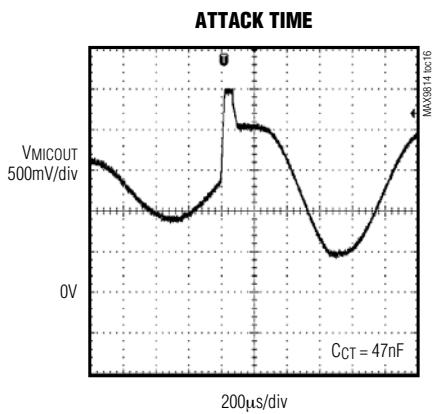
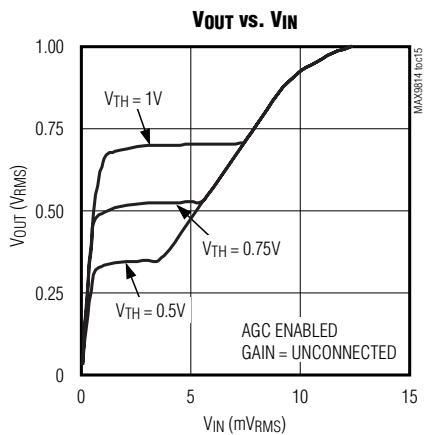
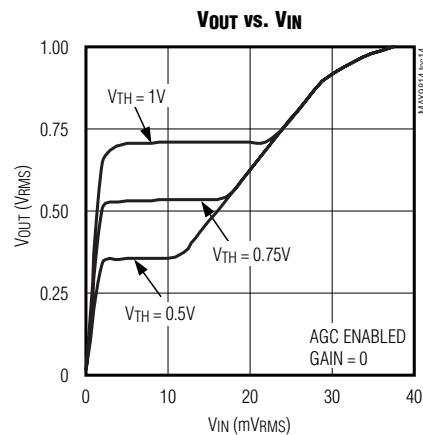
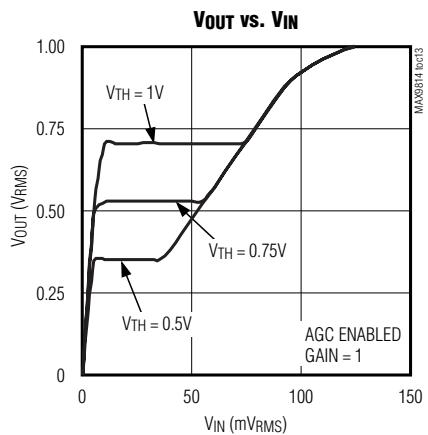
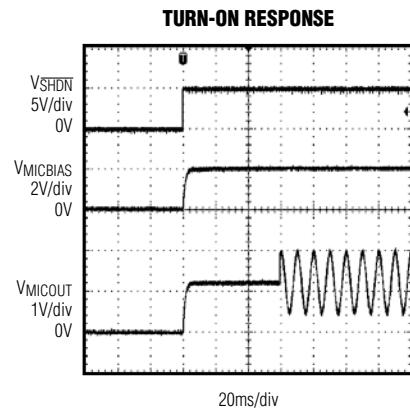
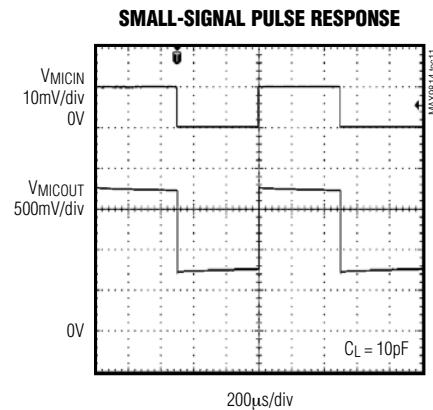
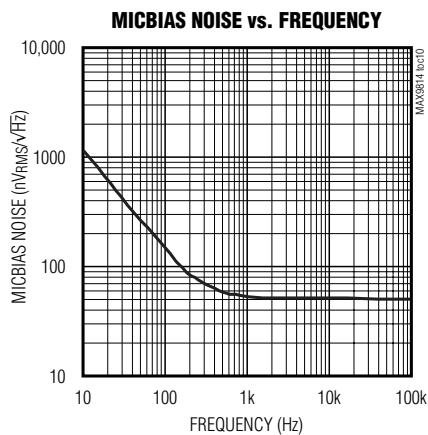
($V_{DD} = 5V$, $C_{CT} = 470nF$, $C_{CG} = 2.2\mu F$, $V_{TH} = V_{MICBIAS} \times 0.4$, GAIN = V_{DD} (40dB), AGC disabled, no load, $R_L = 10k\Omega$, $C_{OUT} = 1\mu F$, $T_A = +25^\circ C$, unless otherwise noted.)



Microphone Amplifier with AGC and Low-Noise Microphone Bias

Typical Operating Characteristics (continued)

($V_{DD} = 5V$, $C_{CT} = 470nF$, $C_{CG} = 2.2\mu F$, $V_{TH} = V_{MICBIAS} \times 0.4$, GAIN = V_{DD} (40dB), AGC disabled, no load, $R_L = 10k\Omega$, $C_{OUT} = 1\mu F$, $T_A = +25^\circ C$, unless otherwise noted.)

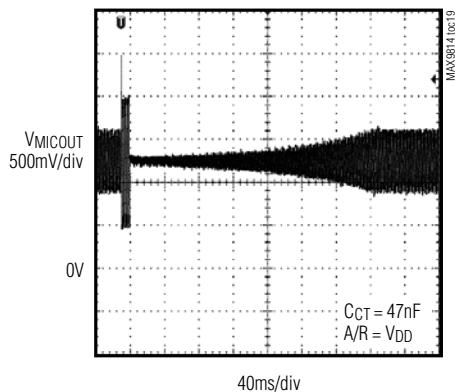


Microphone Amplifier with AGC and Low-Noise Microphone Bias

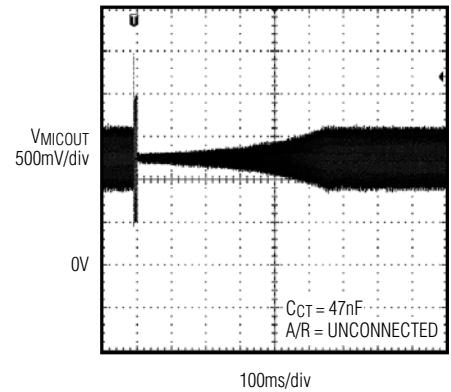
Typical Operating Characteristics (continued)

($V_{DD} = 5V$, $C_{CT} = 470nF$, $C_{CG} = 2.2\mu F$, $V_{TH} = V_{MICBIAS} \times 0.4$, GAIN = V_{DD} (40dB), AGC disabled, no load, $R_L = 10k\Omega$, $C_{OUT} = 1\mu F$, $T_A = +25^\circ C$, unless otherwise noted.)

HOLD AND RELEASE TIME



HOLD AND RELEASE TIME



Pin Description

PIN		NAME	FUNCTION
TDFN	UCSP		
1	A1	CT	Timing Capacitor Connection. Connect a capacitor to CT to control the Attack and Release times of the AGC.
2	B2	SHDN	Active-Low Shutdown Control
3	A2	CG	Amplifier DC Offset Adjust. Connect a $2.2\mu F$ capacitor to GND to ensure zero offset at the output.
4, 11	—	N.C.	No Connection. Connect to GND.
5	A3	V_{DD}	Power Supply. Bypass to GND with a $1\mu F$ capacitor.
6	A4	MICOUT	Amplifier Output
7	B4	GND	Ground
8	C4	MICIN	Microphone Noninverting Input
9	B3	A/R	Tri-Level Attack and Release Ratio Select. Controls the ratio of attack time to release time for the AGC circuit. A/R = GND: Attack/Release Ratio is 1:500 A/R = V_{DD} : Attack/Release Ratio is 1:2000 A/R = BIAS: Attack/Release Ratio is 1:4000
10	C3	GAIN	Tri-Level Amplifier Gain Control. GAIN = V_{DD} , gain set to 40dB. GAIN = GND, gain set to 50dB. GAIN = Unconnected, uncompressed gain set to 60dB.
12	C2	BIAS	Amplifier Bias. Bypass to GND with a $0.47\mu F$ capacitor.
13	C1	MICBIAS	Microphone Bias Output
14	B1	TH	AGC Threshold Control. TH voltage sets gain control threshold. Connect TH to MICBIAS to disable the AGC.

Microphone Amplifier with AGC and Low-Noise Microphone Bias

Detailed Description

The MAX9814 is a low-cost, high-quality microphone amplifier with automatic gain control (AGC) and a low-noise microphone bias. The MAX9814 consists of several distinct circuits: a low-noise preamplifier, a variable gain amplifier (VGA), an output amplifier, a microphone-bias-voltage generator, and AGC control circuitry.

An internal microphone bias voltage generator provides a 2V bias that is suitable for most electret condenser microphones. The MAX9814 amplifies the input in three distinct stages. In the first stage, the input is buffered and amplified through the low-noise preamplifier with a gain of 12dB. The second stage consists of the VGA controlled by the AGC. The VGA/AGC combination is capable of varying the gain from 20dB to 0dB. The output amplifier is the final stage in which a fixed gain of 8dB, 18dB, 20dB is programmed through a single tri-level logic input. With no compression from the AGC, the MAX9814 is capable of providing 40dB, 50dB, or 60dB gain.

Automatic Gain Control (AGC)

A device without AGC experiences clipping at the output when too much gain is applied to the input. AGC prevents clipping at the output when too much gain is applied to the input, eliminating output clipping. Figure 1 shows a comparison of an over-gained microphone input with and without AGC.

The MAX9814's AGC controls the gain by first detecting that the output voltage has exceeded a preset limit. The microphone amplifier gain is then reduced with a selectable time constant to correct for the excessive output-voltage amplitude. This process is known as the attack time. When the output signal subsequently lowers in amplitude, the gain is held at the reduced state for a short period before slowly increasing to the normal value. This process is known as the hold and release time. The speed at which the amplifiers adjust to changing input signals is set by the external timing capacitor C_{CT} and the voltage applied to A/R. The AGC threshold can be set by adjusting V_{TH} . Gain reduction is a function of input signal amplitude with a maximum AGC attenuation of 20dB. Figure 2 shows the effect of an input burst exceeding the preset limit, output attack, hold and release times.

If the attack and release times are configured to respond too fast, audible artifacts often described as "pumping" or "breathing" can occur as the gain is rapidly adjusted to follow the dynamics of the signal. For best results, adjust the time constant of the AGC to accommodate the source material. For applications in which music CDs are the main audio source, a 160 μ s attack time with an 80ms release time is recommended. Music applications typically require a shorter release time than voice or movie content.

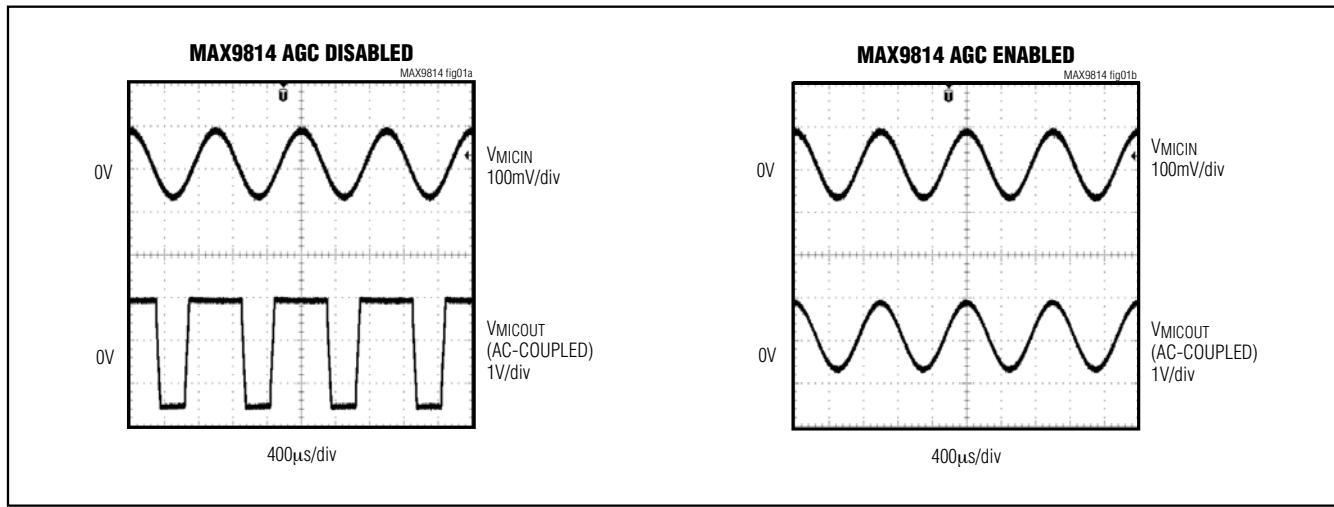


Figure 1. Microphone Input with and Without AGC

Microphone Amplifier with AGC and Low-Noise Microphone Bias

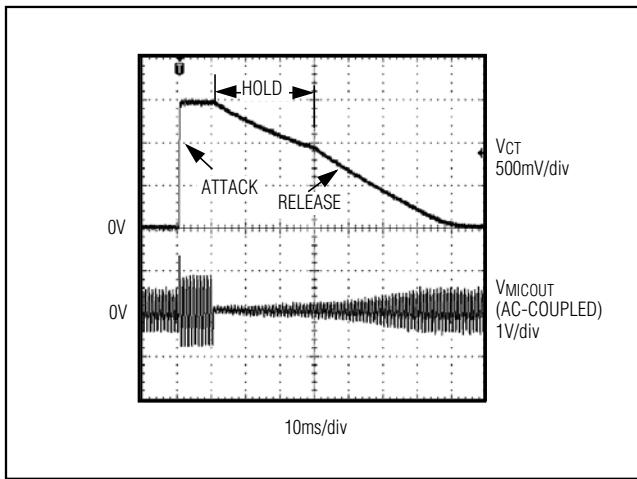


Figure 2. Input Burst Exceeding AGC Limit

Attack Time

The attack time is the time it takes for the AGC to reduce the gain after the input signal has exceeded the threshold level. The gain attenuation during the attack time is exponential, and defined as one-time constant. The time constant of the attack is given by $2400 \times C_{CT}$ seconds (where C_{CT} is the external timing capacitor):

- Use a short attack time for the AGC to react quickly to transient signals, such as snare drum beats (music) or gun shots (DVD).
- Use a longer attack time to allow the AGC to ignore short-duration peaks and only reduce the gain when a noticeable increase in loudness occurs. Short-duration peaks are not reduced, but louder passages are. This allows the louder passages to be reduced in volume, thereby maximizing output dynamic range.

Hold Time

Hold time is the delay after the signal falls below the threshold level before the release phase is initiated. Hold time is internally set to 30ms and nonadjustable. The hold time is cancelled by any signal exceeding the set threshold level, and the attack time is reinitiated.

Release Time

The release time is how long it takes for the gain to return to its normal level after the output signal has fallen below the threshold level and 30ms hold time has expired. Release time is defined as release from a 20dB gain compression to 10% of the nominal gain setting after the input signal has fallen below the TH threshold and the 30ms hold time has expired. Release time is adjustable and has a minimum of 25s. The release time is set by picking an attack time using C_{CT}

and setting the attack-to-release time ratio by configuring A/R as shown in Table 1:

- Use a small ratio to maximize the speed of the AGC.
- Use a large ratio to maximize the sound quality and prevent repeated excursions above the threshold from being independently adjusted by the AGC.

AGC Output Threshold

The output threshold that activates AGC is adjustable through the use of an external resistive divider. Once the divider is set, AGC reduces the gain to match the output voltage to the voltage set at the TH input.

Microphone Bias

The MAX9814 features an internal low-noise microphone bias voltage capable of driving most electret condenser microphones. The microphone bias is regulated at 2V to provide that the input signal to the low-noise preamplifier does not clip to ground.

Applications Information

Programming Attack and Release Times

Attack and release times are set by selecting the capacitance value between CT and GND, and by setting the logic state of A/R (Table 1). A/R is a tri-level logic input that sets the attack-to-release time ratio.

Table 1. Attack-and-Release Ratios

A/R	ATTACK/RELEASE RATIO
GND	1:500
V _{DD}	1:2000
Unconnected	1:4000

The attack and release times can be selected by utilizing the corresponding capacitances listed in Table 2.

Table 2. Attack-and-Release Time

C _{CT}	t _{ATTACK} (ms)	t _{RELEASE} (ms)		
		A/R = GND	A/R = V _{DD}	A/R = UNCONNECTED
22nF	0.05	25	100	200
47nF	0.11	55	220	440
68nF	0.16	80	320	640
100nF	0.24	120	480	960
220nF	0.53	265	1060	2120
470nF	1.1	550	2200	4400
680nF	1.63	815	3260	6520
1μF	2.4	1200	4800	9600

Microphone Amplifier with AGC and Low-Noise Microphone Bias

Setting the AGC Threshold

To set the output-voltage threshold at which the microphone output is clamped, an external resistor-divider must be connected from MICBIAS to ground with the output of the resistor-divider applied to TH. The voltage V_{TH} determines the peak output-voltage threshold at which the output becomes clamped. The maximum signal swing at the output is then limited to two times V_{TH} and remains at that level until the amplitude of the input signal is reduced. To disable AGC, connect TH to MICBIAS.

Microphone Bias Resistor

MICBIAS is capable of sourcing 20mA. Select a value for $R_{MICBIAS}$ that provides the desired bias current for the electret microphone. A value of $2.2\text{k}\Omega$ is usually sufficient for a microphone of typical sensitivity. Consult the microphone data sheet for the recommended bias resistor.

Bias Capacitor

The BIAS output of the MAX9814 is internally buffered and provides a low-noise bias. Bypass BIAS with a 470nF capacitor to ground.

Input Capacitor

The input AC-coupling capacitor (C_{IN}) and the input resistance (R_{IN}) to the microphone amplifier form a highpass filter that removes any DC bias from an input signal (see the *Typical Application Circuit/Functional Diagram*). C_{IN} prevents any DC components from the input-signal source from appearing at the amplifier outputs. The -3dB point of the highpass filter, assuming zero source impedance due to the input signal source, is given by:

$$f_{-3\text{dB_IN}} = \frac{1}{2\pi \times R_{IN} \times C_{IN}}$$

Choose C_{IN} such that $f_{-3\text{dB_IN}}$ is well below the lowest frequency of interest. Setting $f_{-3\text{dB_IN}}$ too high affects the amplifier's low-frequency response. Use capacitors with low-voltage coefficient dielectrics. Aluminum electrolytic, tantalum, or film dielectric capacitors are good choices for AC-coupling capacitors. Capacitors with high-voltage coefficients, such as ceramics (non-C0G dielectrics), can result in increased distortion at low frequencies.

Output Capacitor

The output of the MAX9814 is biased at 1.23V. To eliminate the DC offset, an AC-coupling capacitor (C_{OUT}) must be used. Depending on the input resistance (R_L) of the following stage, C_{OUT} and R_L effectively form a highpass filter. The -3dB point of the highpass filter, assuming zero output impedance, is given by:

$$f_{-3\text{dB_OUT}} = \frac{1}{2\pi \times R_L \times C_{OUT}}$$

Shutdown

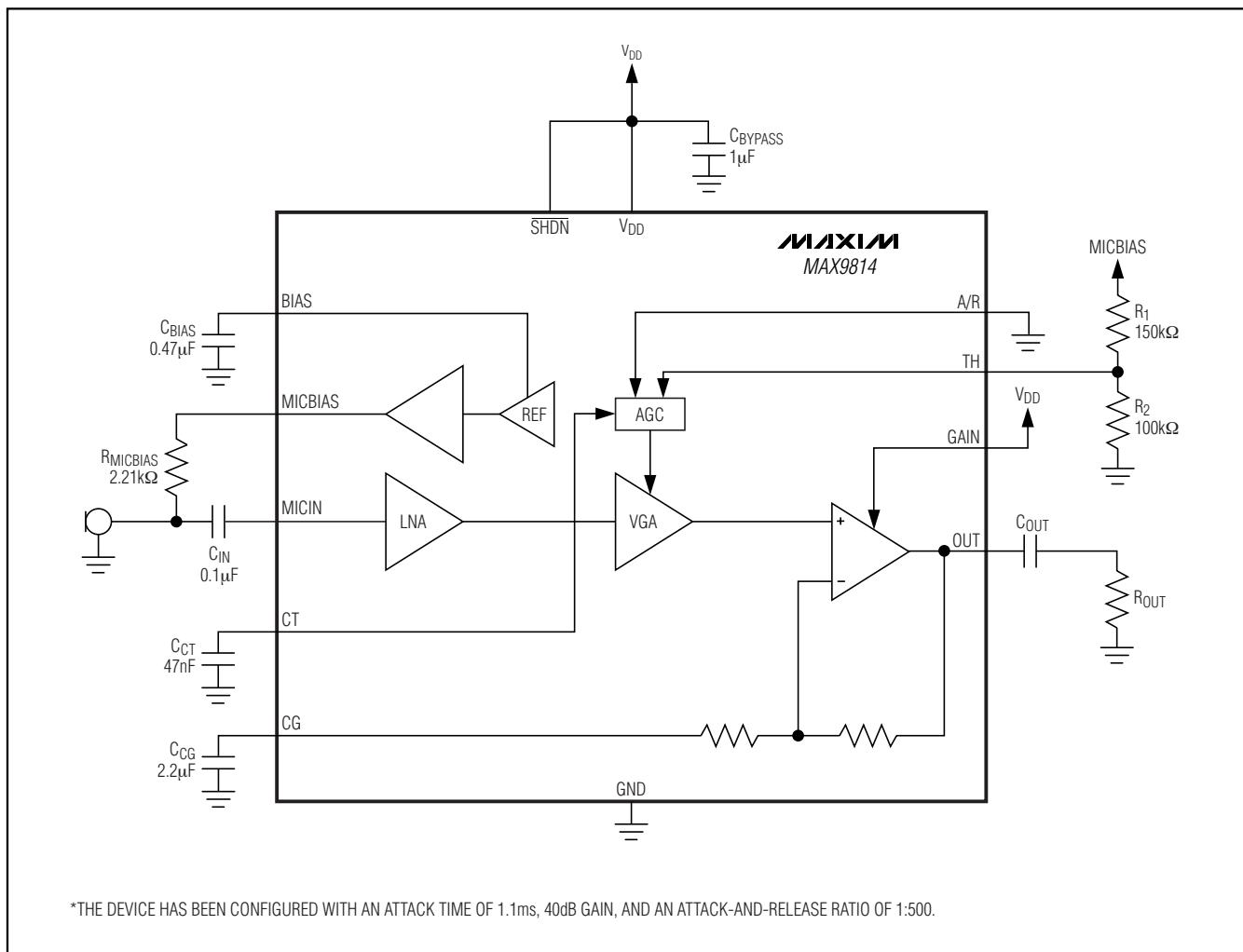
The MAX9814 features a low-power shutdown mode. When \overline{SHDN} goes low, the supply current drops to $0.01\mu\text{A}$, the output enters a high-impedance state, and the bias current to the microphone is switched off. Driving \overline{SHDN} high enables the amplifier. Do not leave \overline{SHDN} unconnected.

Power-Supply Bypassing and PCB Layout

Bypass the power supply with a $0.1\mu\text{F}$ capacitor to ground. Reduce stray capacitance by minimizing trace lengths and place external components as close to the device as possible. Surface-mount components are recommended. In systems where analog and digital grounds are available, connect the MAX9814 to analog ground.

Microphone Amplifier with AGC and Low-Noise Microphone Bias

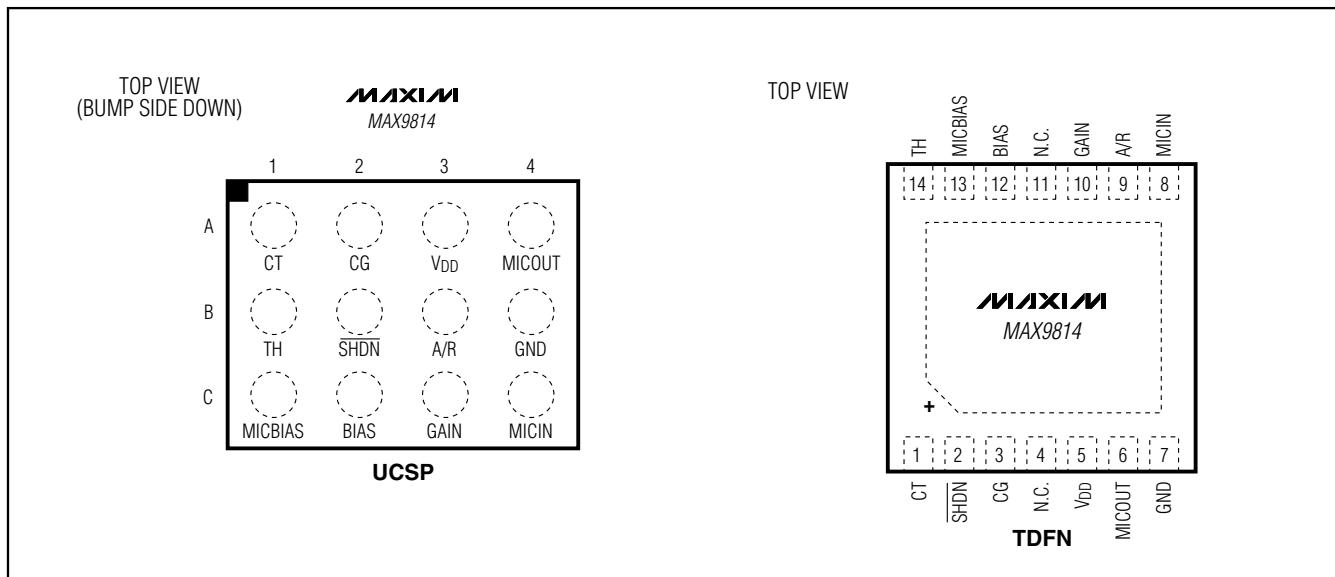
Typical Application Circuit/Functional Diagram



*THE DEVICE HAS BEEN CONFIGURED WITH AN ATTACK TIME OF 1.1ms, 40dB GAIN, AND AN ATTACK-AND-RELEASE RATIO OF 1:500.

Microphone Amplifier with AGC and Low-Noise Microphone Bias

Pin Configurations



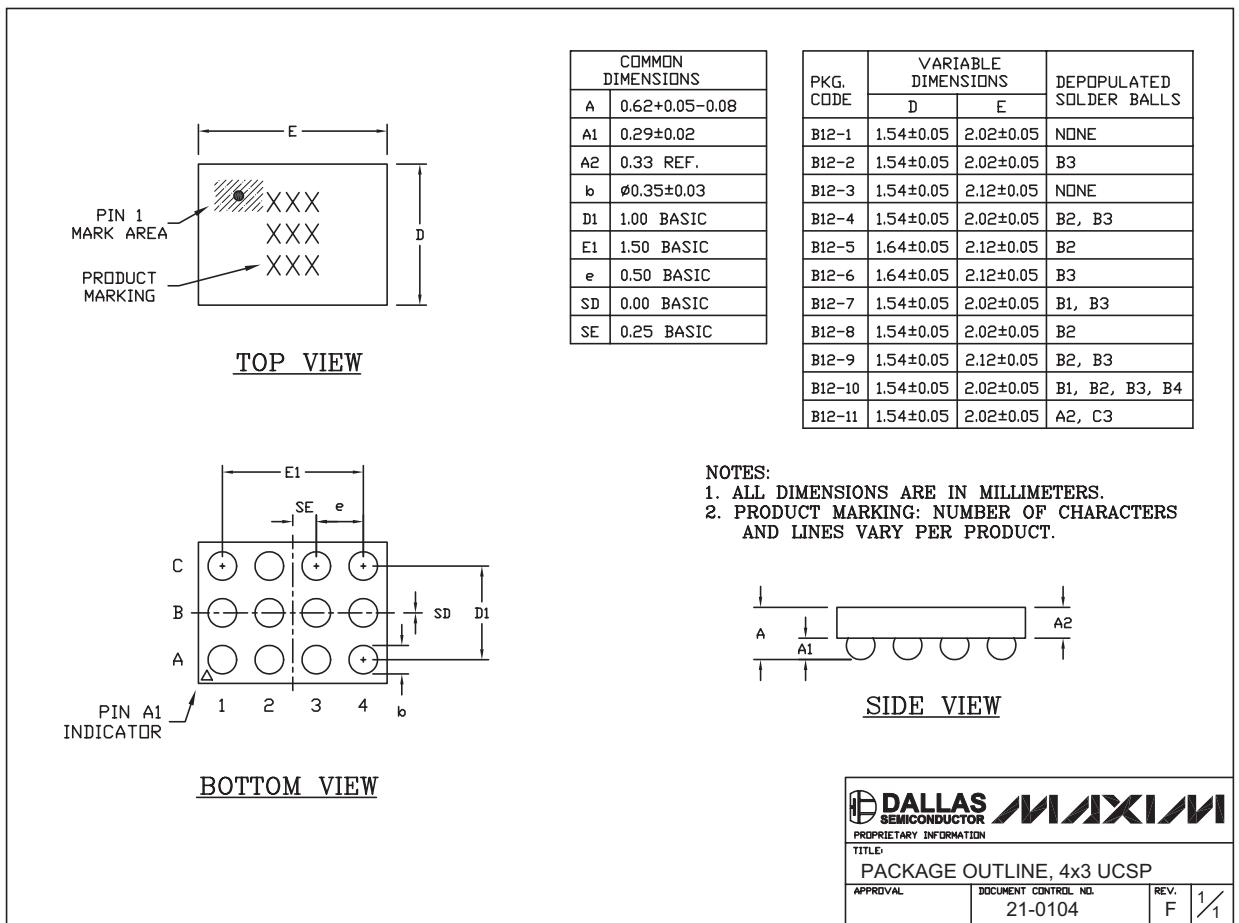
Chip Information

PROCESS: BiCMOS

Microphone Amplifier with AGC and Low-Noise Microphone Bias

Package Information

(The package drawing(s) in this data sheet may not reflect the most current specifications. For the latest package outline information go to www.maxim-ic.com/packages.)



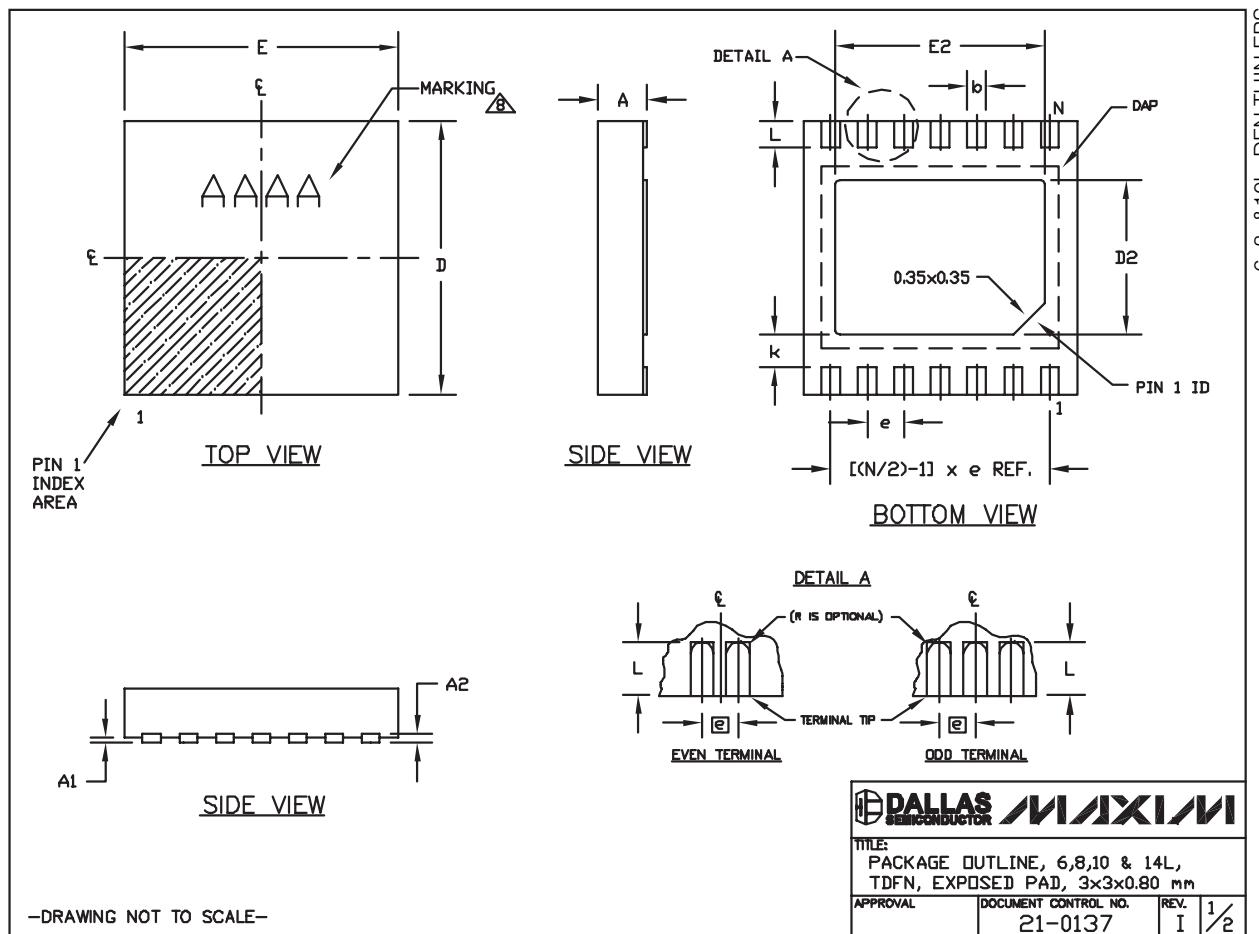
DALLAS SEMICONDUCTOR		MAXIM
PROPRIETARY INFORMATION		
TITLE: PACKAGE OUTLINE, 4x3 UCSP		
APPROVAL	DOCUMENT CONTROL NO. 21-0104	REV. F 1/1

Microphone Amplifier with AGC and Low-Noise Microphone Bias

Package Information (continued)

(The package drawing(s) in this data sheet may not reflect the most current specifications. For the latest package outline information go to www.maxim-ic.com/packages.)

MAX9814



Microphone Amplifier with AGC and Low-Noise Microphone Bias

Package Information (continued)

(The package drawing(s) in this data sheet may not reflect the most current specifications. For the latest package outline information go to www.maxim-ic.com/packages.)

COMMON DIMENSIONS		PACKAGE VARIATIONS								
SYMBOL	MIN.	MAX.	PKG. CODE	N	D2	E2	e	JEDEC SPEC	b	[(N/2)-1] x e
A	0.70	0.80	T633-2	6	1.50±0.10	2.30±0.10	0.95 BSC	MO229 / WEEA	0.40±0.05	1.90 REF
D	2.90	3.10	T833-2	8	1.50±0.10	2.30±0.10	0.65 BSC	MO229 / WEEC	0.30±0.05	1.95 REF
E	2.90	3.10	T833-3	8	1.50±0.10	2.30±0.10	0.65 BSC	MO229 / WEEC	0.30±0.05	1.95 REF
A1	0.00	0.05	T1033-1	10	1.50±0.10	2.30±0.10	0.50 BSC	MO229 / WEED-3	0.25±0.05	2.00 REF
L	0.20	0.40	T1033-2	10	1.50±0.10	2.30±0.10	0.50 BSC	MO229 / WEED-3	0.25±0.05	2.00 REF
k	0.25 MIN.		T1433-1	14	1.70±0.10	2.30±0.10	0.40 BSC	-----	0.20±0.05	2.40 REF
A2	0.20 REF.		T1433-2	14	1.70±0.10	2.30±0.10	0.40 BSC	-----	0.20±0.05	2.40 REF

NOTES:

- ALL DIMENSIONS ARE IN mm. ANGLES IN DEGREES.
- COPLANARITY SHALL NOT EXCEED 0.08 mm.
- WARPAGE SHALL NOT EXCEED 0.10 mm.
- PACKAGE LENGTH/PACKAGE WIDTH ARE CONSIDERED AS SPECIAL CHARACTERISTIC(S).
- DRAWING CONFORMS TO JEDEC MO229, EXCEPT DIMENSIONS "D2" AND "E2", AND T1433-1 & T1433-2.
- "N" IS THE TOTAL NUMBER OF LEADS.
- NUMBER OF LEADS SHOWN ARE FOR REFERENCE ONLY.
- MARKING IS FOR PACKAGE ORIENTATION REFERENCE ONLY.

-DRAWING NOT TO SCALE-

 DALLAS SEMICONDUCTOR			
TITLE: PACKAGE OUTLINE, 6,8,10 & 14L, TDFN, EXPOSED PAD, 3x3x0.80 mm			
APPROVAL	DOCUMENT CONTROL NO.	REV.	21-0137 I 2/2

Maxim cannot assume responsibility for use of any circuitry other than circuitry entirely embodied in a Maxim product. No circuit patent licenses are implied. Maxim reserves the right to change the circuitry and specifications without notice at any time.

14 **Maxim Integrated Products, 120 San Gabriel Drive, Sunnyvale, CA 94086 408-737-7600**

© 2007 Maxim Integrated Products

MAXIM is a registered trademark of Maxim Integrated Products, Inc.

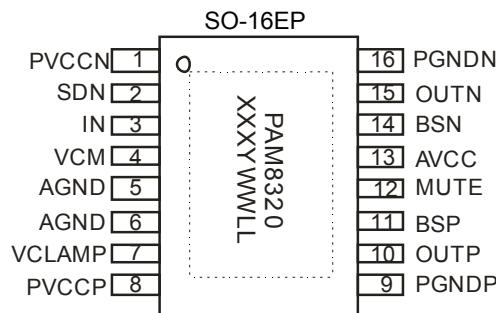
Description

The PAM8320 is an efficient 20W mono Class-D audio power amplifier, designed to drive speakers as low as 4Ω in a bridge-tied-load configuration. Due to the low power dissipation and high efficiency of up to 95%, the device can be used without any external heat sink whilst playing music.

The PAM8320 features short circuit protection, thermal shutdown, over voltage protection and under voltage lock-out.

The PAM8320 is available in a SO-16EP package.

Pin Assignments



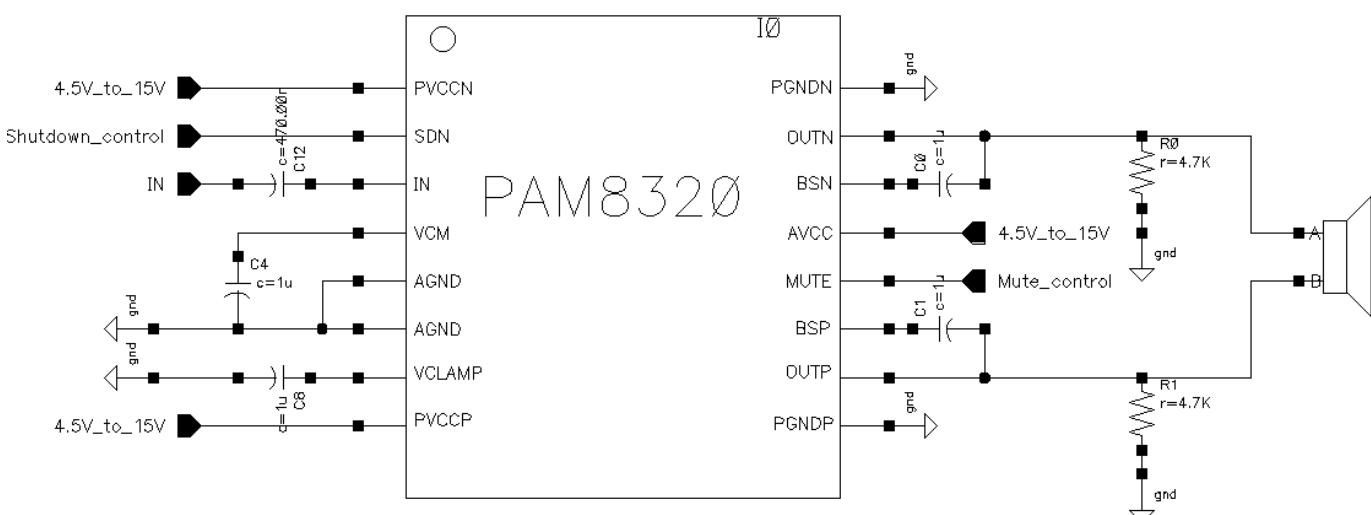
Features

- Operates from 4.5V to 15V
- 20W into 4Ω BTL Load from 12V Supply
- Single-Ended Analog Input
- No Pop Noise for Start-up and Shut-down Sequences
- Internal Oscillator (No External Components Required)
- High Efficient Class-D Operation Eliminates Need for Heat Sinks
- Thermal and Short-Circuit Protection with Auto Recovery
- Over Voltage Protection and Under Voltage Lock-out
- Space-Saving Surface-Mount SO-16EP Package
- Pb-Free Package

Applications

- PC Speaker
- Blue Tooth Speaker
- Home Sound Systems
- Active Speakers
- Docking stations

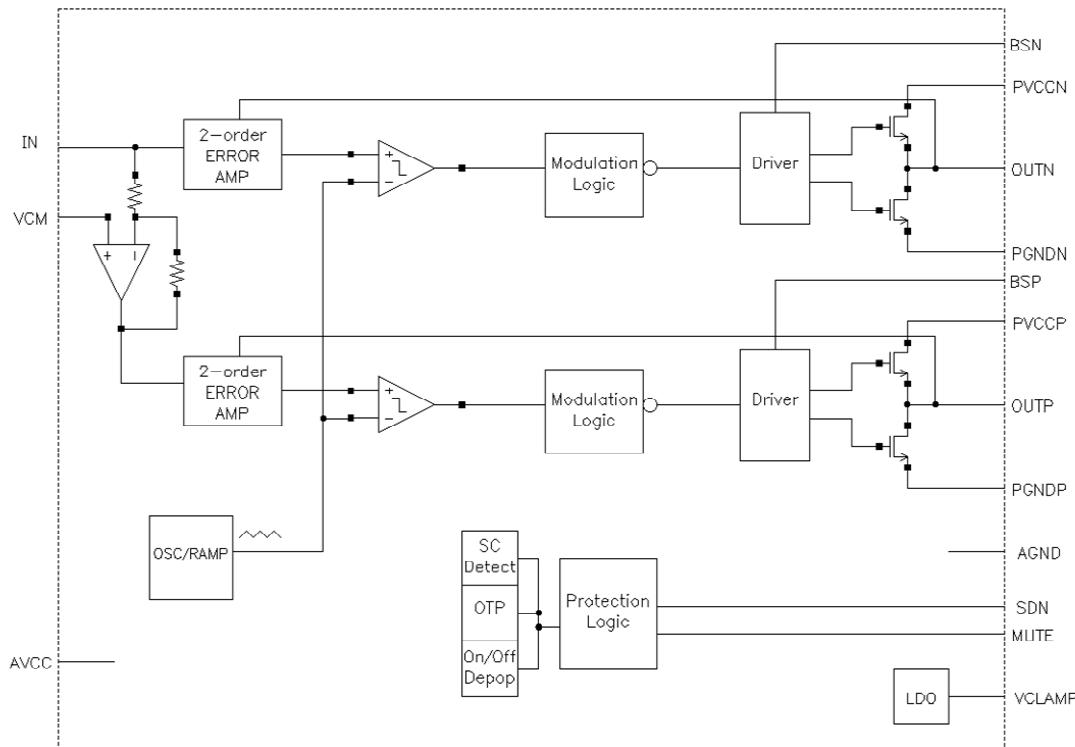
Typical Applications Circuit



Pin Descriptions

Pin	Name	I/O/P	Description
1	PVCCN	P	Power supply for negative H-bridge, not connected to PVCCP or AVCC
2	SDN	I	Shutdown signal for IC (low=shutdown, high=operational). TTL logic levels with compliance to AVCC
3	IN	I	Audio input
4	VCM	O	Reference for analog cells
5,6	AGND	P	Analog ground for digital/analog cells in core
7	VCLAMP	P	Internally generated voltage supply for bootstrap. Not to be used as a supply or connected to any component other than the decoupling capacitor.
8	PVCCP	P	Power supply for positive H-bridge, not connected to PVCCN or AVCC
9	PGNDP	P	Power ground for positive H-bridge
10	OUTP	O	Positive BTL output
11	BSP	I/O	Bootstrap terminal for high-side drive of positive BTL output
12	MUTE	I	A logic high on this pin disables the outputs. A low on this pin enables the outputs. TTL logic levels with compliance to AVCC
13	AVCC	P	High-voltage analog power supply
14	BSN	I/O	Bootstrap terminal for high-side drive of negative BTL output
15	OUTN	O	Negative BTL output
16	PGNDN	P	Power ground for negative H-bridge

Functional Block Diagram



Absolute Maximum Ratings (@ $T_A = +25^\circ\text{C}$, unless otherwise specified.)

Parameter	Rating	Unit
Supply Voltage (V_{CC})	18	V
Logic Input Voltage (SDN, MUTE)	-0.3 to $V_{CC}+0.3$	V
Analog Input Voltage (I_N)	-0.3 to 5.5	V
Storage Temperature	-65 to +150	°C
Maximum Junction Temperature	+150	°C
Junction to ambient thermal resistance	40	°C/W

Recommended Operating Conditions (@ $T_A = +25^\circ\text{C}$, unless otherwise specified.)

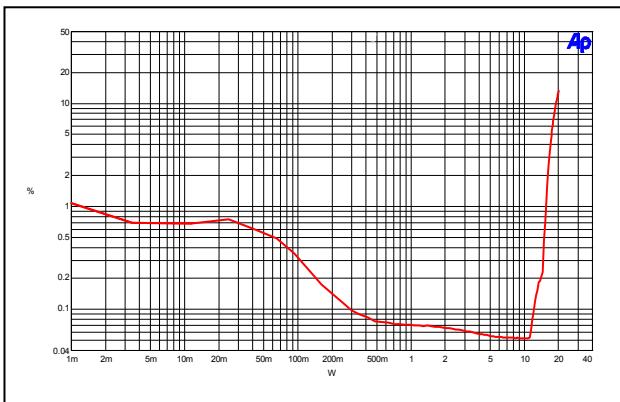
Symbol	Parameter	Min	Max	Unit
V_{CC}	Supply Voltage	4.5	15	V
T_A	Operating Ambient Temperature Range	-40	+85	°C
T_J	Junction Temperature Range	-40	+125	°C

Electrical Characteristics (@ $T_A = +25^\circ\text{C}$, $V_{CC} = 12\text{V}$, Gain = 20dB, $R_L = L(33\mu\text{H}) + R + L(33\mu\text{H})$, unless otherwise noted.)

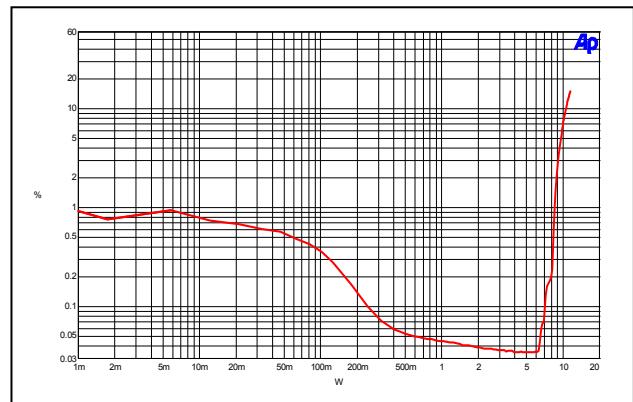
Symbol	PARAMETER	Test Conditions	MIN	TYP	MAX	Units
$ V_{OS} $	Class-D output offset voltage(measured differently)	$V_i=0\text{V}$, $A_v=20\text{dB}$	—	20	100	mV
$I_{CC(q)}$	Quiescent supply current	$\text{SDN}=3.0\text{V}$, $\text{MUTE}=0\text{V}$, No Load	—	15	30	mA
$I_{CC(\text{MUTE})}$	Quiescent supply current in mute mode	$\text{MUTE}=2.0\text{V}$, No load	—	8	20	mA
$I_{CC(\text{SDN})}$	Quiescent current in shutdown mode	$\text{SDN}=0.5\text{V}$, No load	—	20	40	uA
$R_{ds(on)}$	Drain-source on-state resistance	$I_o=0.5\text{A}$	—	150	—	$\text{m}\Omega$
P_{SRR}	Power Supply Rejection Ratio	$V_{\text{ripple}}=200\text{mVpp}$, $f=1\text{kHz}$, gain=20dB	—	-60	—	dB
P_o	Output Power at 1% THD+N	$f=1\text{kHz}$	—	15	—	W
	Output Power at 10% THD+N	$f=1\text{kHz}$	—	20	—	
THD+N	Total harmonic distortion + noise	$f=1\text{kHz}$, $P_o=7\text{W}$	—	0.05	—	%
V_n	Output integrated noise floor	20Hz to 22kHz, A-weighted, Gain=20dB	—	300	—	uV
SNR	Signal-to-noise ratio	Max output at THD+N<1%, $f=1\text{kHz}$, Gain=20dB	—	95	—	dB
OTP	Thermal trip point	—	—	+160	—	°C
OTH	Thermal hysteresis	—	—	+40	—	°C
f_{osc}	Oscillator frequency	—	—	300	—	kHz
V_{IH_SDN}	SDN Input High	—	3	—	—	—
V_{IL_SDN}	SDN Input Low	—	—	—	0.5	—
V_{IH_MUTE}	MUTE Input High	—	2	—	—	—
V_{IL_MUTE}	MUTE Input Low	—	—	—	0.5	—

Performance Characteristics (@ $T_A = +25^\circ\text{C}$, $V_{DD} = 12\text{V}$, Gain = 20dB, $R_L = L(33\mu\text{H}) + R + L(33\mu\text{H})$, unless otherwise noted.)

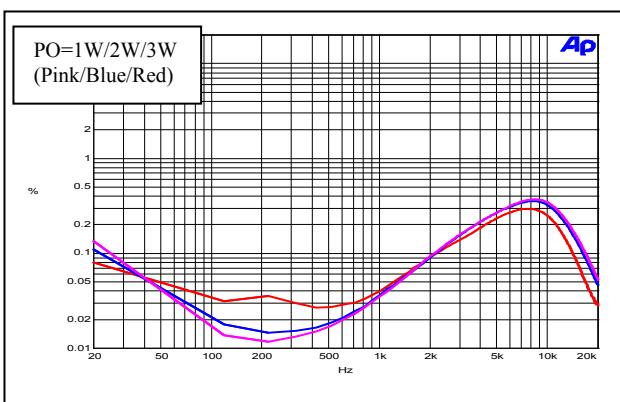
THD+N Vs. Output Power ($RL=4\Omega$)



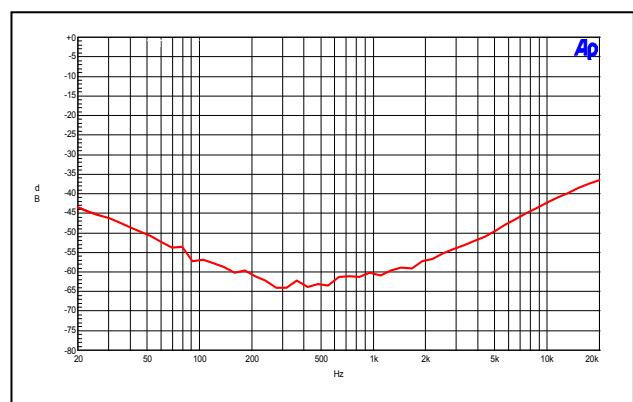
THD+N Vs. Output Power ($RL=8\Omega$)



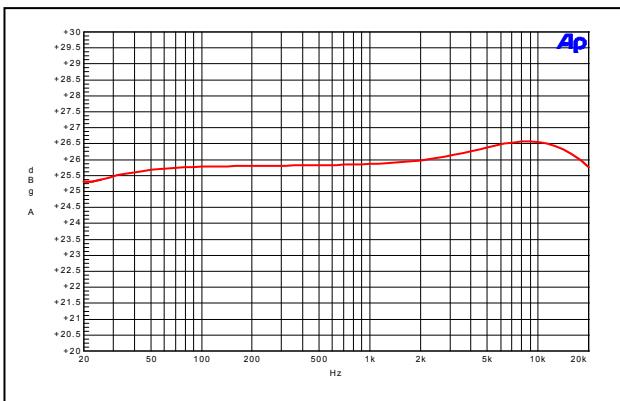
THD+N Vs. Frequency



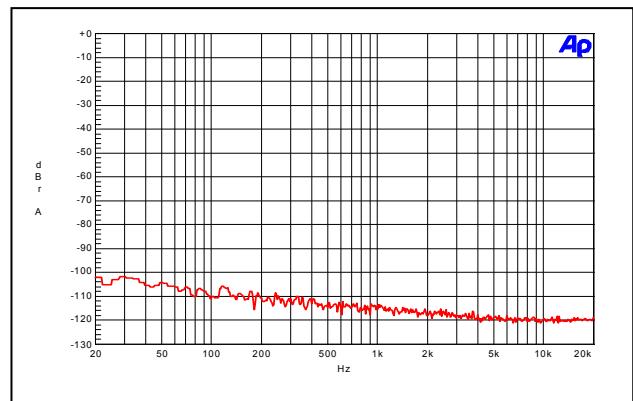
PSRR Vs. Frequency



Frequency Response

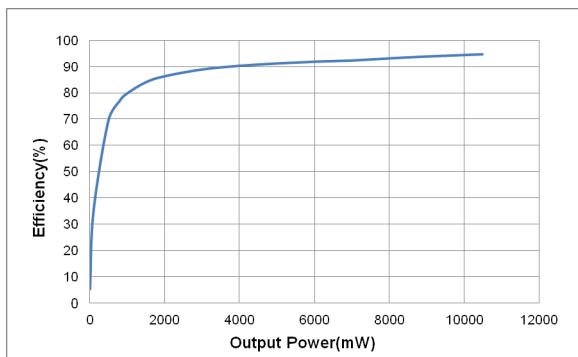


Noise Floor

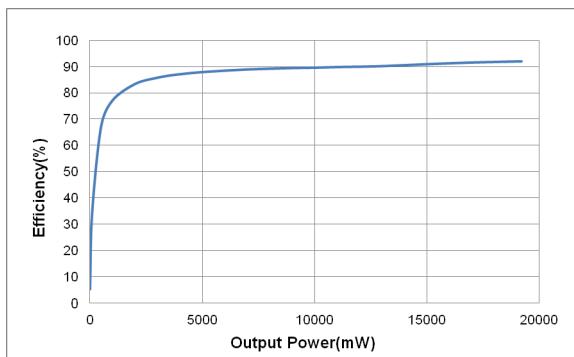


Performance Characteristics (@ $T_A=25^\circ\text{C}$, $V_{DD}=12\text{V}$, Gain=20dB, $R_L=L(33\mu\text{H})+R+L(33\mu\text{H})$, unless otherwise noted.)

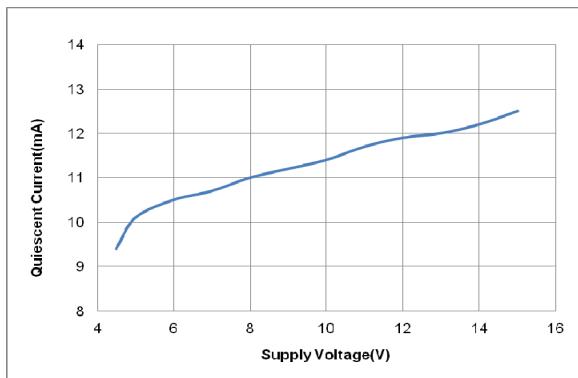
Efficiency Vs. Output Power ($RL=8\Omega$)



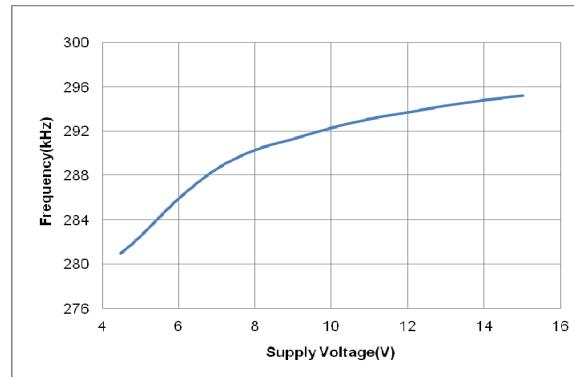
Efficiency Vs. Output Power ($RL=4\Omega$)



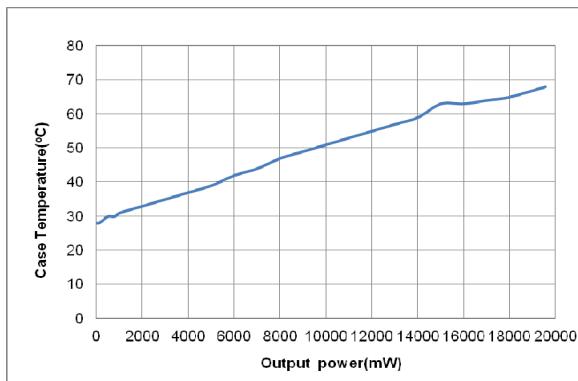
Quiescent Current Vs. Supply Voltage



OSC Frequency Vs. Supply Voltage



Case Temperature Vs. Output Power ($RL=4\Omega$)



Turn-on Response



Application Information

Input Capacitors (Ci)

In the typical application, an input capacitor Ci, is required to allow the amplifier to bias the input signal to the proper DC level for optimum operation. In this case, Ci and the minimum input impedance Ri form a high-pass filter with the corner frequency determined in the follow equation:

$$f_c = \frac{1}{(2\pi R_i C_i)}$$

It is important to consider the value of Ci as it directly affects the low frequency performance of the circuit. For example, when Ri is 40kΩ and the specification calls for a flat bass response are down to 20Hz. The equation is reconfigured as followed to determine the value of Ci:

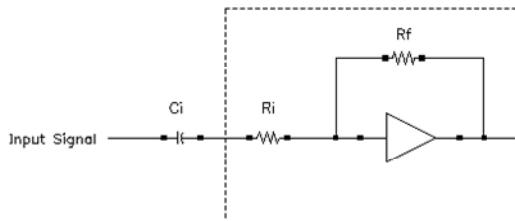
$$C_i = \frac{1}{(2\pi R_i f_c)}$$

When input resistance variation is considered Ci is 200nF, so one would likely choose a value of 220nF. A further consideration for this capacitor is the leakage path from the input source through the input network (Ci, Ri and Rf) to the load. This leakage current creates a DC offset voltage at the input to the amplifier that reduces useful headroom, especially in high gain applications. For this reason, a low-leakage tantalum or ceramic capacitor is the best choice. When polarized capacitors are used, the positive side of the capacitor should face the amplifier input in most applications as the DC level is held at VDD/2, which is likely higher than the source DC level. Please note that it is important to confirm the capacitor polarity in the application.

Input Resistance

The value of the input resistance (Ri) of the amplifier is 40kΩ ±20%. If a single capacitor is added to the input of the high-pass filter the –3dB cutoff frequency can be calculated using equation:

$$f_c = \frac{1}{(2\pi R_i C_i)}$$



Gain Formula with External Input Resistor

The default gain of PAM8320 is 26dB. The gain can be reduced by adding one external resistor between input decoupling capacitor and IN PIN. The gain formula is as below:

$$A_v = \frac{20}{1 + 14 \times \frac{R_x}{400k}}$$

Note: Rx is external input resistor

Power and Heat Dissipation

Speakers must be chosen to withstand the large output power from the PAM8320, otherwise speaker damage may occur.

Heat dissipation is very important when the device works in full power operation. Two factors affect the heat dissipation, the efficiency of the device that determines the dissipation power and the thermal resistance of the package that determines the heat dissipation capability.

The PAM8320 class-D amplifier is highly efficiency and should not need heat sink. Operating at higher powers a heat sink still may not be necessary if the PCB is carefully designed to achieve good thermal dissipation.

Dual-Side PCB

To achieve good heat dissipation the PCB's copper plate should be thicker than 35um and the copper plate on both sides of the PCB should be utilized for heat sink.

The thermal pad on the bottom of the device should be soldered to the plate of the PCB and via holes (usually 9 to 16) should be drilled in the PCB area under the device. Deposited copper on the vias should be thick enough so that the heat can be dissipated to the other side of the plate. There should be no insulation mask on the other side of the copper plate. More vias can and should be added to the PCB around the device for further thermal optimization.

How to Reduce EMI

Most applications require a ferrite bead filter for EMI elimination shown at Figure 1. The ferrite filter reduces EMI around 1MHz and higher. When selecting a ferrite bead it should be chosen with high impedance at high frequencies but low impedance at low frequencies.

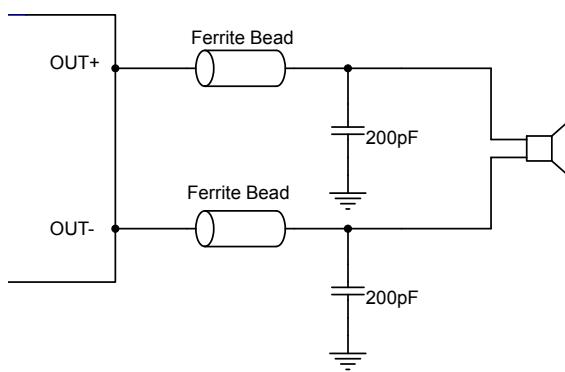


Figure 1: Ferrite Bead Filter to Reduce EMI

Shutdown Operation

The PAM8320 employs a shutdown operation mode to reduce supply current to the absolute minimum level during periods of non-use to save power. The SDN input terminal should be pull high during normal operation. Pulling SDN low causes the outputs to be muted and the amplifier enters a low-current state. SDN should never be left unconnected.

Anti-POP and Anti-Click Circuitry

The PAM8320 contains circuitry to minimize turn-on and turn-off transients or "click and pops", where turn-on refers to either power supply turn-on or device recover from shutdown mode. When the device is turned on, the amplifiers are internally muted. An internal current source ramps up the internal reference voltage. The device will remain in mute mode until the reference voltage reaches half supply voltage. As soon as the reference voltage is stable, the device will begin full operation. For the best power-off pop performance, the amplifier should be set in shutdown mode prior to removing the power supply voltage.

Internal Bias Generator Capacitor Selection

The internal bias generator (VCM) provides the internal bias for the preamplifier stage. The external input capacitors and this internal reference allow the inputs to be biased within the optimal common-mode range of the input preamplifiers.

The selection of the capacitor value on the VCM terminal is critical for achieving the best device performance. During startup or recovery from shutdown state the VCM capacitor determines the rate at which the amplifier starts up. The startup time is not critical for the best de-pop performance since any heard pop sound is the result of the class-D output switching-on other than that of the startup time. However, at least a 0.47µF capacitor is recommended for the VCM capacitor.

Another function of the VCM capacitor is to bypass high frequency noise on the internal bias generator.

Power Supply Decoupling, CS

The PAM8320 is a high-performance CMOS audio amplifier that requires adequate power supply decoupling to ensure the output total harmonic distortion (THD) as low as possible. Power supply decoupling also prevents the oscillations causing by long lead length between the amplifier and the speaker.

Optimum decoupling is achieved by using two different types of capacitors that target different types of noise on the power supply leads. Higher frequency transients, spikes or digital hash should be filtered with a good low equivalent-series-resistance (ESR) ceramic capacitor with a value of typically $0.1\mu F$. This capacitor should be placed as close as possible to the PVCC pin of the device. Lower frequency noise signals should be filtered with a large ceramic capacitor of $470\mu F$ or greater. It's recommended to place this capacitor near the audio power amplifier. The $10\mu F$ capacitor also serves as a local storage capacitor for supplying current during large signal transients on the amplifier outputs.

BSN and BSP Capacitors

The half H-bridge output stages use NMOS transistors therefore requiring bootstrap capacitors for the high side of each output to turn on correctly. A ceramic capacitor $220nF$ or more rated for over 25V must be connected from each output to its corresponding bootstrap input. Specifically, one $220nF$ capacitor must be connected from OUTN to BSN and another $220nF$ capacitor from OUTP to BSP. It is recommended to use $1\mu F$ BST capacitor to replace $220nF$ for lower than 100Hz applications.

VCLAMP Capacitors

To ensure that the maximum gate-to-source voltage for the NMOS output transistors is not exceeded, an internal regulator is used to clamp the gate voltage. A $1\mu F$ capacitor must be connected from VCLAMP to ground and must be rated for at least 25V. The voltages at the VCLAMP terminals vary with VCC and may not be used to power any other circuitry.

Using low-ESR Capacitors

Low-ESR capacitors are recommended throughout this application section. A real (with respect to ideal) capacitor can be modeled simply as a resistor in series with an ideal capacitor. The voltage drop across this resistor minimizes the beneficial effects of the capacitor in the circuit. The lower the equivalent value of this resistance the more the real capacitor behaves as an ideal capacitor.

Short-circuit Protection

The PAM8320 has short circuit protection circuitry on the outputs to prevent damage to the device when output-to-output shorts (BTL mode), output-to-GND shorts, or output-to-VCC shorts occur. Once a short-circuit is detected on the outputs, the output drive is immediately disabled. This is not a latched fault, if the short is removed the normal operation is restored.

Thermal Protection

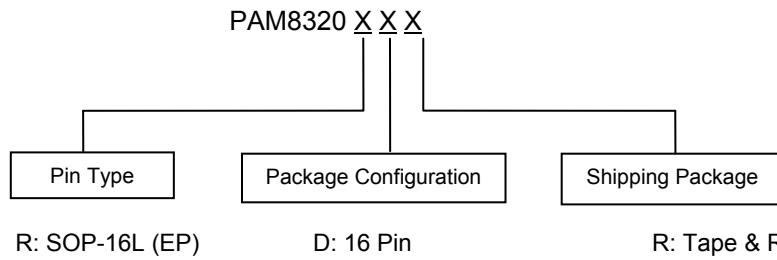
Thermal protection prevents the device from damage. When the internal die temperature exceeds a typical of $160^{\circ}C$ the device will enter a shutdown state and the outputs are disabled. This is not a latched fault, once the thermal fault is cleared and the temperature of the die decreased by $40^{\circ}C$ the device will restart with no external system interaction.

Over Voltage Protection and Under Voltage Lock-out (OVP and UVLO)

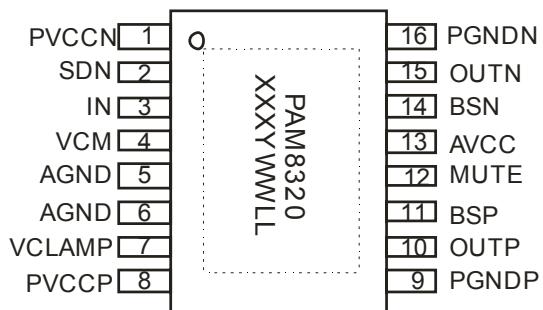
An over voltage protection (OVP) circuit is integrated in PAM8320, when the supply voltage is over 18V the OVP is active and then the output stage is disabled. The PAM8320 will auto recovery when the supply voltage is lower than the OVP threshold.

The PAM8320 incorporates circuitry designed to detect low supply voltage. When the supply voltage drops to 4.4V or below, the PAM8320 goes into a state of shutdown. When the supply voltage is higher than 4.5V normal operation is resumed.

PAM8320

Ordering Information

Part Number	Package	Standard Package
PAM8320RDR	SO-16EP	2,500Units/Tape&Real

Marking Information

PAM8320: Product Code

X: Internal Code

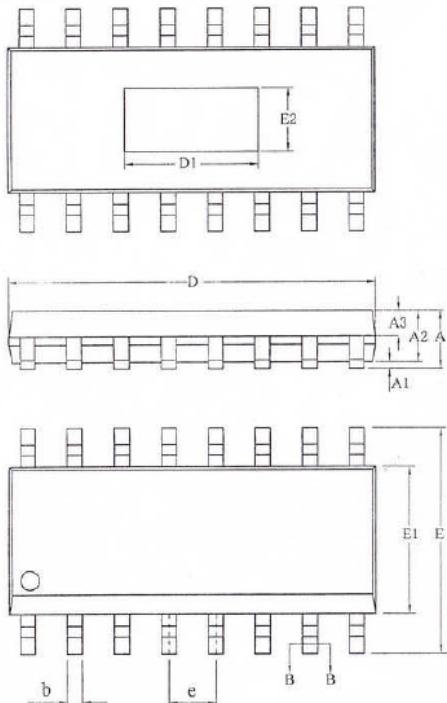
Y: Year

W: Week

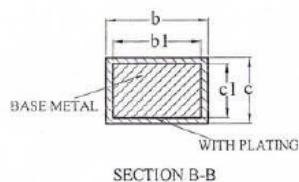
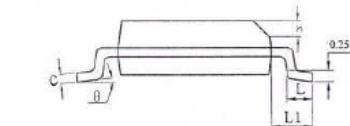
LL: Internal Code

Package Outline Dimensions (All dimensions in mm.)

Package: SO-16EP



SYMBOL	MILLIMETER		
	MIN	NOM	MAX
A	—	—	1.75
A1	0.05	—	0.225
A2	1.30	1.40	1.50
A3	0.60	0.65	0.70
b	0.39	—	0.48
b1	0.38	0.41	0.43
c	0.21	—	0.26
c1	0.19	0.20	0.21
D	9.70	9.90	10.10
E	5.80	6.00	6.20
E1	3.70	3.90	4.10
e	1.27BSC		
h	0.25	—	0.50
L	0.50	—	0.80
L1	1.05BSC		
Ø	Ø	—	8°
D1	3.86REF		
E2	1.67REF		
LT载体尺寸 (mil)	95*180		



IMPORTANT NOTICE

DIODES INCORPORATED MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARDS TO THIS DOCUMENT, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE (AND THEIR EQUIVALENTS UNDER THE LAWS OF ANY JURISDICTION).

Diodes Incorporated and its subsidiaries reserve the right to make modifications, enhancements, improvements, corrections or other changes without further notice to this document and any product described herein. Diodes Incorporated does not assume any liability arising out of the application or use of this document or any product described herein; neither does Diodes Incorporated convey any license under its patent or trademark rights, nor the rights of others. Any Customer or user of this document or products described herein in such applications shall assume all risks of such use and will agree to hold Diodes Incorporated and all the companies whose products are represented on Diodes Incorporated website, harmless against all damages.

Diodes Incorporated does not warrant or accept any liability whatsoever in respect of any products purchased through unauthorized sales channel. Should Customers purchase or use Diodes Incorporated products for any unintended or unauthorized application, Customers shall indemnify and hold Diodes Incorporated and its representatives harmless against all claims, damages, expenses, and attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized application.

Products described herein may be covered by one or more United States, international or foreign patents pending. Product names and markings noted herein may also be covered by one or more United States, international or foreign trademarks.

This document is written in English but may be translated into multiple languages for reference. Only the English version of this document is the final and determinative format released by Diodes Incorporated.

LIFE SUPPORT

Diodes Incorporated products are specifically not authorized for use as critical components in life support devices or systems without the express written approval of the Chief Executive Officer of Diodes Incorporated. As used herein:

A. Life support devices or systems are devices or systems which:

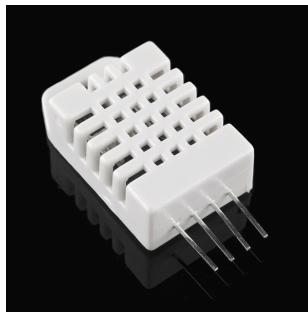
1. are intended to implant into the body, or
2. support or sustain life and whose failure to perform when properly used in accordance with instructions for use provided in the labeling can be reasonably expected to result in significant injury to the user.

B. A critical component is any component in a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or to affect its safety or effectiveness.

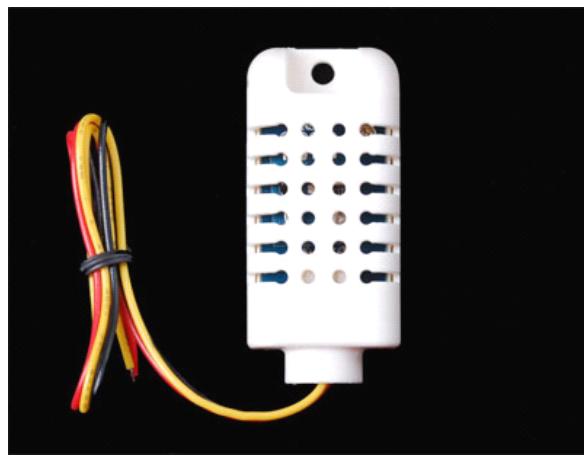
Customers represent that they have all necessary expertise in the safety and regulatory ramifications of their life support devices or systems, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of Diodes Incorporated products in such safety-critical, life support devices or systems, notwithstanding any devices- or systems-related information or support that may be provided by Diodes Incorporated. Further, Customers must fully indemnify Diodes Incorporated and its representatives against any damages arising out of the use of Diodes Incorporated products in such safety-critical, life support devices or systems.

Copyright © 2013, Diodes Incorporated

www.diodes.com



Standard AM2302/DHT22



AM2302/DHT22 with big case and wires

Digital relative humidity & temperature sensor AM2302/DHT22

1. Feature & Application:

- *High precision
- *Capacitive type
- *Full range temperature compensated
- *Relative humidity and temperature measurement
- *Calibrated digital signal
- *Outstanding long-term stability
- *Extra components not needed
- *Long transmission distance, up to 100 meters
- *Low power consumption
- *4 pins packaged and fully interchangeable

2. Description:

AM2302 output calibrated digital signal. It applies exclusive digital-signal-collecting-technique and humidity sensing technology, assuring its reliability and stability. Its sensing elements are connected with 8-bit single-chip computer.

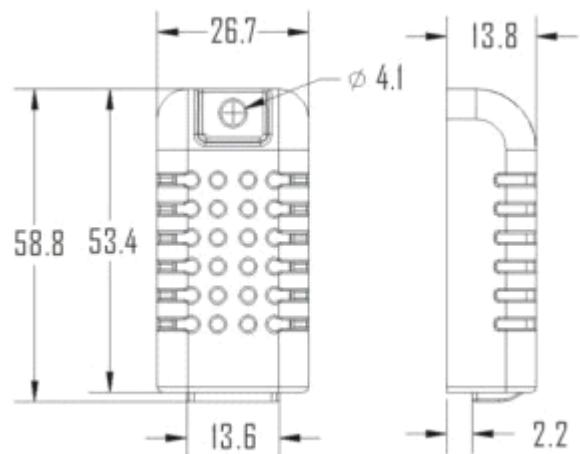
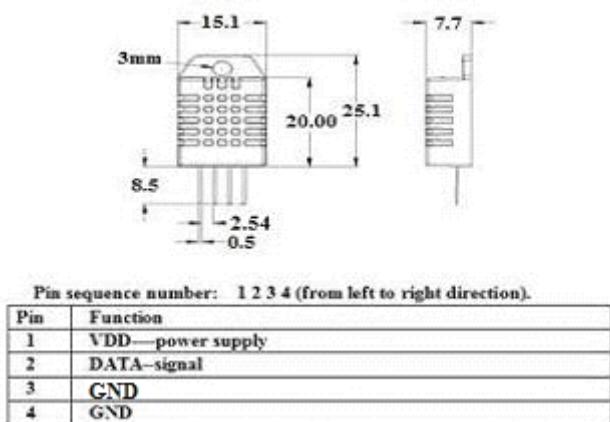
Every sensor of this model is temperature compensated and calibrated in accurate calibration chamber and the calibration-coefficient is saved in type of programme in OTP memory, when the sensor is detecting, it will cite coefficient from memory.

Small size & low consumption & long transmission distance(100m) enable AM2302 to be suited in all kinds of harsh application occasions. Single-row packaged with four pins, making the connection very convenient.

3. Technical Specification:

Model	AM2302	
Power supply	3.3-5.5V DC	
Output signal	digital signal via 1-wire bus	
Sensing element	Polymer humidity capacitor	
Operating range	humidity 0-100%RH;	temperature -40~80Celsius
Accuracy	humidity +2%RH(Max +5%RH);	temperature +-0.5Celsius
Resolution or sensitivity	humidity 0.1%RH;	temperature 0.1Celsius
Repeatability	humidity +-1%RH;	temperature +-0.2Celsius
Humidity hysteresis	+0.3%RH	
Long-term Stability	+0.5%RH/year	
Interchangeability	fully interchangeable	

4. Dimensions: (unit---mm)



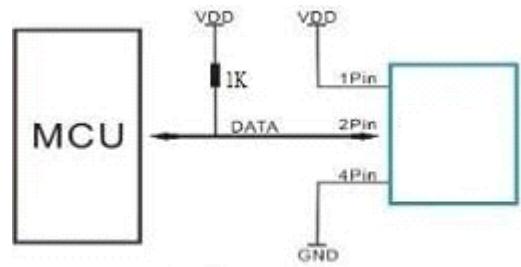
Standard AM2302's dimensions as above

Big case's dimensions as above

Red wire—power supply, Black wire—GND

Yellow wire—Data output

5. Electrical connection diagram:



6. Operating specifications:

(1) Power and Pins

Power's voltage should be 3.3-5.5V DC. When power is supplied to sensor, don't send any instruction to the sensor within one second to pass unstable status. One capacitor valued 100nF can be added between VDD and GND for wave filtering.

(2) Communication and signal

1-wire bus is used for communication between MCU and AM2302. (Our 1-wire bus is specially designed, it's different from Maxim/Dallas 1-wire bus, so it's incompatible with Dallas 1-wire bus.)

Illustration of our 1-wire bus:

DATA=16 bits RH data+16 bits Temperature data+8 bits check-sum

Example: MCU has received 40 bits data from AM2302 as

0000 0010 1000 1100 0000 0001 0101 1111 1110 1110
 16 bits RH data 16 bits T data check sum

Here we convert 16 bits RH data from binary system to decimal system,

0000 0010 1000 1100 → 652
 Binary system Decimal system

RH=652/10=65.2%RH

Here we convert 16 bits T data from binary system to decimal system,

0000 0001 0101 1111 → 351
 Binary system Decimal system

T=351/10=35.1°C

When highest bit of temperature is 1, it means the temperature is below 0 degree Celsius.

Example: 1000 0000 0110 0101, T= minus 10.1°C

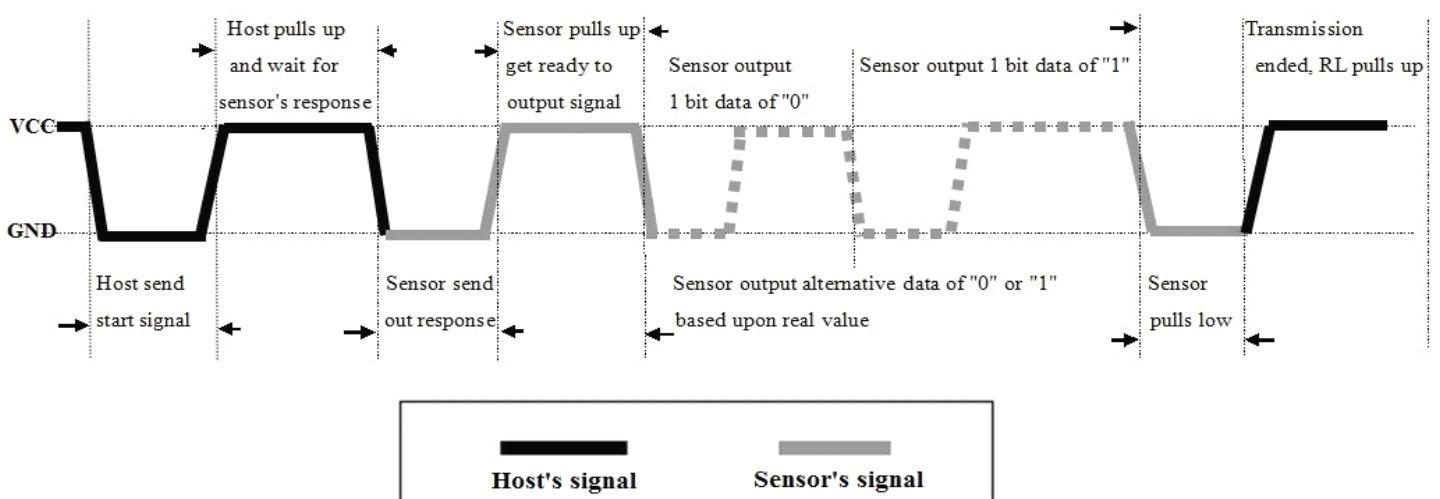
16 bits T data

Sum=0000 0010+1000 1100+0000 0001+0101 1111=1110 1110

Check-sum=the last 8 bits of Sum=1110 1110

When MCU send start signal, AM2302 change from standby-status to running-status. When MCU finishes sending the start signal, AM2302 will send response signal of 40-bit data that reflect the relative humidity and temperature to MCU. Without start signal from MCU, AM2302 will not give response signal to MCU. One start signal for one response data from AM2302 that reflect the relative humidity and temperature. AM2302 will change to standby status when data collecting finished if it don't receive start signal from MCU again.

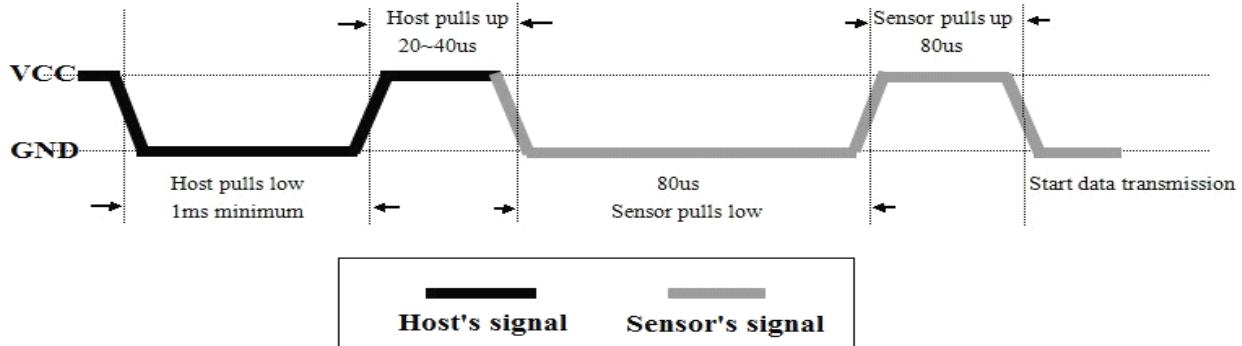
See below figure for overall communication process, **the interval of whole process must beyond 2 seconds**.



- 1) Step 1: MCU send out start signal to AM2302 and AM2302 send response signal to MCU

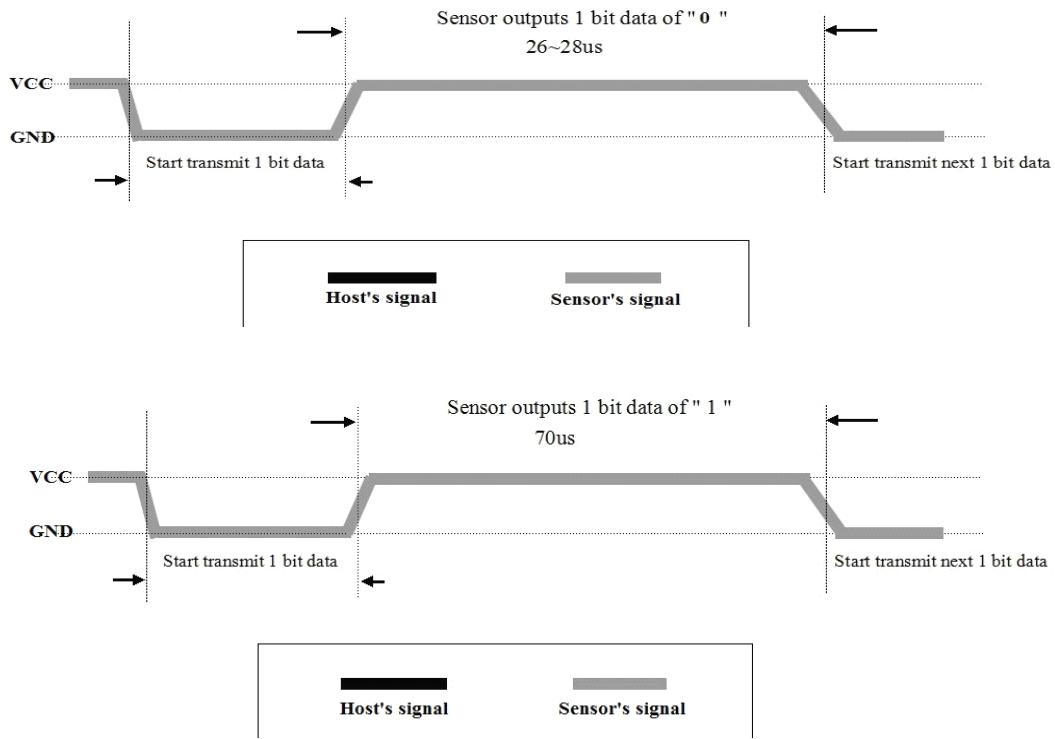
Data-bus's free status is high voltage level. When communication between MCU and AM2302 begins, MCU will pull low data-bus and this process must beyond at least 1~10ms to ensure AM2302 could detect MCU's signal, then MCU will pulls up and wait 20-40us for AM2302's response.

When AM2302 detect the start signal, AM2302 will pull low the bus 80us as response signal, then AM2302 pulls up 80us for preparation to send data. See below figure:



2). Step 2: AM2302 send data to MCU

When AM2302 is sending data to MCU, every bit's transmission begin with low-voltage-level that last 50us, the following high-voltage-level signal's length decide the bit is "1" or "0". See below figures:



Attention:

If signal from AM2302 is always high-voltage-level, it means AM2302 is not working properly, please check the electrical connection status.

7. Electrical Characteristics:

Items	Condition	Min	Typical	Max	Unit
Power supply	DC	3.3	5	6	V
Current supply	Measuring	1		1.5	mA
	Stand-by	40	Null	50	uA
Collecting period	Second		2		Second

8. Attentions of application:

(1) Operating and storage conditions

We don't recommend the applying RH-range beyond the range stated in this specification. The AM2302 sensor can recover after working in abnormal operating condition to calibrated status, but will accelerate sensors' aging.

(2) Attentions to chemical materials

Vapor from chemical materials may interfere AM2302's sensitive-elements and debase AM2302's sensitivity.

(3) Disposal when (1) & (2) happens

Step one: Keep the AM2302 sensor at condition of Temperature 50~60Celsius, humidity <10%RH for 2 hours;

Step two: After step one, keep the AM2302 sensor at condition of Temperature 20~30Celsius, humidity >70%RH for 5 hours.

(4) Attention to temperature's affection

Relative humidity strongly depend on temperature, that is why we use temperature compensation technology to ensure accurate measurement of RH. But it's still be much better to keep the sensor at same temperature when sensing.

AM2302 should be mounted at the place as far as possible from parts that may cause change to temperature.

(5) Attentions to light

Long time exposure to strong light and ultraviolet may debase AM2302's performance.

(6) Attentions to connection wires

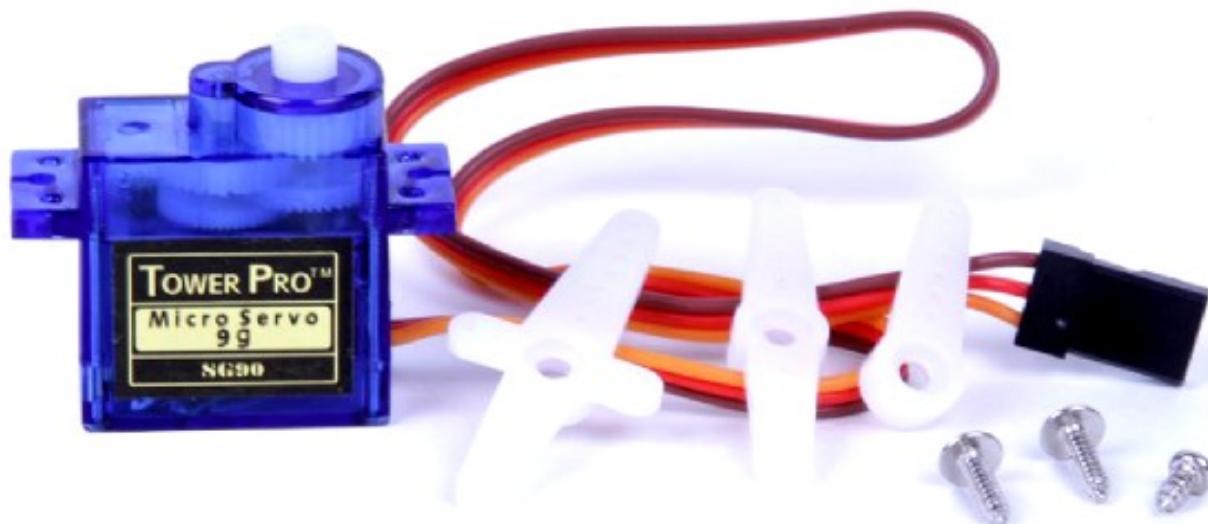
The connection wires' quality will effect communication's quality and distance, high quality shielding-wire is recommended.

(7) Other attentions

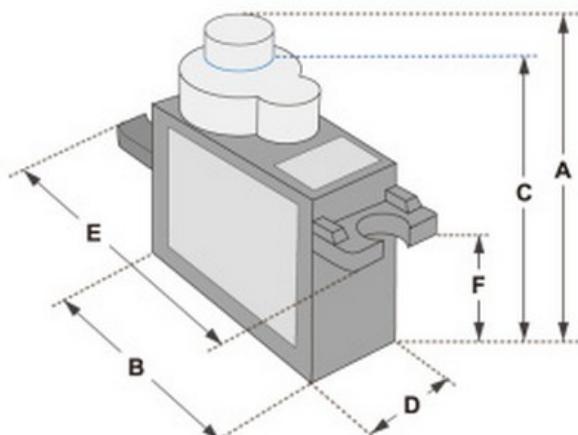
* Welding temperature should be bellow 260Celsius.

* Avoid using the sensor under dew condition.

* Don't use this product in safety or emergency stop devices or any other occasion that failure of AM2302 may cause personal injury.



Tiny and lightweight with high output power. Servo can rotate approximately 180 degrees (90 in each direction), and works just like the standard kinds but smaller. You can use any servo code, hardware or library to control these servos. Good for beginners who want to make stuff move without building a motor controller with feedback & gear box, especially since it will fit in small places. It comes with a 3 horns (arms) and hardware.



Dimensions & Specifications

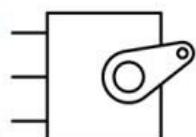
A (mm) : 32
B (mm) : 23
C (mm) : 28.5
D (mm) : 12
E (mm) : 32
F (mm) : 19.5
Speed (sec) : 0.1
Torque (kg-cm) : 2.5
Weight (g) : 14.7
Voltage : 4.8 - 6

Position "0" (1.5 ms pulse) is middle, "90" (~2ms pulse) is middle, is all the way to the right, "-90" (~1ms pulse) is all the way to the left.

PWM=Orange (脉冲)

Vcc=Red (+)

Ground=Brown (-)



1 - 2 ms
Duty Cycle

4.8 V (~5 V)
Power
and Signal



20 ms (50 Hz)
PWM Period