3 Microcontroladores

Un microcontrolador es un pequeño ordenador que contiene en su interior básicamente un procesador, soporte (reloj y reset), memoria y puertos de entrada-salida, todo ello dentro de un pequeño chip que podemos programar con total flexibilidad y relativa facilidad.

3.1 Tipos de microcontroladores

3.1.1 Microcontroladores de 8 bit (“embedded”)

El termino “incrustado”, del inglés “embedded”, define su estructura, es decir, que todos los recursos necesarios a nivel de hardware (memoria, procesador, etc.) están contenidos dentro del microcontrolador, así solo se necesita alimentarlo (pila o fuente de alimentación) y configurar la señal del oscilador de reloj, para que el microcontrolador se ponga en funcionamiento. Proporcionan un sistema programable de bajo coste, con posibilidad de conectar con otros dispositivos externos, así como un control sofisticado en ciertas aplicaciones. Nos encontraremos en este grupo con microcontroladores entre 2€ y 40€ aproximadamente, dependiendo de su velocidad y capacidad de memoria.

Por lo general estos dispositivos gozan de las características que a continuación se enumeran y que se recogen gráficamente en la Figura 3.1:

- Entrada de RESET: pad por el que podemos reiniciar el chip en cualquier momento para que vuelva al inicio del programa.

- RELOJ: El controlador ejecuta el programa a la frecuencia del reloj. El reloj puede ser interno, o externo, usando un cristal de cuarzo o un circuito resonante LC, o incluso un circuito RC. Al alimentar el microcontrolador el reloj comienza a operar.

- Procesador CENTRAL: es la CPU del microcontrolador. Su función es sacar, decodificar y ejecutar las instrucciones almacenadas en la memoria de programa.

- Memoria de programa: contiene el programa a ejecutar. Puede ser de varios tipos: ROM, de sólo lectura, por lo que viene programada de fábrica; EPROM, programable por el usuario, pero difícilmente reprogramable; EEPROM, programable y borrable eléctricamente, lo que permite un control total, cómodo y rápido por parte del usuario.

- Memoria RAM: es la memoria de trabajo, en la que se realizarán las operaciones con las variables de programa definidas.

- Registros hardware: pueden ser de dos tipos, registros internos del procesador y los registros usados para controlar los dispositivos externos.

- Puertos de E/S: son las conexiones con el mundo exterior. Por estas patillas podremos manejar dispositivos externos de salidas (LED, relés, etc.) y leer dispositivos de entrada (pulsadores, interruptores, sensores, etc.).

- Contadores y divisores: empleados en procesos que requieran un control del tiempo, como relojes, alarmas, temporizadores... y cualquier proceso que requiera controlar períodos de tiempo...

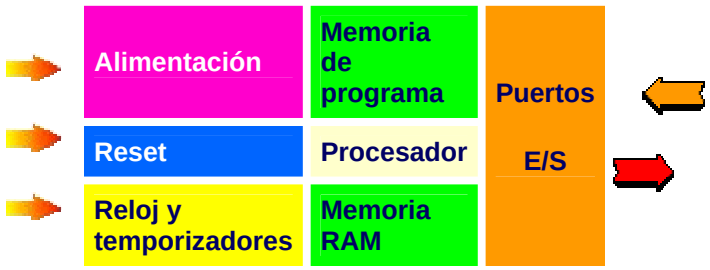


Figura 3.1 ***Representación esquemática de un microcontrolador***

Además de las características básicas descritas, pueden llevar añadidas otras más sofisticadas que incrementan la potencia del microcontrolador notablemente y en ocasiones resultan imprescindibles, como las que se indican a continuación:

- Programa de depuración o monitorización (debugger): permite comprobar en tiempo real el funcionamiento del programa y detectar errores en el mismo de manera sencilla.

- Interrupciones: Eventos que hacen que se ejecuten rutinas concretas. Todo ello independientemente de la instrucción que se este ejecutando.

- Puertos analógicos de E/S: si los microcontroladores incluyen conversores analógico-digitales y/o digitales-analógicos, podremos leer señales analógicas o generar señales analógicas, con lo que la comunicación con el exterior se hace más cómoda y flexible.

- Puertos serie: permiten una comunicación fácil con un ordenador personal o con otros dispositivos que usen puertos serie (RS232, CAN, SPI, etc.).

- Interfaces con memoria externa: si lo incluyen permite ampliar la capacidad de memoria para procesar programas más extensos o disponer de memoria RAM adicional.

3.2 Microcontroladores Atmel AVR

Los microcontroladores usados, siguiendo las especificaciones del proyecto general, son de la serie AVR de Atmel. Se ha optado por esta serie por razones técnicas y por dar continuidad al trabajo realizado en Dept. de Ingeniería de Sistemas y

Automática. Se facilita así el seguimiento de proyectos futuros, reduciendo el gasto de desarrollo de software y hardware, y dejando abierta una línea de investigación.

La tecnología de los microcontroladores AVR es CMOS de 8 bits con bajo consumo, basados en arquitectura RISC. Esto implica que poseen un reducido juego de instrucciones, la mayor parte de las cuales se ejecutan en un único ciclo de reloj, consiguiendo una capacidad de procesamiento cercana a 1MIPS por MHz, permitiendo al diseñador del sistema optimizar el consumo gracias a la gran velocidad de procesamiento. La familia AVR utiliza el concepto de arquitectura Harvard con buses y memorias separados para los datos y el programa, permitiendo que las instrucciones sencillas sean ejecutadas en un ciclo de reloj.

Según el modelo, incorporan diferentes tamaños de memoria flash, RAM y EEPROM. Además es posible usar una SRAM externa mediante el uso de un bus de datos y direcciones multiplexado junto con las señales de control usuales en modo Intel (RD, WR, CS, ALE). La memoria flash permite programar el microcontrolador incluso una vez ya montado en la tarjeta final (ISP – In System Programming) sin recurrir a niveles de tensión especiales. La memoria EEPROM permite almacenar datos para conservarlos aún cuando se apague el dispositivo, útil para guardar configuraciones por ejemplo.

La tecnología AVR combina un gran número de instrucciones con 32 registros de propósito general. Los 32 registros están conectados directamente a la unidad aritmético lógica (ALU), permitiendo que dos registros independientes sean accesibles por una instrucción simple ejecutada en un ciclo de instrucción. Resulta una tecnología muy eficiente, que permite una capacidad de procesamiento hasta 10 veces superior que los microcontroladores CISC convencionales.

La familia AVR permite trabajar con frecuencias hasta 16 MHz. Suelen existir dos versiones por modelo. La versión L que trabaja con una tensión de alimentación 2.7 V, alcanzando velocidades de trabajo entre 4 u 8 MHz, y la versión superior que trabaja con alimentación en el rango 4.0 – 5.5 V, alcanzando hasta 8 o 16 MHz.

Características típicas de la familia AVR de microcontroladores:

- Timers y contadores de 8 y 16 bits flexibles con modos de comparación.
- Interrupciones internas y externas.
- UART serie programable.
- USART serie programable (reemplazando a la UART).
- Puerto serie SPI.
- RTC (Real Time Clock).
- TWI (Two Wire serial Interface).
- Líneas de entrada / salida digitales configurables por software.
- Temporizador Watchdog programable con oscilador interno.
- Detector de Brown – Out. (algunos lo llevan incorporado).
- Comparador Analógico.
- Conversor Analógico Digital.
- Interfaz JTAG.
- PWM (Pulses Width Modulation).

El modo “Idle Mode” detiene la CPU permitiendo que la SRAM, los contadores y temporizadores, el puerto SPI y el sistema de interrupciones continúen funcionando. El modo de bajo consumo guarda el contenido de los registros pero detiene el oscilador, deshabilitando todas las funciones del chip hasta que se produzca una interrupción o un reset.

La memoria Flash on-chip Downloadable (descargable) permite que la memoria del chip sea reprogramada a través del interfaz SPI en el propio sistema o mediante programador JTAG.

Combinando una tecnología RISC de 8 bits con una CPU con memoria Flash, la familia de microcontroladores AVR de Atmel proporcionan una elevada flexibilidad en los diseños a bajo coste, dando una solución bastante efectiva para muchas aplicaciones de control.

La familia AVR se complementa con un completo juego de programas y sistemas de desarrollo incluyendo: compiladores C, ensambladores, simuladores, emuladores en circuito, Kits de evaluación, etcétera.

3.3 Elección del microcontrolador

Entre la amplia variedad de microcontroladores AVR se ha seleccionado el modelo ATmega128[1] representado en la Figura 3.8 de la página 47, pues es uno de los más potentes dentro de su gama y tiene todos los dispositivos y periféricos integrados. Los principales factores que influyeron en la elección de este microcontrolador fueron su velocidad, amplia memoria de programa y RAM internas y, sobre todo, que cumple la necesidad fundamental de incorporar dos UART.

La velocidad es importante porque el sistema va a actuar como módulo inteligente exterior al PLC capaz de gestionar las comunicaciones PLC ⇔ módem GSM para liberar de este modo al autómata programable de dicha tarea y evitar así la continua interrupción del que sería su ciclo de trabajo habitual. Por ello es importante asegurarnos que en ningún caso el PLC tenga que quedarse en ciclos de espera aguardando la respuesta del microcontrolador.

Siempre es interesante contar con una amplia memoria para alojar el código del programa pues es improbable que se sepa a priori la longitud de dicho código y sus requerimientos de memoria RAM para almacenar variables intermedias. La posibilidad de contar con memoria Flash EEPROM resultó también particularmente útil a posteriori. La gran cantidad de memoria disponible (128 KB de memoria flash y 4KB de SRAM garantizaban, desde un primer momento, poder diseñar la circuiteria sin necesidad de incluir chips de memoria externa que complicasen el diseño).

[1] Nos referiremos al microcontrolador blanco de nuestra elección como ATmega128 o ATmega128L indistintamente. El microcontrolador empleado fue el ATmega128L, versión de bajo consumo del anterior. El hecho de que ambos sean utilizables con los diseños tanto hardware como software propuestos, hace que obviemos esta distinción.

Este dispositivo presenta dos puertos USART, uno de ellos se utilizará para comunicarse con el autómata programable y el otro para comunicarse con el módem GSM. Se consideraron otras opciones que se enumeran a continuación para evitar tener que utilizar un microcontrolador con dos UART y tratar de utilizar otra versión que contase con sólo una, pero todas resultaron descartadas finalmente:

- Utilizar una sola UART multiplexada. Esta solución complica el hardware extraordinariamente pues es necesario integrar el multiplexador además de su lógica de selección cableada. Además, y peor aún, impide el acceso simultáneo del módem y el PLC a la tarjeta de comunicaciones y, dada la aleatoriedad de la llegada de mensajes al sistema, obligaría a implementar una gestión de los mensajes muy avanzada.

- Utilizar un microcontrolador con una sola UART y otra externa. Esta solución solventa el problema anterior pero su implementación hardware continúa siendo demasiado compleja.

- Utilizar una sola UART y el puerto USB que los microcontroladores AVR modernos disponen on-board. En este caso, es el software el que se complica pues sería necesario gestionar completamente dicho puerto, incluyendo temporizaciones, gestión de colisiones, etcétera.

Existen en el mercado otros microcontroladores de esta familia que cumplen las especificaciones anteriores, dentro de ellos, resulta particularmente interesante el ATmega162 que además cuenta con la ventaja de contar con encapsulado trough-hole frente al encapsulado de montaje superficial (SMD) en que se ofrece el ATmega128, pero su superior coste hizo descartar esta opción.

En la fase de desarrollo previa a la configuración definitiva de la placa se utilizo un microcontrolador ATmega8515 dado su menor coste, mayor disponibilidad y posibilidad de acceso al módulo de desarrollo STK500 de Atmel que permite programar rápidamente dicho microcontrolador sin necesidad de implantar el hardware de un sistema ISP. De este modo, con una placa de prototipado, dicho microcontrolador y sus correspondientes zócalos para un montaje y desmontaje rápido, un adaptador de niveles TTL ⇔ RS-232, un conector DB9 y leds de prueba, se puede desarrollar un primer montaje de evaluación para familiarizarse con el sistema. Por todo ello, y a pesar de que el sistema definitivo se implementó con ATmega128, se introducirán aquí las características fundamentales del Kit de evaluación STK500 y del microcontrolador ATmega8515.

3.3.1 Kit de evaluación STK500

El STK500 es un completo sistema de evaluación y desarrollo para los microcontroladores de la serie AVR Flash de Atmel. Está diseñado para ofrecer a los diseñadores un rápido comienzo para desarrollar código en los AVR, combinado con características avanzadas para el uso del Kit de evaluación como prototipo y ensayo de nuevos diseños. Las características del Kit de evaluación STK500 son:

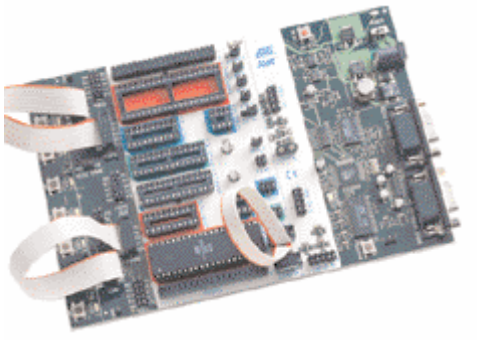


Figura 3.2 ***Kit de evaluación STK 500***

- Compatible con CodeVisionAVR y la mayoría de los compiladores comerciales.
- Interfaz RS-232 con PC para programación y control.
- Alimentación de potencia regulada entre 10-15 V DC.
- Zócalos para dispositivos AVR de 8, 20, 28 y 40 pines.
- Varios modos de programación para dispositivos AVR: “Parallel/Serial High Voltaje” e “ISP (In-system Programming)”.
- Programación ISP para dispositivos AVR en placas externas.
- Reprogramación de dispositivos AVR.
- 8 LEDs y 8 pulsadores para demostración o depuración.
- Todos los puertos entradas/salidas de los microcontroladores AVR son fácilmente accesibles gracias a conectores con pines.
- Puerto RS-232 adicional para uso general.
- Conectores de expansión para ampliaciones.
- Memoria Flash de 2 Mbit para almacenamiento de datos.

Entre todos los dispositivos AVR admitidos por la STK500, se encuentra el microcontrolador usado para adiestramiento, el ATmega8515.

El aspecto físico del Kit con el microcontrolador ya montado se aprecia en la Figura 3.2, pudiendo verse dicho Kit esquematizado en la Figura 3.3, donde se señalan todos los elementos que la componen:

El Kit de evaluación STK500 dispone de un área de zócalos para los dispositivos AVR (‘sockets for target AVR’) que permite conectar varios tipos de microcontroladores AVR, según número de pines. Concretamente, permite programar y testear un microcontrolador ATmega8515 insertado en el zócalo marcado SCKT3000D3 el cual queda indicado más claramente en la Figura 3.3.

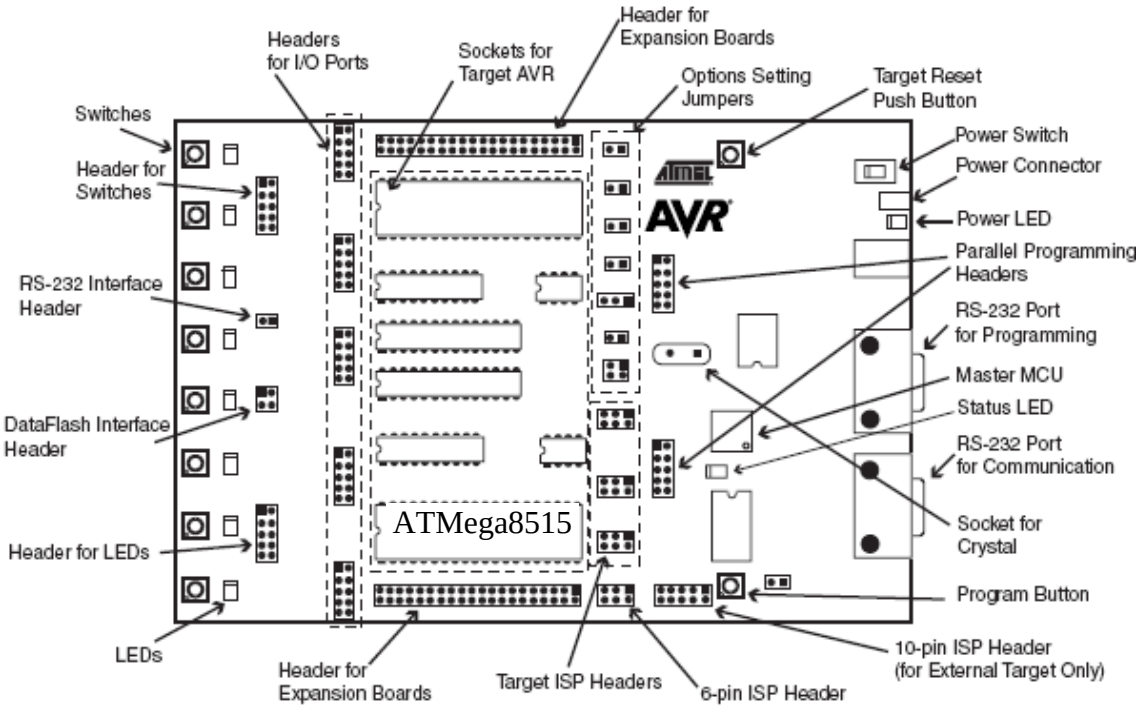


Figura 3.3 ***Esquema del Kit de Evaluación STK500***

El Kit de evaluación tiene una zona de jumpers cuya colocación por defecto permite al microcontrolador trabajar con la fuente de reloj y alimentación de la placa. También podría tener una tensión de referencia, un reset y/o un oscilador, externos.

El sistema de desarrollo requiere una alimentación externa de 10-15 V DC. Se recomienda introducir en 'Power connector' un conector con centro negativo para permitir apagar y encender la alimentación desde el interruptor correspondiente en la placa ('Power switch'), caso de hacerse con un conector con centro positivo, no se garantiza que el switch de la placa funcione correctamente. Al alimentar la placa, el LED de potencia ('Power LED') se ilumina (de color rojo), y el LED de estado ('Status LED') se muestra una luz verde, tras pasar por amarillo, indicando que la alimentación está llegando al microcontrolador.

Para programar el ATmega8515 hay que conectar el cable plano de 6 pines entre el conector "ISP6PIN" y el conector del dispositivo AVR correspondiente, en este caso "SPROG3", que es el más cercano como se ve en la Figura 3.4. Esta conexión marca que el tipo de programación usada es ISP (In-System Programming), utilizando el SPI (Serial Peripheral Interface) interno del microcontrolador AVR para cargar el código en la memoria Flash y EEPROM del dispositivo. Requiere sólo de VCC, GND, RESET y 3 señales para la programación.

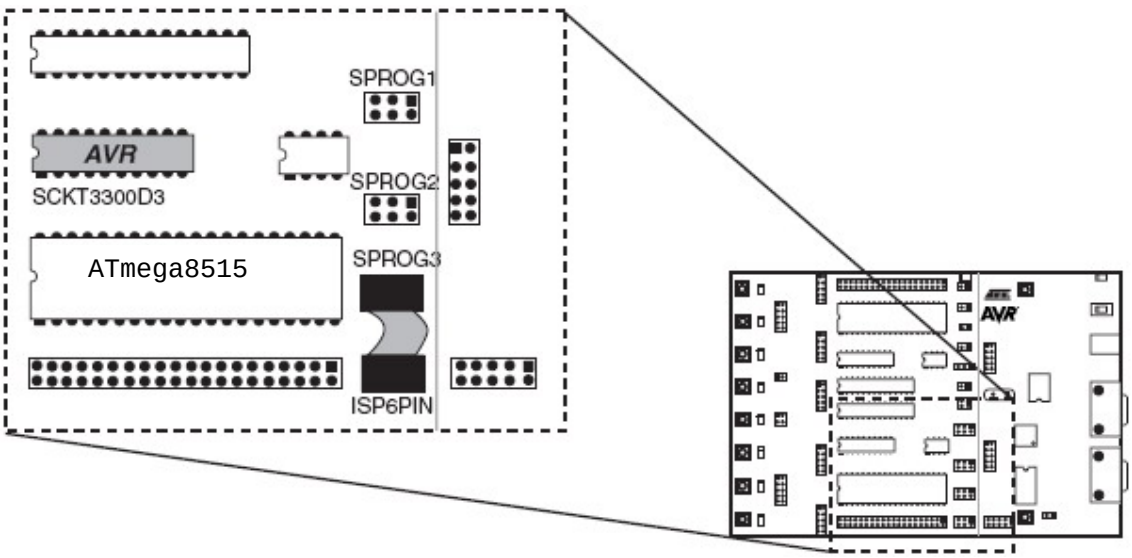


Figura 3.4 ***Conexión cable plano 6 pines para programación ISP***

Para comunicar la placa STK 500 con el PC de trabajo, donde se instalará un compilador adecuado para trabajar, se dispone de un cable serie RS-232, y un par de puertos en la placa: uno para el control, y otro auxiliar. Se conecta el cable entre el puerto RS-232CTRL del Kit y el puerto COM del PC (Figura 3.5).

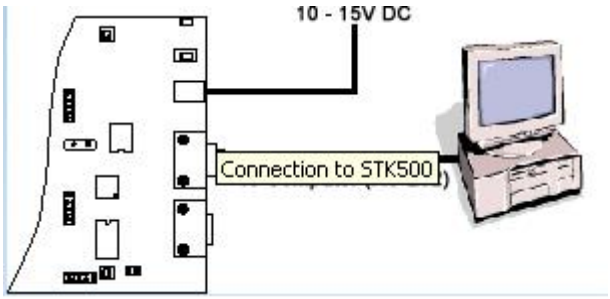


Figura 3.5 ***Conexión entre STK 500 y PC vía RS-232***

Una vez realizadas todas las conexiones de hardware necesarias para la programación del microcontrolador, se abre el programa elegido para compilar el código escrito en C, sólo hemos de tener la precaución de elegir un compilador capaz de dialogar con la STK500, esta discusión se deja para una sección posterior.

El resultado final es que con muy poco hardware (dos cables planos, un cable serie RS-232, la placa STK500, y el microcontrolador) y con un programa software adecuado, (se pueden encontrar versiones totalmente gratuitas en la página web del fabricante www.atmel.com) se realiza la programación de cualquier microcontrolador de la serie AVR fácilmente, siempre que esté admitido por los zócalos de la placa del Kit de evaluación.

3.3.2 ATmega8515

En la Figura 3.6 se muestra una imagen del microcontrolador descrito en esta sección:

Figura 3.6 ***Microcontrolador ATmega8515***

3.3.2.1 Características principales

Las características de este dispositivo son:

- Avanzada arquitectura AVR RISC.
  - 130 potentes instrucciones, la mayoría ejecutadas en un único ciclo de reloj.
  - 32 registros de propósito general de 8 bits.
  - Hasta 16 MIPS trabajando a 16 MHz.

- Memoria de programa no volátil y RAM interna:
  - 8Kbytes de Memoria Flash reprogramable en el sistema final (ISP). Vida útil: 10,000 ciclos escritura / borrado.
  - 512 bytes de memoria RAM estática interna.
  - 512 bytes de memoria EEPROM reprogramable en el sistema final (ISP). Vida útil: 10,000 ciclos escritura / borrado.
  - Protección del programa para seguridad del SW.

- Características de los periféricos:
  - Un temporizador / contador de 8 bits con preescalado y modos de comparación y de captura.
  - Un temporizador / contador de 16 bits con preescalado y modos de comparación y de captura independientes.
  - 3 canales PWM de 8, 9 y 10 bits.
  - Comparador Analógico interno.
  - Temporizador Watchdog programable generado a partir de oscilador interno.
  - UART serie programable.
  - Interfaz SPI Maestro / Esclavo.

- Características especiales:
  - Modos de bajo consumo: Idle, Power-down.
  - Interrupciones externas e internas.

- Especificaciones:
  - Tecnología de procesos CMOS de baja potencia y alta velocidad.
  - Operación completamente estática.

- Consumo de potencia a 4 MHz, 3V, 25ºC:
  - Modo Activo: 3.0 mA.
  - Modo Idle: 1.0 mA.
  - Modo Power-down: <1 μA.

- Líneas de entrada/salida y encapsulados:
  - 32 líneas de entrada/salida programables por SW.
  - Encapsulados 40-lead PDIP, 44-lead PLCC y TQFP.

- Alimentación:
  - 2.7 – 5.5 V para ATmega8515L.
  - 4.5 – 5.5 V para ATmega8515.

- Velocidad:
  - 0 – 8 MHz para ATmega8515L.
  - 0 – 16 MHz para ATmega8515.

3.3.2.2 Descripción de pines

Los pines que presenta el ATmega8515 y que se pueden apreciar en la Figura 3.7 se dividen en dos grupos, los asociados a puertos y los pines para configuración básica del dispositivo (VCC, GND, RESET, XTAL1, XTAL2, ICP, OC1B, ALE). Por otro lado los pines asociados a puertos tienen dos posibles modos de funcionamiento como línea de entrada/salida digital programable por Software o como pin de un determinado periférico interno del microcontrolador. Para más detalle consultar la Figura 3.7.

- **Puerto A (PA7-PA0)**

El puerto A es un puerto de 8-bit bidireccional configurable como entrada o como salida. Los pines del puerto pueden ir provistos de resistencias internas pull-up (seleccionables para cada bit). Cada pin del puerto A como salida puede absorber suficiente intensidad para controlar directamente cargas tipo LED. El puerto A sirve como bus de direcciones y datos de entrada/salida cuando se usa una SRAM externa. Los pines del puerto A se hallan en triestado durante la condición de reset.

- **Puerto B (PB7-PB0)**

El puerto B presenta las mismas características que el puerto A en cuanto a líneas de entrada/salida, presentando adicionalmente funciones relativas al interfaz SPI, el comparador analógico y los contadores.

- **Puerto C (PC7-PC0)**

El puerto C presenta las mismas características que los puertos anteriores para las líneas de entrada/salida digitales. La funcionalidad alternativa que presenta es actuar como byte alto del bus de direcciones en el acceso a una RAM externa.

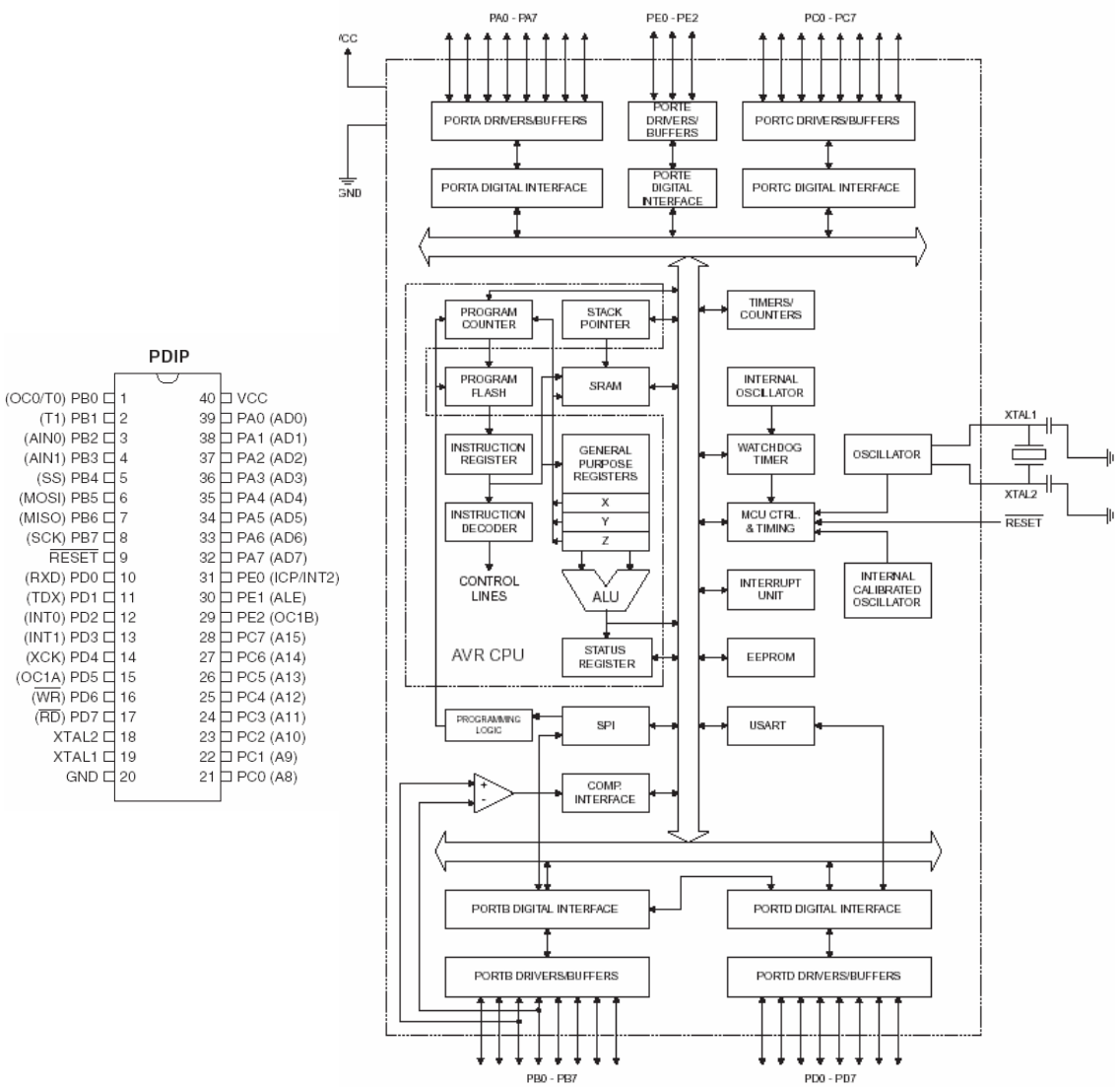


Figura 3.7 ***Configuración de pines y diagrama de bloques del ATmega8515***

- **Puerto D (PD7-PD0)**

El puerto D presenta las mismas características que los puertos anteriores para las líneas de entrada/salida digitales. Las funcionalidades alternativas de los pines de este puerto se refieren al acceso a memorias externas y entradas de interrupción entre otros. Este puerto se considera enormemente interesante pues dos de sus pines permiten la comunicación serie:

PD1 - TXD: UART Output Line.
Línea de Salida de UART.

PD0 - RXD: UART Input Line.
Línea de Entrada de UART.

- **VCC**

Voltaje de alimentación.

- **GND**
  Tierra.

- **RESET**
  Entrada de reset. Una señal por nivel bajo durante más de 50 ns generará un reset, incluso si el reloj no está funcionando. Pequeños pulsos no garantizan la generación de un reset.

- **XTAL1**
  Entrada del oscilador y entrada al circuito que opera con el reloj interno.

- **XTAL2**
  Salida del oscilador.

- **ICP**
  El pin de entrada para la función de captura del Timer / Counter1.

- **OC1B**
  El pin de salida para la función de comparación de salida B del Timer/Counter 1.

- **ALE** (Address Latch Enable)
  Usado cuando la memoria externa está habilitada. El pin habilitador (strobe) ALE es usado para dirigir la dirección de más bajo nivel (8 bits) en un latch de direcciones durante el primer ciclo de acceso.

3.3.2.3 Arquitectura

Una de las ventajas de utilizar microcontroladores de la familia AVR es que tienen una arquitectura común al igual que el juego de instrucciones que es prácticamente el mismo, es por ello que se detallará únicamente la arquitectura para el microcontrolador ATmega128 para evitar multiplicidades, por tanto, se remite al lector a las secciones 3.3.3.3 a 3.3.3.5.

3.3.3 Atmega128

3.3.3.1 Características principales

- Avanzada arquitectura RISC:
  - 133 potentes instrucciones, la mayoría se ejecuta en un ciclo de reloj.
  - 32 registros de trabajo de propósito general de 8 bits.
  - Registros de control de periféricos.
  - Funcionamiento en modo estático.
  - Hasta 16 MIPS funcionando a 16 MHz.
  - Multiplicador de dos ciclos integrado.

- Memorias de trabajo y datos no volátiles
  - 128 KBytes de memoria Flash integrada.

        - Vida útil de 10,000 ciclos de lectura/escritura.
        - Sección de arranque opcional con bits de cerrojo independientes.
    - 4 KBytes de memoria EEPROM.
        - 100,000 ciclos de lectura/escritura de vida útil.
    - 4 KBytes de memoria SRAM.
        - Capacidad de direcionamiento de 64 KBytes de memoria externa.
    - Protección de programa para seguridad del software.
    - Interfaz SPI para programación en sistema final (ISP).

- Interfaz para depuración JTAG
    - Soporte para depuración interna.
    - Programación de memorias (Flash, EEPROM), fusibles y cerrojos a través del puerto JTAG.

- Características de los periféricos:
    - 2 temporizadores/contadores de 8 bits con programación y modos de comparación independientes.
    - 2 temporizadores / contadores de 16 bits con programación, modos de comparación y de captura independientes.

- Contadores en tiempo real.

- 2 canales PWM de 8 bits.

- 6 canales PWM con resolución programable entre 2 y 16 bits.

- Modulador por comparador de salida.

- Conversor analógico digital ADC de 8 canales y 10 bits:
    - 8 canales simples.
    - 7 canales diferenciales.
    - 2 canales diferenciales con ganancia programable 1x, 10x y 200x.

- Interfaz TWI (Two-way Serial Interface) orientado a Byte.

- Dos USART programables.

- Interfaz SPI maestro-esclavo.

- Watchdog programable con temporizador generado a partir de oscilador interno.

- Otras características especiales:
    - Circuito interno de reset sobre alimentación estable.
    - Monitorización de alimentación.
    - Oscilador interno RC calibrado.
    - Interrupciones internas y externas.
    - Modos de bajo consumo: Idle, Power-save, Standby y Extended Standby.
    - Frecuencia de reloj seleccionable por software.
    - Deshabilitación global de resistencias pull-up.

- Líneas de entrada/salida y encapsulados:
  - 53 líneas de entrada/salida programables por SW.
  - Encapsulados 64-lead TQFP y 64-pad MLF.

- Alimentación:
  - 2.7 – 5.5 V para ATmega128L.
  - 4.5 – 5.5 V para ATmega128.

- Velocidad:
  - 0 – 8 MHz para ATmega128L.
  - 0 – 16 MHz para ATmega128.

3.3.3.2 Descripción de pines

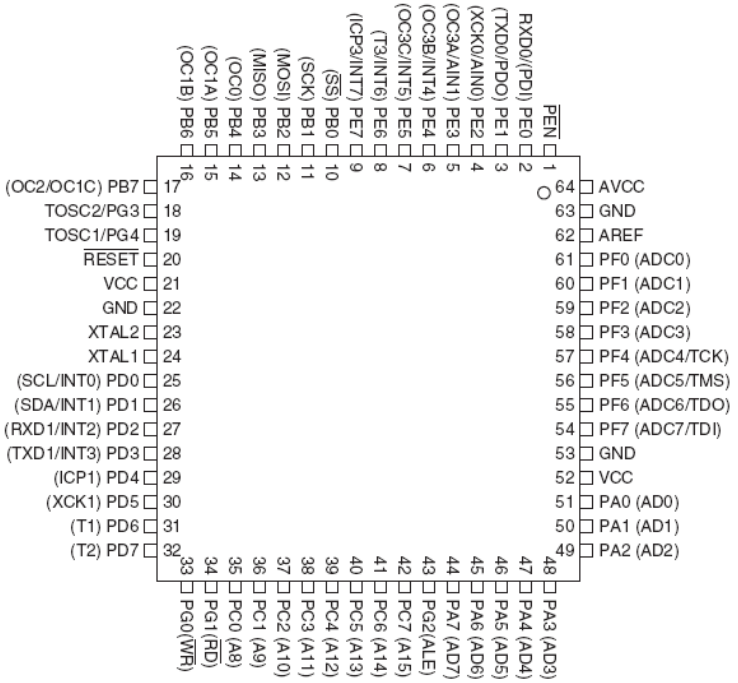


Figura 3.8 ***Configuración de pines del ATmega128***

Los pines que presenta el ATmega128 y que se pueden apreciar en la Figura 3.8 se dividen en dos grupos, los asociados a puertos y los pines para configuración básica del dispositivo (VCC, GND, RESET, XTAL1, XTAL2, ICP, OC1B, ALE). Por otro lado los pines asociados a puertos tienen dos posibles modos de funcionamiento como línea de entrada/salida digital programable por Software o como pin de un determinado periférico interno del microcontrolador. Para más detalle consultar la Figura 3.8 y la Figura 3.9.

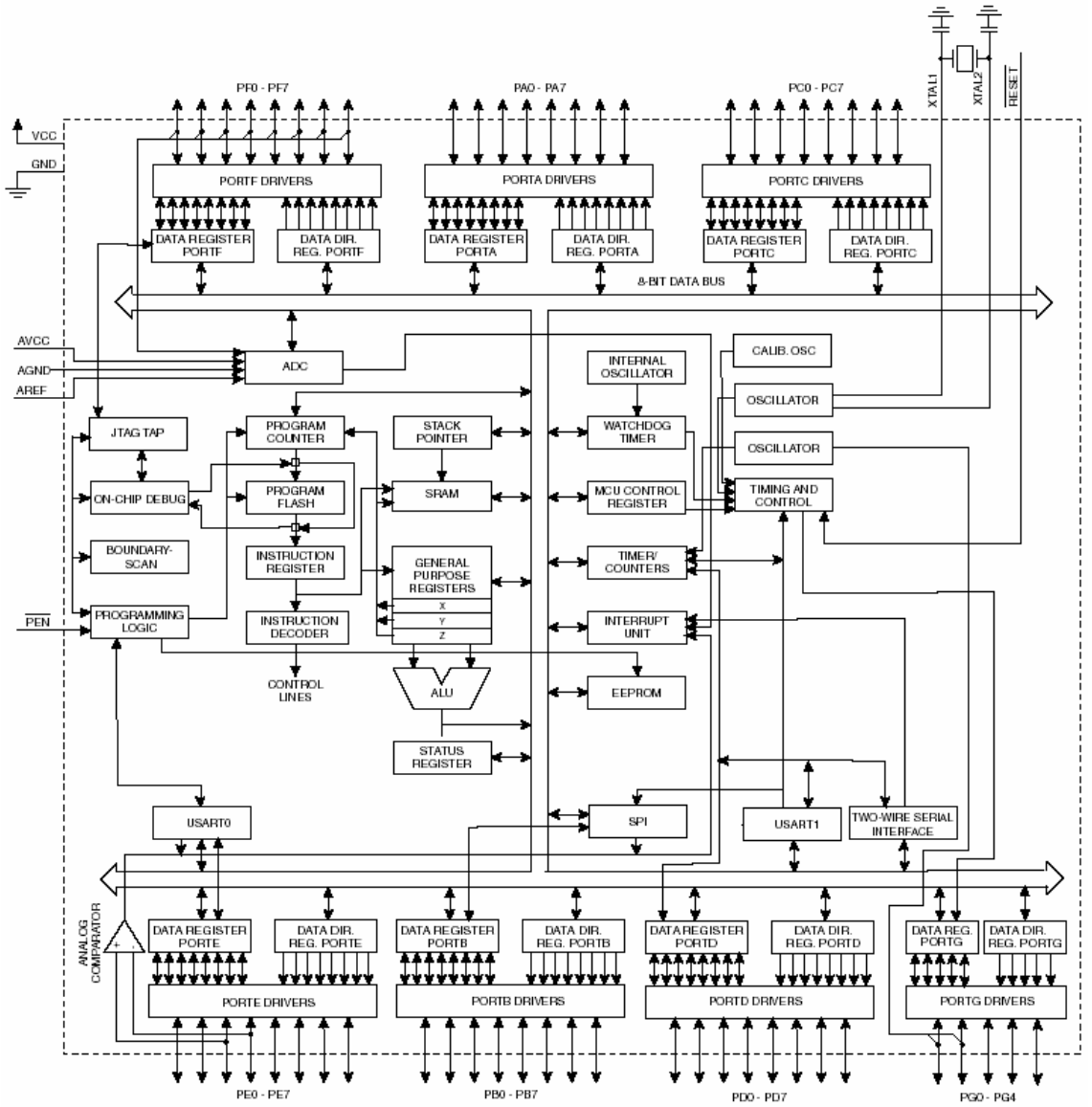


Figura 3.9 ***Diagrama de bloques del ATmega128***

- **Puerto A (PA7-PA0)**

El puerto A es un puerto de 8-bit bidireccional configurable como entrada o como salida. Los pines del puerto pueden ir provistos de resistencias internas pull-up (seleccionables para cada bit). Cada pin del puerto A como salida puede absorber suficiente intensidad para controlar directamente cargas tipo LED. El puerto A sirve como bus de direcciones y datos de entrada/salida cuando se usa una SRAM externa. Los pines del puerto A se hallan en triestado durante la condición de reset.

- **Puerto B (PB7-PB0)**

El puerto B presenta las mismas características que el PORTA en cuanto a líneas de entrada/salida. Las funciones alternativas que presenta cada pin se recogen en la Tabla 3.1.

Pin	Función alternativa
PB7	OC2/OC1C(1) (Output Compare y PWM Output para Timer/Counter2 o Output Compare y PWM Output C para Timer/Counter1)
PB6	OC1B (Output Compare y PWM Output B para Timer/Counter1)
PB5	OC1A (Output Compare y PWM Output A para Timer/Counter1)
PB4	OC0 (Output Compare y PWM Output para Timer/Counter0)
PB3	MISO (SPI Bus Master Input/Slave Output)
PB2	MOSI (SPI Bus Master Output/Slave Input)
PB1	SCK (SPI Bus Serial Clock)
PB0	SS (SPI Slave Select input)

Tabla 3.1 ***Función alternativa de los pines del puerto B***

- **Puerto C (PC7-PC0)**

El puerto C presenta las mismas características que los puertos anteriores para las líneas de entrada/salida digitales. La funcionalidad alternativa que presenta es actuar como byte alto del bus de direcciones en el acceso a una RAM externa.

- **Puerto D (PD7-PD0)**

El puerto D presenta las mismas características que los puertos anteriores para las líneas de entrada/salida digitales. Las funcionalidades alternativas de los pines de este puerto son las especificadas en la Tabla 3.2.

Pin	Función alternativa
PD7	T2 (Timer/Counter2 Clock Input)
PD6	T1 (Timer/Counter1 Clock Input)
PD5	XCK1 (USART1 External Clock Input/Output)
PD4	ICP1 (Timer/Counter1 Input Capture Pin)
PD3	INT3/TXD1 (External Interrupt3 Input o UART1 Transmit Pin)
PD2	INT2/RXD1 (External Interrupt2 Input o UART1 Receive Pin)
PD1	INT1/SDA (External Interrupt1 Input o TWI Serial DAta)
PD0	INT0/SCL (External Interrupt0 Input o TWI Serial CLock)

Tabla 3.2 ***Función alternativa de los pines del puerto D***

- **Puerto E (PE7-PE0)**

El puerto E presenta las mismas características que los puertos anteriores para las líneas de entrada/salida digitales. Las funcionalidades alternativas de los pines de este puerto son las especificadas en la Tabla 3.3.

Pin	Función alternativa
PE7	INT7/ICP3 (External Interrupt 7 Input o Timer/Counter3 Input Capture Pin)
PE6	INT6/ T3 (External Interrupt 6 Input o Timer/Counter3 Clock Input)
PE5	INT5/OC3C (External Int. 5 Input o Output Comp. y PWM Output C para Timer/Cnter3)
PE4	INT4/OC3B (External Int.4 Input o Output Comp. y PWM Output B para Timer/Cnter3)
PE3	AIN1/OC3A (Analog Comp. Neg. Input o Output Comp. y PWM Output A para Timer/Cnter3)
PE2	AIN0/XCK0 (Analog Comparator Positive Input o USART0 external clock input/output)
PE1	PDO/TXD0 (Programming Data Output o UART0 Transmit Pin)
PE0	PDI/RXD0 (Programming Data Input o UART0 Receive Pin)

Tabla 3.3 ***Función alternativa de los pines del puerto E***

- **Puerto F (PF7-PF0)**

El puerto F presenta las mismas características que los puertos anteriores para las líneas de entrada/salida digitales. La funcionalidad alternativa de los pines de este puerto es como líneas de entrada para el conversor analógico digital o señales del interfaz de depuración JTAG. Se recomienda durante la conversión a digital de la entrada que ningún pin de este puerto configurado como salida cambie de valor pues puede perturbar la medida. La descripción de cada pin se detalla en la Tabla 3.4.

Pin	Función alternativa
PF7	ADC7/TDI (ADC channel 7 o JTAG Test Data Input)
PF6	ADC6/TDO (ADC channel 6 o JTAG Test Data Output)
PF5	ADC5/TMS (ADC channel 5 o JTAG Test Mode Select)
PF4	ADC4/TCK (ADC input channel 4 o JTAG Test ClocK)
PF3	ADC3 (ADC input channel 3)
PF2	ADC2 (ADC input channel 2)
PF1	ADC1 (ADC input channel 1)
PF0	ADC0 (ADC input channel 0)

Tabla 3.4 ***Función alternativa de los pines del puerto F***

- **Puerto G (PG4-PG0)**

El puerto G presenta las mismas características que los puertos anteriores para las líneas de entrada/salida digitales. Presenta dos funciones alternativas, por un lado, incorpora a las líneas de control de acceso a la memoria externa y por otro lado, dos de sus pines sirven para conectar un cristal oscilador para aplicaciones donde se necesite RTC. Para más detalle consultar la Tabla 3.5.

Pin	Función alternativa
PG4	TOSC1 (RTC Oscillator Timer/Counter0)
PG3	TOSC2 (RTC Oscillator Timer/Counter0)
PG2	ALE (Address Latch Enable to external memory)
PG1	RD (Read strobe to external memory)
PG0	WR (Write strobe to external memory)

Tabla 3.5 ***Función alternativa de los pines del puerto G***

- **VCC**

  Voltaje de alimentación.

- **GND**

  Tierra.

- **RESET**

Entrada de reset. Una señal por nivel bajo durante más de 50 ns generará un reset, incluso si el reloj no está funcionando. Pequeños pulsos no garantizan la generación de un reset.

- **XTAL1**

  Entrada del oscilador y entrada al circuito que opera con el reloj interno.

- **XTAL2**

  Salida del oscilador.

- **AVCC**

  Tensión de alimentación para el puerto F y el conversor analógico digital. Debe ser conectado externamente a VCC, incluso si no se usa el ADC. En caso de requerir el ADC, se debe conectar a VCC a través de un filtro paso bajo.

- **AREF**

  Referencia de tensión analógica para el ADC.

- **PEN (Pogramming ENable)**

  Habilitación de la programación a través del SPI. Este pin debe estar a nivel bajo durante un reset para iniciarla programación SPI.

3.3.3.3 Arquitectura

Para maximizar el rendimiento, paralelismo y ejecución la familia AVR emplea una arquitectura Harvard con memorias y buses distintos para programa y datos. Las instrucciones de la memoria de programa son ejecutadas con un nivel simple de entrelazado o solapamiento (pipelining). Mientras una instrucción está siendo ejecutada, la siguiente está siendo tomada de la memoria de programa y preparada para su ejecución. Este concepto permite que se ejecute una instrucción cada ciclo de reloj pese a que cada instrucción necesita de dos ciclos de reloj como se aprecia en la Figura 3.10.

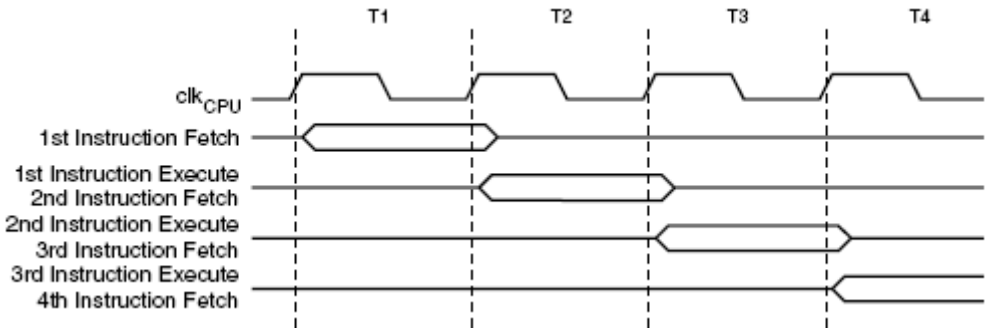


Figura 3.10 ***Ciclos extrae – ejecuta***

El ATmega128 posee 32 registros de trabajo de propósito general con un tiempo de acceso de un ciclo de reloj. Esto significa que en tan sólo un ciclo de reloj, una operación de la ALU (unidad aritmético lógica) es ejecutada. Seis de los 32 registros de trabajo pueden usarse como tres registros de 16 bits que sirven como punteros de direccionamiento indirectos para direccionar el espacio de datos. Estos registros con funciones especiales son el registro X, el Y y el Z.

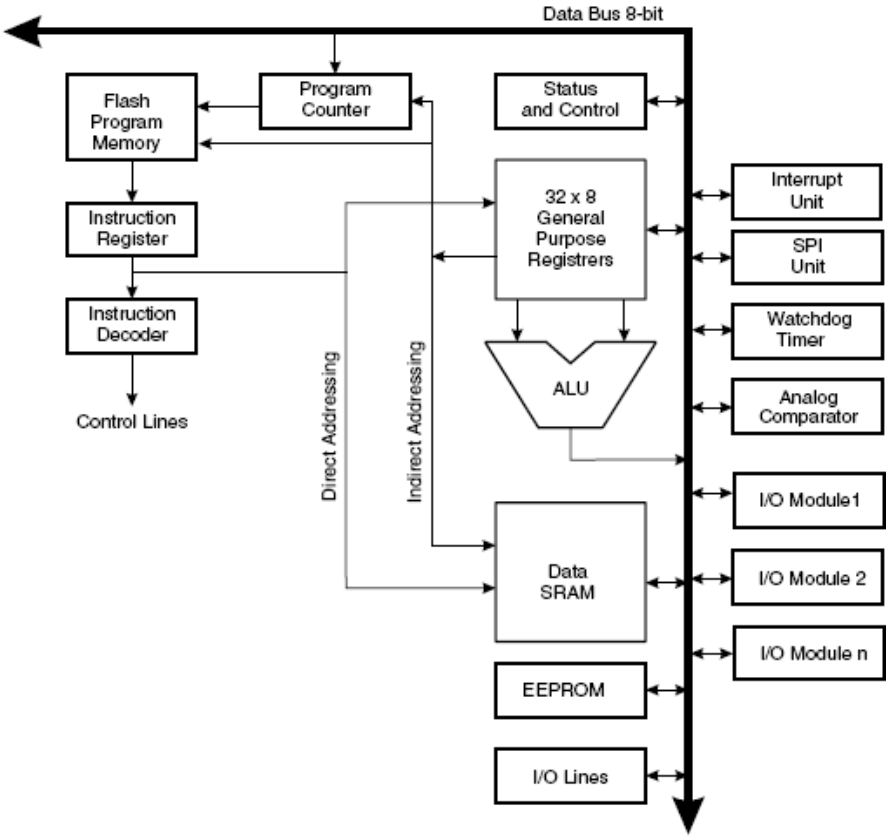


Figura 3.11 ***Arquitectura interna del ATmega128***

La ALU soporta operaciones aritméticas y lógicas entre registros o entre un registro y una constante. Las operaciones simples con registros son también ejecutadas por la ALU, en la Figura 3.11 puede apreciarse un esquema detallado de la estructura interna de un microcontrolador de la familia AVR. Los registros de trabajo están colocados en las 32 primeras posiciones de la memoria de datos ($00 -$1F) como se indica en la Figura 3.12, permitiendo su acceso como si fueran lugares de memoria

convencionales. El espacio de memoria contiene 64 direcciones para las funciones propias de la CPU como son los registros de Control, los Timer/Contadores, conversores A/D, y otras funciones. Estos registros están colocados a continuación de los de trabajo, de la dirección $20 a la $5F. Particularmente, el ATmega128 tiene un espacio extendido de entradas/salidas de $60 a $FF.

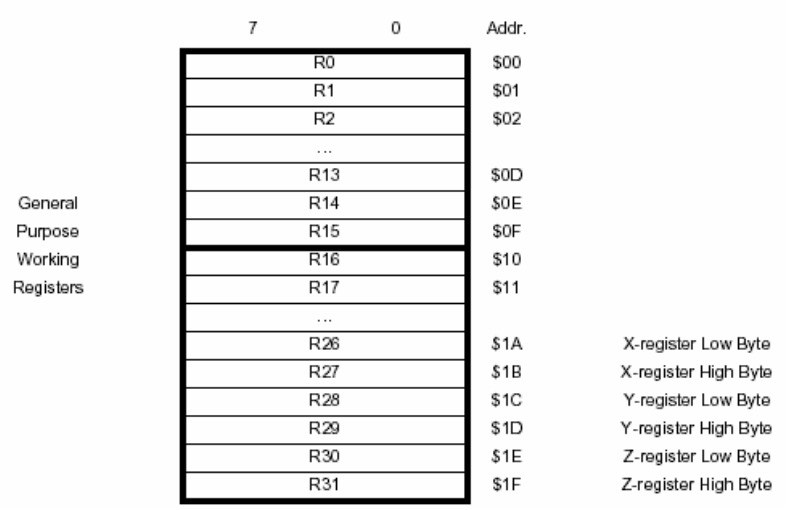


Figura 3.12 ***Registros de propósito general de una CPU AVR***

Con las instrucciones de salto incondicional y con la de llamada, se puede acceder directamente a toda la memoria. Las instrucciones son de 16 o 32 bits, por este motivo, los 128 Kbytes de memoria Flash están estructurados en 64 Kwords de 2 bytes, dividida en memoria de programa de aplicación y memoria de arranque “boot”.

Se aprecia en la Figura 3.13 que existen dos posibles mapas de memoria, según si se va a configurar el ATmega128 en modo normal o en modo compatibilidad con el modelo ATmega163 (modelo anterior de menores prestaciones). La diferencia es que en el modo compatibilidad se anulan ciertos periféricos que no están presentes en el ATmega163 por lo que desaparece el espacio extendido de direcciones E/S.

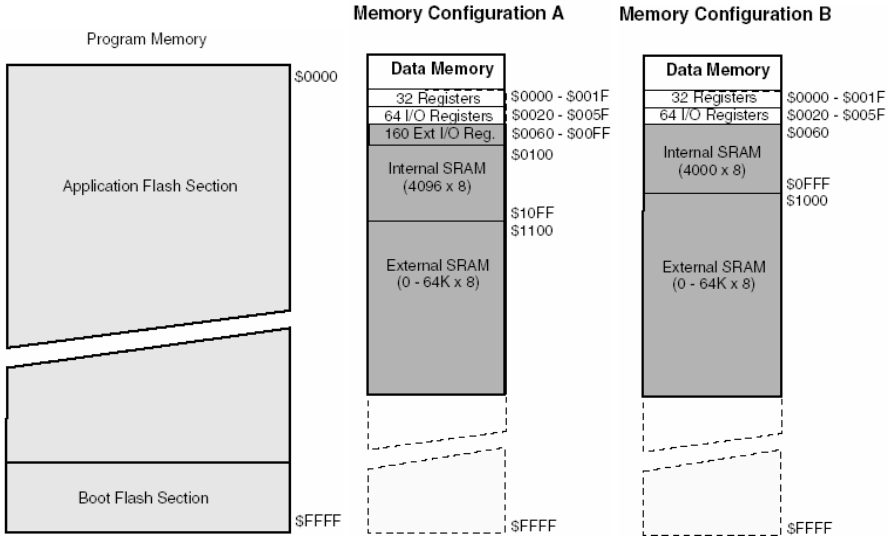


Figura 3.13 ***Mapas de memoria de programa y datos***

3.3.3.4 Registro de estado

El registro de estado (Figura 3.14) almacena información concerniente a las últimas operaciones realizadas. Esta información se emplea para alterar el flujo del programa. La información de este registro se modifica cada vez que termina alguna operación en la unidad aritmético lógica.

Bit	7	6	5	4	3	2	1	0	
	I	T	H	S	V	N	Z	C	SREG
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Figura 3.14 ***Registro de estado SREG***

Todos los bits de este registro son de escritura o lectura. El valor inicial de este registro es 0x00. La descripción de los bits es la siguiente:

- **Bit7-I: Global Interrupt Enable**

El bit 7 permite habilitar las interrupciones cuando está a 1. El control individual de las interrupciones es configurado en registros de control separados. Si el bit 7 está a 0 todas las interrupciones están deshabilitadas independientemente del estado de los registros separados anteriormente citados. Este bit es puesto a 0 vía hardware en el momento en el que se produce una interrupción y es puesto de nuevo a 1 tras la instrucción RETI que sirve para producir el retorno de la interrupción.

- **Bit 6-T: Bit Copy Storage**

Las instrucciones para copiar bits BLD (Bit LoaD) y BST (Bit Store) utilizan el bit T como fuente o destino para u bit. Un bit de un registro del archivo de registros puede ser guardado en el bit T mediante la instrucción BST, igualmente, un bit en T puede ser copiado en un registro del archivo de registros mediante una instrucción BLD.

- **Bit 5-H: Half carry flag:**

Indica que se ha producido un half carry en alguna operación aritmética.

- **Bit 4-S: Sign Bit:**

El bit-S realiza un XOR entre los bits N y V del registro de estado.

- **Bit 3-V: Two's Complement Overflow flag**

Se pone a uno al producirse overflow al realizar alguna operación aritmética de CA2.

- **Bit 2-N: Negative flag**

Indica un resultado negativo tras una operación aritmética o lógica.

- **Bit 1-Z: Zero flag**

Indica un resultado de cero tras una operación aritmética o lógica.

- **Bit 0-C: Carry flag**

Indica que se ha producido acarreo tras una operación aritmética o lógica.

3.3.3.5 Puntero de pila

El Stack Pointer (puntero de la pila) de la tecnología AVR es de 16 bits y está construido sobre dos registros de 8 bits. Dado que el microcontrolador ATmega128 soporta 64 Kbytes de memoria externa SRAM, los 16 bits son utilizados y se detallan en la Figura 3.15

Bit	15	14	13	12	11	10	9	8	
	SP15	SP14	SP13	SP12	SP11	SP10	SP9	SP8	SPH
	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	SPL
	7	6	5	4	3	2	1	0	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

Figura 3.15 ***Puntero de Pila (Stack Pointer)***

Este puntero debe señalar posiciones de memoria SRAM por encima de 0x60. La pila crece hacia direcciones de memoria baja conforme se introducen datos en esta.

3.3.3.6 Puertos de entrada/salida

Cada puerto del microcontrolador tiene una serie de pines que se pueden configurar como E/S digitales. Para configurar cada pin del puerto es necesario utilizar tres registros (PORTx, DDRx y PINx) donde x es A, B, C, D, E, F y G según corresponde a cada puerto, en la Tabla 3.6, puede verse un resumen del uso de estos registros.

Mediante el registro DDRx puede configurarse cada pin del puerto x individualmente como entrada o como salida. Para configurar un pin como salida hay que escribir un cero y viceversa para hacerlo como entrada. Del mismo modo, todos los pines del puerto tienen resistencias de pull-up seleccionables individualmente (consultar la Figura 3.16 para más detalle). Cuando un pin está configurado como entrada y su valor en el registro PORTx es uno, la resistencia de pull-up queda activada. Para desconectar la resistencia de pull-up debe ponerse un cero en el bit correspondiente del registro PORTx o configurarse el pin como salida.

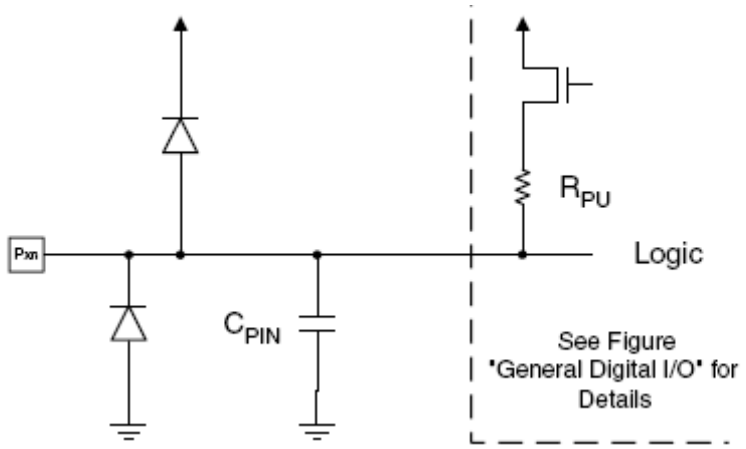


Figura 3.16 ***Esquema de un pin del ATmega128***

DDxn	PORTxn	PUD (en FIOR)	E/S	Pull-up	Comentarios
0	0	X	Entrada	No	Tri-estado (Hi-Z)
0	1	0	Entrada	Yes	
0	1	1	Entrada	No	Tri-estado (Hi-Z)
1	0	X	Salida	No	Salida nivel bajo
1	1	X	Salida	No	Salida nivel alto

Tabla 3.6 ***Configuración de pines del ATmega128***

En el caso del puerto A, sin perdida de generalidad, se tiene que cuando el puerto queda configurado como entrada, el valor de la entrada es guardado en el registro PINA. Del mismo modo, cuando es configurado como salida y se quiere enviar un dato, éste debe ser puesto en el puerto PORTA y no en el PINA que se utiliza sólo para las entradas. En la Tabla 3.7 se pueden observar los bits de los tres registros asociados al puerto A. Recordar por último, que cada pin de los puertos E/S tiene una función alternativa que se puede consultar en la sección Descripción de pines, página 47.

Bit	7	6	5	4	3	2	1	0	
	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0	PORTA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit	7	6	5	4	3	2	1	0	
	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0	DDRA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit	7	6	5	4	3	2	1	0	
	PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0	PINA
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

Tabla 3.7 ***Registros de configuración del puerto A***

3.3.3.7 USARTs

El microcontrolador ATmega128 dispone de dos unidades USART (Universal Synchronous and Asynchronous serial Receiver and Transmitter) integradas independientes que pueden trabajar como puerto serie flexible.

Las principales características de estos dispositivos se detallan a continuación, pudiéndose ver en la Figura 3.17 un esquema completo de una USART:

- Funcionamiento Full Duplex, permite transmisión y recepción independientes y simultáneas.
- Funcionamiento síncrono o asíncrono.
- Modo maestro o esclavo en caso de funcionamiento síncrono.
- Generador de tasas de transmisión de alta resolución.
- Soporta tramas serie de 5, 6, 7, 8 ó 9 bits y 1 ó 2 bits de stop.
- Generador de paridad par o impar y chequeo de paridad hardware.
- Detección de pérdida de datos.
- Detección de error de entramado.
- Filtrado paso bajo digital y detección de bit de start incorrecto.
- Tres interrupciones distintas en caso de transmisión finalizada, registro de datos de transmisor vacío y recepción finalizada.
- Modo de comunicación multiprocesador.
- Doble velocidad en modo de comunicación asíncrona.

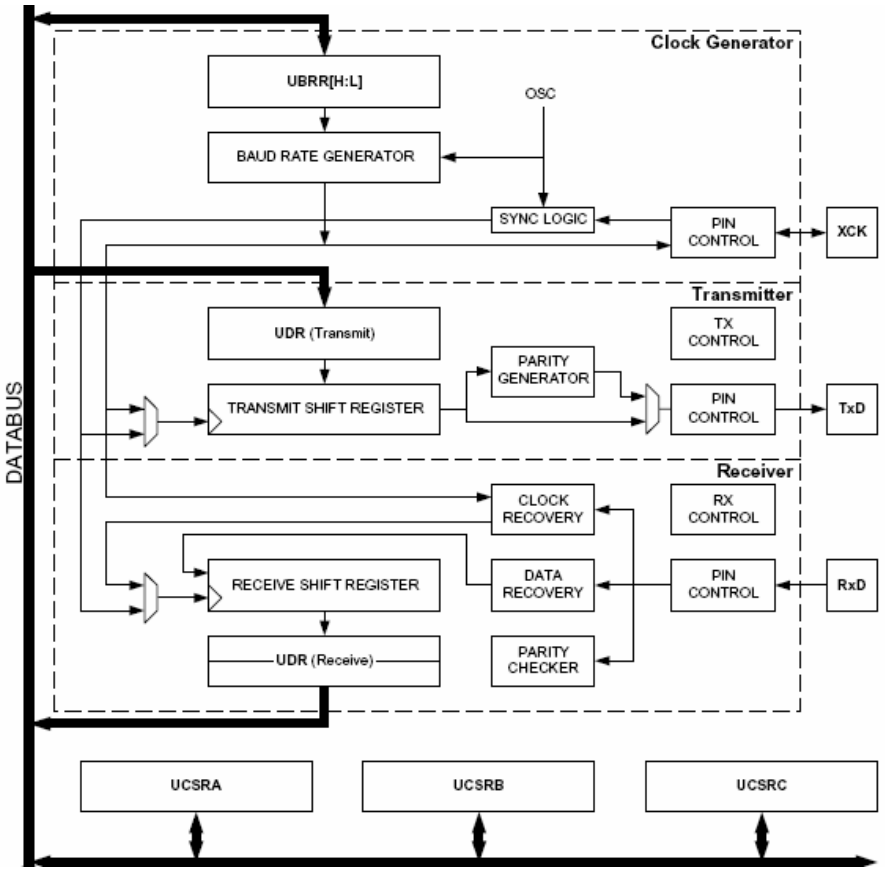


Figura 3.17 ***Diagrama de bloques de la USART***

El registro de tasa de transmisión de la USART (UBRR) y el contador decremental conectado a él funcionan como un preescalador programable. El contador, que corre a la velocidad del reloj del sistema fosc, se precarga con el valor de UBRR cada vez que el contador llega a cero o cuando se escribe en el registro UBRRL. Un reloj se genera a partir de estos pasos por cero, este reloj es el generador de la tasa de transmisión y equivale a $\frac{fosc}{UBRR+1}$. El transmisor divide la velocidad de este reloj por 16 en modo normal asíncrono, que es el que no ocupa, siendo utilizada sin dividir por el receptor para recuperar los datos, sin embargo, utiliza una máquina de estados que consume 16 estados.

El rango de funcionamiento del receptor depende de la similitud entre el reloj que le llega y el internamente generado. Si el transmisor envía muy rápido o muy lento o la velocidad del reloj que le llega presenta una elevada desviación respecto a la nominal, el receptor no será capaz de sincronizarse con los bits de inicio de trama. En las hojas de catálogo de Atmel, aparecen expresiones que se emplean para calcular la relación entre la tasa de datos entrante y la internamente generada. Conocidas estas expresiones, Atmel nos remite a un grupo de tablas para conocer cual debería ser el máximo error porcentual tolerado que, para el caso que nos ocupa, es decir, modo asíncrono a velocidad normal y 8 bits de datos sin paridad es de ± 2.0%.

Atmel suministra tablas con los cálculos para obtener las tasas de datos típicas a partir de las frecuencias más comunes de cristales y resonadores del mercado. A simple vista se aprecia que es preferible utilizar relojes con una frecuencia que no sea múltiplo

de 1 MHz pues el error en estos casos es mucho mayor, sin embargo, puesto que las comunicaciones que vamos a establecer no requerirán de altas tasas de transmisión y utilizaremos el estándar de 9,600 baudios probablemente para dialogar con el módem o 14,400 bps a lo sumo. La comunicación con el PLC se realizará a 19,200 bps por ser esta la velocidad que suele utilizarse por defecto para la programación de este dispositivo y de esto modo evitamos cambiar continuamente la configuración del interfaz PC/PPI que se mostró en la Figura 2.1, pag. 16. Podemos comprobar que los errores producidos a estas velocidades son despreciables (en torno al 0.2% para un reloj de 8 MHz) frente al máximo previsto del 2%. Se muestran a continuación las tablas que contienen las frecuencias más útiles en nuestro caso que son las de 8 MHz en la Tabla 3.8.

Baud Rate (bps)	f_{osc} = 8.0000 MHz				f_{osc} = 11.0592 MHz				f_{osc} = 14.7456 MHz			
	U2X = 0		U2X = 1		U2X = 0		U2X = 1		U2X = 0		U2X = 1	
	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error
2400	207	0.2%	416	-0.1%	287	0.0%	575	0.0%	383	0.0%	767	0.0%
4800	103	0.2%	207	0.2%	143	0.0%	287	0.0%	191	0.0%	383	0.0%
9600	51	0.2%	103	0.2%	71	0.0%	143	0.0%	95	0.0%	191	0.0%
14.4k	34	-0.8%	68	0.6%	47	0.0%	95	0.0%	63	0.0%	127	0.0%
19.2k	25	0.2%	51	0.2%	35	0.0%	71	0.0%	47	0.0%	95	0.0%
28.8k	16	2.1%	34	-0.8%	23	0.0%	47	0.0%	31	0.0%	63	0.0%
38.4k	12	0.2%	25	0.2%	17	0.0%	35	0.0%	23	0.0%	47	0.0%
57.6k	8	-3.5%	16	2.1%	11	0.0%	23	0.0%	15	0.0%	31	0.0%
76.8k	6	-7.0%	12	0.2%	8	0.0%	17	0.0%	11	0.0%	23	0.0%
115.2k	3	8.5%	8	-3.5%	5	0.0%	11	0.0%	7	0.0%	15	0.0%
230.4k	1	8.5%	3	8.5%	2	0.0%	5	0.0%	3	0.0%	7	0.0%
250k	1	0.0%	3	0.0%	2	-7.8%	5	-7.8%	3	-7.8%	6	5.3%
0.5M	0	0.0%	1	0.0%	–	–	2	-7.8%	1	-7.8%	3	-7.8%
1M	–	–	0	0.0%	–	–	–	–	0	-7.8%	1	-7.8%
Max (1)	0.5 Mbps		1 Mbps		691.2 kbps		1.3824 Mbps		921.6 kbps		1.8432 Mbps	

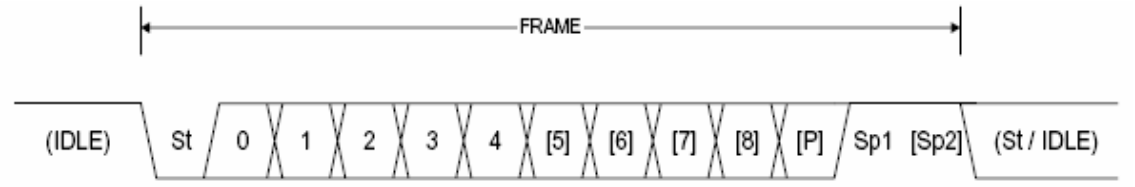
Tabla 3.8 ***Tablas de configuración del registro UBRR***

Formatos de trama

Una trama serie se caracteriza por tener un carácter de datos con bits de sincronización (de start y stop) y un bit de paridad opcional para la detección de errores. La USART acepta las 30 combinaciones que pueden hacerse con los datos que siguen:

- Un bit de start.
- 5, 6, 7, 8 ó 9 bits de datos.
- Paridad par o impar o sin paridad.
- 1 ó 2 bits de stop.

Una trama comienza con el bit de start seguido por el bit de datos menos significativo y todos los demás hasta un máximo de nueve finalizando con el MSB. Si se habilitó, el bit de paridad se inserta tras los bits de datos y antes de los bits de stop. Cuando una trama completa se transmite puede ser seguida directamente por una nueva trama o la línea puede dejarse en estado de reposo (nivel alto) como se aprecia en la Figura 3.18.



St	Start bit, always low.
(n)	Data bits (0 to 8).
P	Parity bit. Can be odd or even.
Sp	Stop bit, always high.
IDLE	No transfers on the communication line (RxD or TxD). An IDLE line must be high.

Figura 3.18 ***Estructura de una trama serie***

El formato de trama utilizado por la USART se configura con los bits UCSZ2:0, UPM1:0 y USBS en los registro UCSRB y UCSRC. El receptor y el transmisor han de usar la misma configuración pues en caso contrarió ambos fallarán.

El USART Character SiZe (UCSZ2:0) selecciona el numero de bits de datos de la trama y el USART Parity mode (UPM1:0) habilita y establece el tipo de paridad. La selección entre uno o dos bits de stop se hace con el USART Stop Bit Select (USBS), aunque el receptor ignora el segundo bit de stop, por lo que un error de trama sólo podrá ser detectado si el primer bit de stop es cero.

Registros asociados al puerto serie

UDR – USART I/O Data Register

Este registro permite introducir datos para que sean transmitidos y extraer los caracteres recibidos. Actúa como interfaz pues físicamente existen búferes separados para transmisión y recepción (TXB y RXB), que comparten la misma dirección en el mapa de memoria del microcontrolador a la cual se accede desde el UDR. Es un registro de 16 bits como se ve en la Tabla 3.9, ocupando RXB la palabra alta y TXB la baja.

Bit	7	6	5	4	3	2	1	0	
	RXB[7:0]								UDR (Read)
	TXB[7:0]								UDR (Write)
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Tabla 3.9 ***Registro de datos de la USART***

UCSRA – USART Control & Status Register A

Este registro contiene flags que indican el estado del dispositivo y bits de configuración para el control. A continuación se muestra la Tabla 3.10 como diagrama del registro para posteriormente indicar más pormenorizadamente la función de cada bit.

Bit	7	6	5	4	3	2	1	0	
	RXC	TXC	UDRE	FE	DOR	UPE	U2X	MPCM	UCSRA
Read/Write	R	R/W	R	R	R	R	R/W	R/W	
Initial Value	0	0	1	0	0	0	0	0	

Tabla 3.10 ***Registro de control y estado A***

Los siguientes bites indican el estado del dispositivo en funcionamiento normal sin errores, reportando información en cuanto a si se ha recibido un nuevo carácter, se ha terminado de transmitir y si esta vacío el registro de datos.

- **Bit 7 – RXCn: USART Receive Complete**
- **Bit 6 – TXCn: USART Transmit Complete**
- **Bit 5 – UDREn: USART Data Register Empty**

Los bits 4, 3 y 2 avisan de posibles errores. El bit 4 avisa de error de entramado, el bit 3 de que se ha perdido un dato porque se ha sobrescrito otro carácter entrante y el bit 2 informa de error de paridad si esta característica ha sido habilitada por el bit UPM.

- **Bit 4 – FEn: Frame Error**
- **Bit 3 – DORn: Data OverRun**
- **Bit 2 – UPEn: Parity Error**

El bit 1 selecciona el número de muestras empleado en recepción para el procesamiento de los caracteres recibidos.

- **Bit 1 – U2Xn: Double the USART Transmissión Speed**

El bit 0 es para habilitar el modo de comunicación multiprocesador.

- **Bit 0 – MPCMn: Multi-Processor Communication Mode**

UCSRB – USART Control & Status Register B

Bit	7	6	5	4	3	2	1	0	
	RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8	UCSRB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Tabla 3.11 ***Registro de control y estado B***

Los bits 7, 6 y 5 son los bits de habilitación de las posibles interrupciones que puede generar la USART por los eventos recepción completa, transmisión completa y registro de datos vacío. Todo ello supeditado a que el bit de habilitación global de interrupciones en el registro SREG este habilitado.

- **Bit 7 – RXCIEn: RX Complete Interrupt Enable**
- **Bit 6 – TXCIEn: TX Complete Interrupt Enable**

- **Bit 5 – UDRIEn: USART Data Register Interrupt Enable**

Los bits 4 y 3 habilitan las funciones de recepción y transmisión de la USART.

- **Bit 4 – RXEn: Receiver Enable**
- **Bit 3 – TXEn: Transmitter Enable**

El bit 2 se emplea en conjunción con UCSZn1:0 del registro UCRSC para definir el número de bits que componen el carácter.

- **Bit 2 – UCSZn2: Character Size**

Los bits 1 y 0 son una extensión del registro UDR cuando se emplean caracteres de 9 bits.

- **Bit 1 – RXB8n: Receive Data Bit 8**
- **Bit 0 – TXB8n: Transmit Data Bit 8**

UCSRC – USART Control & Status Register C

Bit	7	6	5	4	3	2	1	0	
	–	UMSEL	UPM1	UPM0	USBS	UCSZ1	UCSZ0	UCPOL	UCSRC
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	1	1	0	

Tabla 3.12 ***Registro de configuración y estado C***

El bit 6 selecciona el modo de trabajo de la USART, distinguiendo entre modo síncrono (valor 1) o asíncrono (valor 0).

- **Bit 6 – UMSELn: USART Mode SELect**

Los bits 5 y 4 configuran si se emplea paridad y de que tipo. El transmisor genera automáticamente la paridad y el receptor la calcula y compara según el valor de estos bits según se indica en la Tabla 3.13.

- **Bit 5:4 – UPMn1:0: Parity Mode**

UPMn1	UPMn0	Paridad
0	0	Deshabilitado
0	1	Reservado
1	0	Paridad par
1	1	Paridad impar

Tabla 3.13 ***Configuración de los bits UPM***

El bit 3, **USBSn** (Stop Bit Select) indica si se usan uno o dos bits de parada, cero para un bit y uno para dos bits.

Los bits 2 y 1 definen junto con el bit UCSZ2 del UCSRB el número de bits que conforman el carácter según la Tabla 3.14.

- **Bit 2:1 – UCSZn1:0: Character Size**

UCSZn2	UCSZn1	UCSZn0	Tamaño
0	0	0	5-bit
0	0	1	6-bit
0	1	0	7-bit
0	1	1	8-bit
1	0	0	Reservado
1	0	1	Reservado
1	1	0	Reservado
1	1	1	9-bit

Tabla 3.14 ***Bits de configuración de longitud de trama***

El bit 0 sólo se utiliza en el modo síncrono para indicar, cuando se produce el cambio en la entrada y salida respecto al reloj.

UBRR – USART Baud Rate Register

Como ya se indicó, en este registro se escribe el valor que se necesite para obtener la tasa de datos deseada. Esta formado por 2 bytes UBRRH y UBRRL pero sólo se emplean los 12 bits menos significativos como se desprende de la Tabla 3.15.

Bit	15	14	13	12	11	10	9	8	
	–	–	–	–	UBRR[11:8]				UBRRH
	UBRR[7:0]								UBRRL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

Tabla 3.15 ***Registro de configuración de la tasa binaria***

Cabe indicar que ya se mostró en la Tabla 3.8 de la página 59 que los valores recomendados por el Atmel para un reloj de 8 MHz es 51 respectivamente para una velocidad de 9,600 baudios y 25 para el mismo reloj y una velocidad de 19,200 baudios.

3.3.3.8 Interrupciones

El microcontrolador ATmega128 posee 34 fuentes de interrupción más reset. Todas las interrupciones tienen un vector de interrupción propio en el espacio de memoria del programa y de unos bits que debe ser puestos a uno conjuntamente con el bit I (bit 7) del registro de estado para que las interrupciones puedan producirse.

La ejecución de la respuesta a una interrupción es un proceso que tiene una duración mínima de 4 ciclos de reloj. Después de estos 4 ciclos de reloj, el programa ejecuta el código de programa correspondiente a la interrupción producida. Durante este período de 4 ciclos de reloj el Contador de Programa (2 bytes) se coloca en la pila, y el Stack Pointer se decrementa en dos unidades. Tras producirse una interrupción la instrucción que se ejecutará en el vector correspondiente será una de salto RJMP a la rutina de código correspondiente y ésta es una instrucción que tarda dos ciclos de reloj. Si una interrupción es producida mientras se está ejecutando una interrupción multiciclo se esperará a que la instrucción finalice su ejecución antes de atender a la interrupción.

El retorno de una rutina de interrupción dura 4 ciclos de reloj. Durante estos 4 ciclos de reloj el Contador de Programa (2 bytes) se carga con su valor de la pila y el Stack Pointer se incrementado en dos unidades. Cuando el microcontrolador sale de la ejecución del código de interrupción, vuelve al programa principal y ejecuta exactamente la instrucción en la que se encontraba antes de que se produjera la interrupción.

Las direcciones más bajas del espacio de memoria de programa son automáticamente definidas para el vector de Reset y para los vectores de interrupción (a no ser que se trabaje en modo bootload). La lista completa de vectores de interrupción se muestra en la Tabla 3.16. Esta lista también determina los niveles de prioridad de las diferentes interrupciones. La mayor prioridad la tiene el vector de RESET y a continuación el vector INT0 y así sucesivamente.

Vector	Dirección	Fuente	Definición
1	$0000	RESET	External Pin, Power-on Reset
2	$0002	INT0	External Interrupt Request 0
3	$0004	INT1	External Interrupt Request 1
4	$0006	INT2	External Interrupt Request 2
5	$0008	INT3	External Interrupt Request 3
6	$000A	INT4	External Interrupt Request 4
7	$000C	INT5	External Interrupt Request 5
8	$000E	INT6	External Interrupt Request 6
9	$0010	INT7	External Interrupt Request 7
10	$0012	TIMER2 COMP	Timer/Counter2 Compare Match
11	$0014	TIMER2 OVF	Timer/Counter2 Overflow
12	$0016	TIMER1 CAPT	Timer/Counter1 Capture Event
13	$0018	TIMER1 COMPA	Timer/Counter1 Compare Match A
14	$001A	TIMER1 COMPB	Timer/Counter1 Compare Match B
15	$001C	TIMER1 OVF	Timer/Counter1 Overflow
16	$001E	TIMER0 COMP	Timer/Counter0 Compare Match
17	$0020	TIMER0 OVF	Timer/Counter0 Overflow
18	$0022	SPI, STC	SPI Serial Transfer Complete
19	$0024	USART0, RX	USART0, Rx Complete
20	$0026	USART0, UDRE	USART0 Data Register Empty
21	$0028	USART0, TX	USART0, Tx Complete
22	$002A	ADC	ADC Conversion Complete
23	$002C	EE READY	EEPROM Ready
24	$002E	ANALOG COMP	Analog Comparator

25	$0030	TIMER1 COMPC	Timer/Counter1 Compare Match C
26	$0032	TIMER3 CAPT	Timer/Counter3 Capture Event
27	$0034	TIMER3 COMPA	Timer/Counter3 Compare Match A
28	$0036	TIMER3 COMPB	Timer/Counter3 Compare Match B
29	$0038	TIMER3 COMPC	Timer/Counter3 Compare Match C
30	$003A	TIMER3 OVF	Timer/Counter3 Overflow
31	$003C	USART1, RX	USART1, RX complete
32	$003E	USART1, UDRE	USART1 Data Register Empty
33	$0040	USART1, TX	USART1, TX complete
34	$0042	TWI	Two-wire Serial Interface
35	$0044	SPM READY	Store Program Memory Ready

Tabla 3.16 ***Vectores de interrupción del ATmega128***

3.3.3.9 Temporizadores / Contadores

Para el caso que nos ocupa, la gestión de los temporizadores es transparente gracias al compilador de C elegido, si bien, se realizará una breve descripción teórica para conocer su funcionamiento dado que en el código escrito para el microcontrolador, se utilizan abundantes periodos de espera que en realidad se implementan a bajo nivel gestionando temporizadores.

El ATmega128 presenta cuatro Temporizadores/Contadores, dos de 8 bits y dos de 16 bits. Cada temporizador tiene asignado un preescalador individual y pueden ser usados como temporizadores según la base de tiempos del oscilador interno o como contadores sincronizándose con eventos externos. A continuación se detallan las características de ambos.

Temporizador / Contador de 8 bits

El Timer/Counter 0 de 8 bits puede seleccionar su base de tiempos del reloj interno, del reloj interno con preescalador o de un cristal RTC conectado a los pines TOC1 y TOC2 (Timer 0) o de la entrada T2 (Timer 1). Además puede ser activado o detenido y configurado según el registro de control Timer/Counter TCCR. El flag que señaliza el desbordamiento del Timer se encuentra en el registro TIFR. Existe la posibilidad de que se produzca una interrupción cuando este Timer se desborda configurandola, en este caso, en el registro TIMSK.

El Timer0 funciona como un contador ascendente/descendente que va desde el valor 0x00 hasta el valor 0xff. Cuando se pasa del valor 0xff al 0x00, o viceversa, es cuando se habla de que el Timer se ha desbordado (overflow). En el registro TCNT0 de Escritura/Lectura es en el que se puede introducir el valor inicial con el que se quiere que el Temporizador realice su cuenta. Así se puede configurar fácilmente para obtener interrupciones periódicas (llamadas Timer Overflow) que sean la base de un reloj o el aviso para realizar determinadas operaciones.

Los registros empleados en la configuración del Timer 0 son:

TCCR0 – Timer Counter Control Register 0

Bit	7	6	5	4	3	2	1	0	
	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00	TCCR0
Read/Write	W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Tabla 3.17 ***Registro de configuración de cuenta del timer 0***

- **Bit 7 – FOC0: Force Output Compare**

Este bit puesto a uno fuerza un evento de tipo Output compare en las condiciones impuestas por el resto de bits de configuración.

- **Bit 6, 3 – WGM01:0: Waveform generation mode**

Controlan el modo de operación del temporizador, consultar la Tabla 3.18.

Modo	WGM21 (CTC2)	WGM20 (PWM2)	Timer/Counter Modo de Operación	TOP	Actualización de OCR2	TOV2 Flag Set
0	0	0	Normal	0xFF	Immediata	MAX
1	0	1	PWM, Fase Correcta	0xFF	TOP	BOTTOM
2	1	0	CTC	OCR2	Immediata	MAX
3	1	1	PWM Rápido	0xFF	TOP	MAX

Tabla 3.18 ***Modo de trabajo del Timer***

- **Bit 5:4 – COM01:0: Compare match Output Mode**

Controlan el comportamiento del módulo output compare.

- **Bit 2:0 – CS02:0: Clock Select**

Estos tres bits cuya configuración se aprecia en la Tabla 3.19, seleccionan la fuente de reloj que actúa sobre el contador a partir del reloj del sistema. La otra posible fuente de reloj es a partir del cristal RTC en los pines TOSC1 y TOSC2 que se configura con el registro ASSR.

CS22	CS21	CS20	Descripción
0	0	0	Sin fuente de reloj (Timer/Counter stopped)
0	0	1	$clk_{I/O}$/(No prescaling)
0	1	0	$clk_{I/O}$/8 (Del prescaler)
0	1	1	$clk_{I/O}$/64 (Del prescaler)
1	0	0	$clk_{I/O}$/256 (Del prescaler)
1	0	1	$clk_{I/O}$/1024 (Del prescaler)
1	1	0	Fuente Externa en pin T2 pin. Flanco de bajada
1	1	1	Fuente Externa en pin T2 pin. Flanco de subida

Tabla 3.19 **Selección del reloj del Timer**

TIMSK – Timer Interrupt MaSK register

Este registro, detallado en la Tabla 3.20, contiene los bits que habilitan las distintas fuentes de interrupciones asociadas a los temporizadores/contadores. Los bits asociados al temporizador 0 son los bits 0 y 1 que se usa para habilitar la interrupción asociada al evento Output Capture y Timer Overflow.

Bit	7	6	5	4	3	2	1	0	
	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0	TIMSK
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Tabla 3.20 ***Registro de habilitación de interrupciones asociadas al Timer***

TIFR – Timer Interrupt Flag Register

Este registro, detallado en la Tabla 3.21, presenta los "flags" asociados al estado de los temporizadores / contadores. Los bits que informan del estado del temporizador 0 son el bit 1 y el 0, que indica si se ha producido un Output Capture o un Timer Overflow.

Bit	7	6	5	4	3	2	1	0	
	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0	TIFR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Tabla 3.21 ***Registro de banderas asociadas al Timer***

TCNT0 – Timer Counter 0

El temporizador/contador 0 cuenta hacia delante, con acceso de lectura y escritura. Si se escribe un valor sobre él y existe una fuente de reloj, continúa contando con la frecuencia del reloj a partir del valor escrito (se reinicia al nuevo valor).

Bit	7	6	5	4	3	2	1	0	
	TCNT0[7:0]								TCNT0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Tabla 3.22 ***Contador asociado al Timer 0***

Temporizador / Contador de 16 bits

El microcontrolador ATmega128 dispone de dos temporizadores/contadores de 16 bits y cada uno de ellos puede seleccionar con un preescalador independiente. Su funcionamiento puede ser habilitado o interrumpido mediante la configuración de los registros de control asociados: TCCRA, TCCRB y TCCRC. Presenta varios modos de funcionamiento. Además de su uso natural como temporizador/contador, posee un registro de captura de entrada de 16 bits usado para el modo Input Capture (para medir, por ejemplo, anchos de pulsos o capturar tiempos). También tiene dos registros de

comparación de salida de 16 bits que se usan para el modo Output Compare (para sacar, por ejemplo, frecuencias o pulsos por un pin del microcontrolador).

3.4 Compilador de C para AVR

El compilador utilizado es el Codevision AVR, también conocido como CVAVR o simplemente Codevision. Es un sistema completamente desarrollado para las series de AVR, disponible en dos versiones: la versión completa que genera código para todas las series de AVR Classic y Mega, y la versión "ligera" que solo genera código para la serie de AVR classic. Hay una versión gratuita de evaluación, descargable desde el sitio web del fabricante, aunque limita el tamaño del programa que compila a 2 KB.

El entorno de desarrollo incluye un interesantísimo generador automático de código llamado Codewizard AVR (en la Figura 3.19 se muestran las ventanas más interesantes, particularmente destacable es el hecho de mostrar la tasa de error estimada de la USART para la configuración elegida, mostrando un aviso cuando el error esta fuera del margen aconsejado por Atmel), que genera todo el código necesario para la inicialización y configuración de los periféricos internos del microcontrolador utilizando bibliotecas incluidas.

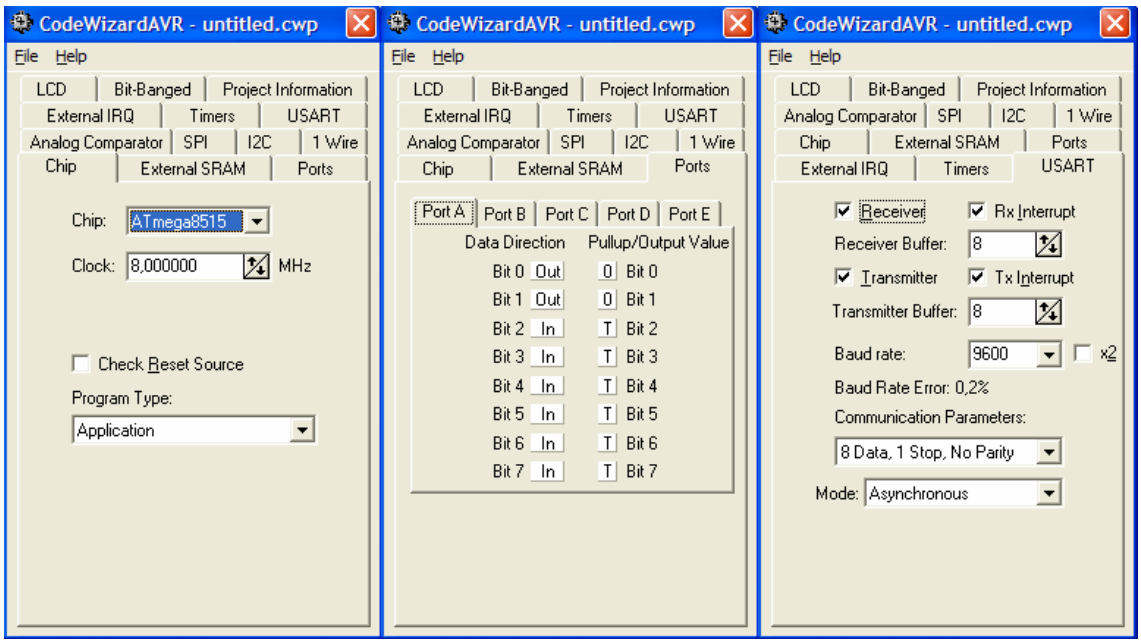


Figura 3.19 ***Ventanas de configuración de Codewizard AVR***

Las ventajas de la utilización de un compilador de C de este tipo saltan a la vista si se piensa en el ingente trabajo que supone configurar "a mano" todos los registros del microcontrolador, labor que el generador de código obvia simplemente configurando ventanas como las anteriores. De este modo también se evita la gestión de las interrupciones manualmente, pues a través de las ventanas del configurador podemos generar todo el código necesario para generar rutinas de interrupción asociadas a todos los acontecimientos sin tener que recurrir a la Tabla 3.16 de la página 65.

Además, trabajamos con un lenguaje de programación mucho más conocido y fácil de depurar que el lenguaje ensamblador propio de este dispositivo. Sirva a modo de ejemplo la declaración de una variable de tipo entera dentro de la EEPROM que en el caso de lenguajes de bajo nivel requiere además de configurar registros, ordenar al microcontrolador generar sobretensiones internas durante tiempos muy concretos, en C basta declarar "EEPROM int x". También contamos con numerosas funciones de librerías ofrecidas por el fabricante, piénsese de nuevo que la ardua tarea de lanzar un temporizador de "n" ms se solventa con la invocación "delay_ms(n)" utilizando la librería adecuada.

Para programar el microcontrolador objetivo, es necesario indicar al programa el tipo de programador de que disponemos (STK500 en nuestro caso) y podemos indicarle que cargue automáticamente todo el código generado después de cada compilación en el microcontrolador o bien desde la ventana del programador que se muestra en la Figura 3.20, indicar desde la pestaña Program si se desea programar sólo la memoria Flash, la EEPROM, etc. Además permite leer el código hexadecimal almacenado en la memoria del microcontrolador, proteger el código para que no pueda ser leído por terceros o chequear el estado del microcontrolador y del programador utilizado para descartar posibles fallos debidos a daños físicos en alguno de los anteriores. También se permite desde esta ventana programar los llamados Fuse bits, los cuales configuran en gran medida el comportamiento del microcontrolador, describen entre otros, la fuente de reloj a utilizar, modo de carga de los programas, gestión de la memoria EEPROM, o activan el modo especial de compatibilidad con microcontrolador más antiguos.

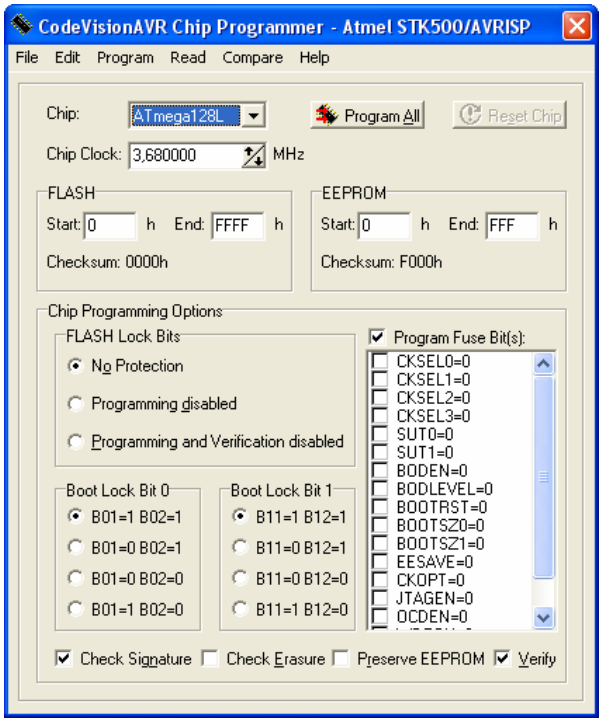


Figura 3.20 ***Ventana de programación de CodeVisionAVR***