TECNICATURA SUPERIOR EN TELECOMUNICACIONES

ELECTRÓNICA MICROCONTROLADA

Docentes: Ing. Jorge E. Morales, Téc. Gonzalo Vera.

**Título:** Estructura del Microcontrolador AVR® de 8 bits

**Grupo 1:**

❖ Birge, Adolfo Federico.

❖ Carunchio, Carlos Javier.

❖ Ferreyra, María Luciana.

❖ Gutiérrez, Emma Vilma.

❖ Merlo, Emmanuel.

❖ Romero, Gisela de Lourdes.

Estructura del Microcontrolador AVR® de 8 bits

Núcleo de CPU AVR

Tiempo de instrucción AVR

Unidad lógica aritmética AVR

Registros de propósito general AVR

Registro de pila AVR

Registro de estado de AVR

Memoria AVR

Optimización del código C en AVR

Memoria de lectura mientras se escribe AVR (gestor de arranque)

Puertos de E/S digitales AVR

Modos de suspensión de bajo consumo AVR

Registro de reducción de potencia periférico AVR

Fusibles AVR

Interfaces de programación AVR

Temporizador AVR Watchdog

AVR USART Introducción

Temporizador AVR Comparar registro doble búfer

Modos de funcionamiento de AVR ADC

Modo diferencial AVR ADC

Modo de reducción de ruido AVR ADC

Voltaje de referencia AVR ADC

Interfaz periférica serie AVR (SPI)

Controlador táctil periférico AVR (PTC)

Comparador interno AVR

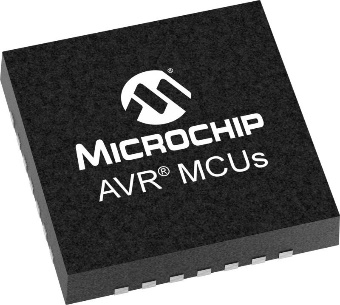
Detección de apagado AVR

Interrupciones AVR

Interrupciones externas AVR

Contador en tiempo real (RTC) de AVR

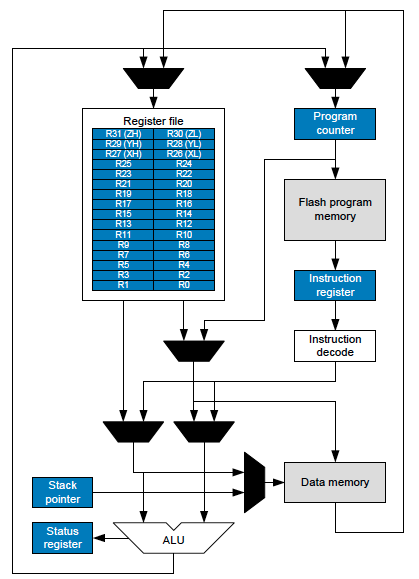
**Estructura del Microcontrolador AVR® de 8 bits**

**Los microcontroladores (MCU) AVR de 8 bits ofrecen una familia de dispositivos únicos y versátiles. Con la gran cantidad de periféricos incorporados, siempre es útil tener capacitación sobre cómo funcionan, cómo configurarlos y ejemplos de cómo usarlos. A continuación, detallaremos su arquitectura:**

**Núcleo de CPU AVR**

La función principal del núcleo de la Unidad Central de Procesamiento (CPU) AVR ® es garantizar la correcta ejecución del programa. Por lo tanto, la CPU debe poder acceder a las memorias, realizar cálculos, controlar periféricos y manejar interrupciones.

Diagrama de bloques de la arquitectura AVR



**Núcleo AVR**

Para maximizar el rendimiento y el paralelismo, el AVR utiliza una arquitectura Harvard con memorias y buses separados para programas y datos. Las instrucciones en la memoria del programa se ejecutan con canalización de un solo nivel. Mientras se ejecuta una instrucción, la siguiente instrucción se obtiene previamente de la memoria del programa. Este concepto permite ejecutar instrucciones en cada ciclo de reloj. La memoria del programa es una memoria Flash reprogramable en el sistema.

**Registros**

El archivo de registro de acceso rápido contiene 32 registros de trabajo de propósito general de 8 bits con un solo tiempo de acceso de ciclo de reloj. Seis de los 32 registros se pueden utilizar como tres punteros de registro de direcciones indirectas de 16 bits para el direccionamiento del espacio de datos, lo que permite cálculos de direcciones eficientes. Uno de estos punteros de dirección también se puede utilizar como puntero de dirección para tablas de búsqueda en la memoria de programa Flash. Estos registros de funciones adicionales son los registros X, Y y Z de 16 bits.

**Unidad Aritmética Lógica (ALU)**

La ALU admite operaciones aritméticas y lógicas entre registros o entre una constante y un registro. El tiempo de acceso de ciclo de reloj único permite operaciones de ALU de ciclo único. En una operación ALU típica, se emiten dos operandos desde el archivo de registro, se ejecuta la operación y el resultado se almacena nuevamente en el archivo de registro en un ciclo de reloj. Las operaciones de registro único también se pueden ejecutar en la ALU. Después de una operación aritmética, el registro de estado se actualiza para reflejar información sobre el resultado de la operación. El flujo del programa lo proporcionan las instrucciones de salto y llamada condicionales e incondicionales, capaces de abordar directamente todo el espacio de direcciones. La mayoría de las instrucciones AVR tienen un solo formato de palabra de 16 bits. Cada dirección de memoria de programa contiene una instrucción de 16 o 32 bits.

**Memoria**

Los espacios de memoria en la arquitectura AVR son todos mapas de memoria lineales y regulares. El espacio de la memoria flash del programa se divide en dos secciones, la sección del programa de arranque y la sección del programa de aplicación. Ambas secciones tienen bits de bloqueo dedicados para protección contra escritura y lectura/escritura. La instrucción Store Program Memory (SPM) que escribe en la sección de memoria Flash de la aplicación debe residir en la sección Boot Program.

Durante las interrupciones y las llamadas a subrutinas, el Contador de programa (PC) de la dirección de retorno se almacena en la pila. La pila se asigna efectivamente en la SRAM de datos generales y, en consecuencia, el tamaño de la pila solo está limitado por el tamaño total de la SRAM y el uso de la SRAM.

Todos los programas de usuario deben inicializar el puntero de pila (SP) en la rutina de reinicio (antes de que se ejecuten las subrutinas o interrupciones). El SP es accesible para lectura/escritura en el espacio de E/S. Se puede acceder fácilmente a la SRAM de datos a través de los cinco modos de direccionamiento diferentes admitidos en la arquitectura AVR.

El espacio de memoria de E/S contiene 64 direcciones para funciones periféricas de la CPU como registros de control, interfaz periférica en serie (SPI) y otras funciones de E/S. Se puede acceder a la memoria de E/S directamente o como ubicaciones de espacio de datos que siguen a las del archivo de registro, 0x20 - 0x5F. Además, este dispositivo ha ampliado el espacio de E/S de 0x60 - 0xFF en SRAM.

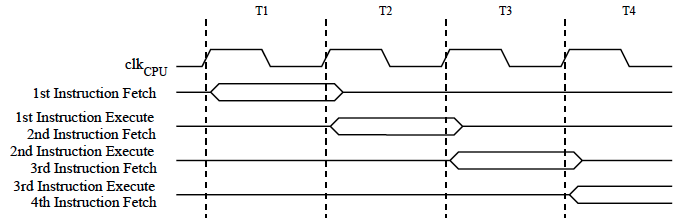
**Interrupciones**

Un módulo de interrupción flexible tiene sus registros de control en el espacio de E/S con un bit de habilitación de interrupción global adicional en el registro de estado. Todas las interrupciones tienen un vector de interrupción separado en la tabla de vectores de interrupción. Las interrupciones tienen prioridad de acuerdo con su posición en el vector de interrupción. Cuanto menor sea la dirección del vector de interrupción, mayor será la prioridad.

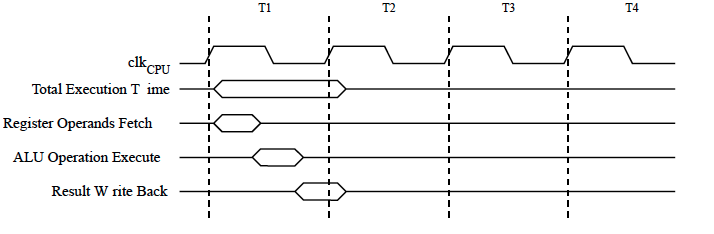
**Tiempo de instrucción AVR**

La unidad central de procesamiento (CPU) AVR es impulsada por el reloj de la CPU, generado directamente desde la fuente de reloj seleccionada para el chip. No se utiliza ninguna división de reloj interna. La arquitectura de Harvard y el concepto de archivo de registro de acceso rápido permiten obtener y ejecutar instrucciones en paralelo. Este es el concepto básico de canalización para obtener hasta 1 MIPS por MHz con los resultados únicos correspondientes para funciones por costo, funciones por relojes y funciones por unidad de potencia.

**Obtención de instrucciones en paralelo y ejecución de instrucciones**

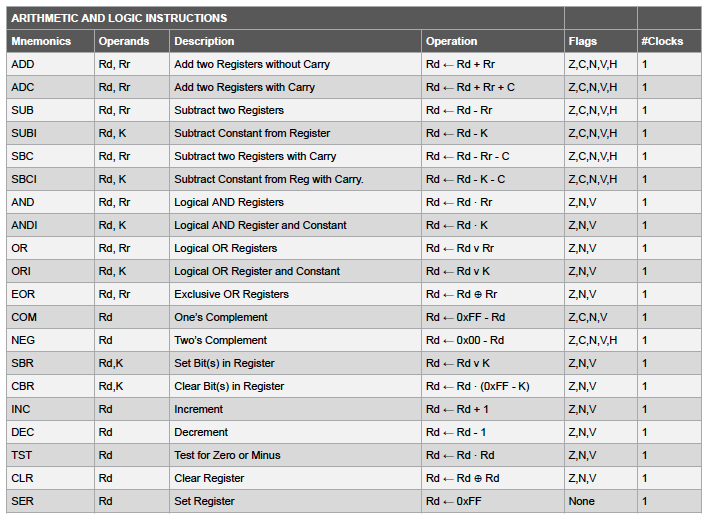
 En un solo ciclo de reloj, se ejecuta una operación de unidad lógica aritmética (ALU) utilizando dos operandos de registro y el resultado se almacena nuevamente en el registro de destino.

**Operación ALU de ciclo único**

****

**Unidad lógica aritmética AVR**

La Unidad Aritmética Lógica (ALU) AVR® de alto rendimiento opera en conexión directa con los 32 registros de trabajo de propósito general. Dentro de un solo ciclo de reloj, se ejecutan operaciones aritméticas entre registros de Propósito General o entre un registro y un inmediato. Las operaciones ALU se dividen en tres categorías principales: funciones aritméticas, lógicas y de bits. Algunas implementaciones de la arquitectura también proporcionan un poderoso multiplicador que admite tanto la multiplicación con signo/sin signo como el formato fraccionario.



**Registros de propósito general AVR**

La estructura del archivo de registro AVR® está optimizada para el conjunto de instrucciones de la computadora del conjunto de instrucciones reducido mejorado (RISC) del AVR. Para lograr el rendimiento y la flexibilidad requeridos, el archivo de registro admite los siguientes esquemas de E/S:

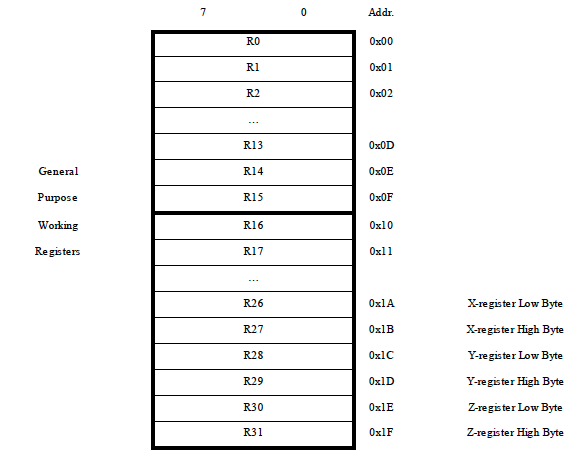
Un operando de salida de 8 bits y una entrada de resultado de 8 bits.

Dos operandos de salida de 8 bits y una entrada de resultado de 8 bits.

Dos operandos de salida de 8 bits y una entrada de resultado de 16 bits.

Un operando de salida de 16 bits y una entrada de resultado de 16 bits.

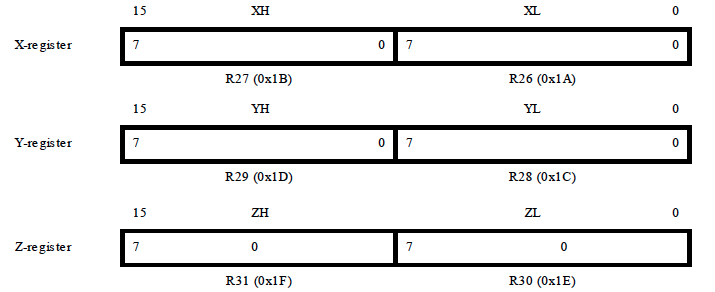
**Conjunto de instrucciones ALU**

Registros de trabajo de propósito general de la CPU AVR

La mayoría de las instrucciones que operan en el archivo de registro tienen acceso directo a todos los registros y la mayoría de ellas son instrucciones de un solo ciclo. A cada registro también se le asigna una dirección de memoria de datos, asignándolos directamente a las primeras 32 ubicaciones del espacio de datos del usuario. Aunque no se implementa físicamente como ubicaciones SRAM, esta organización de memoria proporciona una gran flexibilidad en el acceso a los registros, ya que los registros de puntero X, Y y Z se pueden configurar para indexar cualquier registro en el archivo.

El registro X, el registro Y y el registro Z

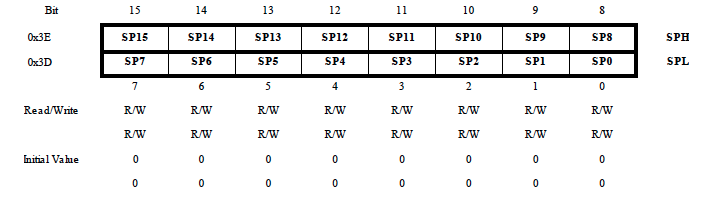
Los registros R26 a R31 tienen algunas funciones adicionales a su uso de propósito general. Estos registros son punteros de dirección de 16 bits para el direccionamiento indirecto del espacio de datos. Los tres registros de direcciones indirectas (X, Y y Z) se definen como se describe en la figura.



**Registro de pila AVR**

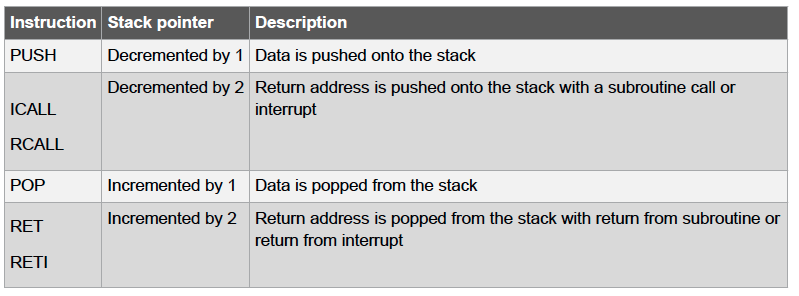
La pila se utiliza principalmente para almacenar datos temporales, variables locales y direcciones de retorno después de interrupciones y llamadas a subrutinas. Se implementa como un crecimiento de ubicaciones de memoria superiores a inferiores. El registro del puntero de pila siempre apunta a la parte superior de la pila; apunta al área de la pila SRAM de datos donde se encuentran las pilas de subrutinas e interrupciones.

Puntero de pila



El AVR Stack Pointer se implementa como dos registros de 8 bits en el espacio de E/S. El número de bits realmente utilizados depende de la implementación.

Conjunto de instrucciones de pila

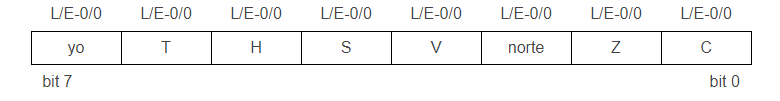


Un comando PUSH de pila disminuirá el puntero de pila. El programa debe definir la pila en la SRAM de datos antes de que se ejecuten las llamadas a subrutinas o se habiliten las interrupciones. El valor inicial del Puntero de pila es igual a la última dirección de la SRAM interna y el Puntero de pila debe configurarse para que apunte por encima del inicio de la SRAM.

**Registro de estado de AVR**

El registro de estado contiene información sobre el resultado de la última instrucción aritmética ejecutada. Esta información se puede utilizar para alterar el flujo del programa con el fin de realizar operaciones condicionales. El registro de estado se actualiza después de todas las operaciones de la unidad lógica aritmética (ALU). En muchos casos, esto eliminará la necesidad de usar las instrucciones de comparación dedicadas, lo que dará como resultado un código más rápido y compacto.

ESTADO: Registro de estado



**bit 7**

I: Habilitación de interrupción global: el bit de habilitación de interrupción global debe establecerse para que se habiliten las interrupciones. El control de habilitación de interrupción individual se realiza entonces en registros de control separados. Si se borra el registro de Habilitación de interrupción global, ninguna de las interrupciones se habilita independientemente de la configuración de habilitación de interrupción individual. El bit I (bit 7) se borra por hardware después de que se ha producido una interrupción y se establece mediante la instrucción RETI (Return from Interrupt) para habilitar las interrupciones subsiguientes. La aplicación también puede establecer y borrar el bit I con las instrucciones Set Global Interrupt Flag (SEI) y Clear Global Interrupt Flag (CLI), como se describe en la referencia del conjunto de instrucciones.

**bit 6**

T: almacenamiento de copia: las instrucciones de copia de bits, carga de bits (BLD) y almacenamiento de bits (BST), utilizan el bit T como fuente o destino para el bit operado. La instrucción BST puede copiar un bit de un registro en el archivo de registro en T y la instrucción BLD puede copiar un bit en T en un bit en un registro en el archivo de registro.

**bit 5**

H: Indicador de medio acarreo: El indicador de medio acarreo, H, indica medio acarreo en algunas operaciones aritméticas. Es útil en la aritmética decimal de código binario (BCD).

**bit 4**

S: Bandera de signo, S = N xor V: El bit S es siempre un exclusivo o entre la Bandera negativa y la Bandera de desbordamiento del complemento a dos.

**bit 3**

V: Indicador de desbordamiento de complemento a dos: El indicador de desbordamiento de complemento a dos, V, es compatible con la aritmética de complemento a dos.

**bit 2**

N: Bandera Negativa: La Bandera Negativa, N, indica un resultado negativo en una operación aritmética o lógica.

**bit 1**

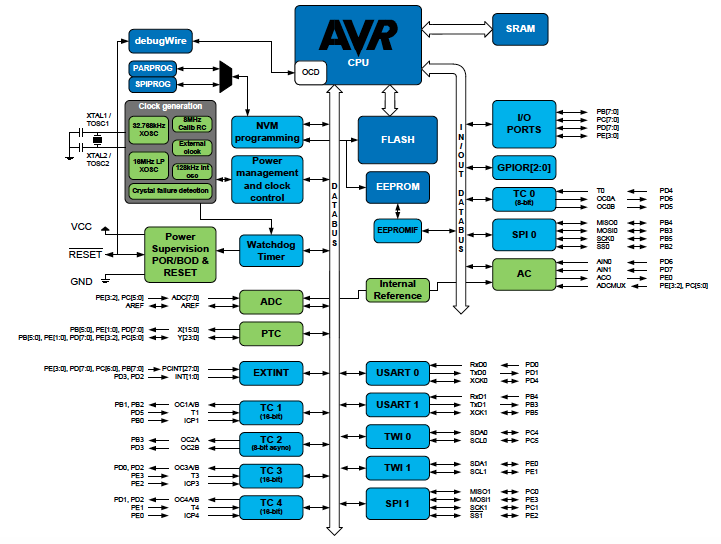
Z: Indicador cero: El indicador cero, Z, indica un resultado cero en una operación aritmética o lógica.

**bit 0**

C: Indicador de acarreo: El indicador de acarreo, C, indica un acarreo en una operación aritmética o lógica.

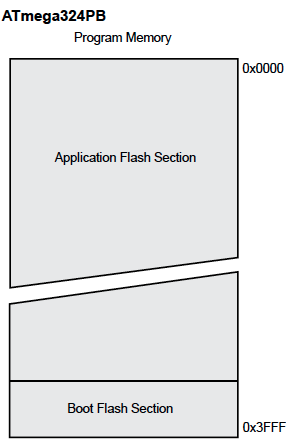
**Memoria AVR**

Los microcontroladores AVR se construyen utilizando una arquitectura Harvard modificada. Esto significa que el espacio de la memoria de programa flash está en un bus de direcciones separado de la memoria estática de acceso aleatorio (SRAM). Hay dos buses de datos, uno que puede acceder a todos los datos y el bus de datos de entrada/salida con acceso limitado a una pequeña sección de memoria.



Memoria de programa

Los microcontroladores AVR contienen memoria flash reprogramable en el sistema en chip para el almacenamiento de programas. Dado que todas las instrucciones AVR tienen 16 o 32 bits de ancho, el Flash está organizado como 32K x 16. Para la seguridad del software, el espacio de la memoria del programa Flash se divide en dos secciones: la sección del cargador de arranque y la sección del programa de aplicación en el dispositivo. La memoria Flash tiene una resistencia típica de al menos 10.000 ciclos de escritura/borrado. Las tablas constantes se pueden asignar dentro de todo el espacio de direcciones de la memoria del programa, utilizando la instrucción Cargar memoria del programa (LPM). También hay una biblioteca de funciones para hacer esto más fácil Biblioteca AVR Libc.



Memoria de datos EEPROM

La memoria de solo lectura programable borrable eléctricamente (EEPROM) de datos se organiza como un espacio de datos separado, en el que se pueden leer y escribir bytes individuales. El acceso desde la CPU a la EEPROM se realiza a través de los registros de direcciones de la EEPROM, el registro de datos de la EEPROM y el registro de control de la EEPROM. La EEPROM tiene una resistencia de al menos 100.000 ciclos de escritura/borrado.

Memoria de datos SRAM

Se puede acceder a los datos a través del bus de datos estándar. Hay un bus de entrada/salida secundario para acceso directo rápido a ubicaciones selectas. La memoria de datos consta de:

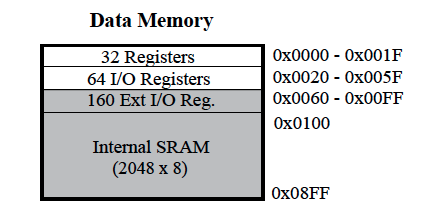
* Registros
* Memoria de E/S
* Memoria de E/S extendida (depende del dispositivo)
* SRAM interna

**Espacio de registro:** consta de 32 registros de trabajo de 8 bits de uso general (R0-R31).

**Memoria de E/S:** contiene espacio direccionable para funciones periféricas, como registros de control y otras funciones de E/S.

**Memoria de E/S extendida:** algunos microcontroladores AVR con más periféricos necesitan más espacio del que puede ocupar la memoria de E/S, por lo que parte de la SRAM se usa como memoria de E/S extendida para manejar los registros de control de periféricos adicionales y otras funciones de E/S.

**SRAM interna (memoria de datos):** se utiliza para almacenar temporalmente variables y resultados intermedios dentro de una aplicación de software.



Hay cinco modos de direccionamiento de bus de datos diferentes (no de entrada/salida) para la memoria de datos:

**Directo:** el direccionamiento directo alcanza todo el espacio de datos.

**Indirecto:** en el archivo de registro, los registros R26 a R31 presentan los registros de puntero de direccionamiento indirecto.

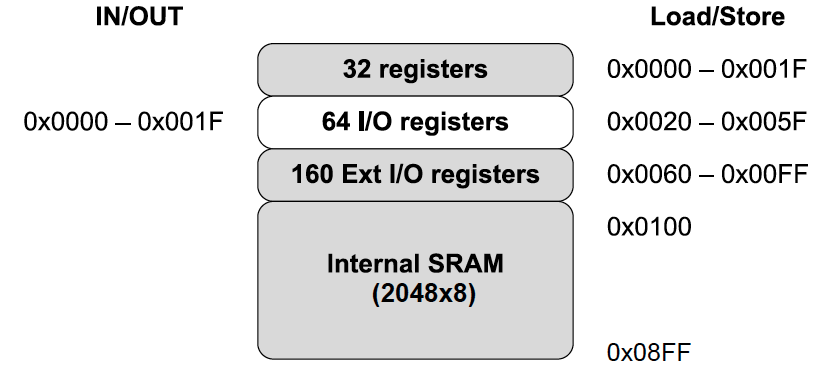
**Indirecto con desplazamiento:** el modo Indirecto con desplazamiento llega a 63 ubicaciones de direcciones desde la dirección base proporcionada por el registro Y o Z.

**Indirecto con decremento previo:** los registros de dirección X, Y y Z se reducen.

**Indirecto con posincremento:** los registros de dirección X, Y y Z se incrementan.

Bus de datos de entrada/salida

Este bus de datos tiene acceso directo a la sección de memoria de E/S de 64 bytes (no extendida) mediante una dirección de 0x00 a 0x1F. También se puede acceder a esta memoria mediante el bus de datos estándar utilizando un desplazamiento de dirección 0x20 en el comando de acceso.



Memoria de E/S

Se puede acceder a todas las ubicaciones de E/S (memoria de E/S y memoria de E/S extendida) mediante las instrucciones de ensamblaje LD/LDS/LDD y ST/STS/STD utilizando el bus de datos estándar. Los datos se transfieren entre los 32 registros de trabajo de propósito general y el espacio de E/S.

Los registros de E/S dentro del rango de direcciones del bus de datos de entrada/salida 0x00-0x1F (memoria de E/S) son accesibles directamente mediante bits mediante las instrucciones SBI y CBI. En estos registros, el valor de los bits individuales se puede comprobar mediante las instrucciones SBIS y SBIC.

Registros de E/S de uso general

Tres registros de E/S de uso general, el registro de E/S de uso general 0/1/2 (GPIOR 0/1/2) se encuentran en la parte superior de la memoria de E/S (0x020-0x022). Estos registros se pueden utilizar para almacenar cualquier información y son particularmente útiles para almacenar variables globales y banderas de estado. Se puede acceder directamente a estos registros mediante las instrucciones SBI, CBI, SBIS y SBIC a través del bus de datos de entrada/salida. Los registros de E/S restantes comienzan después de los registros de E/S de propósito general.

Acceso a registros de 16 bits

El bus de datos AVR tiene 8 bits de ancho, por lo que acceder a registros de 16 bits requiere operaciones atómicas. Se debe acceder a estos registros mediante bytes mediante dos operaciones de lectura o escritura. Los registros de 16 bits están conectados al bus de 8 bits y un registro temporal usando un bus de 16 bits.

Para una operación de escritura, el byte alto del registro de 16 bits debe escribirse antes que el byte bajo. A continuación, el byte alto se escribe en el registro temporal. Cuando se escribe el byte bajo del registro de 16 bits, el registro temporal se copia en el byte alto del registro de 16 bits en el mismo ciclo de reloj.

Para una operación de lectura, el byte bajo del registro de 16 bits debe leerse antes que el byte alto. Cuando la CPU lee el registro de byte bajo, el byte alto del registro de 16 bits se copia en el registro temporal en el mismo ciclo de reloj que se lee el byte bajo. Cuando se lee el byte alto, se lee del registro temporal.

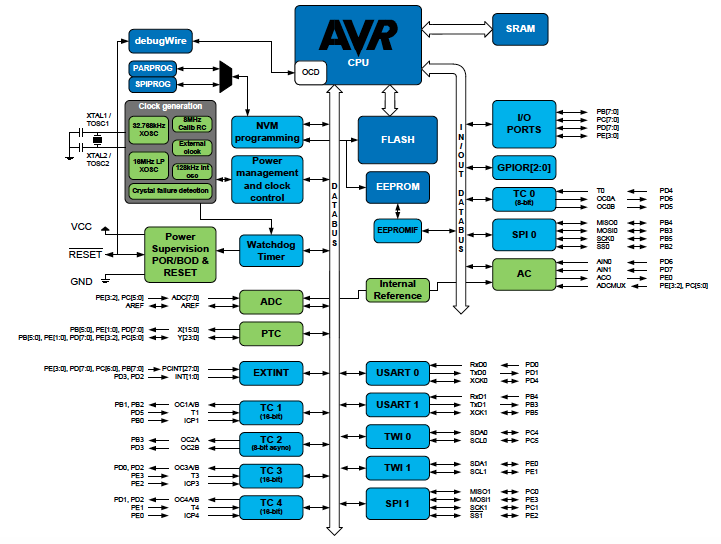
Esto asegura que siempre se acceda simultáneamente a los bytes alto y bajo de los registros de 16 bits al leer o escribir el registro.

**Optimización del código C en AVR**

Antes de optimizar el software de los sistemas integrados, es necesario comprender bien cómo está estructurado el núcleo del microcontrolador (MCU) AVR ® y qué estrategias utiliza AVR GNU Compiler Collection (GCC) para generar código eficiente para el procesador.

Arquitectura Atmel AVR de 8 bits

AVR utiliza la arquitectura Harvard, con memorias y buses separados para programas y datos. Tiene un archivo de registro de acceso rápido de 32 x 8 registros de trabajo de propósito general con un solo tiempo de acceso de ciclo de reloj. Los 32 registros de trabajo son una de las claves para una codificación C eficiente. Estos registros tienen la misma función que el acumulador tradicional, excepto que hay 32 de ellos. Las instrucciones aritméticas y lógicas del AVR funcionan en estos registros, por lo que ocupan menos espacio de instrucción. En un ciclo de reloj, AVR puede enviar dos registros arbitrarios desde el archivo de registro a la ALU, realizar una operación y volver a escribir el resultado en el archivo de registro.



Las instrucciones en la memoria del programa se ejecutan con una canalización de un solo nivel. Mientras se ejecuta una instrucción, la siguiente instrucción se obtiene previamente de la memoria del programa. Este concepto permite ejecutar instrucciones en cada ciclo de reloj. La mayoría de las instrucciones AVR tienen un solo formato de palabra de 16 bits. Cada dirección de memoria de programa contiene una instrucción de 16 o 32 bits.

**AVR CCG**

AVR GCC proporciona varios niveles de optimización. Son -O0, -O1, -O2, -O3 y -Os. En cada nivel, hay diferentes opciones de optimización habilitadas, excepto -O0 que significa que no hay optimización. Además de las opciones habilitadas en los niveles de optimización, también puede habilitar opciones de optimización separadas para obtener una optimización específica. El manual de la colección de compiladores GNU tiene una lista completa de opciones y niveles de optimización.

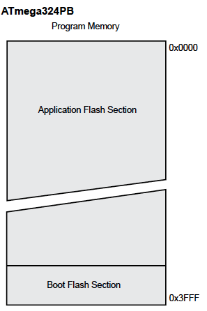
Aparte del AVR GCC, se necesitan muchas otras herramientas trabajando juntas para producir la aplicación ejecutable final para el microcontrolador AVR. El grupo de herramientas se denomina cadena de herramientas. Dentro de la cadena de herramientas AVR, AVR Libc Library, que proporciona muchas de las mismas funciones que se encuentran en una biblioteca C estándar y muchas funciones de biblioteca adicionales específicas para un AVR. Además, la biblioteca proporciona el código de inicio básico que necesitan la mayoría de las aplicaciones. Consulte el Manual de referencia de AVR Libc para obtener más detalles.

**Memoria de lectura mientras se escribe AVR (gestor de arranque)**

En muchos dispositivos AVR ® , la sección del cargador de arranque de la memoria del programa es compatible con la autoprogramación real de lectura mientras escribe. Esta función permite actualizaciones de software de aplicaciones flexibles controladas por la MCU mediante un programa Boot Loader residente en Flash. El programa Boot Loader puede usar cualquier interfaz de datos disponible y protocolo asociado para leer código y escribir (programar) ese código en la memoria Flash o leer el código de la memoria del programa. El código del programa dentro de la sección del cargador de arranque tiene la capacidad de escribir en todo el Flash, incluida la memoria del cargador de arranque. El Boot Loader puede incluso modificarse a sí mismo, y también puede borrarse del código si la función ya no es necesaria.

Memoria flash

La memoria Flash está organizada en dos secciones principales: la sección de la aplicación y la sección del cargador de arranque. El tamaño de las diferentes secciones lo configuran los Fusibles BOOTSZ. Estas dos secciones pueden tener diferentes niveles de protección ya que tienen diferentes conjuntos de bits de bloqueo.

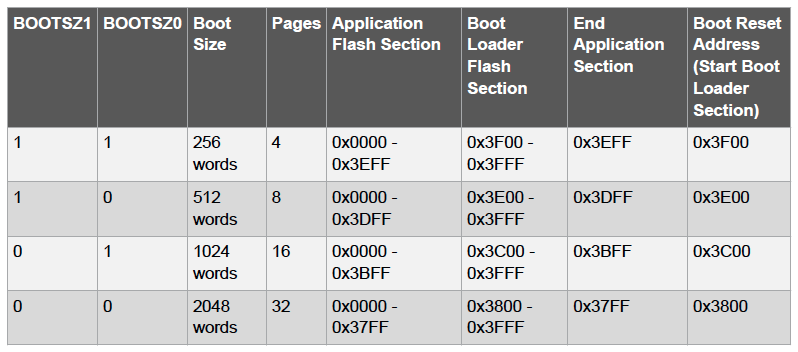


Sección de aplicación

La sección Aplicación es la sección de Flash que se utiliza para almacenar el código de la aplicación. El nivel de protección para la sección Aplicación se puede seleccionar mediante los bits de bloqueo de arranque de la aplicación (bits de bloqueo de arranque 0). La sección de la aplicación nunca puede almacenar ningún código del cargador de arranque, ya que la instrucción Store Program Memory (SPM) está deshabilitada cuando se ejecuta desde la sección de la aplicación.

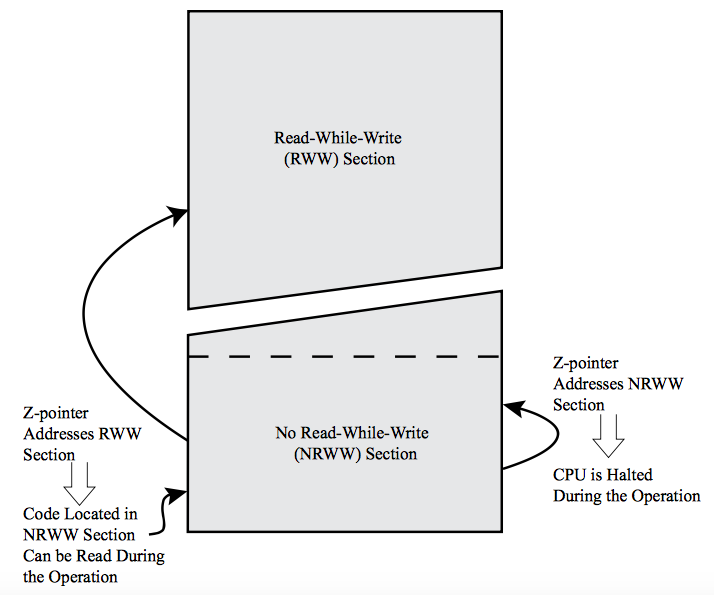
Sección del cargador de arranque (BLS)

Si bien la sección Aplicación se usa para almacenar el código de la aplicación, el software Boot Loader debe estar ubicado en el BLS ya que la instrucción SPM puede iniciar una programación cuando se ejecuta desde el BLS únicamente. La instrucción SPM puede acceder a todo el Flash, incluido el propio BLS. El nivel de protección para la sección del cargador de arranque se puede seleccionar mediante los bits de bloqueo del cargador de arranque. El ejemplo que se muestra es de la hoja de datos AVRmega328PB.



**Secciones flash de lectura mientras escribe (RWW) y sin lectura mientras escribe (NRWW)**

Además de las dos secciones configurables por BOOTSZ Fuses como se describió anteriormente, el flash también se divide en dos secciones fijas, la sección de lectura durante la escritura (RWW) y la sección sin lectura durante la escritura (NRWW).

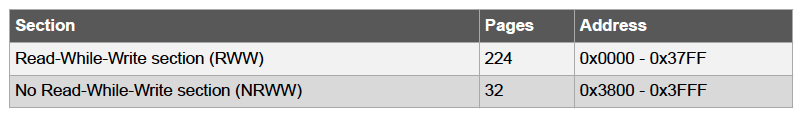


La principal diferencia entre las dos secciones es:

Al borrar o escribir una página ubicada dentro de la sección RWW, la sección NRWW se puede leer durante la operación.

Al borrar o escribir una página ubicada dentro de la sección NRWW, la CPU se detiene durante toda la operación.

El límite entre las secciones RWW y NRWW se define en la ficha técnica del dispositivo. Estos son los ajustes para el AVRmega328PB.



Si el dispositivo AVR es compatible con RWW o si el dispositivo AVR se detiene durante una actualización del software Boot Loader depende de la dirección que se esté programando.

El software del usuario nunca puede leer ningún código ubicado dentro de la sección RWW durante una operación del software Boot Loader. La sintaxis "Sección de lectura mientras se escribe" se refiere a qué sección se está programando (borrando o escribiendo), no a qué sección se está leyendo durante una actualización de software del cargador de arranque.