Universidad de Costa Rica
Facultad de Ingeniería
Escuela de Ingeniería Eléctrica
IE0624 – Laboratorio de Microcontroladores
III ciclo 2022

Reporte 3

Arduino: GPIO,ADC y comunicaciones

Ana Eugenia Sánchez Villalobos B87382
Adrián Montero Bonilla B85092
Grupo 01

Profesor: Marco Villalta

23/01/2023

Índice

Índice de figuras

1. Resumen

En este laboratorio se desarrolla un voltímetro de 4 canales utilizando Arduino UNO, que consiste en recibir 4 tensiones de rangos desde los -24V hasta los 24V en forma AC o DC, la elección se realiza por medio de un switch, y se visualiza los valores en la pantalla LCD PCD8544. En caso de sobrepasar una tensión mayor a los 20V o menor a los -20V se debe de encender una luz LED, que avise como método de precaución. Se utilizan componentes pasivos y se diseña un divisor de tensión con el fin de que el Arduino pueda entender y al mismo tiempo sea seguro para este. A partir de estos datos se realiza una comunicación con una PC, en donde se guardan los datos en un archivo `.csv`, para de esta manera poder graficar y comparar los datos mediante el uso de un archivo `.py`. Estos datos viajan a través del puerto serial con el bloque USART.

2. Nota Teórica

El ATmega328, es un microcontrolador creador por la compañía Atmel, y forma parte de la familia de microcontroladores basadas en AVR. Se compone de una arquitectura Harvard con un procesador de tipo RISC de 8 bits, y se considera como bastante potente, que puede realizar sus instrucciones en un solo ciclo de reloj. Aparte este microcontrolador tiene la ventaja de consumir menos energía debido a que cuenta con la tecnología picopower. En este laboratorio se utilizara el Arduino UNO, esta es una placa electrónica de hardware libre, que contiene al microcontrolador ATmega328P. Las características generales del microcontrolador se pueden observar en la siguiente tabla:

Microcontrolador ATmega328P	
Tipo de programa de memoria	Flash
Tamaño de memoria	32 KB
Data EEPROM (bytes)	1024
Numero de pines	32
Tensión de Operación (min - máx)	1.8V - 5.5V
Velocidad CPU (MIPS)	20
Timers	2 x8bits - 1x 16bits
Rango de temperatura (min - máx)	-40°C -85°C
Canales ADC	8
Maxima resolucion ADC	10 bits
Corriente DC por Vcc y GND	200 mA
Corriente DC por entrada cada PIN I/O	40 mA

Tabla 1: Características generales del ATmega328P

El diagrama de pines del microcontrolador ATmega328P se puede observar en la figura 1.

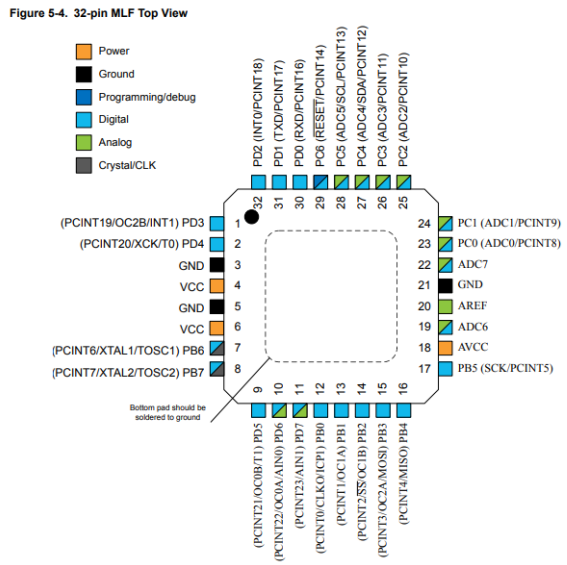


Figura 1: Diagrama de bloques del ATmega328, con 32 pines

Debido a que se va a utilizar los pines del Arduino UNO, se pueden observar en la figura 2 los pines del Arduino UNO.

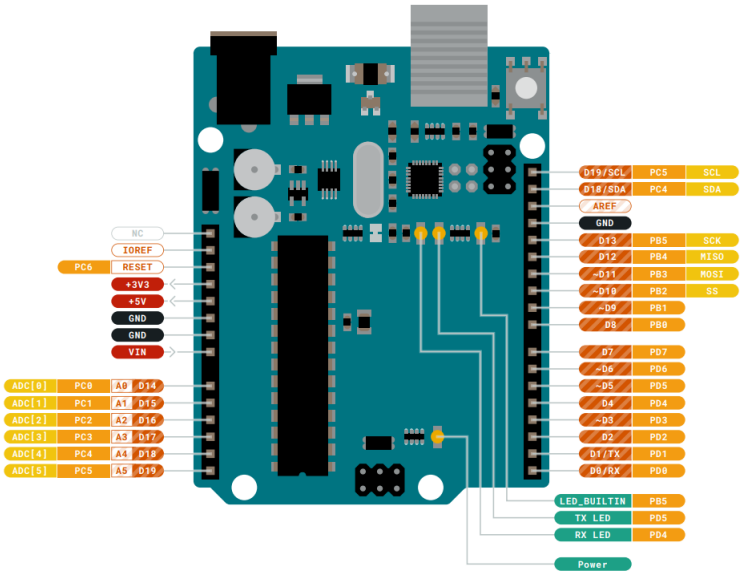


Figura 2: Diagrama de bloques del Arduino UNO

2.1. Periféricos utilizados.

2.1.1. API Arduino

API Arduino (Application Programming Interface Arduino), es el conjunto de instrucciones que facilitan la comunicación entre el programador y el Arduino. Hay distintos tipos de instrucciones que se van a listar a continuación, cada una de estas implica cierto comportamiento en el Arduino, dentro del listado podemos observar su función y su sintaxis:
Entradas y salidas digitales:

- pinMode(numeropin, modo): Esta función recibe el numero de pin el que se quiere configurar, y el modo de como se desea configurar el pin dado, y se le indica si se desea que funcione como OUTPUT o INPUT.

- digitalRead(numeropin): Esta funcion lee el estado de un pin en un momento dado, este puede estar en HIGH o LOW. Depende de la tensión medida podrá asignar un valor u otro. Para el rango de 0V a 1.5V el estado sera LOW, para el rango 3.3V a 5V el estado será HIGH.

- digitalWrite(numeropin, valor): Permite asignar el valor HIGH o LOW al pin que se le pase como parámetro. Siendo 5V el valor para HIGH y 0V el valor para LOW, esto en configuración OUTPUT. Sin embargo existe el caso de que sea configurado como INPUT, al asignar un valor HIGH a la entrada se habilita la resistencia de pull-up del pin y al asignar LOW dejara de funcionar dicha resistencia.

Entradas y salidas analógicas

- analogReference(tipo): Esta función recibe un parámetro el cual sirve para configurar la tensión de referencia al medir un pin analógico.

- analogRead(numeropin): Al igual que la función digitalRead, esta función lee el estado del pin que se le pasa como parámetro, pero no le asigna un valor de alto o bajo, sino que esta lee el valor analógico en un rango de 0 a 1023 dependiendo del valor de tensión que se tenga en la entrada.

Entradas y salidas avanzadas:

- tone(): Genera una onda cuadrada con un ciclo de trabajo del 50 %.

- noTone(): Esta función detiene la función que se genera con tone().

Tiempo:

- millis(): Función que devuelve el tiempo en milisegundos.

- micros(): Función que devuelve en tiempo en microsegundos.

- delay(milisegundos): Detiene la ejecución del programa en milisegundos.

- delayMicroseconds(microsegundos): Detiene la ejecución del programa en microsegundos.

Interrupciones:

- attachInterrupt(interrupción, ISR, modo): Con esta función se indica que función se va a llamar cuando se produce una interrupción por los pines que permiten dicha interrupción. En Arduino UNO los pines digitales 2 y 3 permiten esta funcionalidad. Hay cuatro posibilidades de manejo para las interrupciones: LOW, CHANGE, RISING y FALLING. Cada una de estas tiene una distinta definición, LOW, llama a la interrupción cuando el pin se lee bajo. RISING y FALLING llaman a la interrupción cuando el flanco es positivo o negativo respectivamente. Por el otro lado CHANGE llama a la interrupción cuando el pin cambia de nivel.

- noInterrupts(): Desabilita las interrupciones del sistema.

Comunicación:

- Serial.begin(baud): Se inicio la comunicación serial con dispositivos fuera Arduino.

- Serial.availabe(): Esta función devuelve el número de bytes disponibles para lectura en el puerto serial.

- Serial.read(): Lee los datos en el puerto en serie.
- Serial.write(): Escribe en el puente en serie, datos binarios.
- Serial.prinIn(): Misma función que Serial.print(), a diferencia de que esta incluye un salto de línea.

2.1.2. Conversión Analógica/Digital

Consiste en la conversión de señales analógicas a señales digitales, esto con el propósito de facilitar la codificación y la compresión. Las señales analógicas no son constantes, debido a que provienen de una fuente de información, esto es un efecto no deseado, el cual hace muy difícil trabajar con ellas, por esto se debe transformar en una señal cuantizada. Por otro lado las señales digitales, al ser codificadas cuentan con sistemas y algoritmos de corrección de errores, esto con el fin de preservar el mensaje original y eliminar el ruido de la señal. [2] La conversión analógica a digital, consiste en distintos pasos que son los siguientes:

1. Muestreo: Lee la señal analógica y la procesa obteniendo pequeñas muestras de esta.
2. Retención: Guarda las muestras en un tiempo estimado para poder realizar una evaluación de errores.
3. Cuantificación: Mide la tensión obtenida en cada una de las muestra, y procede a decidir si tomarlo como ceros o unos lógicos.
4. Codificación: Asigna una secuencia de bits, para los símbolos disponibles.

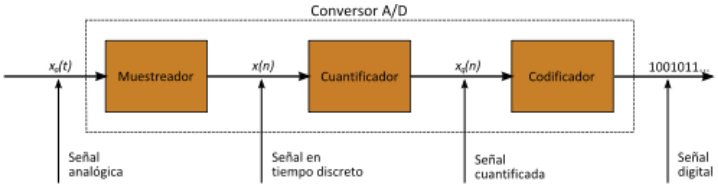


Figura 3: Diagrama de conversor de señal analógica a digital

2.2. Pantalla LCD PCD8544

Esta pantalla se toma de la librería Adafruit_PCD8544, esta tipo de pantalla tiene la ventaja de consumir poca energía lo que la hace económica a la hora de implementar en un proyecto real. Este tipo de pantallas, se pueden encontrar en el celular Nokia 5110. En el siguiente código se puede observar un pedazo de código en donde se utiliza.

```
1 #include <Adafruit_GFX.h>
2 #include <Adafruit_PCD8544>
3
4 Adafruit_PCD8544 display = Adafruit_PCD8544(7,6,5,4,3);
```

Código 1: Pantalla PCD8544,captionpos=b, label=cod1

Cada pin al cual el display puede ser conectado significa una utilidad distinta. El pin 7, en este caso está conectado, al Serial clock out (SCLK), el pin 6, esta conectado a Serial data out (DIN), el pin 5 esta conectado al Data/Command select (D/C), el pin 4 esta conectado al LCD chip select (CS), y el pin 3 esta conectado a LCD reset (RST). La instancia se realiza en la línea cuatro del **??**. En esto se utiliza el protocolo de comunicación del SPI, el cual define lo que es cada una. [3]

SYMBOL	DESCRIPTION
R0 to R47	LCD row driver outputs
C0 to C83	LCD column driver outputs
V_{SS1}, V_{SS2}	ground
V_{DD1}, V_{DD2}	supply voltage
V_{LCD1}, V_{LCD2}	LCD supply voltage
T1	test 1 input
T2	test 2 output
T3	test 3 input/output
T4	test 4 input
SDIN	serial data input
SCLK	serial clock input
D/$\overline{C}$	data/$\overline{\text{command}}$
$\overline{SCE}$	chip enable
OSC	oscillator
$\overline{RES}$	external reset input
dummy1, 2, 3, 4	not connected

Figura 4: Características eléctricas de la pantalla

En este puede contar con un poco más de entradas ya que la pantalla que utilizamos es de dimensión 48x48, y esta es de dimensión 48x84, pero las definiciones mencionadas siguen siendo las mismas.

2.3. SPI

Bus SPI (Serial Peripheral Interface), es un protocolo que se realiza transferencias de información entre circuitos integrados en equipos electrónicos. Este se encarga de que un dispositivo electrónico digital acepte un flujo de bits regulado por una señal de reloj. Este estándar se basa en multiplexar las líneas de reloj. [1]

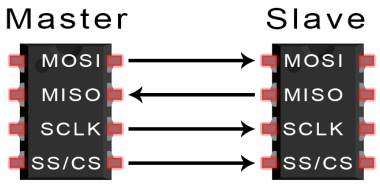


Figura 5: Diagrama general de la comunicación SPI [1]

2.4. UART

Asynchronous Receiver-Transmitter es el dispositivo que controla los puertos y dispositivos en serie. Se encuentra integrado a la placa, este es un puerto de comunicación paralela, que transfiere bits por ciclo de reloj. Un ejemplo de este son los cables VGA. [1]

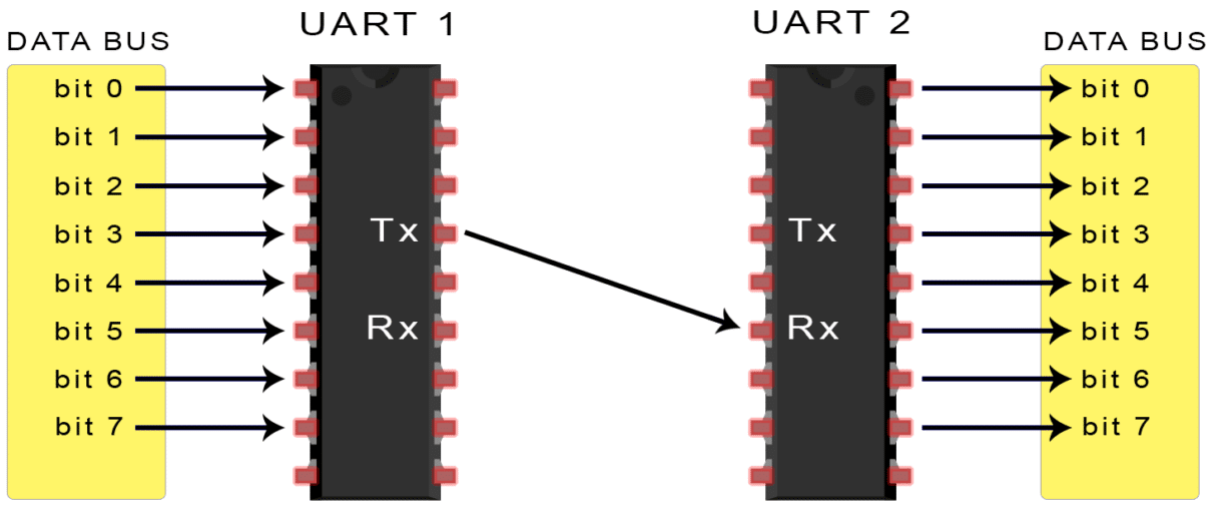


Figura 6: Diagrama general de la comunicación UART [1]

2.5. Divisor de tensión para transformar el rango de tensión de entrada

Uno de los requerimientos es que el voltímetro pueda medir tensiones en un rango de -24V a 24V y las mismas presentan dos problemas que hay que resolver, primero que los pines analógicos del Arduino solo pueden leer en un rango de 0V a 5V, por lo que hay que encontrar una forma de convertir la tensión para que esta pueda ser leída. Para resolver el problema se piensa en solucionarlo a través de un divisor de tensión con dos fuentes; es posible pasar de una tensión que varía de -24V a 24V a una de 0V a 5V. Para realizar el análisis del circuito y poder obtener los valores de las resistencias que se necesitan, se utiliza el siguiente diagrama:

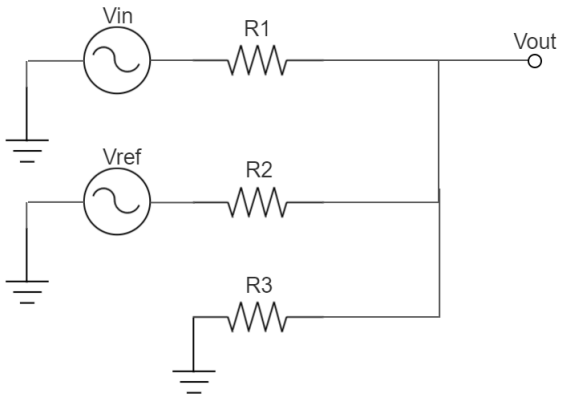


Figura 7: Diagrama de divisor de tensión. Elaboración propia

Al aplicar análisis circuital, podemos encontrar una ecuación que modele el comportamiento de la salida **Vout**. Se pueden aplicar muchos métodos para llegar a la misma respuesta, sin embargo, se escogió utilizar el teorema de la superposición. En este, se apagan las fuentes independientes una a una y se saca el aporte que cada una tiene sobre **Vout**. Comenzamos apagando la fuente V_{ref} y se encuentra que tanto R_2 como R_3 quedan en paralelo y con esto se obtiene un divisor de tensión perfecto. Al aplicar la fórmula de divisor de tensión para calcular cuánto es **Vout** se obtiene que:

$$V_{out'} = \frac{R_2||R_3}{R_2||R_3 + R_1} \cdot V_{in} \tag{1}$$

Luego, aplicamos la misma técnica para la otra fuente y en este caso R_1 y R_3 quedan en paralelo. Al calcular el segundo aporte, tenemos que:

$$V_{out''} = \frac{R_1||R_3}{R_1||R_3 + R_2} \cdot V_{ref} \tag{2}$$

Por último, se sabe que:

$$V_{out} = V_{out'} + V_{out''} = \frac{R_2||R_3}{R_2||R_3 + R_1} \cdot V_{in} + \frac{R_1||R_3}{R_1||R_3 + R_2} \cdot V_{ref} \tag{3}$$

Con la ecuación anterior, podemos empezar a dimensionar los componentes que necesitamos para poder transformar las tensiones. En este caso, de forma arbitraria se escoge que $R_1 = 1500\Omega$ y $V_{ref} = 4{,}75V$ y asimismo, creamos un sistema de ecuaciones con 2 incógnitas y con 2 ecuaciones para encontrar los valores de R_2 como R_3 que necesitamos para que cuando la tensión $V_{in} = 24V$ la salida sea de $V_{out} = 5V$ y viceversa para cuando se invierte la polaridad en la tensión de entrada. De esta forma, tenemos el siguiente sistema de ecuaciones:

$$5 = \frac{R_2||R_3}{R_2||R_3 + 1500} \cdot 24 + \frac{1500||R_3}{1500||R_3 + R_2} \cdot 4{,}75 \tag{4}$$

$$0 = \frac{R_2||R_3}{R_2||R_3 + 1500} \cdot -24 + \frac{1500||R_3}{1500||R_3 + R_2} \cdot 4{,}75 \tag{5}$$

Al resolver el sistema, se encuentra que $R_2 = 297\Omega$ y $R_3 = 423\Omega$ para cumplir con ambas igualdades. De esta forma, los valores de los componentes se resumen en el siguiente desglose:

- $R_1 = 1500\Omega$
- $R_2 = 297\Omega$
- $R_3 = 423\Omega$

Los precios se pueden ver a continuación:

Componente	**Cantidad**	**Precio en Colones**
Resistencia 1	12	₡ 180
Resistencia 1k	4	₡ 100
Resistencia 220	8	₡ 480
Resistencia 100	5	₡ 500
Resistencia 12	4	₡ 60

Tabla 2: Valor de componentes. [4]

Dando hasta ahora un total de: 1320 colones.

2.6. Componentes para tratar el efecto rebote de los switches de entrada

Considerando el efecto rebote que puede existir al pulsar un botón/switch, se diseña un capacitor con un tiempo de descarga y carga de unos cuantos milisegundos, de forma que las pequeñas fluctuaciones de los estados lógicos de los pines causados por el efecto rebote sean filtrados y se tenga una transición suave y sin picos grandes. En la siguiente imagen se puede observar el filtro que elimina el efecto rebote del switch.

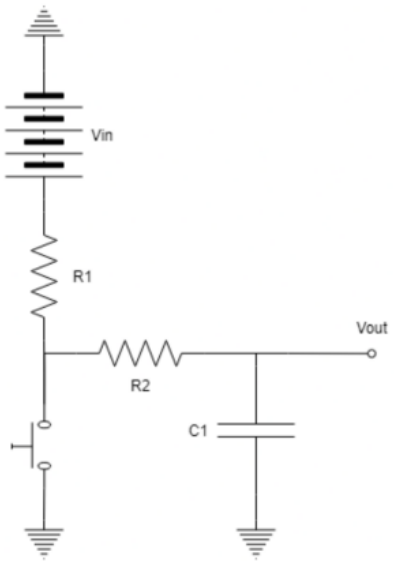


Figura 8: Filtro que elimina el efecto rebote del switch.

Por lo tanto se elige un capacitor comercial de 10μF y una resistencia de 100Ω. Donde si sacamos el tiempo τ, obtenemos que.

$$\tau = RC = 100\Omega * 10\mu F = 1ms \tag{6}$$

Este tiempo es adecuado, ya que no es muy pequeño, pero tampoco muy grande. Este tiempo es captado por el usuario y al mismo tiempo no tiene reacción directa en el resto del funcionamiento del circuito.

2.7. Resistores de protección para los LEDs de precaución

El diseño de los resistores escogidos vienen de un análisis circuital con las leyes de Kirchhoff. Se escogieron LEDs con umbrales de 1.65V y se aplicó un LTK; para una corriente de 20 mA (valor típico especificado en la hoja del fabricante de los LEDs escogidos) y con una tensión de pin de 4V (se le resta un Volt para considerar pérdidas internas), se tiene la siguiente ecuación:

$$-4 + V_{Rext} + 1{,}65 = 0 \tag{7}$$

Al cambiar V_{Rext} con ley de Ohm y despejar para encontrar el valor de R_{ext} se consigue lo siguiente:

$$R_{ext} = \frac{4 - 1{,}65}{20 \cdot 10^{-3}} \tag{8}$$

$$= 118\Omega \tag{9}$$

Esta resistencia se forma fácilmente con una resistencia de 100Ω, una de 10Ω y 8 de 1Ω en serie. Teniendo un costo total de 305 colones [4]. Esto da un costo total del proyecto de 1605 colones, sin contar los LEDs de emergencia. Contando los LEDs de emergencia, sabiendo que cada uno tiene un valor de 120 colones (ocupando 4 de estos, para cada canal), tenemos un costo total de 2085 colones. Los costos se sacaron de la siguiente referencia: [4].

2.8. Diseño lógico del firmware utilizado

Se van a enseñar algunas de las funciones implementadas para poder cumplir con todas las pautas del laboratorio.

2.8.1. Determinación del valor RMS para medición AC

En este caso, se crea una función que cuando es llamada, hace un muestreo pequeño (variando el iterador i de 0 a 100) y cada vez que encuentra un valor máximo nuevo, actualiza el mismo y retorna dicho máximo. Se implementan 4 funciones similares para los 4 canales existentes.

```
...
float get_max_vA() {
  float max_v = 0.00;
  for(int i = 0; i < 100; i++) {
    float r = analogRead(A0);
    if(max_v < r) max_v = r;
    delayMicroseconds(200);
  }
  return max_v;
}
...
```

Código 2: Función `get_max_vA()`

Una vez se encuentra el valor de amplitud máximo de la señal de entrada, esta se divide entre $\sqrt{2}$ para obtener su valor cuadrático medio.

2.8.2. Disparado de LEDs de precaución para medición DC

En esta función simplemente se revisa el valor de la medición de tensión obtenida y si supera los límites (menor a -20 o mayor a 20V) se disparan las LEDs.

```

void precaucion_DC(float vA, float vB, float vC, float vD){
  if( vA > 20 || vA < -20) digitalWrite(9, HIGH);
  else digitalWrite(9, LOW);

  if( vB > 20 || vB < -20) digitalWrite(10, HIGH);
  else digitalWrite(10, LOW);

  if( vC > 20 || vC < -20) digitalWrite(11, HIGH);
  else digitalWrite(11, LOW);

  if( vD > 20 || vD < -20) digitalWrite(12, HIGH);
  else digitalWrite(12, LOW);
}
```

Código 3: Función `precaucion_DC()`

Se realiza lo mismo para las mediciones en AC pero los límites están dados en RMS.

2.8.3. Recuperación de valor real

Inicialmente, se tenía una tensión de entrada dentro del rango de [-24V, 24V] la cual es reducida mediante el circuito divisor de tensión a un rango de [0V, 5V]; debemos recuperar el valor original de la señal de entrada para poder reflejarla en la pantalla LCD.

Para ello, se utiliza una constante llamada *converting_factor* la cual escala la señal de entrada convertida de [0V, 5V] a un rango de [0V, 48V] y, de mismo modo, se le restan 24 unidades y de esta forma se puede regresar al rango de valores entre [-24V, 24V]; se logra desarrollar el proceso inverso del inicio.

```
1
2  void loop(){
3  ...
4
5  else{ // si comms esta cerrado mide DC
6
7      //Valores DC
8      vA = analogRead(A0);
9      vB = analogRead(A1);
10     vC = analogRead(A2);
11     vD = analogRead(A3);
12
13
14     vAk = (((vA*5)/1023)*converting_factor)-24;
15     vBk = (((vB*5)/1023)*converting_factor)-24;
16     vCk = (((vC*5)/1023)*converting_factor)-24;
17     vDk = (((vD*5)/1023)*converting_factor)-24;
18
19 ...
20 }
```

Código 4: Función `loop()`

3. Desarrollo y Análisis de Resultados

La figuras 9 y 10 muestra el esquemático del circuito creado en el simulador Simulide. Al comenzar a tomar mediciones, nótese que las tensiones se muestran en la pantalla LCD ubicada en la esquina superior izquierda y, asimismo, esas mediciones son enviadas mediante el puerto serial y las recibe el script de Python para ser procesadas y guardadas en un archivo .csv; la figura 12 muestra un par de mediciones "arregladas" en formato .csv mientras que la figura 11 muestra la ejecución del script llamado *config.sh* el cual se encarga de conectar los puertos seriales con los dispositivos correspondientes por medio de comunicación USART (una vez se conectan los puertos seriales se debe ejecutar el archivo *archivocsv.py* para que comience la extracción de las mediciones).

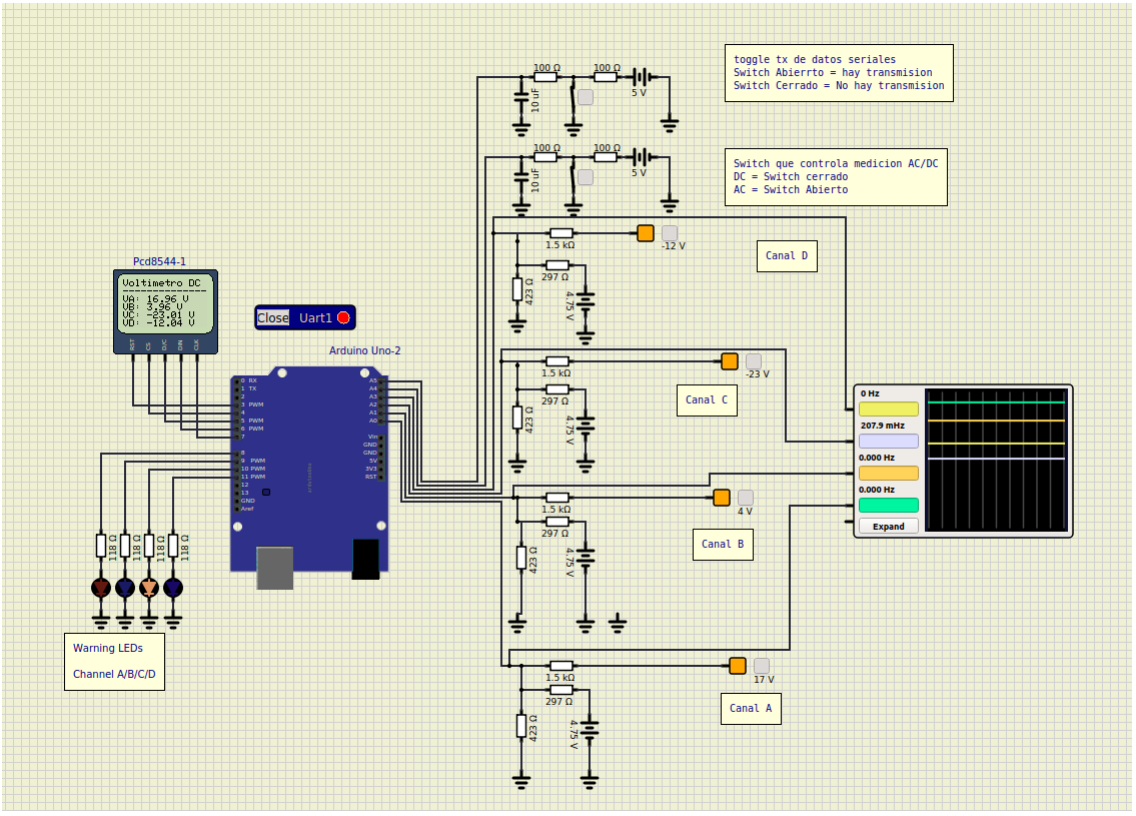


Figura 9: Resultados de las mediciones DC.

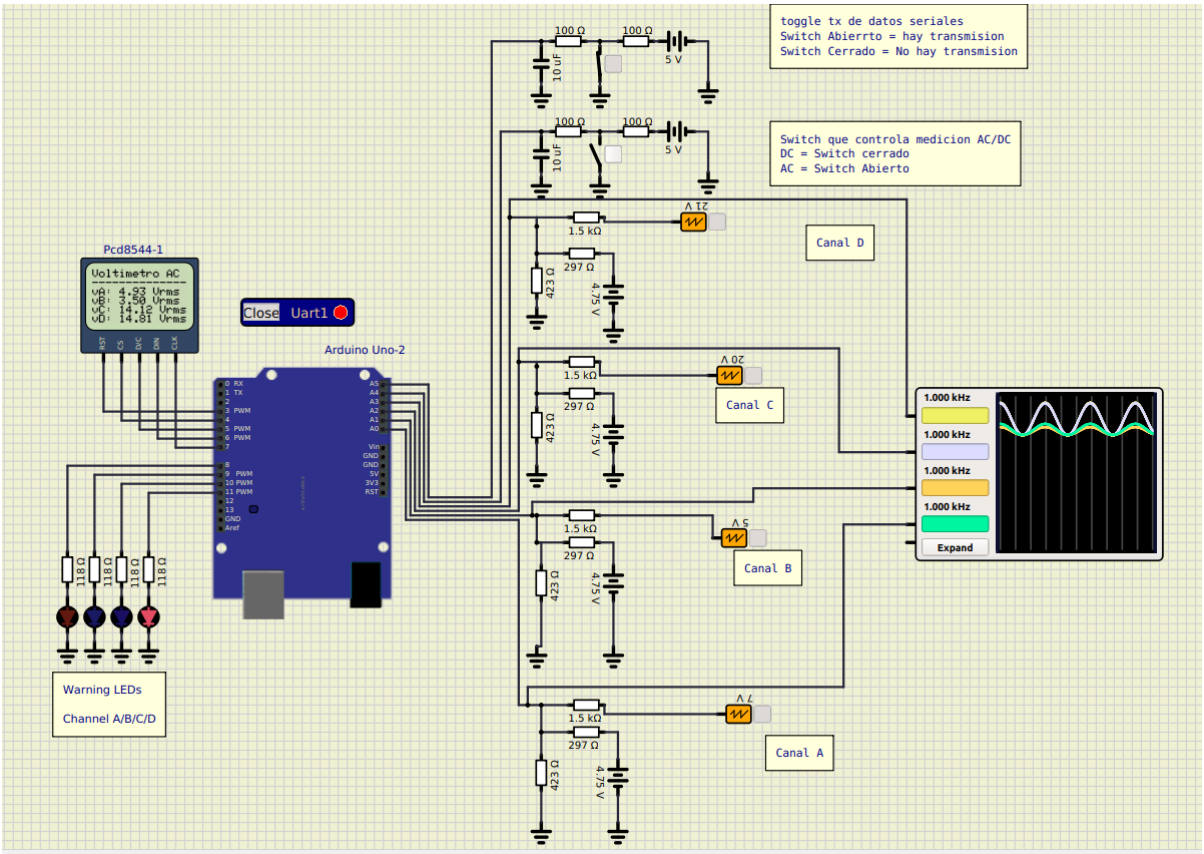


Figura 10: Resultados de las mediciones AC.

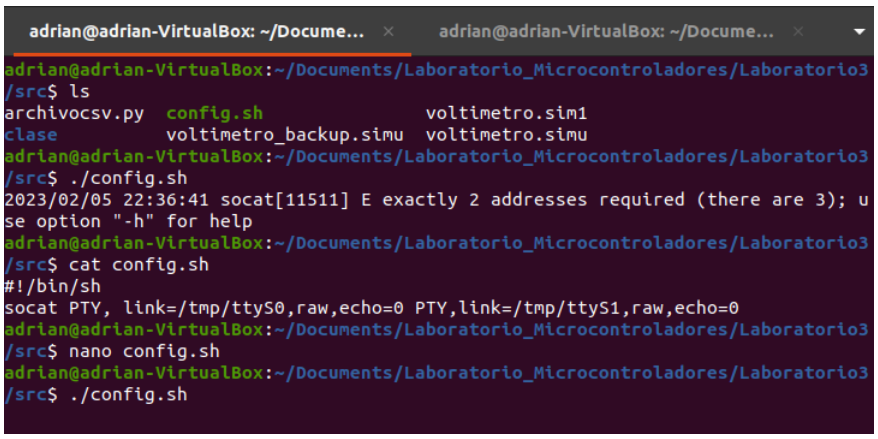


Figura 11: Conexión entre PC y script de Python para la recolección de datos.

	A	B	C
1	2.92,DC,24.00,24.00,24.00		
2	23.95,DC,24.00,24.00,24.00		
3	21.75,DC,24.00,24.00,24.00		
4	0.49,DC,24.00,24.00,24.00		
5	17.76,DC,24.00,24.00,24.00		
6	6.26,DC,24.00,24.00,24.00		
7	3.03,DC,24.00,24.00,24.00		
8	21.94,DC,24.00,24.00,24.00		
9	2.93,DC,24.00,24.00,24.00		
10	20.15,DC,24.00,24.00,24.00		
11	10.25,DC,24.00,24.00,24.00		
12	1.76,DC,24.00,24.00,24.00		
13	1.48,DC,24.00,24.00,24.00		
14	5.98,DC,24.00,24.00,24.00		
15	23.95,DC,24.00,24.00,24.00		
16	21.65,DC,24.00,24.00,24.00		
17	7.16,DC,24.00,24.00,24.00		
18	7.06,DC,24.00,24.00,24.00		
19	21.33,DC,24.00,24.00,24.00		
20	2.32,DC,24.00,24.00,24.00		
21	19.92,DC,24.00,24.00,24.00		

Figura 12: Archivo .csv con las mediciones del voltímetro.

A simple vista, se puede notar que el valor mostrado en la pantalla LCD posee una pequeña discrepancia con el valor de tensión de entrada, lo cual es de esperar debido a que los simuladores siempre manejan un cierto grado de incertidumbre a la hora de poner a operar dichos elementos en conjunto, pero son lo suficientemente pequeños como para no afectar a gran escala el funcionamiento general del voltímetro y, por lo tanto, se puede concluir que el laboratorio fue realizado con éxito.

4. Conclusiones y recomendaciones

- Se concluye que trabajar con SimulIDE es una buena aproximación al trabajo con el hardware real. Es una buena práctica realizar el circuito en simulaciones con esta, para luego ejecutarlas en la vida real.
- Se recomienda hacer uso de medidores de tensión, de corriente y osciloscopios para poder verificar que todo este funcionando adecuadamente.
- Se aprende a realizar una conexión entre el Arduino y el código Python que produce los resultados que se exportan en el archivo csv.

Referencias

[1] Scott Campbell. Basics of the spi communication protocol, Nov 2021.

[2] Www.facebook.com/mecafenix. ¿cómo funciona el convertidor analógico a digital?, Jan 2023.

[3] Graphic lcd 84x48 - nokia 5110.

[4] Steren. Venta de potenciómetros y resistencias.

5. Anexos

5.1. Repositorio Git

En este repositorio `eusanchez y amonbon GIT`, se adjunta el proyecto donde se puede observar el progreso y los archivos de `.simu` y `.c`.

Nota: El desarrollo del laboratorio está en la branch llamada `Laboratorio3`.

5.2. Hojas del Fabricante del ATmega328

8-bit AVR Microcontrollers

ATmega328/P

DATASHEET COMPLETE

Introduction

The Atmel® picoPower® ATmega328/P is a low-power CMOS 8-bit microcontroller based on the AVR® enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the ATmega328/P achieves throughputs close to 1MIPS per MHz. This empowers system designer to optimize the device for power consumption versus processing speed.

Feature

High Performance, Low Power Atmel®AVR® 8-Bit Microcontroller Family

- Advanced RISC Architecture
  - 131 Powerful Instructions
  - Most Single Clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 20 MIPS Throughput at 20MHz
  - On-chip 2-cycle Multiplier
- High Endurance Non-volatile Memory Segments
  - 32KBytes of In-System Self-Programmable Flash program Memory
  - 1KBytes EEPROM
  - 2KBytes Internal SRAM
  - Write/Erase Cycles: 10,000 Flash/100,000 EEPROM
  - Data Retention: 20 years at 85°C/100 years at 25°C(1)
  - Optional Boot Code Section with Independent Lock Bits
    - In-System Programming by On-chip Boot Program
    - True Read-While-Write Operation
  - Programming Lock for Software Security
- Atmel® QTouch® Library Support
  - Capacitive Touch Buttons, Sliders and Wheels
  - QTouch and QMatrix® Acquisition
  - Up to 64 sense channels

- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescaler and Compare Mode
  - One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Six PWM Channels
  - 8-channel 10-bit ADC in TQFP and QFN/MLF package
    - Temperature Measurement
  - 6-channel 10-bit ADC in PDIP Package
    - Temperature Measurement
  - Two Master/Slave SPI Serial Interface
  - One Programmable Serial USART
  - One Byte-oriented 2-wire Serial Interface (Philips I^2C compatible)
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - One On-chip Analog Comparator
  - Interrupt and Wake-up on Pin Change
- Special Microcontroller Features
  - Power-on Reset and Programmable Brown-out Detection
  - Internal Calibrated Oscillator
  - External and Internal Interrupt Sources
  - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby, and Extended Standby
- I/O and Packages
  - 23 Programmable I/O Lines
  - 28-pin PDIP, 32-lead TQFP, 28-pad QFN/MLF and 32-pad QFN/MLF
- Operating Voltage:
  - 1.8 - 5.5V
- Temperature Range:
  - -40°C to 105°C
- Speed Grade:
  - 0 - 4MHz @ 1.8 - 5.5V
  - 0 - 10MHz @ 2.7 - 5.5V
  - 0 - 20MHz @ 4.5 - 5.5V
- Power Consumption at 1MHz, 1.8V, 25°C
  - Active Mode: 0.2mA
  - Power-down Mode: 0.1µA
  - Power-save Mode: 0.75µA (Including 32kHz RTC)

Table of Contents

1. Description

The Atmel AVR® core combines a rich instruction set with 32 general purpose working registers. All the 32 registers are directly connected to the Arithmetic Logic Unit (ALU), allowing two independent registers to be accessed in a single instruction executed in one clock cycle. The resulting architecture is more code efficient while achieving throughputs up to ten times faster than conventional CISC microcontrollers.

The ATmega328/P provides the following features: 32Kbytes of In-System Programmable Flash with Read-While-Write capabilities, 1Kbytes EEPROM, 2Kbytes SRAM, 23 general purpose I/O lines, 32 general purpose working registers, Real Time Counter (RTC), three flexible Timer/Counters with compare modes and PWM, 1 serial programmable USARTs , 1 byte-oriented 2-wire Serial Interface (I2C), a 6-channel 10-bit ADC (8 channels in TQFP and QFN/MLF packages) , a programmable Watchdog Timer with internal Oscillator, an SPI serial port, and six software selectable power saving modes. The Idle mode stops the CPU while allowing the SRAM, Timer/Counters, SPI port, and interrupt system to continue functioning. The Power-down mode saves the register contents but freezes the Oscillator, disabling all other chip functions until the next interrupt or hardware reset. In Power-save mode, the asynchronous timer continues to run, allowing the user to maintain a timer base while the rest of the device is sleeping. The ADC Noise Reduction mode stops the CPU and all I/O modules except asynchronous timer and ADC to minimize switching noise during ADC conversions. In Standby mode, the crystal/resonator oscillator is running while the rest of the device is sleeping. This allows very fast start-up combined with low power consumption. In Extended Standby mode, both the main oscillator and the asynchronous timer continue to run.

Atmel offers the QTouch® library for embedding capacitive touch buttons, sliders and wheels functionality into AVR microcontrollers. The patented charge-transfer signal acquisition offers robust sensing and includes fully debounced reporting of touch keys and includes Adjacent Key Suppression® (AKS™) technology for unambiguous detection of key events. The easy-to-use QTouch Suite toolchain allows you to explore, develop and debug your own touch applications.

The device is manufactured using Atmel’s high density non-volatile memory technology. The On-chip ISP Flash allows the program memory to be reprogrammed In-System through an SPI serial interface, by a conventional nonvolatile memory programmer, or by an On-chip Boot program running on the AVR core. The Boot program can use any interface to download the application program in the Application Flash memory. Software in the Boot Flash section will continue to run while the Application Flash section is updated, providing true Read-While-Write operation. By combining an 8-bit RISC CPU with In-System Self-Programmable Flash on a monolithic chip, the Atmel ATmega328/P is a powerful microcontroller that provides a highly flexible and cost effective solution to many embedded control applications.

The ATmega328/P is supported with a full suite of program and system development tools including: C Compilers, Macro Assemblers, Program Debugger/Simulators, In-Circuit Emulators, and Evaluation kits.

2. Configuration Summary

Features	ATmega328/P
Pin Count	28/32
Flash (Bytes)	32K
SRAM (Bytes)	2K
EEPROM (Bytes)	1K
General Purpose I/O Lines	23
SPI	2
TWI (I^2C)	1
USART	1
ADC	10-bit 15kSPS
ADC Channels	8
8-bit Timer/Counters	2
16-bit Timer/Counters	1

3. Ordering Information

3.1. ATmega328

Speed [MHz][3]	Power Supply [V]	Ordering Code[2]	Package[1]	Operational Range
20	1.8 - 5.5	ATmega328-AU ATmega328-AUR[5] ATmega328-MMH[4] ATmega328-MMHR[4][5] ATmega328-MU ATmega328-MUR[5] ATmega328-PU	32A 32A 28M1 28M1 32M1-A 32M1-A 28P3	Industrial (-40°C to 85°C)

Note:

1. This device can also be supplied in wafer form. Please contact your local Atmel sales office for detailed ordering information and minimum quantities.
2. Pb-free packaging, complies to the European Directive for Restriction of Hazardous Substances (RoHS directive). Also Halide free and fully Green.
3. Please refer to *Speed Grades* for Speed vs. V_{CC}
4. Tape & Reel.
5. NiPdAu Lead Finish.

Package Type	
28M1	28-pad, 4 x 4 x 1.0 body, Lead Pitch 0.45mm Quad Flat No-Lead/Micro Lead Frame Package (QFN/MLF)
28P3	28-lead, 0.300” Wide, Plastic Dual Inline Package (PDIP)
32M1-A	32-pad, 5 x 5 x 1.0 body, Lead Pitch 0.50mm Quad Flat No-Lead/Micro Lead Frame Package (QFN/MLF)
32A	32-lead, Thin (1.0mm) Plastic Quad Flat Package (TQFP)

3.2. ATmega328P

Speed [MHz][3]	Power Supply [V]	Ordering Code[2]	Package[1]	Operational Range
20	1.8 - 5.5	ATmega328P-AU ATmega328P-AUR[5] ATmega328P-MMH[4] ATmega328P-MMHR[4][5] ATmega328P-MU ATmega328P-MUR[5] ATmega328P-PU	32A 32A 28M1 28M1 32M1-A 32M1-A 28P3	Industrial (-40°C to 85°C)
		ATmega328P-AN ATmega328P-ANR[5] ATmega328P-MN ATmega328P-MNR[5] ATmega328P-PN	32A 32A 32M1-A 32M1-A 28P3	Industrial (-40°C to 105°C)

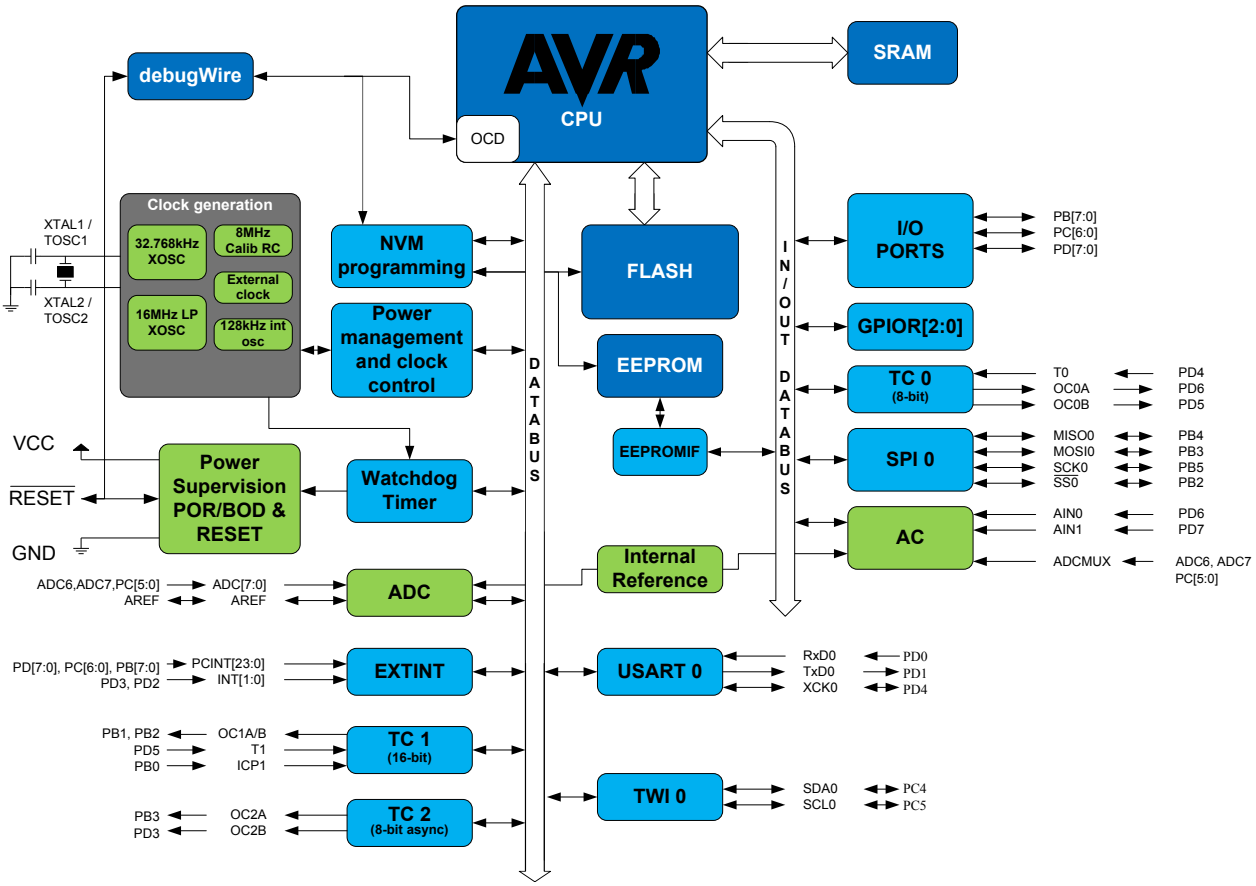
Note:

1. This device can also be supplied in wafer form. Please contact your local Atmel sales office for detailed ordering information and minimum quantities.
2. Pb-free packaging, complies to the European Directive for Restriction of Hazardous Substances (RoHS directive). Also Halide free and fully Green.
3. Please refer to *Speed Grades* for Speed vs. V_{CC}
4. Tape & Reel.
5. NiPdAu Lead Finish.

Package Type	
28M1	28-pad, 4 x 4 x 1.0 body, Lead Pitch 0.45mm Quad Flat No-Lead/Micro Lead Frame Package (QFN/MLF)
28P3	28-lead, 0.300” Wide, Plastic Dual Inline Package (PDIP)
32M1-A	32-pad, 5 x 5 x 1.0 body, Lead Pitch 0.50mm Quad Flat No-Lead/Micro Lead Frame Package (QFN/MLF)
32A	32-lead, Thin (1.0mm) Plastic Quad Flat Package (TQFP)

4. Block Diagram

Figure 4-1. Block Diagram



5. Pin Configurations

5.1. Pin-out

Figure 5-1. 28-pin PDIP

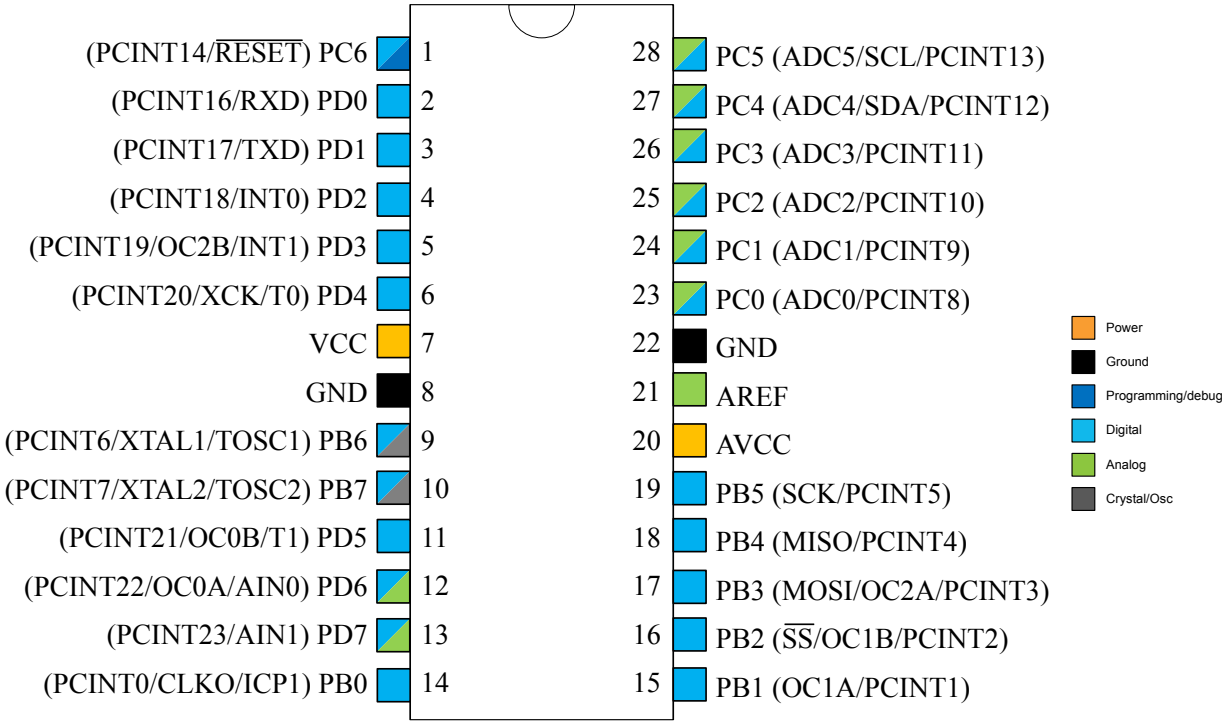


Figure 5-2. 28-pin MLF Top View

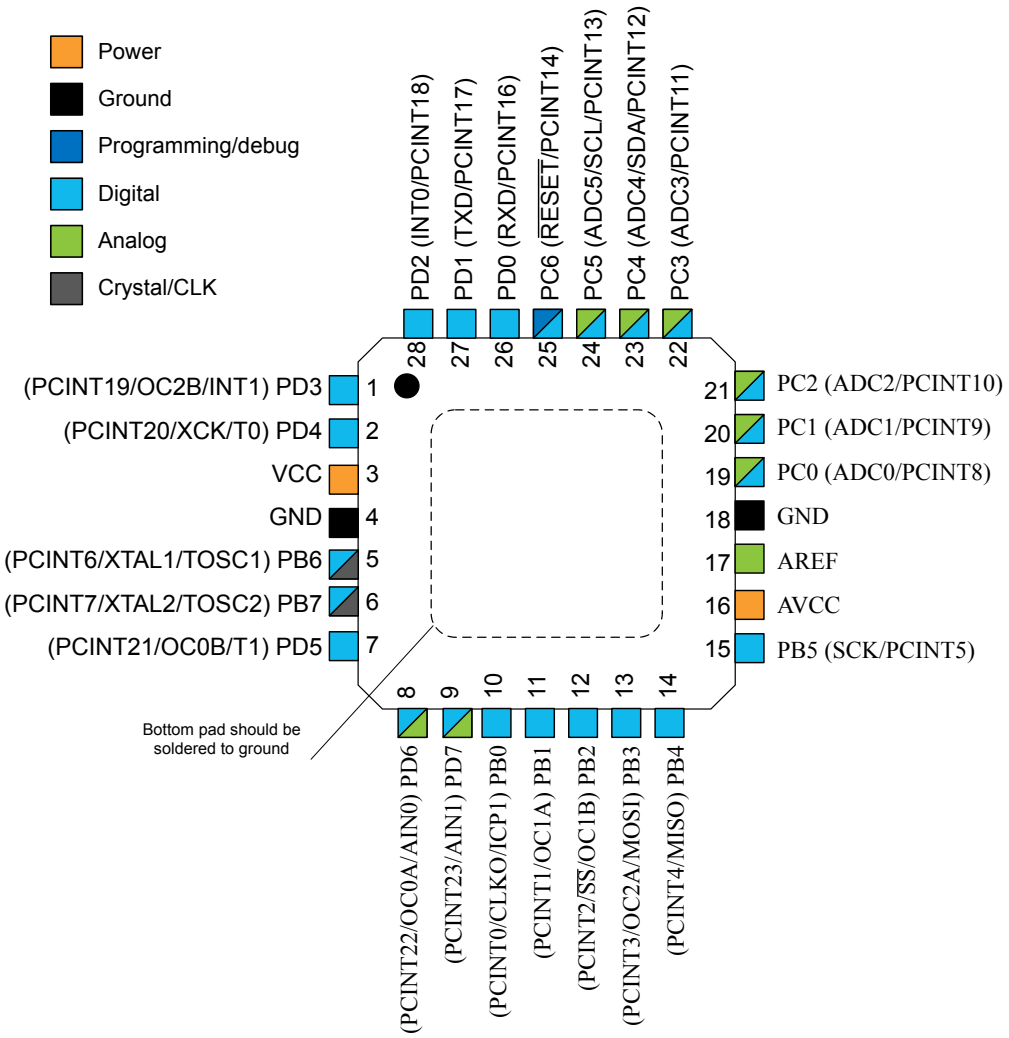


Figure 5-3. 32-pin TQFP Top View

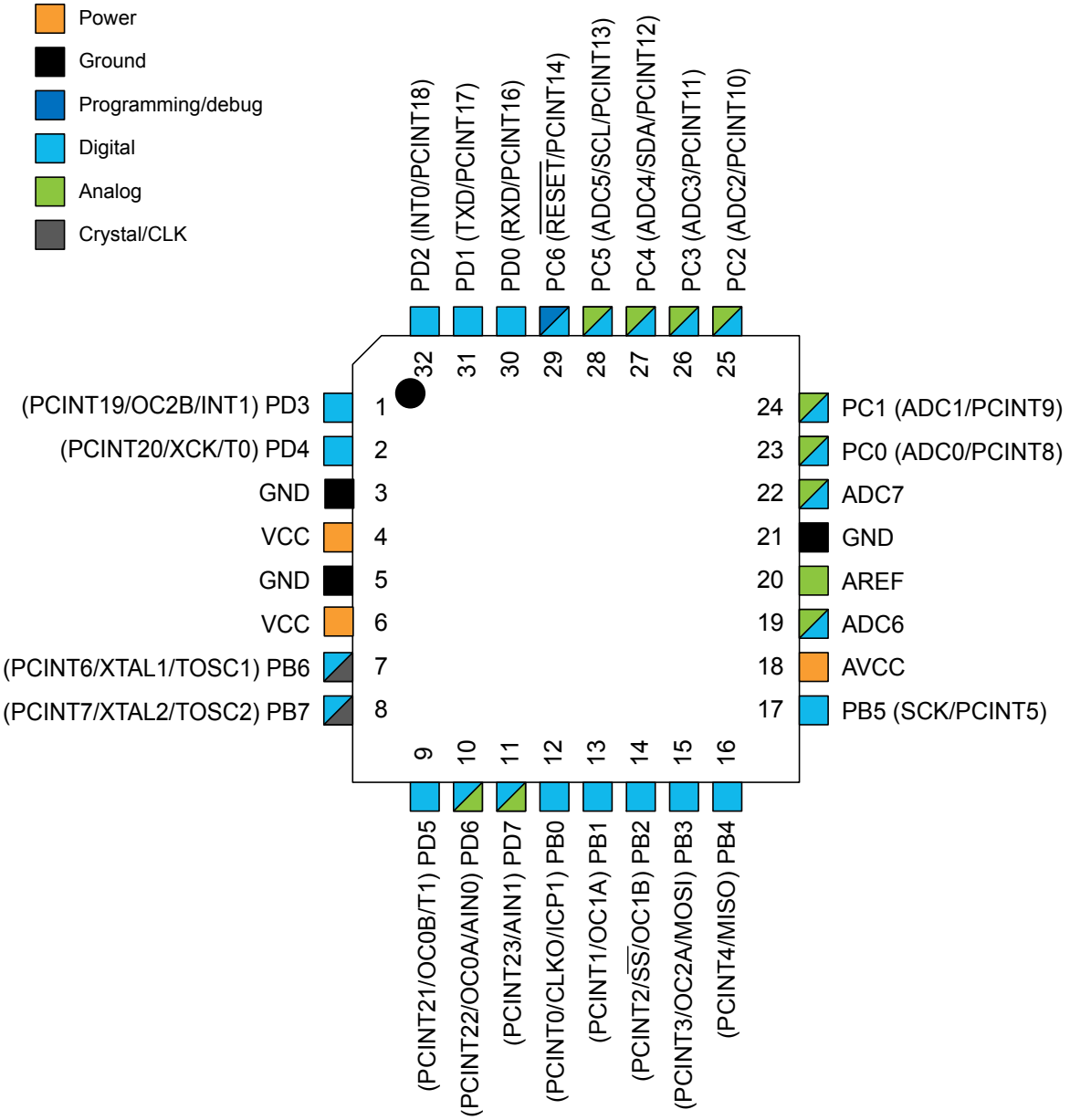
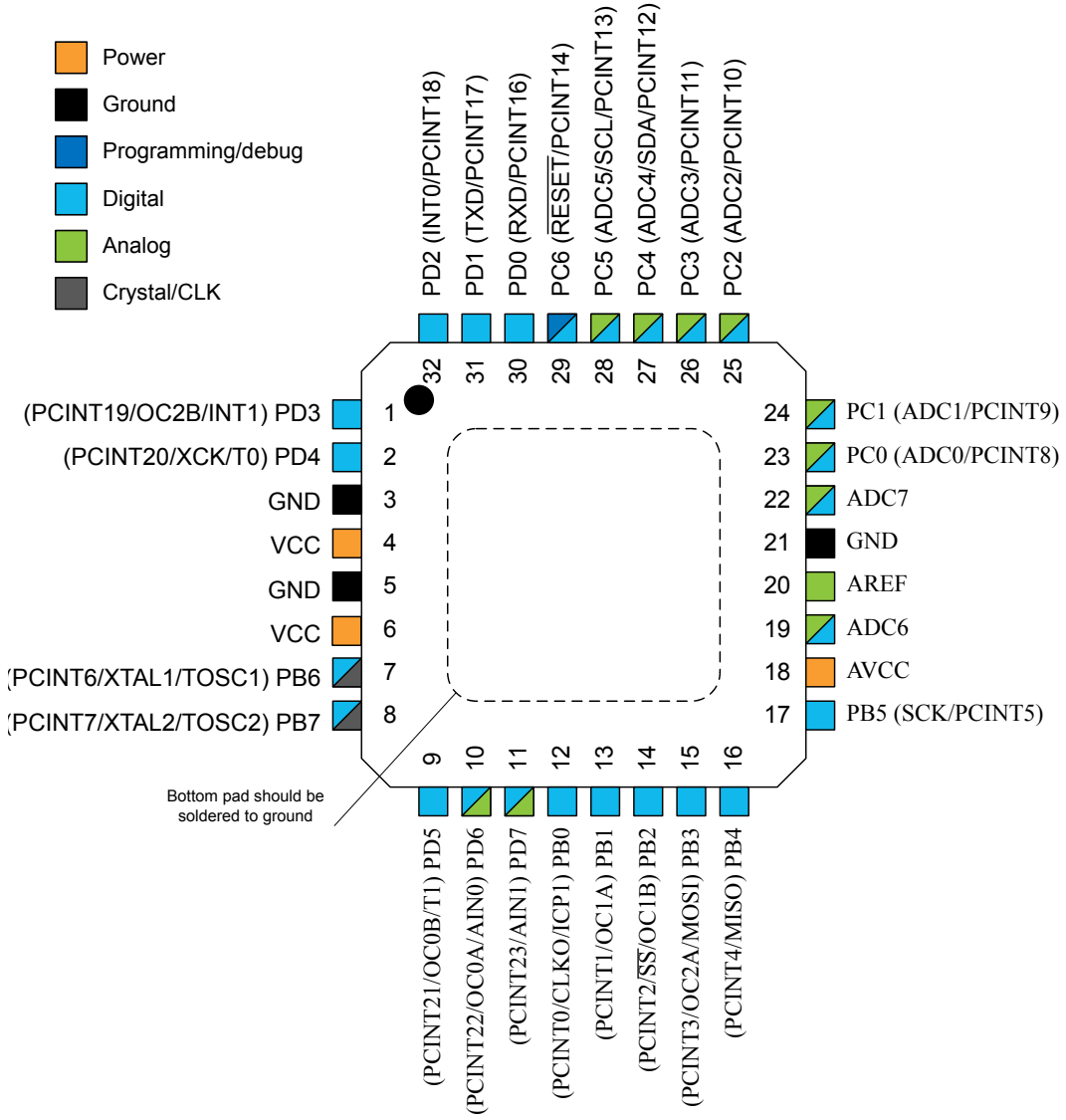


Figure 5-4. 32-pin MLF Top View



5.2. Pin Descriptions

5.2.1. VCC

Digital supply voltage.

5.2.2. GND

Ground.

5.2.3. Port B (PB[7:0]) XTAL1/XTAL2/TOSC1/TOSC2

Port B is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port B output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port B pins that are externally pulled low will source current if the pull-up resistors are activated. The Port B pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Depending on the clock selection fuse settings, PB6 can be used as input to the inverting Oscillator amplifier and input to the internal clock operating circuit.

Depending on the clock selection fuse settings, PB7 can be used as output from the inverting Oscillator amplifier.

If the Internal Calibrated RC Oscillator is used as chip clock source, PB[7:6] is used as TOSC[2:1] input for the Asynchronous Timer/Counter2 if the AS2 bit in ASSR is set.

5.2.4. Port C (PC[5:0])

Port C is a 7-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The PC[5:0] output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port C pins that are externally pulled low will source current if the pull-up resistors are activated. The Port C pins are tri-stated when a reset condition becomes active, even if the clock is not running.

5.2.5. PC6/$\overline{\text{RESET}}$

If the RSTDISBL Fuse is programmed, PC6 is used as an I/O pin. Note that the electrical characteristics of PC6 differ from those of the other pins of Port C.

If the RSTDISBL Fuse is unprogrammed, PC6 is used as a Reset input. A low level on this pin for longer than the minimum pulse length will generate a Reset, even if the clock is not running. Shorter pulses are not guaranteed to generate a Reset.

The various special features of Port C are elaborated in the *Alternate Functions of Port C* section.

5.2.6. Port D (PD[7:0])

Port D is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port D output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port D pins that are externally pulled low will source current if the pull-up resistors are activated. The Port D pins are tri-stated when a reset condition becomes active, even if the clock is not running.

5.2.7. AV_{CC}

AV_{CC} is the supply voltage pin for the A/D Converter, PC[3:0], and PE[3:2]. It should be externally connected to V_{CC}, even if the ADC is not used. If the ADC is used, it should be connected to V_{CC} through a low-pass filter. Note that PC[6:4] use digital supply voltage, V_{CC}.

5.2.8. AREF

AREF is the analog reference pin for the A/D Converter.

5.2.9. ADC[7:6] (TQFP and VFQFN Package Only)

In the TQFP and VFQFN package, ADC[7:6] serve as analog inputs to the A/D converter. These pins are powered from the analog supply and serve as 10-bit ADC channels.

6. I/O Multiplexing

Each pin is by default controlled by the PORT as a general purpose I/O and alternatively it can be assigned to one of the peripheral functions.

The following table describes the peripheral signals multiplexed to the PORT I/O pins.

Table 6-1. PORT Function Multiplexing

(32-pin MLF/TQFP) Pin#	(28-pin MLF) Pin#	(28-pin PIPD) Pin#	PAD	EXTINT	PCINT	ADC/AC	OSC	T/C #0	T/C #1	USART 0	I2C 0	SPI 0
1	1	5	PD[3]	INT1	PCINT19			OC2B				
2	2	6	PD[4]		PCINT20			T0		XCK0		
3	3	7	VCC									
4	4	8	GND									
5	-	-	VCC									
6	-	-	GND									
7	5	9	PB[6]		PCINT6		XTAL1/ TOSC1					
8	6	10	PB[7]		PCINT7		XTAL2/ TOSC2					
9	7	11	PD[5]		PCINT21			OC0B	T1			
10	8	12	PD[6]		PCINT22	AIN0		OC0A				
11	9	13	PD[7]		PCINT23	AIN1						
12	10	14	PB[0]		PCINT0		CLKO	ICP1				
13	11	15	PB[1]		PCINT1			OC1A				
14	12	16	PB[2]		PCINT2			OC1B				$\overline{SS0}$
15	13	17	PB[3]		PCINT3			OC2A				MOSI0
16	14	18	PB[4]		PCINT4							MISO0
17	15	19	PB[5]		PCINT5							SCK0
18	16	20	AVCC									
19	-	-	ADC6			ADC6						
20	17	21	AREF									
21	18	22	GND									
22	-	-	ADC7			ADC7						
23	19	13	PC[0]		PCINT8	ADC0						
24	20	24	PC[1]		PCINT9	ADC1						
25	21	25	PC[2]		PCINT10	ADC2						
26	22	26	PC[3]		PCINT11	ADC3						
27	23	27	PC[4]		PCINT12	ADC4					SDA0	
28	24	28	PC[5]		PCINT13	ADC5					SCL0	
29	25	1	PC[6]/ $\overline{RESET}$		PCINT14							

(32-pin MLF/TQFP) Pin#	(28-pin MLF) Pin#	(28-pin PIPD) Pin#	PAD	EXTINT	PCINT	ADC/AC	OSC	T/C #0	T/C #1	USART 0	I2C 0	SPI 0
30	26	2	PD[0]		PCINT16					RXD0		
31	27	3	PD[1]		PCINT17					TXD0		
32	28	4	PD[2]	INT0	PCINT18							

7. Resources

A comprehensive set of development tools, application notes, and datasheets are available for download on http://www.atmel.com/avr.

8. Data Retention

Reliability Qualification results show that the projected data retention failure rate is much less than 1 PPM over 20 years at 85°C or 100 years at 25°C.

9. About Code Examples

This documentation contains simple code examples that briefly show how to use various parts of the device. These code examples assume that the part specific header file is included before compilation. Be aware that not all C compiler vendors include bit definitions in the header files and interrupt handling in C is compiler dependent. Confirm with the C compiler documentation for more details.

For I/O Registers located in extended I/O map, “IN”, “OUT”, “SBIS”, “SBIC”, “CBI”, and “SBI” instructions must be replaced with instructions that allow access to extended I/O. Typically “LDS” and “STS” combined with “SBRS”, “SBRC”, “SBR”, and “CBR”.

10. Capacitive Touch Sensing

10.1. QTouch Library

The Atmel® QTouch® Library provides a simple to use solution to realize touch sensitive interfaces on most Atmel AVR® microcontrollers. The QTouch Library includes support for the Atmel QTouch and Atmel QMatrix® acquisition methods.

Touch sensing can be added to any application by linking the appropriate Atmel QTouch Library for the AVR Microcontroller. This is done by using a simple set of APIs to define the touch channels and sensors, and then calling the touch sensing API's to retrieve the channel information and determine the touch sensor states.

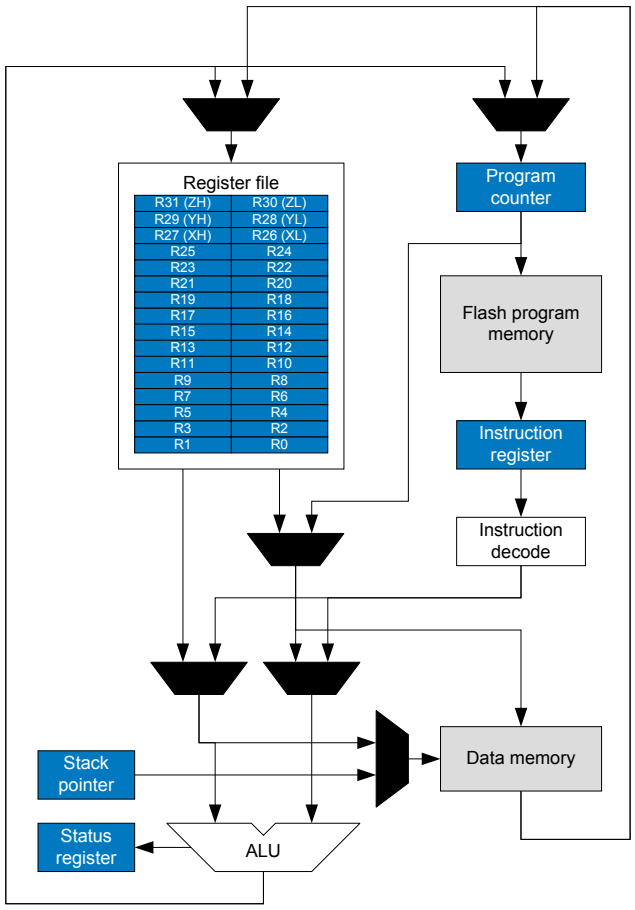
The QTouch Library is FREE and downloadable from the Atmel website at the following location: http://www.atmel.com/technologies/touch/. For implementation details and other information, refer to the Atmel QTouch Library User Guide - also available for download from the Atmel website.

11. AVR CPU Core

11.1. Overview

This section discusses the AVR core architecture in general. The main function of the CPU core is to ensure correct program execution. The CPU must therefore be able to access memories, perform calculations, control peripherals, and handle interrupts.

Figure 11-1. Block Diagram of the AVR Architecture



In order to maximize performance and parallelism, the AVR uses a Harvard architecture – with separate memories and buses for program and data. Instructions in the program memory are executed with a single level pipelining. While one instruction is being executed, the next instruction is pre-fetched from the program memory. This concept enables instructions to be executed in every clock cycle. The program memory is In-System Reprogrammable Flash memory.

The fast-access Register File contains 32 x 8-bit general purpose working registers with a single clock cycle access time. This allows single-cycle Arithmetic Logic Unit (ALU) operation. In a typical ALU operation, two operands are output from the Register File, the operation is executed, and the result is stored back in the Register File – in one clock cycle.

Six of the 32 registers can be used as three 16-bit indirect address register pointers for Data Space addressing – enabling efficient address calculations. One of the these address pointers can also be used as an address pointer for look up tables in Flash program memory. These added function registers are the 16-bit X-, Y-, and Z-register, described later in this section.

The ALU supports arithmetic and logic operations between registers or between a constant and a register. Single register operations can also be executed in the ALU. After an arithmetic operation, the Status Register is updated to reflect information about the result of the operation.

Program flow is provided by conditional and unconditional jump and call instructions, able to directly address the whole address space. Most AVR instructions have a single 16-bit word format. Every program memory address contains a 16- or 32-bit instruction.

Program Flash memory space is divided in two sections, the Boot Program section and the Application Program section. Both sections have dedicated Lock bits for write and read/write protection. The SPM instruction that writes into the Application Flash memory section must reside in the Boot Program section.

During interrupts and subroutine calls, the return address Program Counter (PC) is stored on the Stack. The Stack is effectively allocated in the general data SRAM, and consequently the Stack size is only limited by the total SRAM size and the usage of the SRAM. All user programs must initialize the SP in the Reset routine (before subroutines or interrupts are executed). The Stack Pointer (SP) is read/write accessible in the I/O space. The data SRAM can easily be accessed through the five different addressing modes supported in the AVR architecture.

The memory spaces in the AVR architecture are all linear and regular memory maps.

A flexible interrupt module has its control registers in the I/O space with an additional Global Interrupt Enable bit in the Status Register. All interrupts have a separate Interrupt Vector in the Interrupt Vector table. The interrupts have priority in accordance with their Interrupt Vector position. The lower the Interrupt Vector address, the higher the priority.

The I/O memory space contains 64 addresses for CPU peripheral functions as Control Registers, SPI, and other I/O functions. The I/O Memory can be accessed directly, or as the Data Space locations following those of the Register File, 0x20 - 0x5F. In addition, this device has Extended I/O space from 0x60 - 0xFF in SRAM where only the ST/STS/STD and LD/LDS/LDD instructions can be used.

11.2. ALU – Arithmetic Logic Unit

The high-performance AVR ALU operates in direct connection with all the 32 general purpose working registers. Within a single clock cycle, arithmetic operations between general purpose registers or between a register and an immediate are executed. The ALU operations are divided into three main categories – arithmetic, logical, and bit-functions. Some implementations of the architecture also provide a powerful multiplier supporting both signed/unsigned multiplication and fractional format. See *Instruction Set Summary* section for a detailed description.

Related Links

Instruction Set Summary on page 432

11.3. Status Register

The Status Register contains information about the result of the most recently executed arithmetic instruction. This information can be used for altering program flow in order to perform conditional operations. The Status Register is updated after all ALU operations, as specified in the Instruction Set Reference. This will in many cases remove the need for using the dedicated compare instructions, resulting in faster and more compact code.

The Status Register is not automatically stored when entering an interrupt routine and restored when returning from an interrupt. This must be handled by software.

11.3.1. Status Register

When addressing I/O Registers as data space using LD and ST instructions, the provided offset must be used. When using the I/O specific commands IN and OUT, the offset is reduced by 0x20, resulting in an I/O address offset within 0x00 - 0x3F.

Name: SREG
Offset: 0x5F
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x3F

Bit	7	6	5	4	3	2	1	0
	I	T	H	S	V	N	Z	C
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 7 – I: Global Interrupt Enable
The Global Interrupt Enable bit must be set for the interrupts to be enabled. The individual interrupt enable control is then performed in separate control registers. If the Global Interrupt Enable Register is cleared, none of the interrupts are enabled independent of the individual interrupt enable settings. The I-bit is cleared by hardware after an interrupt has occurred, and is set by the RETI instruction to enable subsequent interrupts. The I-bit can also be set and cleared by the application with the SEI and CLI instructions, as described in the instruction set reference.

Bit 6 – T: Copy Storage
The Bit Copy instructions BLD (Bit LoaD) and BST (Bit STore) use the T-bit as source or destination for the operated bit. A bit from a register in the Register File can be copied into T by the BST instruction, and a bit in T can be copied into a bit in a register in the Register File by the BLD instruction.

Bit 5 – H: Half Carry Flag
The Half Carry Flag H indicates a Half Carry in some arithmetic operations. Half Carry Flag is useful in BCD arithmetic. See the *Instruction Set Description* for detailed information.

Bit 4 – S: Sign Flag, S = N ⊕ V

The S-bit is always an exclusive or between the Negative Flag N and the Two's Complement Overflow Flag V. See the *Instruction Set Description* for detailed information.

Bit 3 – V: Two's Complement Overflow Flag
The Two's Complement Overflow Flag V supports two's complement arithmetic. See the *Instruction Set Description* for detailed information.

Bit 2 – N: Negative Flag
The Negative Flag N indicates a negative result in an arithmetic or logic operation. See the *Instruction Set Description* for detailed information.

Bit 1 – Z: Zero Flag
The Zero Flag Z indicates a zero result in an arithmetic or logic operation. See the *Instruction Set Description* for detailed information.

Bit 0 – C: Carry Flag

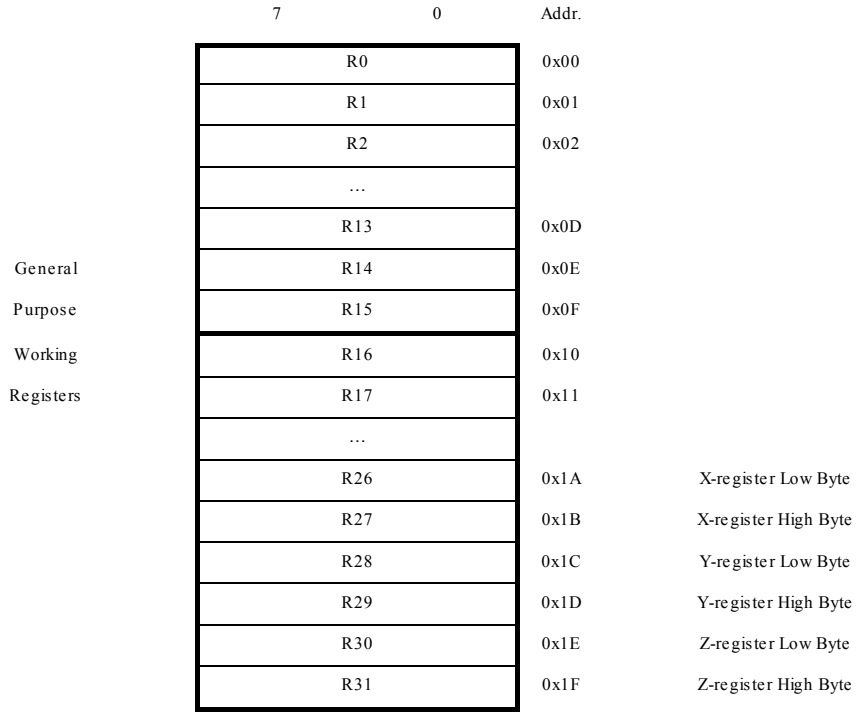
The Carry Flag C indicates a carry in an arithmetic or logic operation. See the *Instruction Set Description* for detailed information.

11.4. General Purpose Register File

The Register File is optimized for the AVR Enhanced RISC instruction set. In order to achieve the required performance and flexibility, the following input/output schemes are supported by the Register File:

- One 8-bit output operand and one 8-bit result input
- Two 8-bit output operands and one 8-bit result input
- Two 8-bit output operands and one 16-bit result input
- One 16-bit output operand and one 16-bit result input

Figure 11-2. AVR CPU General Purpose Working Registers



Most of the instructions operating on the Register File have direct access to all registers, and most of them are single cycle instructions. As shown in the figure, each register is also assigned a data memory address, mapping them directly into the first 32 locations of the user Data Space. Although not being physically implemented as SRAM locations, this memory organization provides great flexibility in access of the registers, as the X-, Y-, and Z-pointer registers can be set to index any register in the file.

11.4.1. The X-register, Y-register, and Z-register

The registers R26...R31 have some added functions to their general purpose usage. These registers are 16-bit address pointers for indirect addressing of the data space. The three indirect address registers X, Y, and Z are defined as described in the figure.

Figure 11-3. The X-, Y-, and Z-registers

In the different addressing modes these address registers have functions as fixed displacement, automatic increment, and automatic decrement (see the instruction set reference for details).

Related Links

Instruction Set Summary on page 432

11.5. Stack Pointer

The Stack is mainly used for storing temporary data, for storing local variables and for storing return addresses after interrupts and subroutine calls. The Stack is implemented as growing from higher to lower memory locations. The Stack Pointer Register always points to the top of the Stack.

The Stack Pointer points to the data SRAM Stack area where the Subroutine and Interrupt Stacks are located. A Stack PUSH command will decrease the Stack Pointer. The Stack in the data SRAM must be defined by the program before any subroutine calls are executed or interrupts are enabled. Initial Stack Pointer value equals the last address of the internal SRAM and the Stack Pointer must be set to point above start of the SRAM. See the table for Stack Pointer details.

Table 11-1. Stack Pointer Instructions

Instruction	Stack pointer	Description
PUSH	Decremented by 1	Data is pushed onto the stack
CALL ICALL RCALL	Decremented by 2	Return address is pushed onto the stack with a subroutine call or interrupt
POP	Incremented by 1	Data is popped from the stack
RET RETI	Incremented by 2	Return address is popped from the stack with return from subroutine or return from interrupt

The AVR Stack Pointer is implemented as two 8-bit registers in the I/O space. The number of bits actually used is implementation dependent. Note that the data space in some implementations of the AVR architecture is so small that only SPL is needed. In this case, the SPH Register will not be present.

11.5.1. Stack Pointer Register High byte

When using the I/O specific commands IN and OUT, the I/O addresses 0x00 - 0x3F must be used. When addressing I/O Registers as data space using LD and ST instructions, 0x20 must be added to these offset addresses.

Name: SPH
Offset: 0x5E
Reset: RAMEND
Property: When addressing I/O Registers as data space the offset address is 0x3E

Bit	7	6	5	4	3	2	1	0
						(SP[10:8]) SPH		
Access						RW	RW	RW
Reset						0	0	0

Bits 2:0 – (SP[10:8]) SPH: Stack Pointer Register
SPH and SPL are combined into SP. It means SPH[2:0] is SP[10:8].

11.5.2. Stack Pointer Register Low byte

When using the I/O specific commands IN and OUT, the I/O addresses 0x00 - 0x3F must be used. When addressing I/O Registers as data space using LD and ST instructions, 0x20 must be added to these offset addresses.

Name: SPL
Offset: 0x5D
Reset: 0x11111111
Property: When addressing I/O Registers as data space the offset address is 0x3D

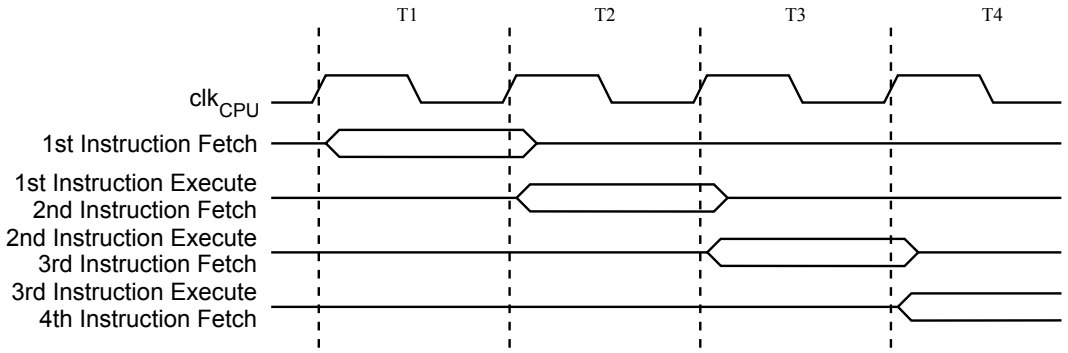
Bit	7	6	5	4	3	2	1	0
	(SP[7:0]) SPL							
Access	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	1

Bits 7:0 – (SP[7:0]) SPL: Stack Pointer Register
SPH and SPL are combined into SP. It means SPL[7:0] is SP[7:0].

11.6. Instruction Execution Timing

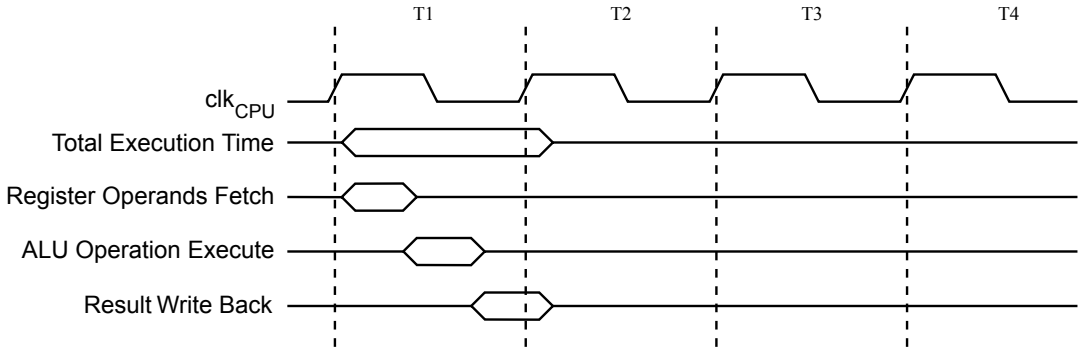
This section describes the general access timing concepts for instruction execution. The AVR CPU is driven by the CPU clock clk_{CPU}, directly generated from the selected clock source for the chip. No internal clock division is used. The Figure below shows the parallel instruction fetches and instruction executions enabled by the Harvard architecture and the fast-access Register File concept. This is the basic pipelining concept to obtain up to 1 MIPS per MHz with the corresponding unique results for functions per cost, functions per clocks, and functions per power-unit.

Figure 11-4. The Parallel Instruction Fetches and Instruction Executions



The following Figure shows the internal timing concept for the Register File. In a single clock cycle an ALU operation using two register operands is executed, and the result is stored back to the destination register.

Figure 11-5. Single Cycle ALU Operation



11.7. Reset and Interrupt Handling

The AVR provides several different interrupt sources. These interrupts and the separate Reset Vector each have a separate program vector in the program memory space. All interrupts are assigned individual enable bits which must be written logic one together with the Global Interrupt Enable bit in the Status Register in order to enable the interrupt. Depending on the Program Counter value, interrupts may be automatically disabled when Boot Lock bits BLB02 or BLB12 are programmed. This feature improves software security.

The lowest addresses in the program memory space are by default defined as the Reset and Interrupt Vectors. They have determined priority levels: The lower the address the higher is the priority level. RESET has the highest priority, and next is INT0 – the External Interrupt Request 0. The Interrupt Vectors can be moved to the start of the Boot Flash section by setting the IVSEL bit in the MCU Control Register (MCUCR). The Reset Vector can also be moved to the start of the Boot Flash section by programming the BOOTRST Fuse.

When an interrupt occurs, the Global Interrupt Enable I-bit is cleared and all interrupts are disabled. The user software can write logic one to the I-bit to enable nested interrupts. All enabled interrupts can then interrupt the current interrupt routine. The I-bit is automatically set when a Return from Interrupt instruction – RETI – is executed.

There are basically two types of interrupts:

The first type is triggered by an event that sets the Interrupt Flag. For these interrupts, the Program Counter is vectored to the actual Interrupt Vector in order to execute the interrupt handling routine, and hardware clears the corresponding Interrupt Flag. Interrupt Flags can also be cleared by writing a logic one to the flag bit position(s) to be cleared. If an interrupt condition occurs while the corresponding interrupt enable bit is cleared, the Interrupt Flag will be set and remembered until the interrupt is enabled, or the flag is cleared by software. Similarly, if one or more interrupt conditions occur while the Global Interrupt Enable bit is cleared, the corresponding Interrupt Flag(s) will be set and remembered until the Global Interrupt Enable bit is set, and will then be executed by order of priority.

The second type of interrupts will trigger as long as the interrupt condition is present. These interrupts do not necessarily have Interrupt Flags. If the interrupt condition disappears before the interrupt is enabled, the interrupt will not be triggered. When the AVR exits from an interrupt, it will always return to the main program and execute one more instruction before any pending interrupt is served.

The Status Register is not automatically stored when entering an interrupt routine, nor restored when returning from an interrupt routine. This must be handled by software.

When using the CLI instruction to disable interrupts, the interrupts will be immediately disabled. No interrupt will be executed after the CLI instruction, even if it occurs simultaneously with the CLI instruction.

The following example shows how this can be used to avoid interrupts during the timed EEPROM write sequence.

Assembly Code Example

```
in r16, SREG ; store SREG value
cli ; disable interrupts during timed sequence
sbi EECR, EEMPE ; start EEPROM write
sbi EECR, EEPE
out SREG, r16 ; restore SREG value (I-bit)
```

C Code Example

```
char cSREG;
cSREG = SREG; /* store SREG value */
/* disable interrupts during timed sequence */
_CLI();
EECR |= (1<<EEMPE); /* start EEPROM write */
EECR |= (1<<EEPE);
SREG = cSREG; /* restore SREG value (I-bit) */
```

Note: Please refer to *About Code Examples*.

When using the SEI instruction to enable interrupts, the instruction following SEI will be executed before any pending interrupts, as shown in this example.

Assembly Code Example

```
sei ; set Global Interrupt Enable
sleep ; enter sleep, waiting for interrupt
; note: will enter sleep before any pending interrupt(s)
```

C Code Example

```
__enable_interrupt(); /* set Global Interrupt Enable */
__sleep(); /* enter sleep, waiting for interrupt */
/* note: will enter sleep before any pending interrupt(s) */
```

Note: Please refer to *About Code Examples*.

Related Links

Memory Programming on page 347
Boot Loader Support – Read-While-Write Self-Programming on page 330

11.7.1. Interrupt Response Time

The interrupt execution response for all the enabled AVR interrupts is four clock cycles minimum. After four clock cycles the program vector address for the actual interrupt handling routine is executed. During this four clock cycle period, the Program Counter is pushed onto the Stack. The vector is normally a jump to the interrupt routine, and this jump takes three clock cycles. If an interrupt occurs during execution of a multi-cycle instruction, this instruction is completed before the interrupt is served. If an interrupt occurs when the MCU is in sleep mode, the interrupt execution response time is increased by four clock cycles. This increase comes in addition to the start-up time from the selected sleep mode. A return from an interrupt handling routine takes four clock cycles. During these four clock cycles, the Program Counter (two bytes) is popped back from the Stack, the Stack Pointer is incremented by two, and the I-bit in SREG is set.

12. AVR Memories

12.1. Overview

This section describes the different memory types in the device. The AVR architecture has two main memory spaces, the Data Memory and the Program Memory space. In addition, the device features an EEPROM Memory for data storage. All memory spaces are linear and regular.

12.2. In-System Reprogrammable Flash Program Memory

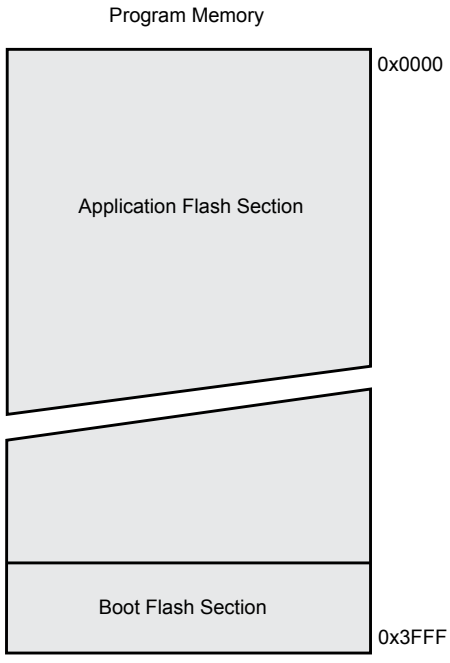
The ATmega328/P contains 32Kbytes On-chip In-System Reprogrammable Flash memory for program storage. Since all AVR instructions are 16 or 32 bits wide, the Flash is organized as 16K x 16. For software security, the Flash Program memory space is divided into two sections - Boot Loader Section and Application Program Section in the device .

The Flash memory has an endurance of at least 10,000 write/erase cycles. The ATmega328/P Program Counter (PC) is 14 bits wide, thus addressing the 16K program memory locations. The operation of Boot Program section and associated Boot Lock bits for software protection are described in detail in *Boot Loader Support – Read-While-Write Self-Programming*. Refer to *Memory Programming* for the description on Flash data serial downloading using the SPI pins.

Constant tables can be allocated within the entire program memory address space, using the Load Program Memory (LPM) instruction.

Timing diagrams for instruction fetch and execution are presented in *Instruction Exectution Timing*.

Figure 12-1. Program Memory Map ATmega328/P



Related Links

BTLDR - Boot Loader Support – Read-While-Write Self-Programming on page 330

MEMPROG- Memory Programming on page 347

Instruction Execution Timing on page 31

12.3. SRAM Data Memory

The following figure shows how the device SRAM Memory is organized.

The device is a complex microcontroller with more peripheral units than can be supported within the 64 locations reserved in the Opcode for the IN and OUT instructions. For the Extended I/O space from 0x60 - 0xFF in SRAM, only the ST/STS/STD and LD/LDS/LDD instructions can be used.

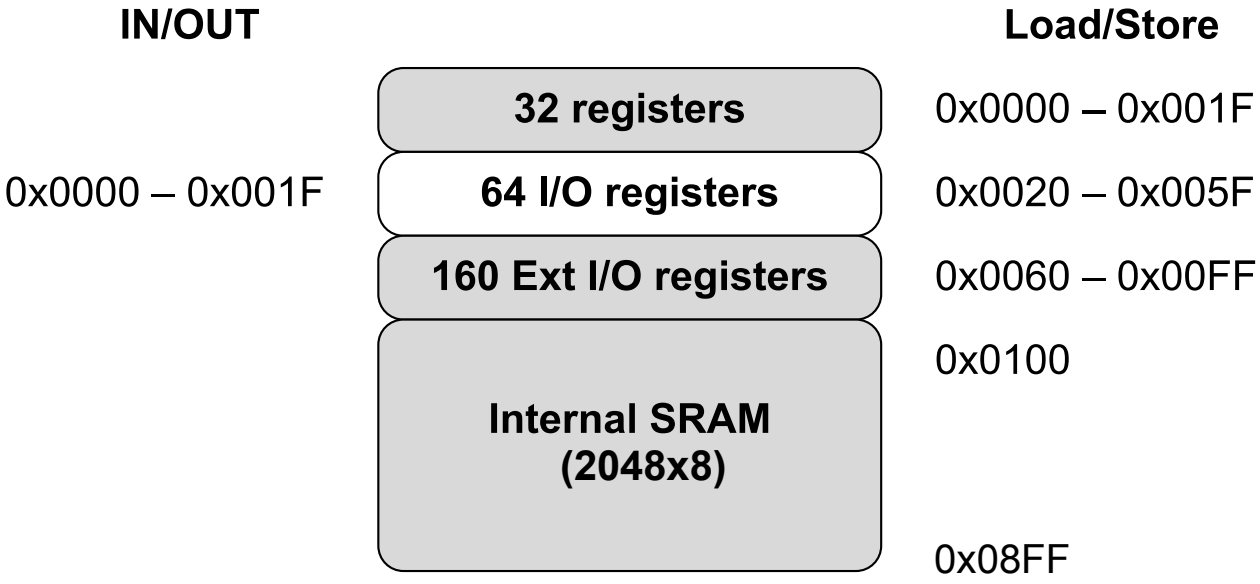
The lower 2303 data memory locations address both the Register File, the I/O memory, Extended I/O memory, and the internal data SRAM. The first 32 locations address the Register File, the next 64 location the standard I/O memory, then 160 locations of Extended I/O memory, and the next 2K locations address the internal data SRAM.

The five different addressing modes for the data memory cover:

- Direct
  - The direct addressing reaches the entire data space.
- Indirect with Displacement
  - The Indirect with Displacement mode reaches 63 address locations from the base address given by the Y- or Z-register.
- Indirect
  - In the Register File, registers R26 to R31 feature the indirect addressing pointer registers.
- Indirect with Pre-decrement
  - The address registers X, Y, and Z are decremented.
- Indirect with Post-increment
  - The address registers X, Y, and Z are incremented.

The 32 general purpose working registers, 64 I/O Registers, 160 Extended I/O Registers, and the 2K bytes of internal data SRAM in the device are all accessible through all these addressing modes.

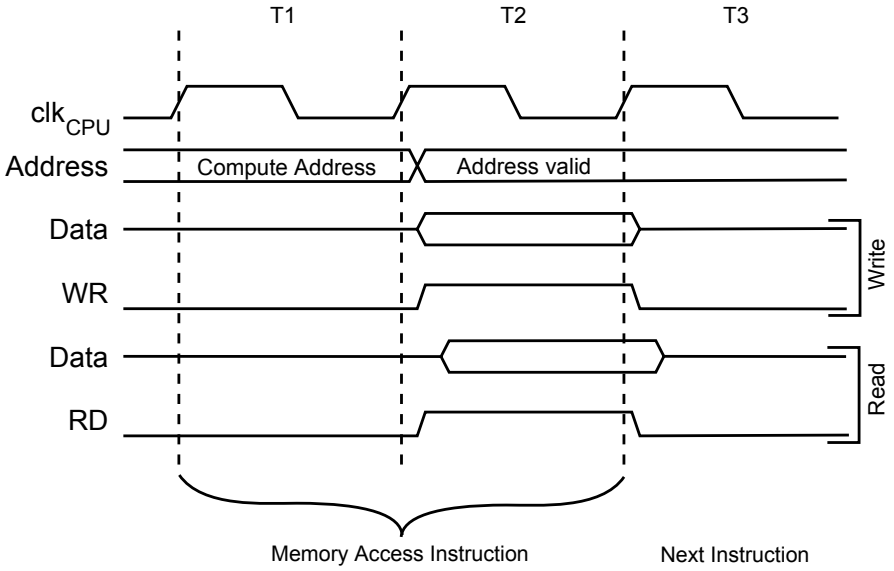
Figure 12-2. Data Memory Map with 2048 byte internal data SRAM



12.3.1. Data Memory Access Times

The internal data SRAM access is performed in two clk_{CPU} cycles as described in the following Figure.

Figure 12-3. On-chip Data SRAM Access Cycles



12.4. EEPROM Data Memory

The ATmega328/P contains 1K bytes of data EEPROM memory. It is organized as a separate data space, in which single bytes can be read and written. The EEPROM has an endurance of at least 100,000 write/erase cycles. The access between the EEPROM and the CPU is described in the following, specifying the EEPROM Address Registers, the EEPROM Data Register, and the EEPROM Control Register.

See the related links for a detailed description on EEPROM Programming in SPI or Parallel Programming mode.

Related Links

MEMPROG- Memory Programming on page 347

12.4.1. EEPROM Read/Write Access

The EEPROM Access Registers are accessible in the I/O space.

The write access time for the EEPROM is given in Table 12-2. A self-timing function, however, lets the user software detect when the next byte can be written. If the user code contains instructions that write the EEPROM, some precautions must be taken. In heavily filtered power supplies, V_{CC} is likely to rise or fall slowly on power-up/down. This causes the device for some period of time to run at a voltage lower than specified as minimum for the clock frequency used. Please refer to Preventing EEPROM Corruption for details on how to avoid problems in these situations.

In order to prevent unintentional EEPROM writes, a specific write procedure must be followed. Refer to the description of the EEPROM Control Register for details on this.

When the EEPROM is read, the CPU is halted for four clock cycles before the next instruction is executed. When the EEPROM is written, the CPU is halted for two clock cycles before the next instruction is executed.

12.4.2. Preventing EEPROM Corruption

During periods of low $V_{CC,}$ the EEPROM data can be corrupted because the supply voltage is too low for the CPU and the EEPROM to operate properly. These issues are the same as for board level systems using EEPROM, and the same design solutions should be applied.

An EEPROM data corruption can be caused by two situations when the voltage is too low. First, a regular write sequence to the EEPROM requires a minimum voltage to operate correctly. Secondly, the CPU itself can execute instructions incorrectly, if the supply voltage is too low.

EEPROM data corruption can easily be avoided by following this design recommendation:

Keep the AVR $\overline{\text{RESET}}$ active (low) during periods of insufficient power supply voltage. This can be done by enabling the internal Brown-out Detector (BOD). If the detection level of the internal BOD does not match the needed detection level, an external low V_{CC} reset Protection circuit can be used. If a reset occurs while a write operation is in progress, the write operation will be completed provided that the power supply voltage is sufficient.

12.5. I/O Memory

The I/O space definition of the device is shown in the *Register Summary*.

All device I/Os and peripherals are placed in the I/O space. All I/O locations may be accessed by the LD/LDS/LDD and ST/STS/STD instructions, transferring data between the 32 general purpose working registers and the I/O space. I/O Registers within the address range 0x00-0x1F are directly bit-accessible using the SBI and CBI instructions. In these registers, the value of single bits can be checked by using the SBIS and SBIC instructions.

When using the I/O specific commands IN and OUT, the I/O addresses 0x00-0x3F must be used. When addressing I/O Registers as data space using LD and ST instructions, 0x20 must be added to these addresses. The device is a complex microcontroller with more peripheral units than can be supported within the 64 location reserved in Opcode for the IN and OUT instructions. For the Extended I/O space from 0x60..0xFF in SRAM, only the ST/STS/STD and LD/LDS/LDD instructions can be used.

For compatibility with future devices, reserved bits should be written to zero if accessed. Reserved I/O memory addresses should never be written.

Some of the Status Flags are cleared by writing a '1' to them; this is described in the flag descriptions. Note that, unlike most other AVRs, the CBI and SBI instructions will only operate on the specified bit, and can therefore be used on registers containing such Status Flags. The CBI and SBI instructions work with registers 0x00-0x1F only.

The I/O and Peripherals Control Registers are explained in later sections.

Related Links

MEMPROG- Memory Programming on page 347
Register Summary on page 428
Instruction Set Summary on page 432

12.5.1. General Purpose I/O Registers

The device contains three General Purpose I/O Registers, General Purpose I/O Register 0/1/2 (GPIOR 0/1/2). These registers can be used for storing any information, and they are particularly useful for storing global variables and Status Flags. General Purpose I/O Registers within the address range 0x00 - 0x1F are directly bit-accessible using the SBI, CBI, SBIS, and SBIC instructions.

12.6. Register Description

12.6.1. EEPROM Address Register High

When addressing I/O Registers as data space using LD and ST instructions, the provided offset must be used. When using the I/O specific commands IN and OUT, the offset is reduced by 0x20, resulting in an I/O address offset within 0x00 - 0x3F.

Name: EEARH
Offset: 0x42
Reset: 0x0X
Property: When addressing as I/O Register: address offset is 0x22

Bit	7	6	5	4	3	2	1	0
							EEAR9	EEAR8
Access							R/W	R/W
Reset							x	x

Bit 1 – EEAR9: EEPROM Address 9
Refer to EEARL.

Bit 0 – EEAR8: EEPROM Address 8
Refer to EEARL.

12.6.2. EEPROM Address Register Low

When addressing I/O Registers as data space using LD and ST instructions, the provided offset must be used. When using the I/O specific commands IN and OUT, the offset is reduced by 0x20, resulting in an I/O address offset within 0x00 - 0x3F.

Name: EEARL
Offset: 0x41
Reset: 0xXX
Property: When addressing as I/O Register: address offset is 0x21

Bit	7	6	5	4	3	2	1	0
	EEAR7	EEAR6	EEAR5	EEAR4	EEAR3	EEAR2	EEAR1	EEAR0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	x	x	x	x	x	x	x	x

Bits 7:0 – EEARn: EEPROM Address

The EEPROM Address Registers – EEARH and EEARL specify the EEPROM address in the 1K Bytes EEPROM space. The EEPROM data bytes are addressed linearly between 0 and 255/511/511. The initial value of EEAR is undefined. A proper value must be written before the EEPROM may be accessed.

12.6.3. EEPROM Data Register

When addressing I/O Registers as data space using LD and ST instructions, the provided offset must be used. When using the I/O specific commands IN and OUT, the offset is reduced by 0x20, resulting in an I/O address offset within 0x00 - 0x3F.

Name: EEDR
Offset: 0x40
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x20

Bit	7	6	5	4	3	2	1	0
	EEDR[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – EEDR[7:0]: EEPROM Data

For the EEPROM write operation, the EEDR Register contains the data to be written to the EEPROM in the address given by the EEAR Register. For the EEPROM read operation, the EEDR contains the data read out from the EEPROM at the address given by EEAR.

12.6.4. EEPROM Control Register

When addressing I/O Registers as data space using LD and ST instructions, the provided offset must be used. When using the I/O specific commands IN and OUT, the offset is reduced by 0x20, resulting in an I/O address offset within 0x00 - 0x3F.

Name: EECR
Offset: 0x3F
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x1F

Bit	7	6	5	4	3	2	1	0
			EEPM1	EEPM0	EERIE	EEMPE	EEPE	EERE
Access			R/W	R/W	R/W	R/W	R/W	R/W
Reset			x	x	0	0	x	0

Bits 5:4 – EEPMn: EEPROM Programming Mode Bits [n = 1:0]

The EEPROM Programming mode bit setting defines which programming action that will be triggered when writing EEPE. It is possible to program data in one atomic operation (erase the old value and program the new value) or to split the Erase and Write operations in two different operations. The Programming times for the different modes are shown in the table below. While EEPE is set, any write to EEPMn will be ignored. During reset, the EEPMn bits will be reset to 0b00 unless the EEPROM is busy programming.

Table 12-1. EEPROM Mode Bits

EEPM[1:0]	Programming Time	Operation
00	3.4ms	Erase and Write in one operation (Atomic Operation)
01	1.8ms	Erase Only
10	1.8ms	Write Only
11	-	Reserved for future use

Bit 3 – EERIE: EEPROM Ready Interrupt Enable

Writing EERIE to one enables the EEPROM Ready Interrupt if the I bit in SREG is set. Writing EERIE to zero disables the interrupt. The EEPROM Ready interrupt generates a constant interrupt when EEPE is cleared. The interrupt will not be generated during EEPROM write or SPM.

Bit 2 – EEMPE: EEPROM Master Write Enable

The EEMPE bit determines whether writing EEPE to '1' causes the EEPROM to be written.
When EEMPE is '1', setting EEPE within four clock cycles will write data to the EEPROM at the selected address.

If EEMPE is zero, setting EEPE will have no effect. When EEMPE has been written to '1' by software, hardware clears the bit to zero after four clock cycles. See the description of the EEPE bit for an EEPROM write procedure.

Bit 1 – EEPE: EEPROM Write Enable

The EEPROM Write Enable Signal EEPE is the write strobe to the EEPROM. When address and data are correctly set up, the EEPE bit must be written to '1' to write the value into the EEPROM. The EEMPE bit

must be written to '1' before EEPE is written to '1', otherwise no EEPROM write takes place. The following procedure should be followed when writing the EEPROM (the order of steps 3 and 4 is not essential):

1. Wait until EEPE becomes zero.
2. Wait until SPMEN in SPMCSR becomes zero.
3. Write new EEPROM address to EEAR (optional).
4. Write new EEPROM data to EEDR (optional).
5. Write a '1' to the EEMPE bit while writing a zero to EEPE in EECR.
6. Within four clock cycles after setting EEMPE, write a '1' to EEPE.

The EEPROM can not be programmed during a CPU write to the Flash memory. The software must check that the Flash programming is completed before initiating a new EEPROM write. Step 2 is only relevant if the software contains a Boot Loader allowing the CPU to program the Flash. If the Flash is never being updated by the CPU, step 2 can be omitted.

Caution:
An interrupt between step 5 and step 6 will make the write cycle fail, since the EEPROM Master Write Enable will time-out. If an interrupt routine accessing the EEPROM is interrupting another EEPROM access, the EEAR or EEDR Register will be modified, causing the interrupted EEPROM access to fail. It is recommended to have the Global Interrupt Flag cleared during all the steps to avoid these problems.

When the write access time has elapsed, the EEPE bit is cleared by hardware. The user software can poll this bit and wait for a zero before writing the next byte. When EEPE has been set, the CPU is halted for two cycles before the next instruction is executed.

Bit 0 – EERE: EEPROM Read Enable

The EEPROM Read Enable Signal EERE is the read strobe to the EEPROM. When the correct address is set up in the EEAR Register, the EERE bit must be written to a '1' to trigger the EEPROM read. The EEPROM read access takes one instruction, and the requested data is available immediately. When the EEPROM is read, the CPU is halted for four cycles before the next instruction is executed.

The user should poll the EEPE bit before starting the read operation. If a write operation is in progress, it is neither possible to read the EEPROM, nor to change the EEAR Register.

The calibrated Oscillator is used to time the EEPROM accesses. See the following table for typical programming times for EEPROM access from the CPU.

Table 12-2. EEPROM Programming Time

Symbol	Number of Calibrated RC Oscillator Cycles	Typ. Programming Time
EEPROM write (from CPU)	26,368	3.3ms

The following code examples show one assembly and one C function for writing to the EEPROM. The examples assume that interrupts are controlled (e.g. by disabling interrupts globally) so that no interrupts will occur during execution of these functions. The examples also assume that no Flash Boot Loader is present in the software. If such code is present, the EEPROM write function must also wait for any ongoing SPM command to finish.

Assembly Code Example[1]

```
EEPROM_write:
      ; Wait for completion of previous write
      sbic     EECR,EEPE
```

```
      rjmp     EEPROM_write
      ; Set up address (r18:r17) in address register
      out     EEARH, r18
      out     EEARL, r17
      ; Write data (r16) to Data Register
      out     EEDR,r16
      ; Write logical one to EEMPE
      sbi     EECR,EEMPE
      ; Start eeprom write by setting EEPE
      sbi     EECR,EEPE
      ret
```

C Code Example(1)

```
void EEPROM_write(unsigned int uiAddress, unsigned char ucData)
{
    /* Wait for completion of previous write */
    while(EECR & (1<<EEPE))
    ;
    /* Set up address and Data Registers */
    EEAR = uiAddress;
    EEDR = ucData;
    /* Write logical one to EEMPE */
    EECR |= (1<<EEMPE);
    /* Start eeprom write by setting EEPE */
    EECR |= (1<<EEPE);
}
```

Note: (1) Please refer to *About Code Examples*

The next code examples show assembly and C functions for reading the EEPROM. The examples assume that interrupts are controlled so that no interrupts will occur during execution of these functions.

Assembly Code Example(1)

```
EEPROM_read:
   ; Wait for completion of previous write
   sbic     EECR,EEPE
   rjmp     EEPROM_read
   ; Set up address (r18:r17) in address register
   out     EEARH, r18
   out     EEARL, r17
   ; Start eeprom read by writing EERE
   sbi     EECR,EERE
   ; Read data from Data Register
   in      r16,EEDR
   ret
```

C Code Example(1)

```
unsigned char EEPROM_read(unsigned int uiAddress)
{
    /* Wait for completion of previous write */
    while(EECR & (1<<EEPE))
    ;
    /* Set up address register */
    EEAR = uiAddress;
    /* Start eeprom read by writing EERE */
    EECR |= (1<<EERE);
    /* Return data from Data Register */
    return EEDR;
}
```

Note: (1) Please refer to *About Code Examples*

12.6.5. GPIOR2 – General Purpose I/O Register 2

When addressing I/O Registers as data space using LD and ST instructions, the provided offset must be used. When using the I/O specific commands IN and OUT, the offset is reduced by 0x20, resulting in an I/O address offset within 0x00 - 0x3F.

Name: GPIOR2
Offset: 0x4B
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x2B

Bit	7	6	5	4	3	2	1	0
	GPIOR2[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – GPIOR2[7:0]: General Purpose I/O

12.6.6. GPIOR1 – General Purpose I/O Register 1

When addressing I/O Registers as data space using LD and ST instructions, the provided offset must be used. When using the I/O specific commands IN and OUT, the offset is reduced by 0x20, resulting in an I/O address offset within 0x00 - 0x3F.

Name: GPIOR1
Offset: 0x4A
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x2A

Bit	7	6	5	4	3	2	1	0
	GPIOR1[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – GPIOR1[7:0]: General Purpose I/O

12.6.7. GPIOR0 – General Purpose I/O Register 0

When addressing I/O Registers as data space using LD and ST instructions, the provided offset must be used. When using the I/O specific commands IN and OUT, the offset is reduced by 0x20, resulting in an I/O address offset within 0x00 - 0x3F.

Name: GPIOR0
Offset: 0x3E
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x1E

Bit	7	6	5	4	3	2	1	0
	GPIOR0[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – GPIOR0[7:0]: General Purpose I/O

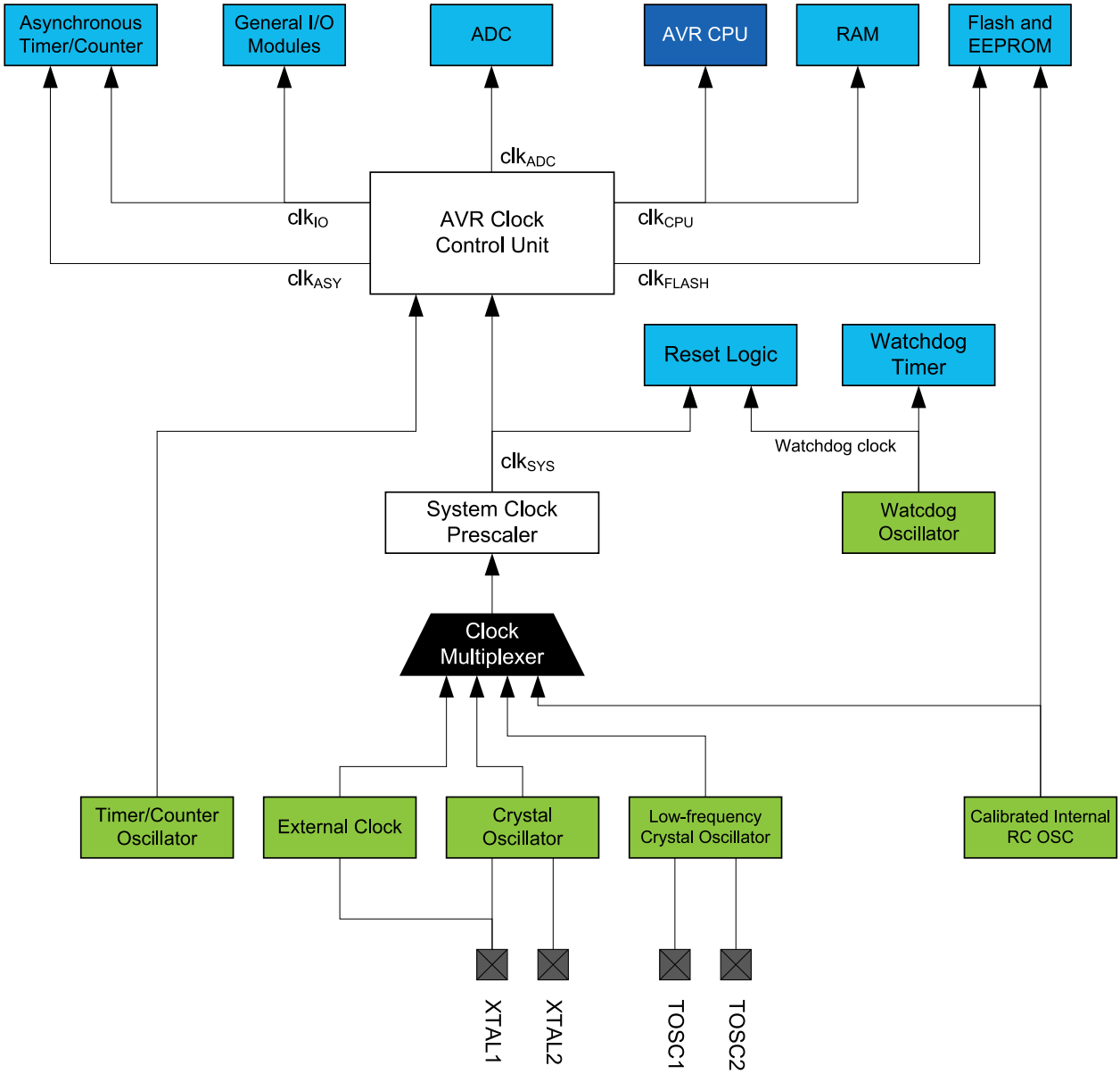
13. System Clock and Clock Options

13.1. Clock Systems and Their Distribution

The following figure illustrates the principal clock systems in the device and their distribution. All the clocks need not be active at a given time. In order to reduce power consumption, the clocks to modules not being used can be halted by using different sleep modes. The clock systems are described in the following sections.

The system clock frequency refers to the frequency generated from the System Clock Prescaler. All clock outputs from the AVR Clock Control Unit runs in the same frequency.

Figure 13-1. Clock Distribution



13.1.1. CPU Clock – clk_{CPU}

The CPU clock is routed to parts of the system concerned with operation of the AVR core. Examples of such modules are the General Purpose Register File, the Status Register and the data memory holding the Stack Pointer. Halting the CPU clock inhibits the core from performing general operations and calculations.

13.1.2. I/O Clock – $clk_{I/O}$

The I/O clock is used by the majority of the I/O modules, like Timer/Counters, SPI, and USART. The I/O clock is also used by the External Interrupt module, but the start condition detection in the USI module is carried out asynchronously when $clk_{I/O}$ is halted, TWI address recognition in all sleep modes.

Note: If a level triggered interrupt is used for wake-up from Power-down, the required level must be held long enough for the MCU to complete the wake-up to trigger the level interrupt. If the level disappears before the end of the Start-up Time, the MCU will still wake up, but no interrupt will be generated. The start-up time is defined by the SUT and CKSEL Fuses.

13.1.3. Flash Clock – clk_{FLASH}

The Flash clock controls operation of the Flash interface. The Flash clock is usually active simultaneously with the CPU clock.

13.1.4. Asynchronous Timer Clock – clk_{ASY}

The Asynchronous Timer clock allows Asynchronous Timer/Counters to be clocked directly from an external clock or an external 32kHz clock crystal. The dedicated clock domain allows using this Timer/Counter as a real-time counter even when the device is in sleep mode.

13.1.5. ADC Clock – clk_{ADC}

The ADC is provided with a dedicated clock domain. This allows halting the CPU and I/O clocks in order to reduce noise generated by digital circuitry. This gives more accurate ADC conversion results.

13.2. Clock Sources

The device has the following clock source options, selectable by Flash Fuse bits as shown below. The clock from the selected source is input to the AVR clock generator, and routed to the appropriate modules.

Table 13-1. Device Clocking Options Select

Device Clocking Option	CKSEL[3:0]
Low Power Crystal Oscillator	1111 - 1000
Full Swing Crystal Oscillator	0111 - 0110
Low Frequency Crystal Oscillator	0101 - 0100
Internal 128kHz RC Oscillator	0011
Calibrated Internal RC Oscillator	0010
External Clock	0000
Reserved	0001

Note: For all fuses, '1' means unprogrammed while '0' means programmed.

13.2.1. Default Clock Source

The device is shipped with internal RC oscillator at 8.0MHz and with the fuse CKDIV8 programmed, resulting in 1.0MHz system clock. The startup time is set to maximum, and the time-out period is enabled: CKSEL=0010, SUT=10, CKDIV8=0. This default setting ensures that all users can make their desired clock source setting using any available programming interface.

13.2.2. Clock Startup Sequence

Any clock source needs a sufficient V_{CC} to start oscillating and a minimum number of oscillating cycles before it can be considered stable.

To ensure sufficient V_{CC}, the device issues an internal reset with a time-out delay (t_{TOUT}) after the device reset is released by all other reset sources. See the Related Links for a description of the start conditions for the internal reset. The delay (t_{TOUT}) is timed from the Watchdog Oscillator and the number of cycles in the delay is set by the SUTx and CKSELx fuse bits. The selectable delays are shown in the Table below. The frequency of the Watchdog Oscillator is voltage dependent.

Table 13-2. Number of Watchdog Oscillator Cycles

Typ. Time-out (V_{CC} = 5.0V)	Typ. Time-out (V_{CC} = 3.0V)	Number of Cycles
0ms	0ms	0
4.1ms	4.3ms	512
65ms	69ms	8K (8,192)

Main purpose of the delay is to keep the device in reset until it is supplied with minimum V_{CC}. The delay will not monitor the actual voltage, so it is required to select a delay longer than the V_{CC} rise time. If this is not possible, an internal or external Brown-Out Detection circuit should be used. A BOD circuit will ensure sufficient V_{CC} before it releases the reset, and the time-out delay can be disabled. Disabling the time-out delay without utilizing a Brown-Out Detection circuit is not recommended.

The oscillator is required to oscillate for a minimum number of cycles before the clock is considered stable. An internal ripple counter monitors the oscillator output clock, and keeps the internal reset active for a given number of clock cycles. The reset is then released and the device will start to execute. The recommended oscillator start-up time is dependent on the clock type, and varies from 6 cycles for an externally applied clock to 32K cycles for a low frequency crystal.

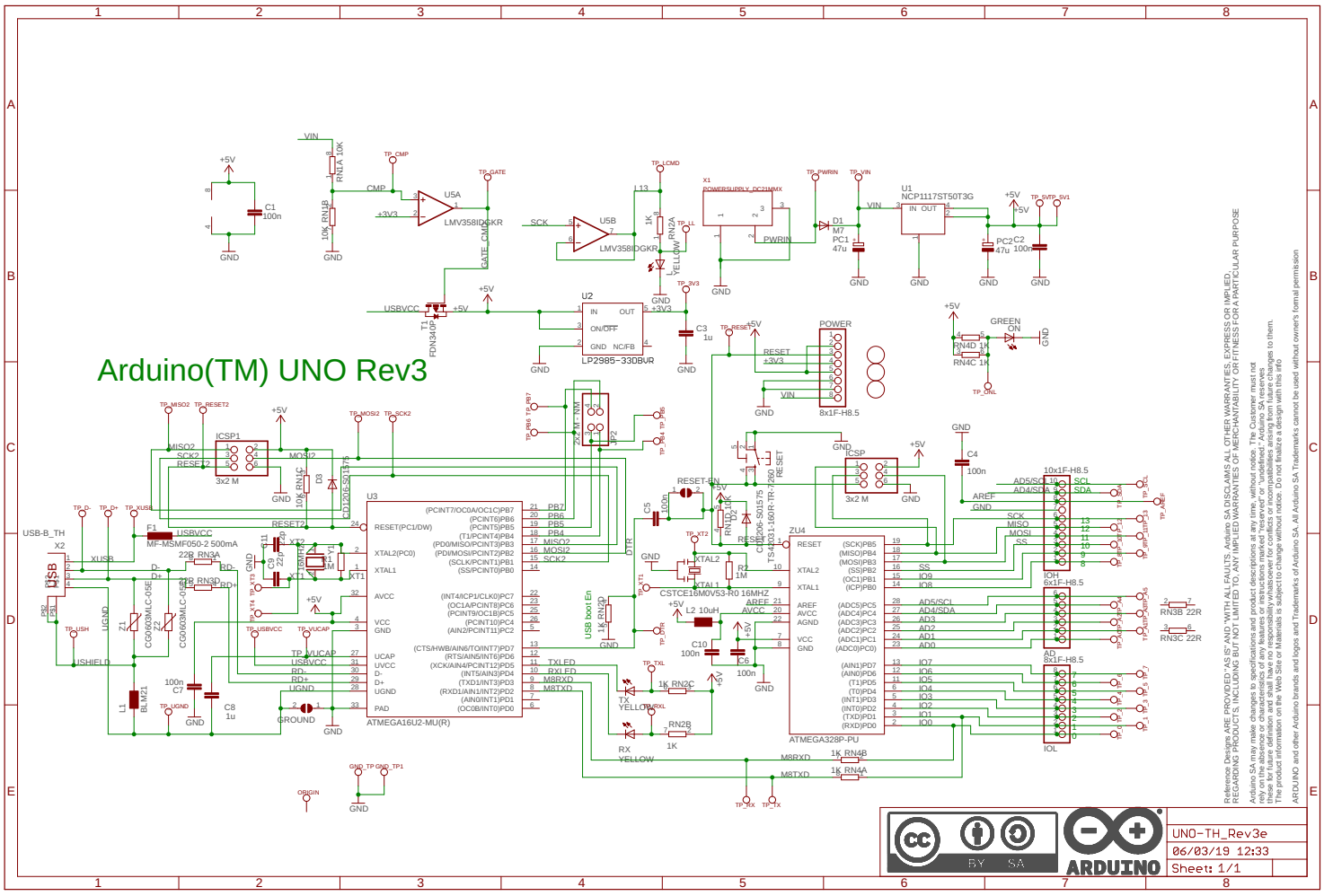
The start-up sequence for the clock includes both the time-out delay and the start-up time when the device starts up from reset. When starting up from Power-save or Power-down mode, V_{CC} is assumed to be at a sufficient level and only the start-up time is included.

13.2.3. Low Power Crystal Oscillator

Pins XTAL1 and XTAL2 are input and output, respectively, of an inverting amplifier which can be configured for use as an On-chip Oscillator, as shown in the Figure below. Either a quartz crystal or a ceramic resonator may be used.

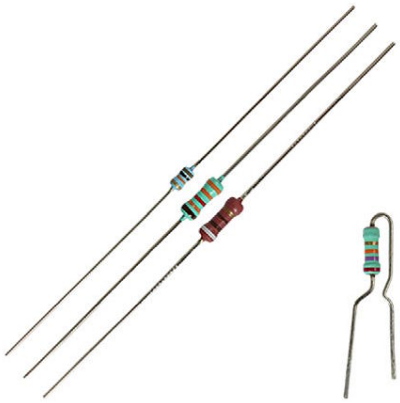
C1 and C2 should always be equal for both crystals and resonators. The optimal value of the capacitors depends on the crystal or resonator in use, the amount of stray capacitance, and the electromagnetic noise of the environment. Some initial guidelines for choosing capacitors for use with crystals are given in the next Table. For ceramic resonators, the capacitor values given by the manufacturer should be used.

5.3. Hojas de Pines del Arduino UNO



5.4. Hojas del fabricante de las resistencias

Standard Metal Film Leaded Resistors



FEATURES

- Small size (SFR16S: 0204, SFR25 / SFR25H: 0207)
- Low noise (max. 1.5 µV/V for $R > 1$ MΩ)
- Compatible to both lead (Pb)-free and lead containing soldering processes
- Material categorization: for definitions of compliance please see www.vishay.com/doc?99912

Pb-free

RoHS COMPLIANT

HALOGEN FREE

APPLICATIONS

- General purpose resistors

A homogeneous film of metal alloy is deposited on a high grade ceramic body. After a helical groove has been cut in the resistive layer, tinned connecting leads of electrolytic copper are welded to the end-caps.

The resistors are coated with a colored lacquer (light-blue for type SFR16S; light-green for type SFR25 and red-brown for type SFR25H) which provides electrical, mechanical, and climatic protection. The encapsulation is resistant to all cleaning solvents in accordance with IEC 60068-2-45.

TECHNICAL SPECIFICATIONS			
DESCRIPTION	**SFR16S**	**SFR25**	**SFR25H**
DIN size	0204	0207	0207
Resistance range	1 Ω to 3 MΩ; jumper (0 Ω)	0.22 Ω to 10 MΩ; jumper (0 Ω)	0.22 Ω to 10 MΩ
Resistance tolerance	± 5 %; ± 1 %		
Temperature coefficient	± 250 ppm/K; ± 100 ppm/K		
Rated dissipation, P_{70}	0.5 W	0.4 W	0.5 W
Thermal resistance	170 K/W	200 K/W	150 K/W
Operating voltage, $U_{max.}$ AC/DC	200 V	250 V	350 V
Operating temperature range	-55 °C to 155 °C		
Permissible film temperature	155 °C		
Max. resistance change at rated dissipation $\lvert\Delta R/R$ max.$\rvert$, after 1000 h	± (2 % R + 0.05 Ω)		

Note

- R value is measured with probe distance of 24 mm ± 1 mm using 4-terminal method.

TEMPERATURE COEFFICIENT AND RESISTANCE RANGE

TYPE	TOLERANCE	TCR	RESISTANCE	E-SERIES
SFR16S	± 5 %	± 250 ppm/K	1 Ω to ≤ 4.7 Ω	E24
		± 100 ppm/K	4.7 Ω to 100 kΩ	
		± 250 ppm/K	> 100 kΩ to 3 MΩ	
	± 1 %	± 100 ppm/K	5.6 Ω to 100 kΩ	E24; E96
		± 250 ppm/K	> 100 kΩ to 976 kΩ	
	Jumper (0 Ω)	-	≤ 30 mΩ; $I_{max.}$ = 3 A	-
SFR25, SFR25H	± 5 %	± 250 ppm/K	0.22 Ω to 4.7 Ω	E24
		± 100 ppm/K	> 4.7 Ω to 1 MΩ	
		± 250 ppm/K	> 1 MΩ to 10 MΩ	
	± 1 %	± 250 ppm/K	1 Ω to 4.7 Ω	E24; E96
		± 100 ppm/K	> 4.7 Ω to 1 MΩ	
		± 250 ppm/K	> 1 MΩ to 10 MΩ	
	Jumper (0 Ω) (1)	-	≤ 30 mΩ; $I_{max.}$ = 5 A	-

Note

(1) Jumper is only available for SFR25.

PART NUMBER AND PRODUCT DESCRIPTION

PART NUMBER: SFR2500001001FA500

S	F	R	2	5	0	0	0	0	1	0	0	1	F	A	5	0	0

TYPE	VARIANT	TCR/MATERIAL	RESISTANCE	TOLERANCE	PACKAGING	SPECIAL
SFR16S0 **SFR2500** **SFR25H0**	**0** = neutral **Z** = value overflow (special)	**0** = standard **Z** = jumper	**3 digit value** **1 digit multiplier** MULTIPLIER **7** = $*10^{-3}$ **2** = $*10^{2}$ **8** = $*10^{-2}$ **3** = $*10^{3}$ **9** = $*10^{-1}$ **4** = $*10^{4}$ **0** = $*10^{0}$ **5** = $*10^{5}$ **1** = $*10^{1}$ **Z** = 0000	**F** = ± 1 % **J** = ± 5 % **Z** = jumper	**N4** **A5** **A1** **R5**	The 2 digits are used for all special parts. **00** = standard

PRODUCT DESCRIPTION: SFR25 1 % A5 1K0

SFR25	1 %	A5	1K0
TYPE	TOLERANCE	PACKAGING (1)	RESISTANCE VALUE
SFR16S **SFR25** **SFR25H**	**± 1 %** **± 5 %**	**N4** **A5** **A1** **R5**	**47K** = 47 kΩ **51R1** = 51.1 Ω

Notes

- The products can be ordered using either the PRODUCT DESCRIPTION or the PART NUMBER.

(1) N4 packaging indicates SFR25 and SFR25H radial version.

PACKAGING						
TYPE	CODE	QUANTITY	PACKAGING STYLE	WIDTH	PITCH	DIMENSIONS
SFR16S	A5	5000	Taped acc. to IEC 60286-1 fan-folded in a box	52 mm	5 mm	75 mm x 73 mm x 270 mm
	R5	5000	Taped acc. to IEC 60286-1 on a reel			92 mm x 278 mm x 278 mm
	A1 (1)	1000	Taped acc. to IEC 60286-1 fan-folded in a box			75 mm x 28 mm x 262 mm
SFR25, SFR25H	A5	5000	Taped acc. to IEC 60286-1 fan-folded in a box	52 mm	5 mm	75 mm x 98 mm x 270 mm
	R5	5000	Taped acc. to IEC 60286-1 on a reel			93 mm x 300 mm x 298 mm
	A1 (1)	1000	Taped acc. to IEC 60286-1 fan-folded in a box			75 mm x 28 mm x 262 mm
	N4 (2)	4000	Taped acc. to IEC 60286-2 fan-folded in a box	-	12.7 mm	45 mm x 262 mm x 330 mm

Notes

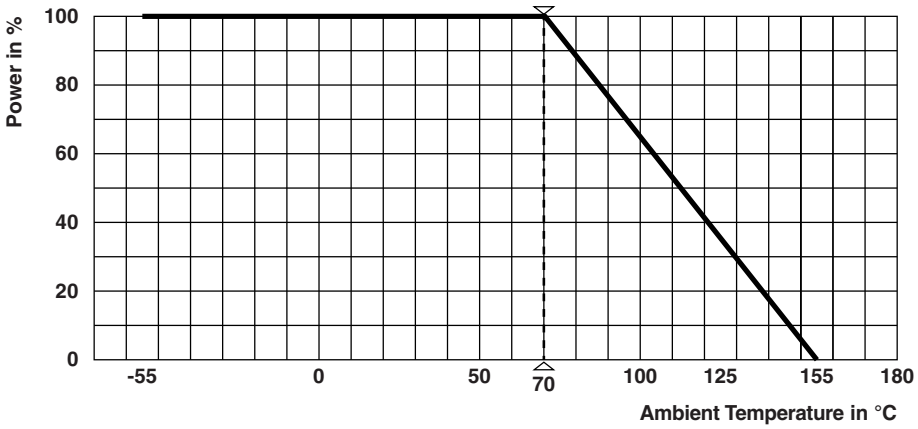
(1) A1 packaging only available for resistors with ± 5 % tolerance.

(2) N4 packaging only available for SFR25 and SFR25H radial version.

MARKING

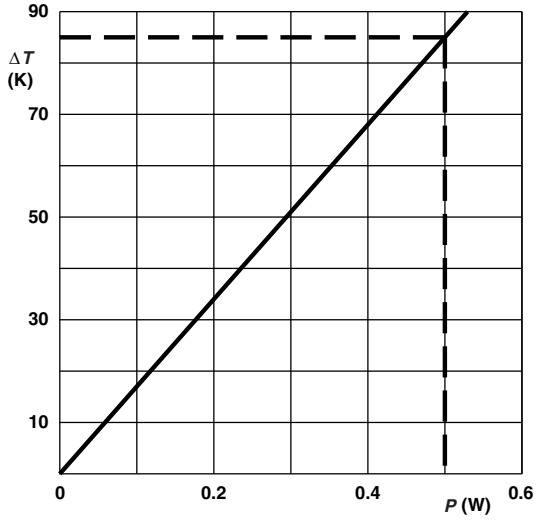
The nominal resistance and tolerance are marked on the resistor using four or five colored bands in accordance with IEC 60062, marking codes for resistors and capacitors.

FUNCTIONAL PERFORMANCE

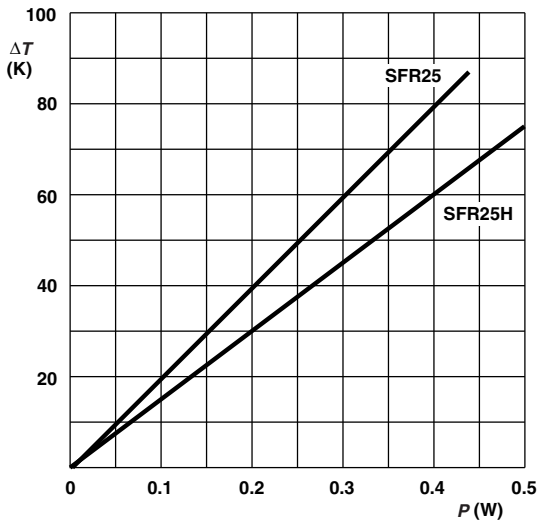


Derating

Maximum dissipation ($P_{max.}$) in percentage of rated power as a function of the ambient temperature (T_{amb})



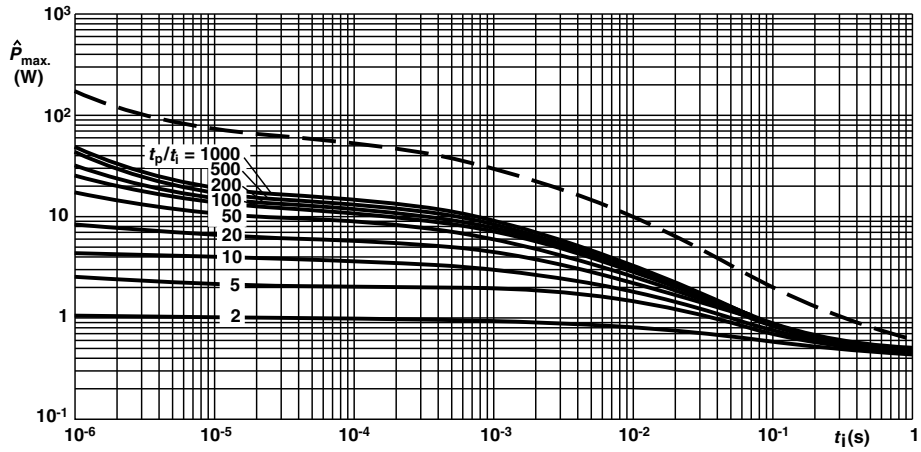
SFR16S Hot-spot temperature rise (ΔT) as a function of dissipated power



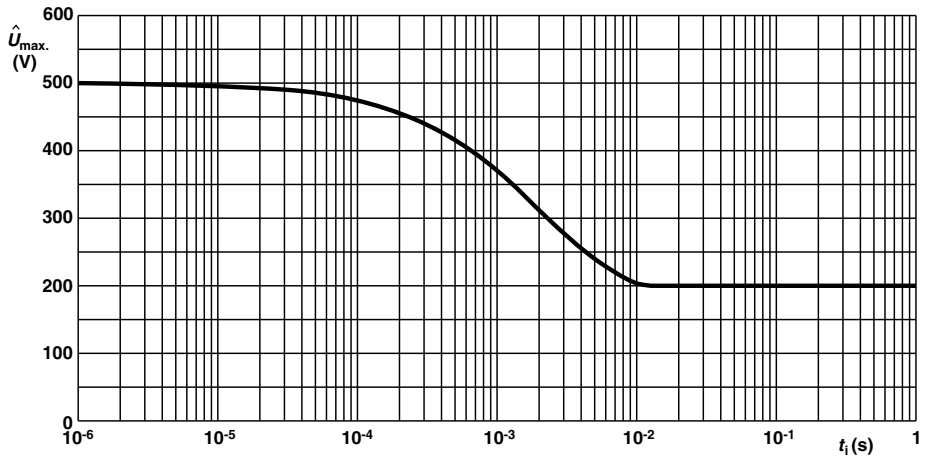
SFR25/SFR25H Hot-spot temperature rise (ΔT) as a function of dissipated power

Note

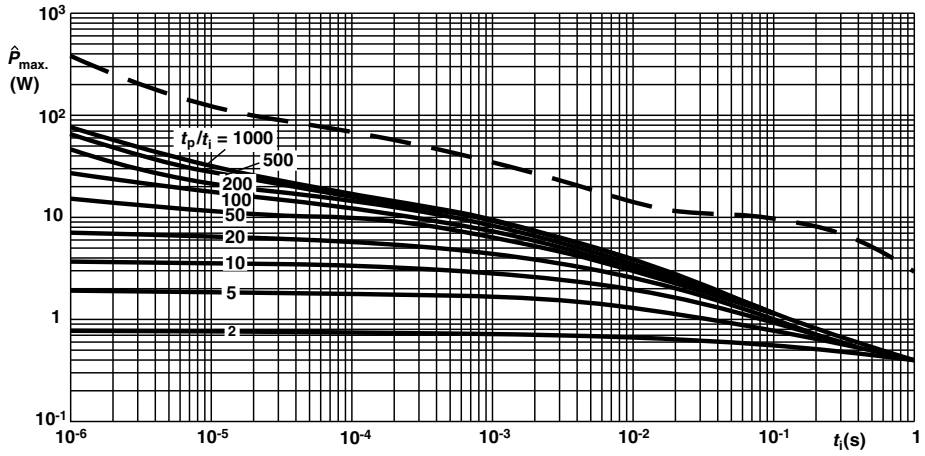
- The maximum permissible hot-spot temperature is 155 °C.



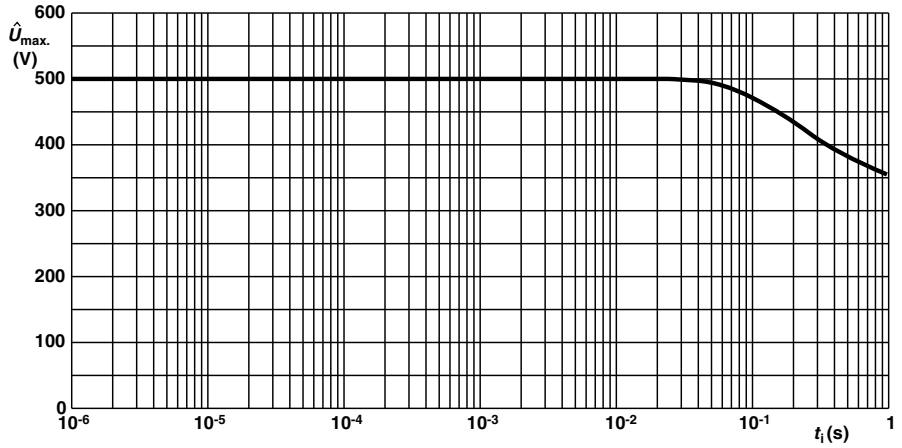
SFR16S Pulse on a regular basis; maximum permissible peak pulse power ($\hat{P}_{max.}$) as a function of pulse duration (t_i)



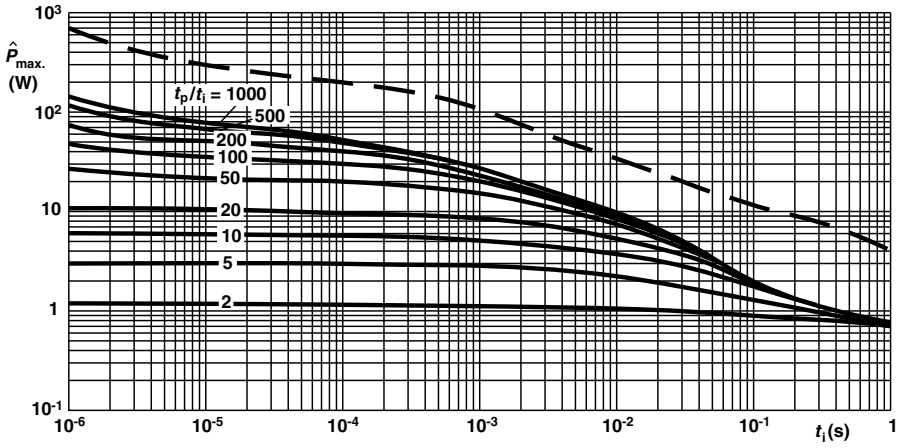
SFR16S Pulse on a regular basis; maximum permissible peak pulse voltage ($\hat{U}_{max.}$) as a function of pulse duration (t_i)



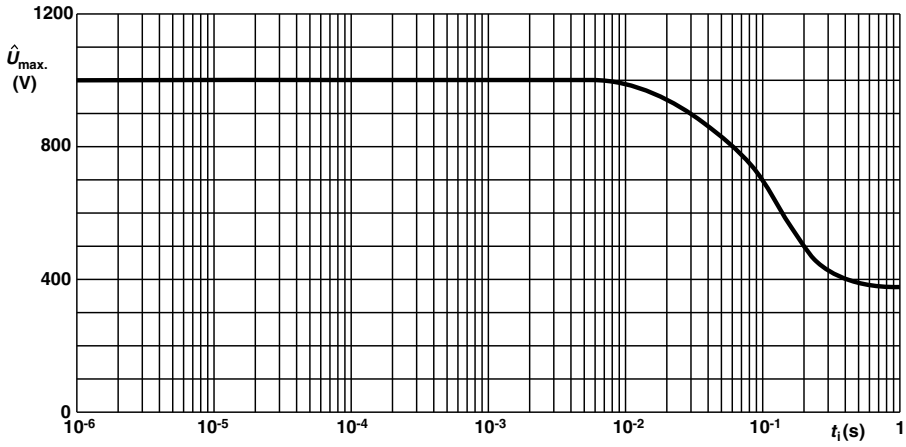
SFR25 Pulse on a regular basis; maximum permissible peak pulse power ($\hat{P}_{max.}$) as a function of pulse duration (t_i)



SFR25 Pulse on a regular basis; maximum permissible peak pulse voltage ($\hat{U}_{max.}$) as a function of pulse duration (t_i)



SFR25H Pulse on a regular basis; maximum permissible peak pulse power ($\hat{P}_{max.}$) as a function of pulse duration (t_i)



SFR25H Pulse on a regular basis; maximum permissible peak pulse voltage ($\hat{U}_{max.}$) as a function of pulse duration (t_i)

TESTS PROCEDURES AND REQUIREMENTS

All tests are carried out in accordance with the following specifications:

- EN 60115-1, generic specification (includes tests)

The test and requirements table contains only the most important tests. For the full test schedule refer to the documents listed above.

The tests are carried out in accordance with IEC 60068-2-xx test method and under standard atmospheric conditions in accordance with IEC 60068-1, 5.3.

Unless otherwise specified the following values apply:

- Temperature: 15 °C to 35 °C
- Relative humidity: 45 % to 75 %
- Air pressure: 86 kPa to 106 kPa (860 mbar to 1060 mbar).

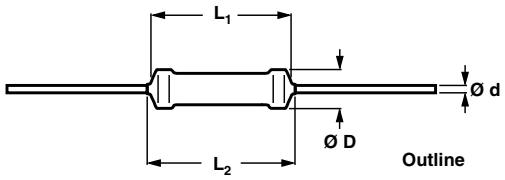
For performing some of the tests, the components are mounted on a test board in accordance with IEC 60115-1, 4.31. In test procedures and requirements table, only the tests and requirements are listed with reference to the relevant clauses of IEC 60115-1 and IEC 60068-2-xx test methods. A short description of the test procedure is also given.

TEST PROCEDURES AND REQUIREMENTS

IEC 60115-1 CLAUSE	IEC 60068-2 TEST METHOD	TEST	PROCEDURE	REQUIREMENTS PERMISSIBLE CHANGE ($\Delta R_{max.}$)				
4.5	-	Resistance	-	± 5 %; ± 1 %				
4.8	-	Temperature coefficient	At (20 / -55 / 20) °C and (20 / 155 / 20) °C	± 250 ppm/K; ± 100 ppm/K				
4.12	-	Noise	IEC 60195		< 68 kΩ	68 kΩ to 100 kΩ	> 100 kΩ to 1 MΩ	> 1 MΩ
				SFR16S	≤ 0.1 µV/V	≤ 0.5 µV/V	≤ 1.5 µV/V	≤ 1.5 µV/V
				SFR25, SFR25H	≤ 0.1 µV/V	≤ 0.1 µV/V	≤ 0.1 µV/V	≤ 1.5 µV/V
4.13	-	Short time overload	Room temperature; P = 6.25 x P_n; (voltage not more than 2 x limiting voltage); 5 s	± (0.25 % R + 0.05 Ω)				
4.16	21 (Ua1) 21 (Ub) 21 (Uc)	Robustness of terminations	Tensile, bending, and torsion	± (0.25 % R + 0.05 Ω)				
4.17	20 (Ta)	Solderability	at +235 °C; 2 s; solder bath method; SnPb40	Good tinning (≥ 95 % covered); no damage				
			at +245 °C; 3 s; solder bath method; SnAg3Cu0.5					
4.18	20 (Tb)	Resistance to soldering heat	Unmounted components (260 ± 5) °C; (10 ± 1) s	± (0.25 % R + 0.05 Ω)				
4.19	14 (Na)	Rapid change of temperature	30 min at -55 °C and 30 min at +155 °C; 5 cycles	± (0.25 % R + 0.05 Ω)				
4.20	29 (Eb)	Bump	3 x 1500 bumps in 3 directions; 40 g	± (0.25 % R + 0.05 Ω); no damage				
4.22	6 (Fc)	Vibration	10 sweep cycles per direction; 10 Hz to 2000 Hz 1.5 mm or 200 m/s^2	± (0.25 % R + 0.05 Ω); no damage				
4.23		Climatic sequence:						
4.23.2	2 (Ba)	Dry heat	155 °C; 16 h					
4.23.3	30 (Db)	Damp heat, cyclic	55 °C; 24 h; 90 % to 100 % RH; 1 cycle					
4.23.4	1 (Aa)	Cold	-55 °C; 2 h					
4.23.5	13 (M)	Low air pressure	8.5 kPa; 2 h; 15 °C to 35 °C					
4.23.6	30 (Db)	Damp heat, cyclic	55 °C; 5 days; 95 % to 100 % RH; 5 cycles	**SFR16S, SFR25, SFR25H**	± (1 % R + 0.05 Ω); no visible damage ± (1 % R + 0.05 Ω); no visible damage ± 2 % R; no visible damage			
4.23.7		DC load	apply rated power for 1 min					

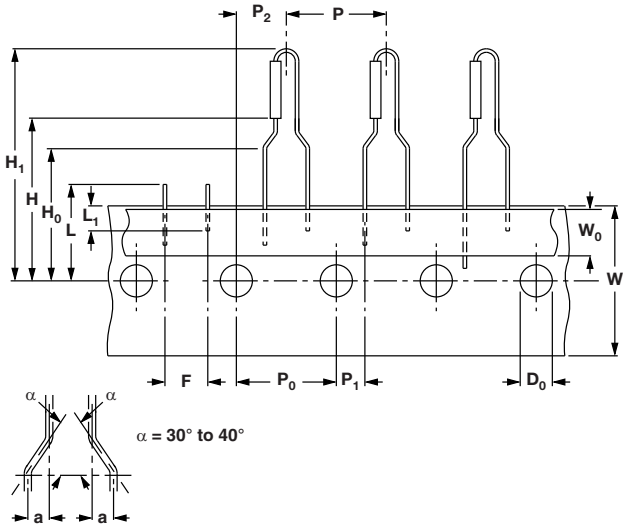
TEST PROCEDURES AND REQUIREMENTS				
IEC 60115-1 CLAUSE	IEC 60068-2 TEST METHOD	TEST	PROCEDURE	REQUIREMENTS PERMISSIBLE CHANGE ($\Delta R_{max.}$)
4.24	78 (Cab)	Damp heat (steady state)	(40 ± 2) °C; 56 days; (93 ± 3) % RH	± (2 % R + 0.05 Ω)
4.25.1		Endurance (at 70 °C)	$U = \sqrt{P_{70} \times R}$ or $U = U_{max.}$; 1.5 h on; 0.5 h off 70 °C; 1000 h	± (2 % R + 0.05 Ω)

DIMENSIONS



DIMENSIONS - Leaded resistor types, mass and relevant physical dimensions					
TYPE	Ø $D_{max.}$ (mm)	$L_{1\ max.}$ (mm)	$L_{2\ max.}$ (mm)	Ø d (mm)	MASS (mg)
SFR16S	1.9	3.5	4.1	0.45 ± 0.05	102
SFR25	2.5	6.5	7.5	0.58 ± 0.05	205
SFR25H	2.5	6.5	7.5	0.58 ± 0.05	205

SFR25, SFR25H WITH RADIAL TAPING



DIMENSIONS in millimeters		
Pitch of components	P	12.7 ± 1.0
Feed-hole pitch	P_0	12.7 ± 0.2
Feed-hole center to lead at topside at the tape	P_1	3.85 ± 0.5
Feed-hole center to body center	P_2	6.35 ± 1.0
Lead-to-lead distance	F	4.8 + 0.7 / - 0
Tape width	W	18.0 ± 0.5
Minimum hold down tape width	W_0	5.5
Maximum component height	H1	29
Lead wire clinch height	H_0	16.5 ± 0.5
Height of component from tape center	H	19.5 ± 1
Feed-hole diameter	D_0	4.0 ± 0.2
Maximum length of snipped lead	L	11.0
Minimum lead wire (tape portion) shortest lead	L_1	2.5

Note

- Please refer to document “Packaging” for more detail (www.vishay.com/doc?28721).

HISTORICAL 12NC INFORMATION

- The resistors had a 12-digit numeric code starting with 23.
- The subsequent 6 digits for 1 % or 7 digits for 5 % indicated the resistor type and packaging.
- The remaining digits indicated the resistance value:
  - The first 3 digits for 1 % or 2 digits for 5 % indicated the resistance value.
  - The last digit indicated the resistance decade.

Resistance Decade for ± 5 % Tolerance

RESISTANCE DECADE	LAST DIGIT
0.10 Ω to 0.91 Ω	7
1 Ω to 9.1 Ω	8
10 Ω o 91 Ω	9
100 Ω to 910 Ω	1
1 kΩ to 9.1 kΩ	2
10 kΩ to 91 kΩ	3
100 kΩ to 910 kΩ	4
1 MΩ to 9.1 MΩ	5
= 10 MΩ	6

Resistance Decade for ± 1 % Tolerance

RESISTANCE DECADE	LAST DIGIT
1 Ω to 9.76 Ω	8
10 Ω to 97.6 Ω	9
100 Ω to 976 Ω	1
1 kΩ to 9.76 kΩ	2
10 kΩ to 97.6 kΩ	3
100 kΩ to 976 kΩ	4
1 MΩ to 9.76 MΩ	5
= 10 MΩ	6

12NC Example

The 12NC of a SFR25 resistor, value 5600 Ω ± 5 %, taped on a bandolier of 5000 units in ammopack was:
2322 181 43562.

HISTORICAL 12NC - Resistor type and packaging

TYPE	TOL.	23..			
		BANDOLIER IN AMMOPACK			BANDOLIER ON REEL
		RADIAL TAPED	STRAIGHT LEADS		STRAIGHT LEADS
		4000 UNITS	1000 UNITS	5000 UNITS	5000 UNITS
SFR16S	± 5 %	-	..22 187 73...	..22 187 53...	..06 187 23...
	± 1 %	-	-	..06 187 3...	..06 187 1....
	Jumper	-	-	..06 187 90013	..22 187 90346
SFR25	± 5 %	..06 184 03...	..22 181 53...	..22 181 43...	..22 181 63...
	± 1 %	-	-	..22 188 2...	..06 181 8....
	Jumper	-	..22 181 90018	..22 181 90019	..06 181 90011
SFR25H	± 5 %	..06 186 03...	..22 186 16...	..22 186 76...	..06 186 63...
	± 1 %	-	-	..22 186 3....	..06 186 8....

Disclaimer

ALL PRODUCT, PRODUCT SPECIFICATIONS AND DATA ARE SUBJECT TO CHANGE WITHOUT NOTICE TO IMPROVE RELIABILITY, FUNCTION OR DESIGN OR OTHERWISE.

Vishay Intertechnology, Inc., its affiliates, agents, and employees, and all persons acting on its or their behalf (collectively, “Vishay”), disclaim any and all liability for any errors, inaccuracies or incompleteness contained in any datasheet or in any other disclosure relating to any product.

Vishay makes no warranty, representation or guarantee regarding the suitability of the products for any particular purpose or the continuing production of any product. To the maximum extent permitted by applicable law, Vishay disclaims (i) any and all liability arising out of the application or use of any product, (ii) any and all liability, including without limitation special, consequential or incidental damages, and (iii) any and all implied warranties, including warranties of fitness for particular purpose, non-infringement and merchantability.

Statements regarding the suitability of products for certain types of applications are based on Vishay’s knowledge of typical requirements that are often placed on Vishay products in generic applications. Such statements are not binding statements about the suitability of products for a particular application. It is the customer’s responsibility to validate that a particular product with the properties described in the product specification is suitable for use in a particular application. Parameters provided in datasheets and / or specifications may vary in different applications and performance may vary over time. All operating parameters, including typical parameters, must be validated for each customer application by the customer’s technical experts. Product specifications do not expand or otherwise modify Vishay’s terms and conditions of purchase, including but not limited to the warranty expressed therein.

Except as expressly indicated in writing, Vishay products are not designed for use in medical, life-saving, or life-sustaining applications or for any other application in which the failure of the Vishay product could result in personal injury or death. Customers using or selling Vishay products not expressly indicated for use in such applications do so at their own risk. Please contact authorized Vishay personnel to obtain written terms and conditions regarding products designed for such applications.

No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted by this document or by any conduct of Vishay. Product names and markings noted herein may be trademarks of their respective owners.

5.5. Hojas del fabricante de los LEDs

ELECTRONICS, INC.
44 FARRAND STREET
BLOOMFIELD, NJ 07003
(973) 748–5089
http://www.nteinc.com

NTE3019
Light Emitting Diode (LED)
Red Diffused, 5mm

Features:

- Tapered Barrel T–1 3/4 Package
- High Intensity Red light source with various lens colors and effects
- Versatile Mounting on PC Board or Panel
- T–1 3/4 with Stand–off

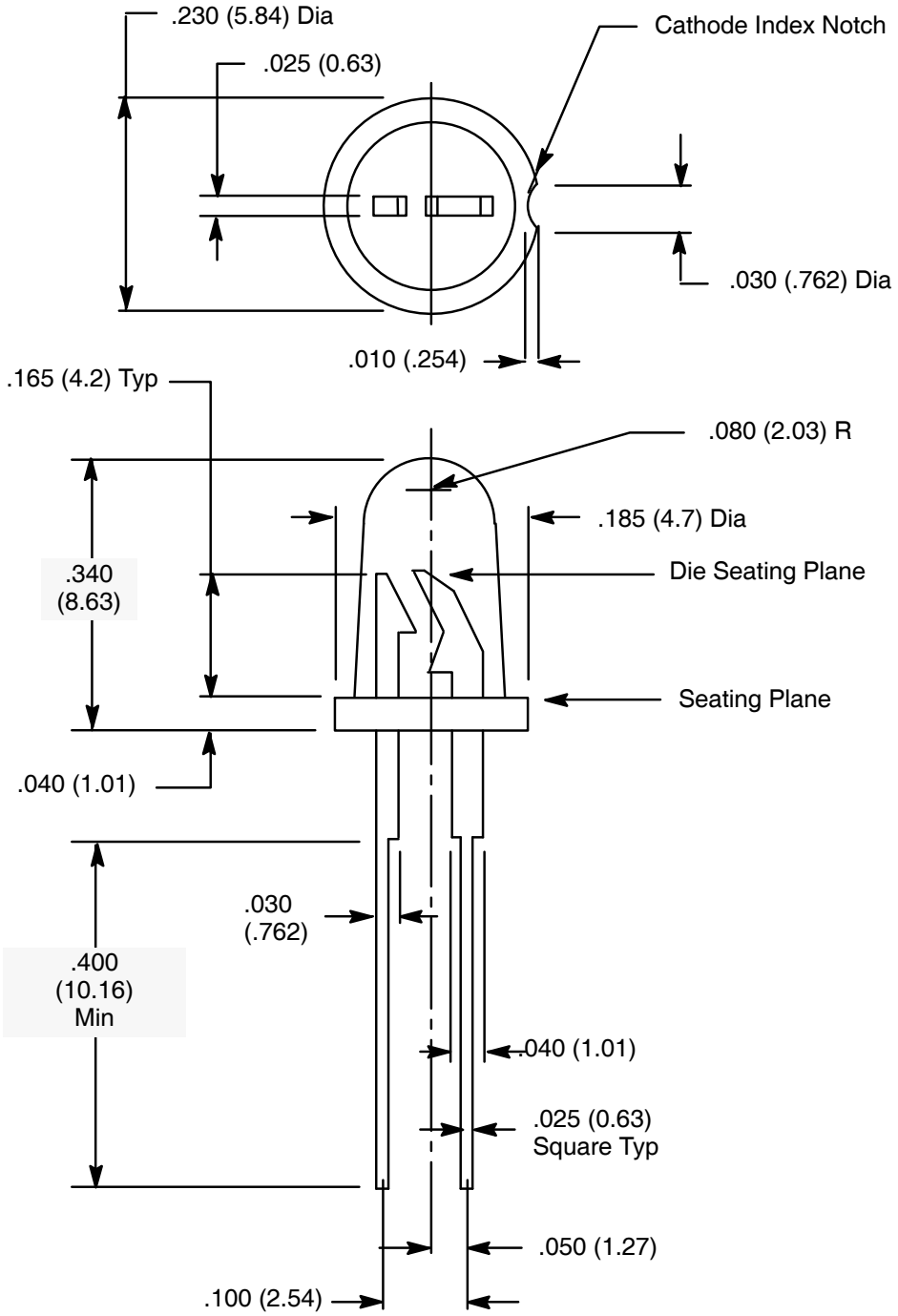
Absolute Maximum Ratings: (T_A = +25°C unless otherwise specified)

Reverse Voltage, V_R 5V
Peak Forward Current (Note 1, I_F 1A
Power Dissipation (T_A = +25°C), P_D 180mW
Derate linearly from 25°C . . . . . 2mW/°C
Operating Temperature Range, T_{opr} . . . . . –55° to +100°C
Storage Temperature Range, T_{stg} . . . . . –55° to +100°C
Lead Temperature (During Soldering, 1/16” (1.6mm) from case, 5sec max), T_L . . . . . +260°C

Note 1. Pulse Width = 1μs, 0.3% duty cycle.

Electrical Characteristics: (T_A = +25°C unless otherwise specified)

Parameter	Symbol	Test Conditions	Min	Typ	Max	Unit
Luminous Intensity	I_V	I_F = 20 mA	0.9	3.0	–	mcd
Peak Wavelength	λ_p	I_F = 20 mA	–	–	660	nm
Spectral Line Half Width	$\Delta\lambda$	I_F = 20 mA	–	20	–	nm
Forward Voltage	V_F	I_F = 20 mA	–	1.65	2.0	V
Reverse Current	I_n	V_R = 5.0V	–	–	100	λA
Reverse Voltage	λA	I_R = 100 λA	–	5.0	–	V
Capacitance	C	V = 0	–	35	–	pF
Viewing Angle	2θ1/2	Between 50% Points	–	60	–	degree
Rise Time	t_r	10% – 90% 50Ω	–	50	–	ns
Fall Time	t_f	90% – 10% 50Ω	–	50	–	ns


.230 (5.84) Dia
.025 (0.63)
Cathode Index Notch
.030 (.762) Dia
.010 (.254)
.165 (4.2) Typ
.080 (2.03) R
.185 (4.7) Dia
.340
(8.63)
Die Seating Plane
Seating Plane
.040 (1.01)
.030
(.762)
.400
(10.16)
Min
.040 (1.01)
.025 (0.63)
Square Typ
.050 (1.27)
.100 (2.54)