Universidad de La Laguna

TRABAJO DE FIN DE GRADO

Control de periféricos a través de microcontrolador ATMEL

Grado en Ingeniería electrónica industrial y automática

Año académico: 2018/2019
Autor: Álvaro Iván Arcia González
Tutor: Evelio J. González González

ÍNDICE

Capítulo 1.
Introducción

Capítulo 1. Introducción

1.1 Abstract

Nos disponemos a realizar la programación de una aplicación que controle los distintos componentes de la placa EasyAVRv7. Se irán programando en primer lugar los componentes más sencillos de utilizar e iremos aumentando la dificultad poco a poco hasta llegar a entender cada uno de los componentes en toda su complejidad. Comenzaremos programando los leds de la placa, continuando así con el display a 7 segmentos, la pantalla LCD, GLCD y por último el panel táctil.

El lenguaje utilizado para la programación ha sido ensamblador y nuestro microcontrolador es ATmega32. Trabajaremos con un entorno de programación llamado AVR Studio 5 y a través de él desarrollaremos poco a poco el código. Reforzaremos el aprendizaje a través de la búsqueda de información por las distintas páginas que nos ofrece internet así como por la hoja de especificaciones de los propios componentes que nos darán las herramientas para su correcto funcionamiento. Las páginas de apoyo utilizadas se verán reflejadas en el capítulo de Referencias.

We dispose to do a program what controls all components of EasyAVRv7 board. During this Project, we will program first the easier peripherals of the board and then we will improve our techniques increasing the difficulty with hardest components. The first we will programs are the leds, continuing with the 7 segments display, the LCD and GLCD screen, ending with de touch panel controller.

The programming language that we will use is Assembler and our microcontroller is ATmega32. We will work with the Integrated Development Environment (IDE) called AVR Studio 5 and we will develop the code first slowly and later increasing the difficulty. We will reinforce the knowledge searching a bit of information through the web sites and component's datasheet. They will give us the required information to the correct work of the different components. The websites that we use will be shown in the chapter "Referencias".

1.2 Objetivos

A continuación expondremos los distintos objetivos y los pasos llevados a cabo para nuestro proyecto. El objetivo general del proyecto será conseguir una aplicación gestionada por microcontrolador y la pantalla táctil GLCD que consiste en un desbloqueo de pantalla a través de distintas pulsaciones en un teclado numérico.

La pantalla mostrará el teclado y marcará los distintos números que vayamos pulsando, en caso de que el pin introducido sea correcto saldrá una imagen de desbloqueado de lo contrario saldrá un mensaje que hemos fallado la contraseña.

Para poder conseguir este objetivo final se llevó a cabo el proceso de ir aprendiendo cada uno de los periféricos de la placa, familiarizándonos poco a poco con el lenguaje ensamblador y sus distintas herramientas. Además trataremos con distintos software como AVR Flash para comunicar el microcontrolador con nuestro ordenador, la herramienta Inkscape para dibujar las distintas imágenes que más tarde serán mostradas por la GLCD.

Capítulo 2.
Software

Capítulo 2. Software

2.1 AVR Studio 5

El AVR Studio 4 es un IDE (Entorno de Desarrollo Interactivo) que nos permite escribir, compilar y depurar códigos tanto en lenguaje C como en Ensamblador para microcontroladores Atmel. En él se encuentran incluidas una gran cantidad de herramientas que nos facilitan el desarrollo del código. A través del mismo crearemos los distintos proyectos que serán introducidos en nuestra placa.

2.1.1 Creación del proyecto en AVR Studio 5

La creación de un proyecto es bastante sencilla. Solo deberemos iniciar el programa y clicar en la opción de “New Project…”.

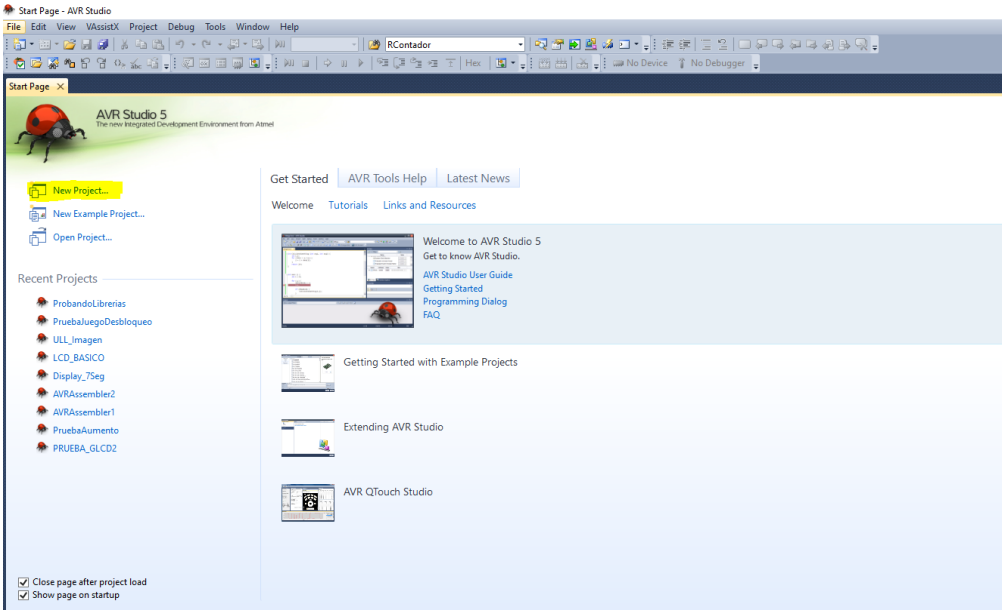


Figura 2.1.1.a Creando un nuevo proyecto.

Esto nos llevará a una nueva pantalla emergente, en esta deberemos seleccionar el idioma de programación que utilizaremos, este caso será “AVR Assembler Project”. También elegiremos el nombre de nuestro proyecto así como su localización.

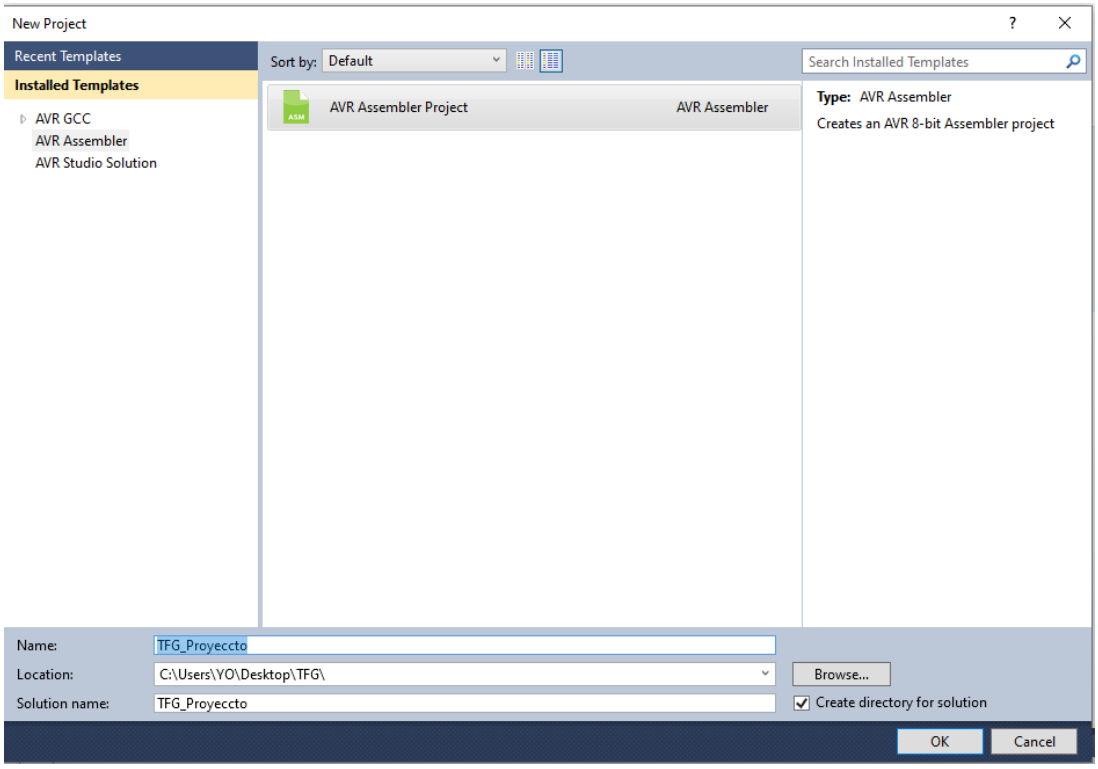


Figura 2.1.1.b Especificaciones del lenguaje y localización.

Por último debemos especificar el dispositivo para el cual programaremos. Esto se debe a que la programación varía según el microcontrolador, como por ejemplo en las interrupciones utilizadas.

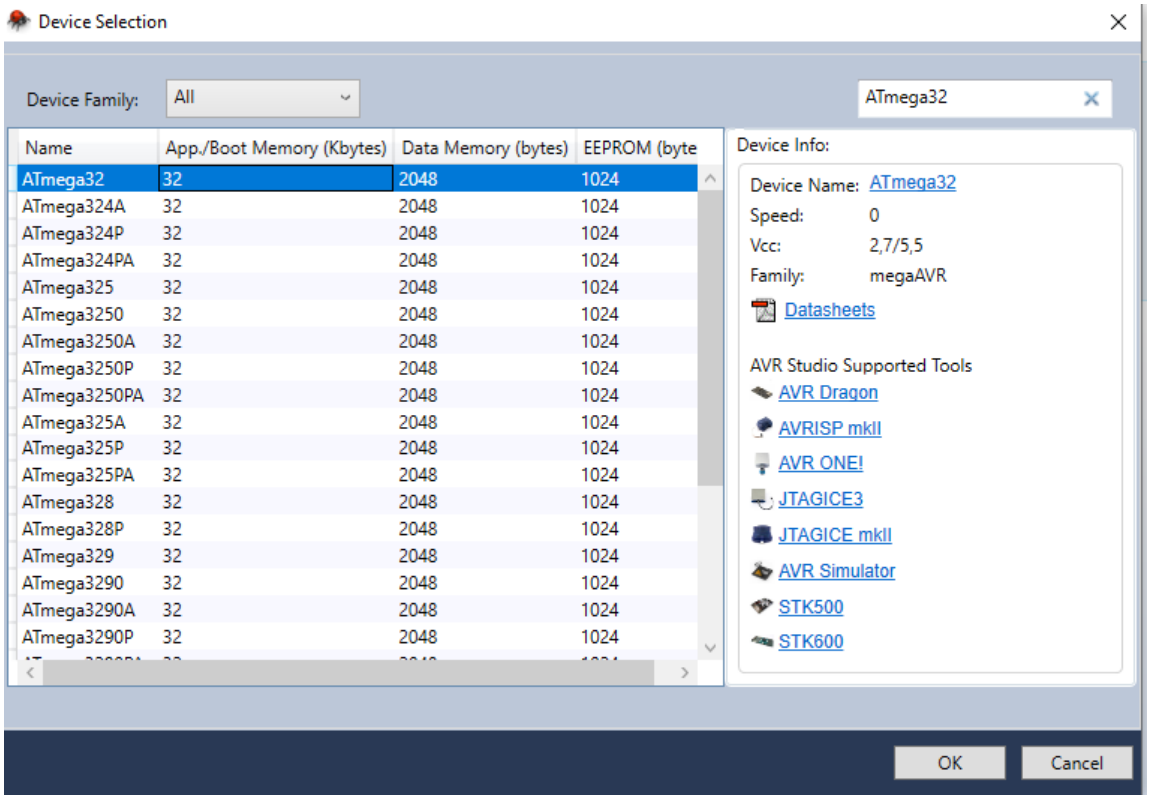


Figura 2.1.1.c Selección de dispositivo.

Nos da la opción de filtrar por familia de microcontrolador así como por texto, en nuestro caso seleccionaremos el ATmega32. Observamos que automáticamente nos muestra tanto el Datasheet como las distintas herramientas que podemos utilizar.Una vez generado el proyecto procederemos a escribir el código.

2.1.2 Herramientas del AVR Studio 5.

Una vez escrito el código para compilarlo solo debemos realizar un click en la barra de herramientas la opción de Build y elegir Build a solution.Automáticamente nos generará un archivo HEX en la carpeta del proyecto, usaremos dicho archivo para enviárselo al microcontrolador para su posterior procesado.

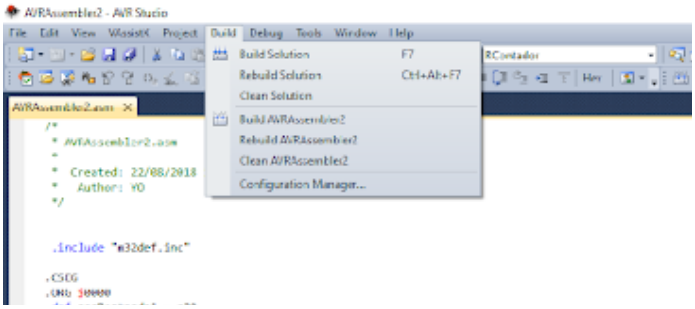


Figura 2.1.2.a Compilando el código

Otra herramienta útil que nos provee es la opción de Debug. A través de ella podemos depurar el código e ir observando cómo va ejecutándose el código manualmente pasando por cada una de las instrucciones. Para ello seleccionamos el simulador de microcontrolador que viene por defecto en el entorno llamado AVR Simulator. Esto ha sido altamente utilizado durante el desarrollo del proyecto debido a su potencial a la hora de encontrar fallos de ejecución.

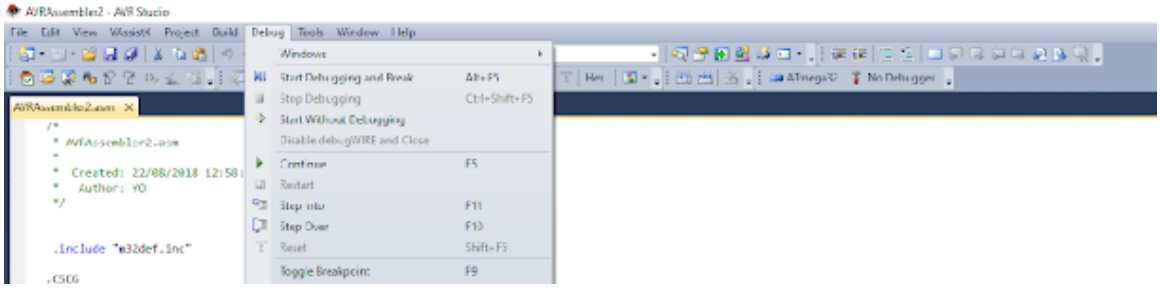


Figura 2.1.2.b Depurando el código

Algo realmente útil de esta herramienta es la opción de observar los valores de los registros así como introducir manualmente los valores que se desee en los mismos. De esta manera podemos ver cómo se comporta nuestro programa. En la Figura 2.1.2.c, observamos a la izquierda una flecha indicando en qué instrucción nos encontramos. Esta se puede controlar a través de F10 y F11. A la derecha observamos los distintos registros con sus valores, clicando en ellos podemos modificarlos a nuestro gusto. Podemos detener el modo Debug través de la opción Stop Debugging.

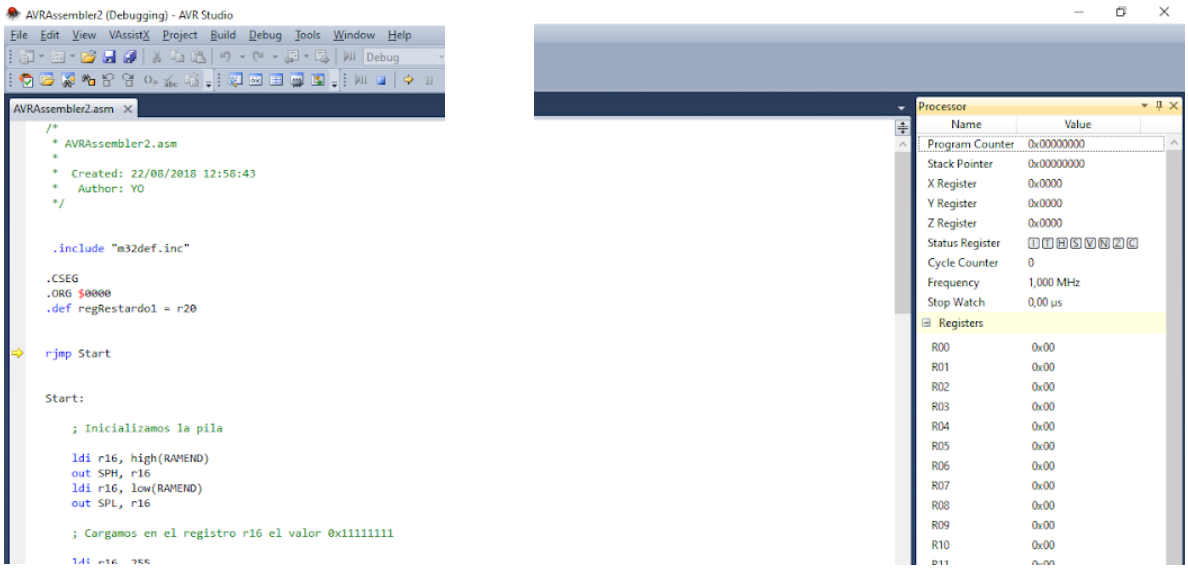


Figura 2.1.2.c Depurando el código, valores de los registros

En la ventana inferior de nuestro entorno podemos observar los errores y warnings de nuestro código, clicando en ellos nos lleva directamente a la línea de error. También nos muestra una descripción de los errores.

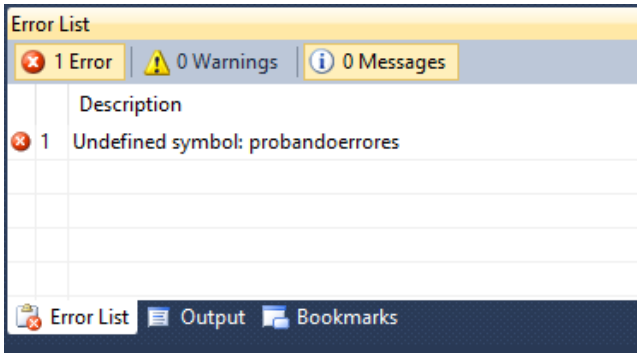


Figura 2.1.2.d Mensajes de error.

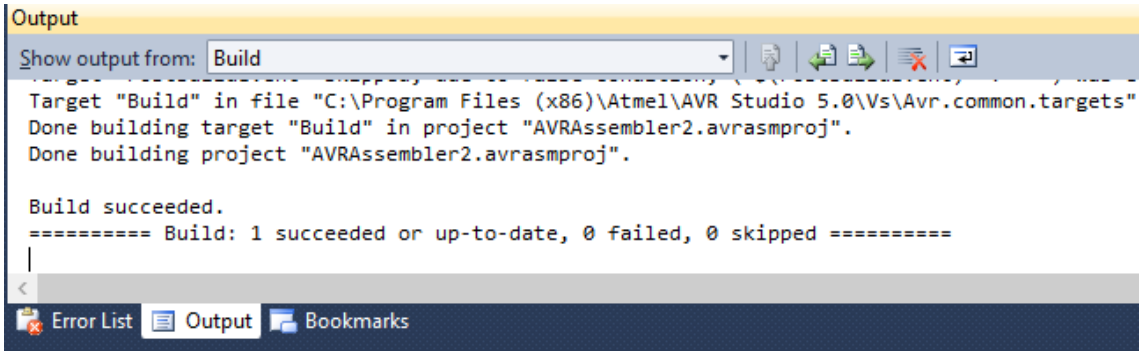


Figura 2.1.2.e Compilación exitosa.

2.2 AVRFlash

Esta herramienta nos permite enviar nuestros programas hacia nuestra placa a través de un cable USB. También nos permite realizar la función inversa que sería extraer archivos de nuestro microcontrolador. Los archivos tanto enviados y recibidos serán con extensión .HEX. Estos se generan automáticamente en la carpeta de nuestro proyecto cuando compilamos nuestro código.

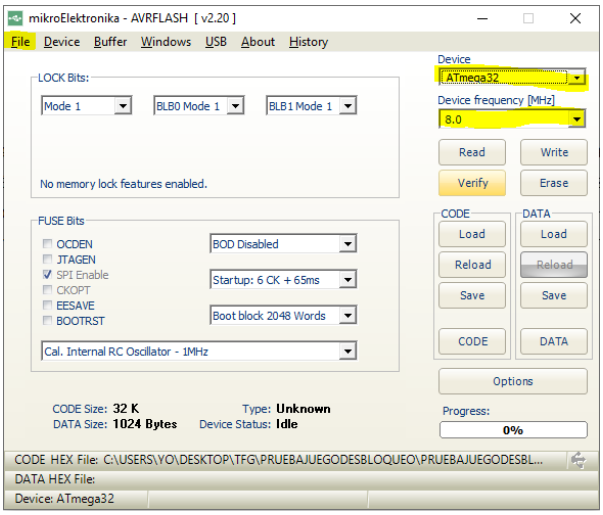


Figura 2.2.a Panel de AVRFlash.

Una vez abierto el programa debemos seleccionar el dispositivo que utilizaremos en nuestro caso ATmega32 y la frecuencia utilizada por el mismo. En nuestro caso el microcontrolador utiliza una frecuencia de 8MHz. Una vez seleccionadas las opciones, vamos a la pestaña **File➔ Load Hex**, y seleccionamos nuestro archivo.

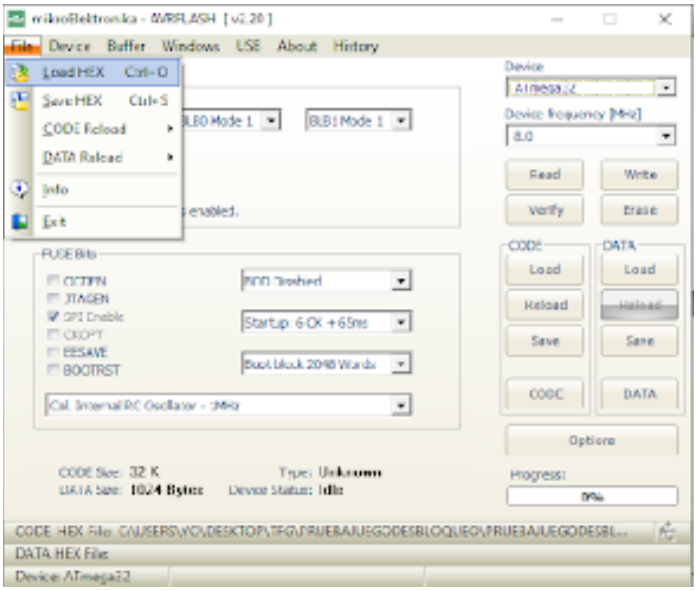


Figura 2.2.b Cargando archivos con AVRFlash.

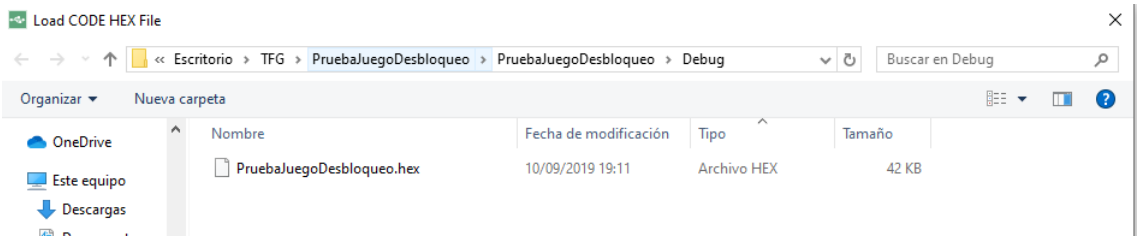


Figura 2.2.c Selección de archivos con AVRFlash.

Una vez seleccionado el archivo que deseamos enviar a nuestro microcontrolador clicamos en la opción de Write. Debemos cerciorarnos de que la placa está debidamente conectada y encendida.

2.3 LCD Assistant

Este programa nos permite transformar imágenes a arrays de números hexagecimales, esto será bastante útil en la utilización de tablas durante la programación de la pantalla GLCD. Posteriormente se hablará más en detalle sobre el tema. Para transformar la imagen solo debemos ejecutar la aplicación y seleccionar en la pestaña **File**. Nos dará la opción de cargar una imagen, esta debe estar en formato BPM. Una vez cargada la imagen tan solo debemos seleccionar **Save Output** y elegir la ubicación del archivo.

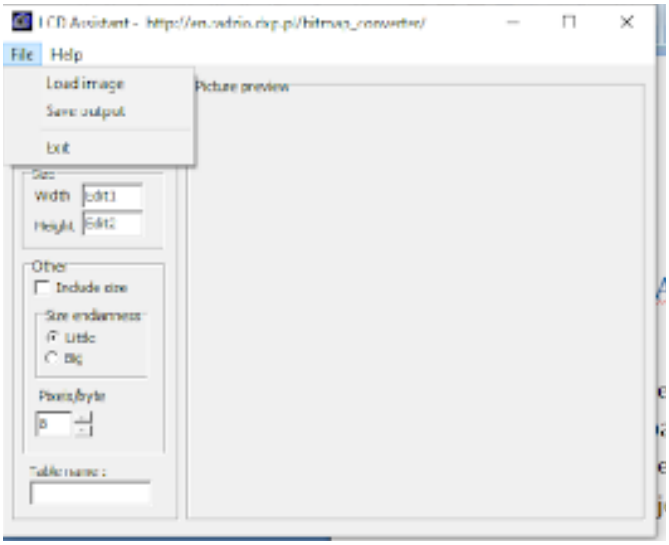


Figura 2.3. LCD Assistant

2.4 Notepad++

Esta aplicación es un editor de texto de software libre. Durante el proyecto ha sido utilizado para editar los distintos arrays que obteníamos a través del LCD Assistant. Esto se debe a que dicho programa nos lo daba en un formato para lenguaje C en vez de ensamblador por lo que debíamos realizar una serie de cambios. Las tablas eran de 64 líneas y en todas debíamos realizar cambios. El motivo por el que se usó Notepadd++ y no otra herramienta es debido a su atajo de teclado **CTRL + SHIFT + ALT**. Este atajo nos permite seleccionar tan solo una parte de las distintas líneas a la vez y poder modificarlas de forma conjunta sin

tener que ir de una en una. Dado a la sencillez y rapidez de su instalación se valoró de forma positiva el tiempo que requería en contraposición de cambiar las líneas de las tablas individualmente, a 64 líneas por tabla y 15 tablas utilizadas hacían un total de mil líneas donde los cambios eran iguales.

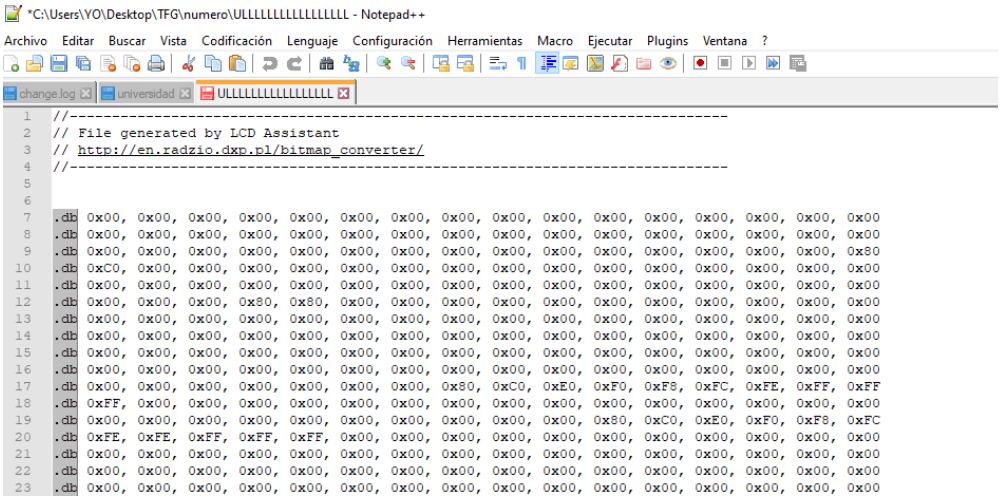


Figura 2.4. Utilización de Notepad++

2.5. Inkscape

Inkscape es una herramienta gratuita y sencilla de utilizar que nos permite realizar ilustraciones de una forma cómoda y profesional. A través de ella creamos las distintas imágenes que más tarde imprimiríamos en la pantalla GLCD. Este software nos habilita la posibilidad de elegir el tamaño en píxeles y guardar el archivo en formato PNG que más tarde convertiríamos a través de un convertidor online (https://www.onlineconverter.com/png-to-bmp) a BMP. Su utilización se basó en la necesidad de crear imágenes de unos pocos píxeles con gran exactitud y a su alta valoración en cuanto a calidad-dificultad. Además de ofrecer un mayor control sobre las imágenes que no nos ofrecen otras herramientas que nos provee Windows.

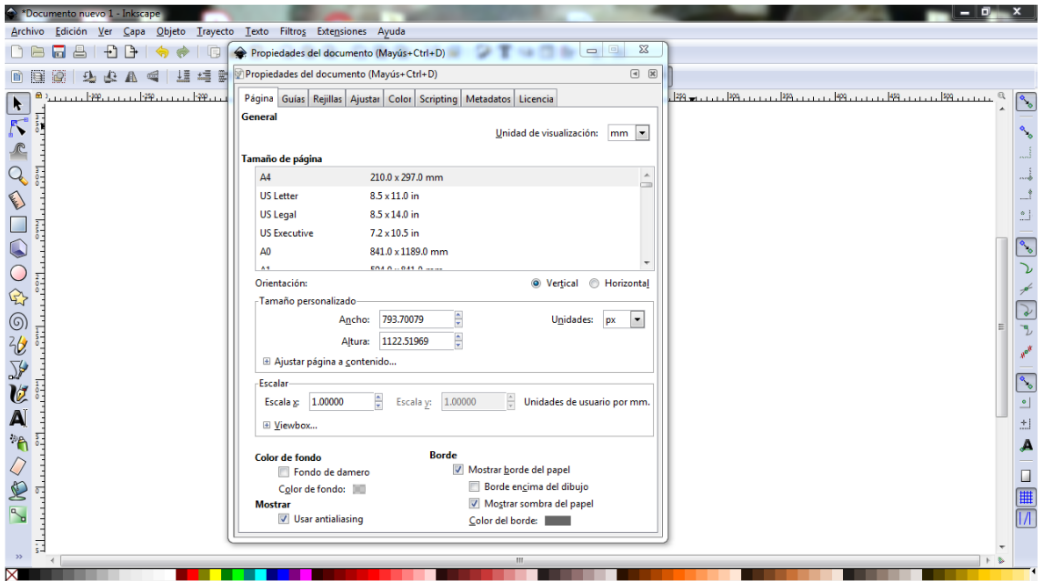


Figura 2.5.a Modificando las propiedades de Inkscape.

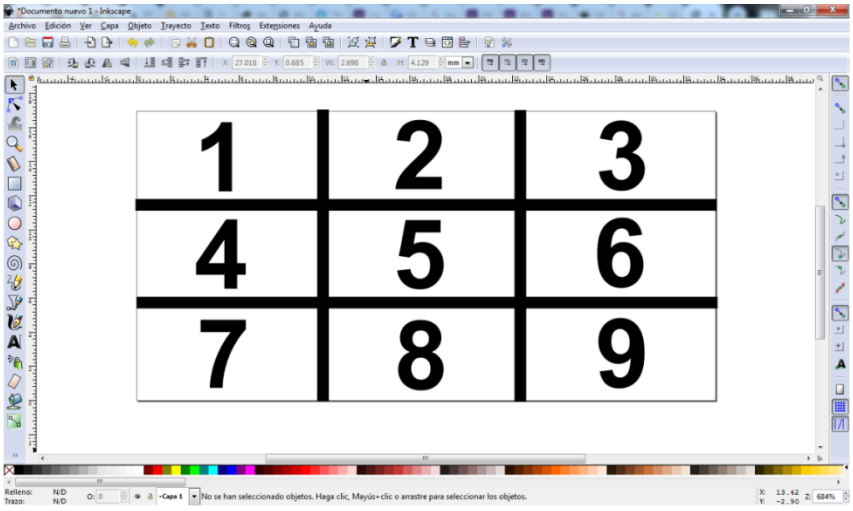


Figura 2.5.b Modificando las propiedades de Inkscape.

2.6 Herramienta GIT

GitHub es un software altamente utilizado por los desarrolladores debido a su utilidad como programa de control de versiones. Se creó un repositorio local a través del cual se controlaban con una serie de comandos las distintas versiones de cada proyecto. Con él somos capaces de guardar cambios en nuestro programa pudiendo volver a versiones anteriores en caso de errores en la programación o pérdidas de archivos. En este proyecto su aplicación no tuvo gran relevancia y no se llegó a utilizar todo su potencial pero es una herramienta extremadamente útil que merece la pena utilizar. También es posible crear un repositorio remoto y guardar todas las versiones de forma online.

2.7 AVR Delay Loop Calculator.

Esta es una herramienta online extremadamente útil que nos permite crear Delays o retardos que serán muy utilizados a lo largo del desarrollo del código. Más adelante entraremos en detalle en qué es un Delay y por qué debemos usarlos. La herramienta se utilizó para crear de forma precisa bucles que produjeran un retardo en el código. Estos se pueden hacer de forma manual y no es necesario usar un programa, como desarrolladores deberíamos tener la capacidad para crearlos pero entraremos en más detalle en capítulos posteriores.

La forma de utilizar la herramienta es muy fácil tan solo se ha de elegir la frecuencia en MHz de nuestro microcontrolador y el tiempo que queremos que dure el delay. A continuación mostramos el enlace a la herramienta: http://www.bretmulvey.com/avrdelay.html

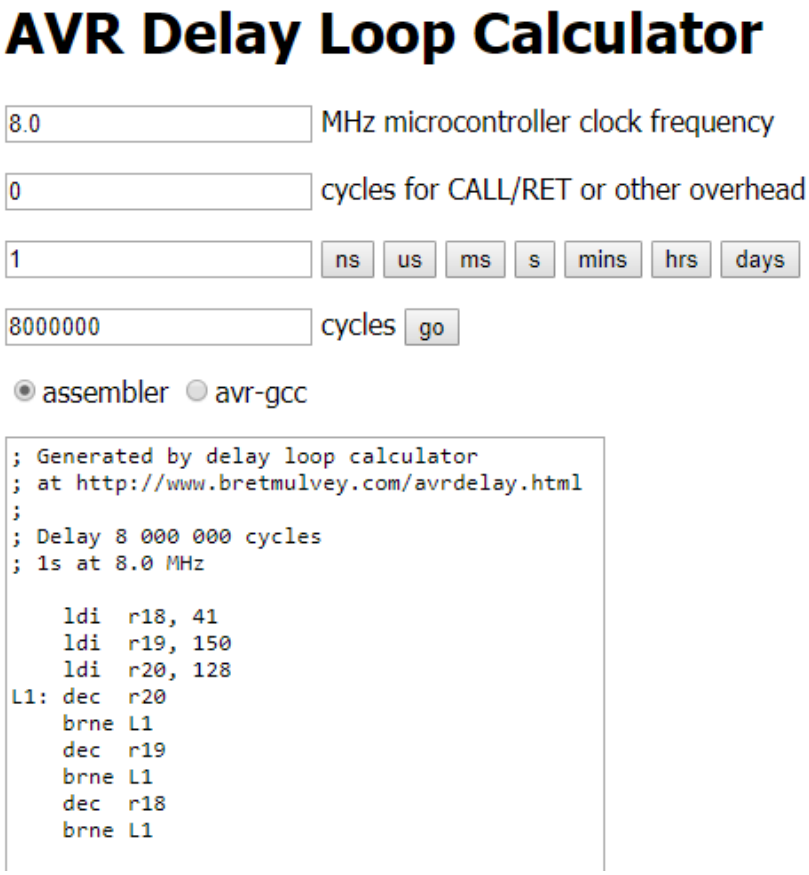


Figura 2.7. AVR Delay Loop Calculator.

2.8 Instalación de los drivers.

Para poder trabajar con AVRFlash se debe realizar la instalación de los drivers propios la placa. Para ello debemos tener en posesión el DVD que viene incluido en el kit de EasyAVR. Este nos llevará a un enlace y a través de él descargamos un archivo zip que descomprimimos para más tarde ejecutar el archivo .exe, seguimos los pasos de instalación.

Capítulo 3.
Hardware

Capítulo 3. Hardware

3.1 Placa EasyAVR™v7.

EasyAVR es una placa de desarrollo que nos permite la creación de proyectos de electrónica propios a través de sus distintos módulos y la escalabilidad que posee. Es una herramienta práctica y fácil de utilizar tanto para estudiantes como para profesionales en el sector. Dispone de conexiones USB UART/RS-232, 32 leds utilizables, 4 displays a 7 segmentos, una pantalla LCD, una pantalla GLCD, posibilidad de alimentación a través de USB o adaptador tanto para 5V como para 3.3v, sensor de temperatura y varios módulos complementarios. También es capaz de soportar 65 microcontroladores distintos aunque por defecto nos viene un ATmega32. La placa se puede programar a través de USB.

Figura 3.1 Placa EasyAVR™v7

3.1.1 MikroProg™

MikroProg es un programador a través de USB que permite la programación de 65 microcontroladores distintos a través del software AVRFlash. Posee 3 leds; el primer led (Link) se enciende cuando el cable USB se encuentra conectado, el led ACTIVE nos indica cuando el programador está realizando su función y el tercer led DATA, nos informa de cuando mikroProg se encuentra recibiendo o enviando datos a través de AVRFlash.

Figura 3.1.1 Programador mikroProg

3.1.2 Leds

Los leds son componentes electrónicos los cuales pueden emitir luz cuando circula por ellos una cierta cantidad de corriente. Nuestra placa EasyAVR consta de 32 leds que se encuentran en 4 filas de 8, cada fila asociada a un puerto de nuestro microcontrolador. Para encender los leds debemos configurar el Switch 10 y así poder habilitar su encendido.

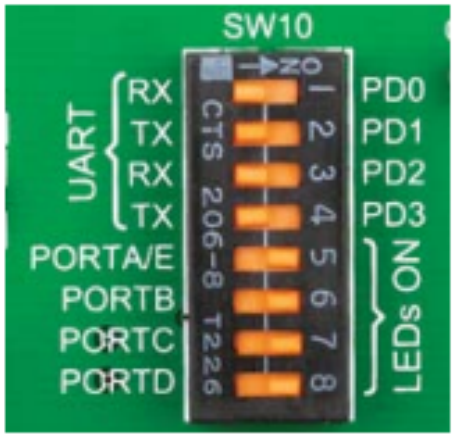


Figura 3.1.2 Configuración del SW10.

3.1.3 Display de 7 segmentos.

El display o visualizador de 7 segmentos es un componente electrónico utilizado para la representación de caracteres (principalmente números), a través de unos segmentos que se pueden controlar de forma individual. Nuestra placa posee un total de 4 displays. A través del puerto C del microcontrolador podemos seleccionar qué segmentos encendemos y a través de los pines del 0 al 3 del puerto A podemos seleccionar qué display encendemos.

Para su utilización debemos configurar el SW8 habilitando los switchs 8.1, 8.2, 8.3 y 8.4.

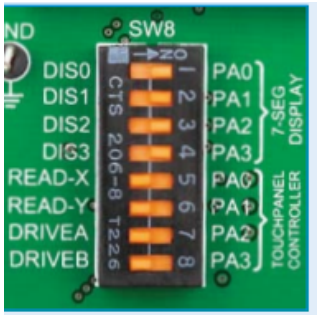


Figura 3.1.3 Configuración del SW8 para el uso del display a 7 segmentos.

3.1.4 Pantalla LCD 2X16 Caracteres.

Las pantallas LCDs o LiquidCrystal Display, son finas pantallas con píxeles monocromáticos que se sitúan delante de una luz para mostrar caracteres, imágenes, números, etc. A continuación se presentarán los pines para las conexiones que vienen por defecto a la pantalla:

GND y Vcc	Son los encargados de proporcionar alimentación de 5V a la pantalla.
V_o	Regula el contraste de la pantalla a través de un potenciómetro.
Rs	Registro de selección de línea. Controlado por el bit 2 del puerto A.
E	Habilita el envío de información a la pantalla. Asociado al Pin 6 del puerto D del microcontrolador
R/W	Determina el modo de la LCD, read/write. Siempre se encuentra en modo Write ya que Read está conectado a tierra
D0-D3	Están conectados a tierra ya que la LCD se encuentra en modo 4 bits
D4-D7	La LCD recibirá información de escritura a través de estos pines. Irá asociado al puerto C del microcontrolador.
LED+	Conexión con el ánodo del led encargado de la luz de fondo
LED-	Conexión con el cátodo del led encargado de la luz de fondo

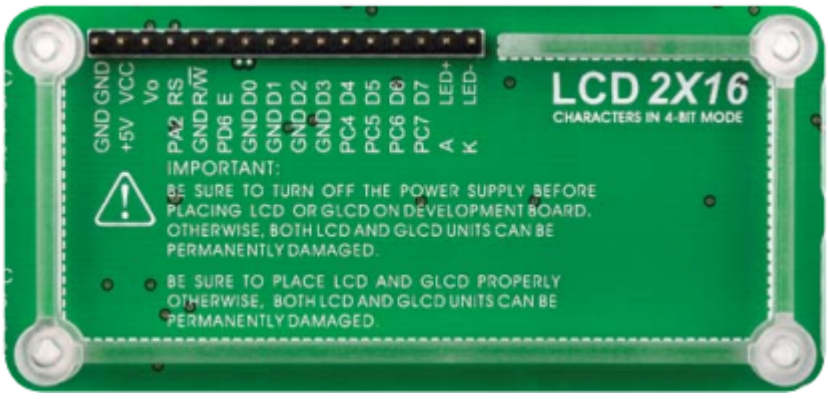


Figura 3.1.4.a Conexiones LCD.

Para habilitar el funcionamiento de la pantalla LCD debemos configurar los Switch 3 activando los sw.1 y sw.3.

Figura 3.1.4.b Configuración del SW3 para el uso de la LCD.

3.1.5 Pantalla GLCD 128X64 píxeles.

Una pantalla GLCD (GraphicLiquidCrystal Display) es una LCD formada por un conjunto de píxeles formando una matriz. Esta utiliza una pequeña cantidad de corriente para encender y apagar los píxeles a nuestro gusto. Esta pantalla dispone de una memoria interna con la que podemos almacenar el estado de la pantalla.

La pantalla se divide en dos controladores en el eje horizontal, cada una de 64 píxeles y en 8 páginas por el eje vertical de 8 píxeles cada una haciendo un total de 64.

Conexiones de la pantalla

CS1 y CS2	Controladores de línea. Estos definen sobre qué pantalla procederemos a escribir. Ambos controlados por los pines 0 y 1 del puerto B respectivamente. CS1 controla la pantalla izquierda y CS2 la derecha.
GND y Vcc	Son los encargados de proporcionar alimentación de 5V a la pantalla.
V0	Se encarga del contraste de la pantalla. Está regulado a través del potenciómetro 4 de la pantalla
R/W	Determina el modo de Gla LCD, read/write. Está determinada a través del bit 3 del puerto A.

RS	Indica el modo de la pantalla. En modo datos se encuentra en 1 y en modo instrucción en 0. Esta función estará controlada a través del puerto A el pin 2.
E	Habilita el display. Está controlado a través del pin 6 del puerto D.
D0-D7	La GLCD recibirá valores escritura a través de estos pines. Irá asociado al puerto C del microcontrolador. Esto controlan los píxeles del eje vertical
RST	Reset del display. Se controla a través del pin 7 del puerto D.
Vee	Voltaje de referencia para el contraste de la GLCD. Está controlado a través del potenciómetro 3.
LED+	Conexión con el ánodo del led encargado de la luz de fondo
LED-	Conexión con el cátodo del led encargado de la luz de fondo

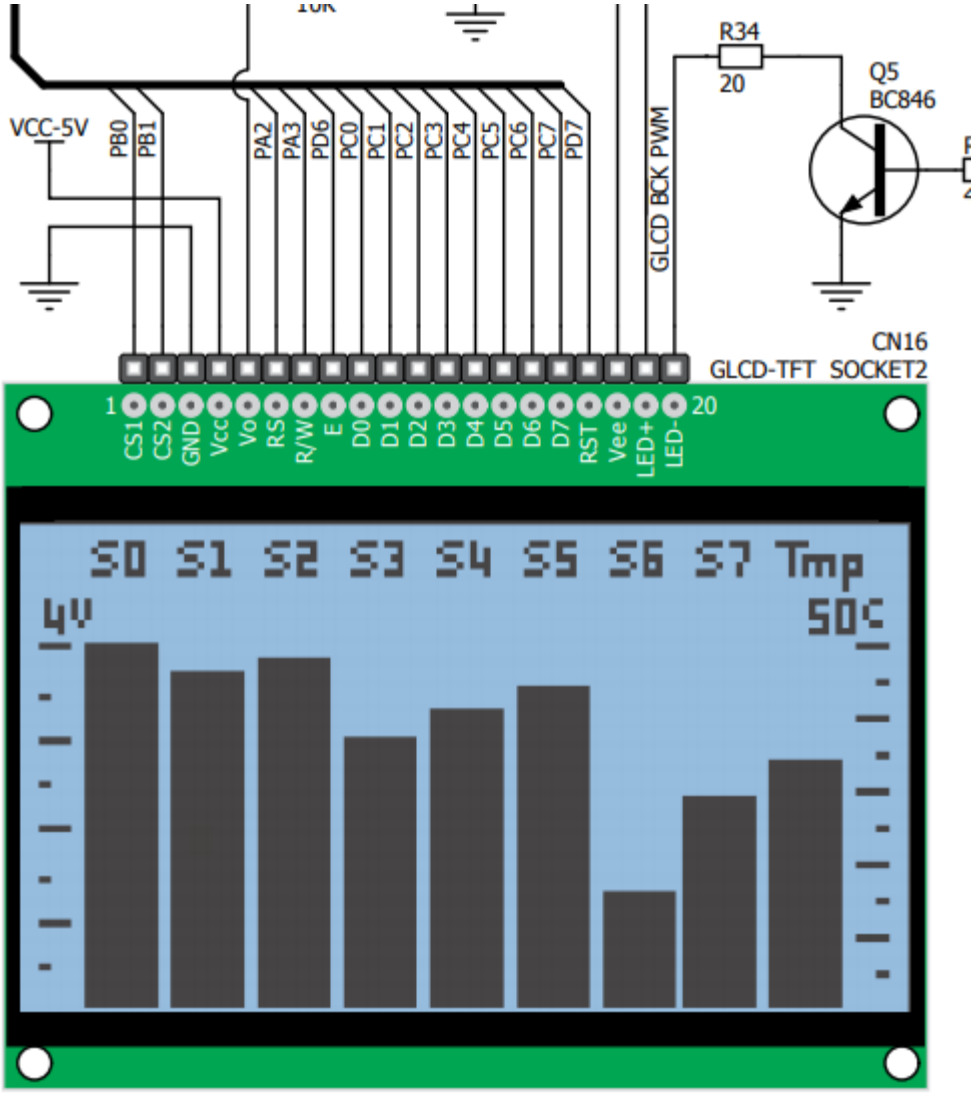


Figura 3.1.5. Conexiones Microcontrolador-GLCD.

Podemos determinar la iluminación de fondo de la pantalla a través del SWITCH 3, para que esta se encuentre en su máxima intensidad debemos utilizar el SW3.1. También podemos habilitar la función para controlar nosotros mismo la iluminación a través del SW3.2. Como dato importante debemos tener en cuenta que para habilitar el modo back-light, así llamado el modo para controlar la luz de fondo), debemos tener activados ambos SWITCH. No solo el segundo.

3.1.6 Panel de control táctil.

Tenemos la opción de habilitar este panel de la pantalla GLCD, con dicho panel habilitado la pantalla GLCD enviará un cierto voltaje según el eje vertical u horizontal. Nosotros a través de los pines de nuestro controlador podemos recibir este voltaje y procesarlo. Estos voltajes serán enviados a través del bit 0 del puerto A del controlador en función del eje vertical y a través del bit 1 del mismo puerto en función de la horizontal. Debemos qué voltaje deseamos leer a través de los bits 2 y 3 del puerto A. Tener en cuenta que no podemos activar ambos pines a la vez y que el voltaje que entra a la placa llega con un valor analógico, nosotros debemos realizar una transformación analógica-digital para poder procesar y trabajar con dicha información. Esto es lo más complejo de trabajar en nuestro proyecto.

Para habilitar este panel debemos activar los switchs del 5 al 8 del SW8.

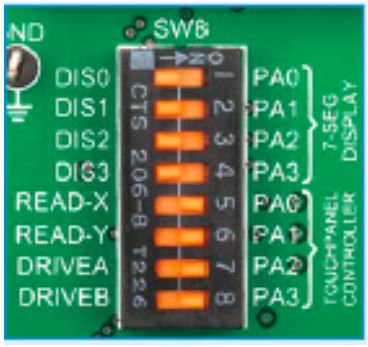


Figura 3.1.6 Configuración del SW3 para el uso de la LCD.

3.2 Multímetro Digital.

Un multímetro digital es un instrumento utilizado para la medida de valores eléctricos. Con este podemos medir desde tensiones, corrientes, resistencias, capacitancia, etc. A continuación, comentamos el motivo de su utilización.

Como se trató en el punto anterior, según donde pulsemos en la pantalla recibiremos un voltaje en función de qué punto de los ejes pulsamos. Por desconocimiento y falta de experiencia se supuso al principio que los voltajes de entrada serían de 0 a 5v debido a la alimentación de la pantalla. Esto no es

correcto a nivel práctico ya que el rango de voltajes de entrada era distinto al comentado por lo que se tuvo que disponer de un multímetro digital para medir los distintos voltajes.

Su utilización fue sencilla, se configuró en primer lugar la pantalla para que leyese el voltaje en función a la pulsación sobre el eje horizontal, desactivando el bit 1 y activando el 2 del puerto A. Conectamos el multímetro a la tierra de la placa y al pin 1 del microcontrolador, se pulsó a ambos extremos de la pantalla y se anotó los datos. Lo mismo se hizo para el eje horizontal, entraremos en más detalle en los capítulos posteriores.

Capítulo 4.
Desarrollo del código

Capítulo 4. Desarrollo del código.

4.1. Lenguaje ensamblador.

El lenguaje ensamblador es un lenguaje de programación de bajo nivel. Consiste principalmente en un conjunto de instrucciones lógicas fáciles de entender para un ser humano y cercanas al lenguaje de máquina. A través de este podemos programar desde ordenadores hasta microcontroladores y cualquier componente programable. A la hora de compilar nuestro código fuente (archivo principal de nuestro programa), se genera un archivo normalmente con extensión .HEX, que más tarde será enviado a nuestra placa. Este lenguaje ha sido el utilizado para nuestro proyecto y a continuación se resumen los aspectos más importantes o utilizados en el mismo.

4.1.1. Registros en ensamblador.

Los registros ensamblador son porciones de memoria que se emplean como si fueran variables, con ellos ejecutamos las distintas instrucciones. Existen 32 registros de 8 bits que podemos utilizar, aunque solo 16 son capaces de ejecutar operaciones inmediatas (r16-r31).

También existen pares de registros que actúan como punteros para acceder a la memoria SRAM de nuestro microcontrolador r26:27 y 28:r29 o a la memoria del programa r30:r31.

4.1.2. Pila.

La pila es un espacio de memoria que se reserva para almacenamiento de información de forma temporal. A través de las instrucciones PUSH y POP, podemos acceder a la pila. Esto es principalmente utilizado en subrutinas, guardamos datos en la pila al entrar en la subrutina para más tarde recuperar dichos datos, de esta forma podemos reutilizar registros sin que se pierda

información. Para utilizarla debemos inicializarla a través de las siguientes instrucciones.

```
ldi regUsoGeneral, high(RAMEND)
out SPH, regUsoGeneral
ldi regUsoGeneral, low(RAMEND)
out SPL, regUsoGeneral
```

Figura 4.1.1. Inicialización de la pila.

4.1.3. Instrucciones.

Las instrucciones son las indicaciones que realizamos a nuestro programa para implementar la lógica deseada. A continuación, se describen las principales instrucciones utilizadas.

4.1.3.1. Instrucciones aritmético-lógicas.

ADD Rd, Rs	Suma de dos registros. Rd = Rs + Rd
INC Rd	Suma 1 a un registro. Rd = Rd + 1
DEC Rd	Resta 1 a un registro. Rd = Rd -1
SUB Rd, Rs	Resta de dos registros Rd= Rd - Rs

4.1.3.2. Instrucciones delógica binaria.

AND Rd, Rs	Operación lógica AND entre dos registros
ANDI Rd, k	Operación lógica AND entre un registro y una constante
OR Rd, Rs	Operación lógica OR entre dos registros
ORI Rd, k	Operación lógica OR entre un registro y una constante

4.1.3.3. Instrucciones para enmascarar información.

SBR Rd, k	Pone en alto el bit k del registro Rd
CBR	Pone en bajo el bit k del registro Rd

4.1.3.4. Instrucciones de salto o control de flujo.

RJMP K	Salto relativo a otra parte del código.
SBRC Rd, b	Si el bit b del registro Rd es igual a 0 salta la siguiente instrucción.
SBRS Rd, b	Si el bit b del registro Rd es igual a 1 salta la siguiente instrucción
BREQ K	Si la última operación es igual a 0, salta a K.
BRNE K	Salta a Ksi la última operación es distinta de 0.
BRLO K	Salta a K si la operación anterior es negativa
BRSH K	Salta a K si la operación anterior es cero o positiva

4.1.3.5. Instrucciones para subrutinas.

RCALL K	Llama a la subrutina CALL
RET	Retorna de una subrutina

4.1.3.6. Instrucciones para enmascarar información.

LDIRd, K	Carga inmediata el valor k en el registro Rd

4.1.3.7. Instrucciones para trabajar con la pila.

POP Rd	Inserta el registro Rd en la pila
PUSH Rd	Recoge el registro Rd de la pila

4.1.3.8. Instrucciones de comparación.

CP Rd, Rs	Comparación de dos registros, Rd – Rs.
CPI Rd, k	Comparación de un registro y una constante Rd - k

4.1.4. Comentarios.

Los comentarios son líneas de código para explicar o hacer anotaciones en el mismo. Es una buena práctica realizar comentarios concisos y aclaratorios sobre qué papel realiza cada parte del código para de esta forma se tenga una mayor claridad a la hora de leerlo. Estas líneas serán ignoradas por el compilador y tan solo tiene objetivos de comprensión.

Para realizar comentarios podemos usar un punto y coma para comentar una línea y un asterisco con barra para comentar varias líneas.

```
; Ejemplo de comentario a una línea.


/*
 * Ejemplo de comentarios
 * varias líneas.
 * Álvaro Iván
 * Arcia González
 */
```

Figura 4.1.4. Ejemplo de comentarios en ensamblador.

4.1.5. Directivas.

Las directivas son comandos utilizados para organizar el código y los datos. En nuestro proyecto solo se ha hecho uso de dos directivas principales:

.INCLUDE ➔ Permite incluir librerías dentro de nuestro programa, ya que para cada microcontrolador puede variar el lenguaje utilizado en cuanto a las instrucciones, necesitamos incluir la librería específica de nuestro microcontrolador: m32def.inc.

.CSEG ➔ Indica el comienzo del segmento del código.

ORG $0000 ➔ Asigna la dirección de memoria.

.DEF nombreRegistro = Rx ➔ Se utiliza para asociar valores con palabras. En nuestro proyecto fue altamente utilizado para la organización del mismo. Asignamos nombres a los distintos tipos de registros para distinguir para qué se utilizaba cada registro.

```
.def regUsoGeneral = r16
.def regVecesPulsado = r29
.def regDesbloqueo = r19
```

Figura 4.1.5. Ejemplo de uso directiva .DEF

4.1.6. Etiquetas y subrutinas.

Las etiquetas en ensamblador son nombres que asociamos a ciertas partes del código a las que nos interesa acceder en un determinado momento. Por ejemplo con condicionales, si se cumple cierta condición saltamos a una parte del código.

Las subrutinas son un conjunto de instrucciones agrupadas que realizan una determinada labor en nuestro programa. Normalmente queremos realizar dicha labor varias veces por lo que para no repetir el grupo de instrucciones constantemente las encapsulamos en una subrutina y la llamamos a través de la instrucción CALL o RCALL cada vez que nos interese.

Ambas son bastante necesarias cuando queremos crear una lógica con cierta complejidad y serán muy utilizadas en nuestro proyecto. En el capítulo de **Errores y soluciones durante el desarrollo**, se comentarán varios errores a tener en cuenta que se tuvieron durante el uso de estas herramientas.

```
Start:
    rcall EjemploSubrutina
    rjmp Start

EjemploSubrutina:
    Instruccion1
    Instruccion2
    Instruccion3
    .
```

Figura 4.1.6. Ejemplo etiqueta y subrutina.

4.1.7. Retardos en ensamblador.

Los retardos o delays son un tipo de rutina muy utilizada en programación a través de las cuales se consigue un tiempo de espera en el cual el programa no realiza ninguna función aparente. En el caso de ensamblador realizamos retardos a través de bucles haciendo que el programa se mantenga sin hacer nada durante un cierto tiempo. En nuestro caso realizamos dichas rutinas a través de una aplicación online llamada AVR Delay Loop Calculator pero nosotros como desarrolladores debemos ser capaces de crear estas rutinas manualmente y de forma precisa. Para ello tenemos que realizar una serie de cálculos que se explicarán a continuación.

Para generar un retardo debemos conocer la frecuencia reloj de nuestro controlador, sabiendo que:

$$T = \frac{1}{f}$$

Siendo T el periodo o tiempo de ejecución de una instrucción y f la frecuencia reloj de nuestro microcontrolador. En nuestro caso sabemos que la frecuencia es de 8 MHz, haciendo un período de 0.125 µs. Este es el tiempo que tarda el microcontrolador en ejecutar una instrucción.

Supongamos que queremos obtener un tiempo de retardo de 100 milisegundos. Para ello debemos realizar un total de 800.000 instrucciones o ciclos ya que:

$$800.000 = \frac{100ms}{0.125\times10^{-3}ms}$$

Para esto realizamos una serie de loops o repeticiones, ya que no resulta viable escribir manualmente tantas veces lo mismo. A continuación haremos un ejemplo de cómo se consigue esto.

```
    ldi  r18, 2
    ldi  r19, 9
L1: dec  r19
    brne L1
    dec  r18
    brne L1
```

Figura 4.1.9. Ejemplo retardo.

En primer lugar tenemos 3 tipos de instrucciones distintas:

LDI: Asignamos un valor inmediato a un registro.

BRNE: Salto hacia otra parte del código si el resultado de la última operación es distinto de 0.

DEC: Disminuye en una unidad el registro.

Entramos al retardo, asignamos al registro r18 un valor de 2 y al r19 un valor de 9. Acto seguido disminuimos en una unidad el valor de r19, comprobamos si el valor de la última operación es 0. Al no serlo volvemos a la etiqueta L1 y volvemos a disminuir el valor de r19 un total de 9 veces. La novena vez el valor del registro es 0 por lo que seguimos a la siguiente instrucción. Disminuimos el valor de r18 haciendo que sea igual a 1, ya que no es 0 volvemos a saltar a L1. Disminuimos el valor de r19 (anteriormente en 0), en una unidad, ya que los registros miden valores enteros positivos del 0 al 255, tendremos que r19 será 255. Se volverá a realizar el bucle hasta llegar a 0 y salir de él. Disminuimos nuevamente el valor de r18 siendo esta vez 0 y así finalizando el retardo. Ahora procederemos a calcular el tiempo del mismo.

Sabiendo que BRNE son dos ciclos cuando se cumple la condición (comparación + salto) y 1 ciclo cuando no lo hace, tenemos la siguiente fórmula:

$$Ciclos = [255\times 3 + 1\times (1+1)]N\times 1 + 8\times 3 + 8\times 1 + 1$$

Siendo N el número de veces que se repite el primer bucle cuando el registro es igual a 0, 1 en nuestro caso. Ya que el registro que mide el número de veces que se repite es igual a 2 y solo en la segunda condición se cumple que es igual a 0 en la etiqueta L1. Eso hace un total de 800 ciclos, sabiendo que cada ciclo tarda 0.125 µs el delay será de 100 µs. Para realizar la creación del mismo solo debemos realizar el cálculo inverso.

4.2. Programando leds.

Ahora entraremos a explicar los distintos códigos utilizados para programar cada componente, al principio empezaremos con códigos sencillos y conforme avancemos en el proyecto continuaremos con códigos extensos y complicados.

```
.def registroUsoGeneral = r16


Start:

    ; Inicializamos la pila

    ldi registroUsoGeneral, high(RAMEND)
    out SPH, registroUsoGeneral
    ldi registroUsoGeneral, low(RAMEND)
    out SPL, registroUsoGeneral

    ; Cargamos en el registro r16 el valor 0x11111111

    ldi registroUsoGeneral, 255

    ; Configuramos todos los puertos como pines de salida

    out DDRA, registroUsoGeneral
    out DDRB, registroUsoGeneral
    out DDRC, registroUsoGeneral
    out DDRD, registroUsoGeneral

bucle0:

    ; Encendemos los leds asociados a los pines del PORTA y apagamos el resto
    ldi registroUsoGeneral, 0b11111111
    out PORTA, registroUsoGeneral
    ldi r16, 0b00000000
    out PORTB, registroUsoGeneral
    out PORTC, registroUsoGeneral
    out PORTD, registroUsoGeneral
    rcall Retardo
```

Figura 4.2.a Código programación de leds.

El funcionamiento del código es bastante sencillo, lo primero que haremos será incluir la librería del microcontrolador y definir el nombre del registro a utilizar, a través la directiva "**.def**". Esto se debe a que cuando el código se haga muy extenso podríamos confundir los distintos registros utilizados y para ir creando una buena práctica, aunque al principio no sea relevante. A continuación procedemos a declarar cada puerto como salida ya que cada uno está asociado a una fila de leds. Encendemos una fila de leds y apagamos el resto de filas. Llamamos a la rutina de retardo y realizamos la misma acción encendiendo el resto de puertos.

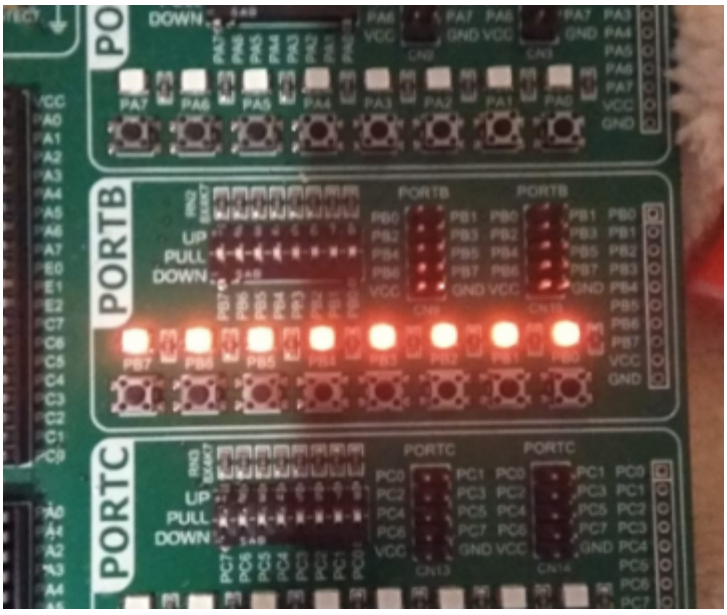


Figura 4.2.b Encendido de leds.

4.3. Programando display a 7 segmentos.

Para la programación del display, se propuso realizar un código en el cuál se realizara una cuenta del 0 al 9, pero en vez de encender un solo display

encenderemos primero el de la derecha, luego el siguiente y así sucesivamente hasta llegar al último de la izquierda e ir hacia atrás. Con este código sencillo podríamos trabajar el encendido y apagado de los distintos displays, además encenderemos todos conjuntamente para mostrar el número 0. El código es bastante fácil, su dificultad principal se basa en saber cómo escribir el número deseado, lo que no es demasiado complicado si se tiene conocimiento al respecto.

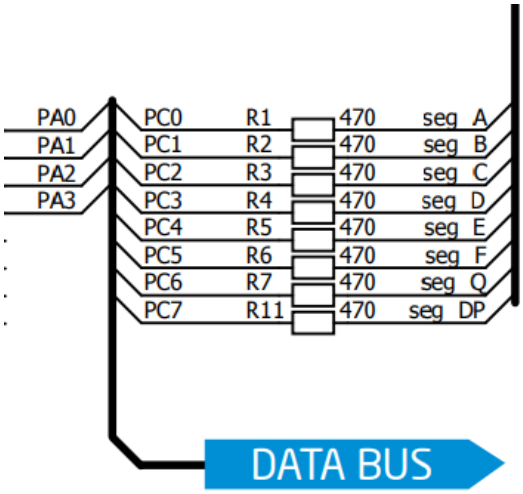


Figura 4.3.a Conexiones del display.

Observamos que cada segmento está controlado por un bit del puerto C de nuestro microcontrolador. Esto quiere decir que si deseamos mostrar en el display un 0 no podemos enviar por el puerto C un 0 simplemente, ya que esto lo que haría sería apagar todo el display.

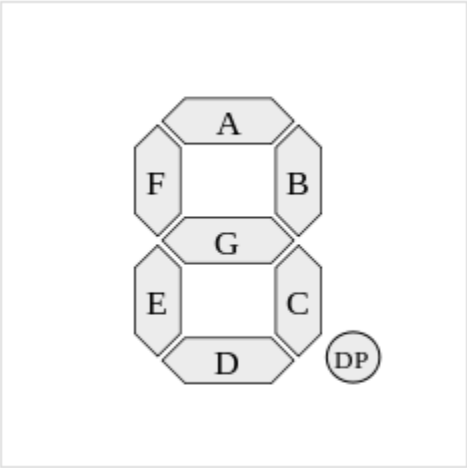


Figura 4.3.b Segmentos del display.

Por ejemplo, si deseamos mostrar un 0, debemos encender los segmentos A, F, E, D, C y B del display. La salida del puerto C quedaría como 0x00111111. Siendo A el bit menos significativo del puerto y G el segundo más significativo.

Usando subrutinas para encender los distintos números quedaría de la siguiente forma:

```
Numero0:

    ldi r16, 0b00111111
    out PORTC, r16
    ret
```

Figura 4.3.c Subrutina para mostrar el número 0.

A través del puerto A controlamos el display que deseamos encender.

4.4. Programando la pantalla LCD.

A continuación explicaremos el proceso para programar la pantalla LCD. Para ello haremos uso de la hoja de características o datasheet del componente. El

objetivo es mostrar el mensaje “TFG Alvaro Ivan”, de forma que se vayan mostrando las letras de una en una para simular una especie de escritura. Una vez escrita la última letra se esperará un segundo y se mostrará todo el mensaje de golpe.

El primer paso consiste en declarar si nuestros pines serán de entrada o de salida, ya que no vamos a recibir ninguna respuesta de la LCD u otro componente externo declaramos los puertos A, C y D como salida. Aunque solo serán usados los 4 bits más significativos del puerto C para el envío de información, el bit 2 del puerto A para activar el modo línea y el bit 6 del puerto D para el envío de información.

```
ldi regEnvioInfor, 0b11111111
out DDRA, regEnvioInfor
; Establecemos el puerto C como salida
; aunque solo será utilizado los 4 bits más
; significativos
out DDRC, regUsoGeneral
out DDRD, regUsoGeneral
```

Figura 4.4.a Instrucciones para inicializar los puertos.

A continuación procedemos a inicializar la configuración de la LCD. Hay que tener en cuenta que la pantalla está configurada en modo 4 bits, esto quiere decir que debemos realizar dos veces el envío de información a través de los bits más significativos del puerto C.

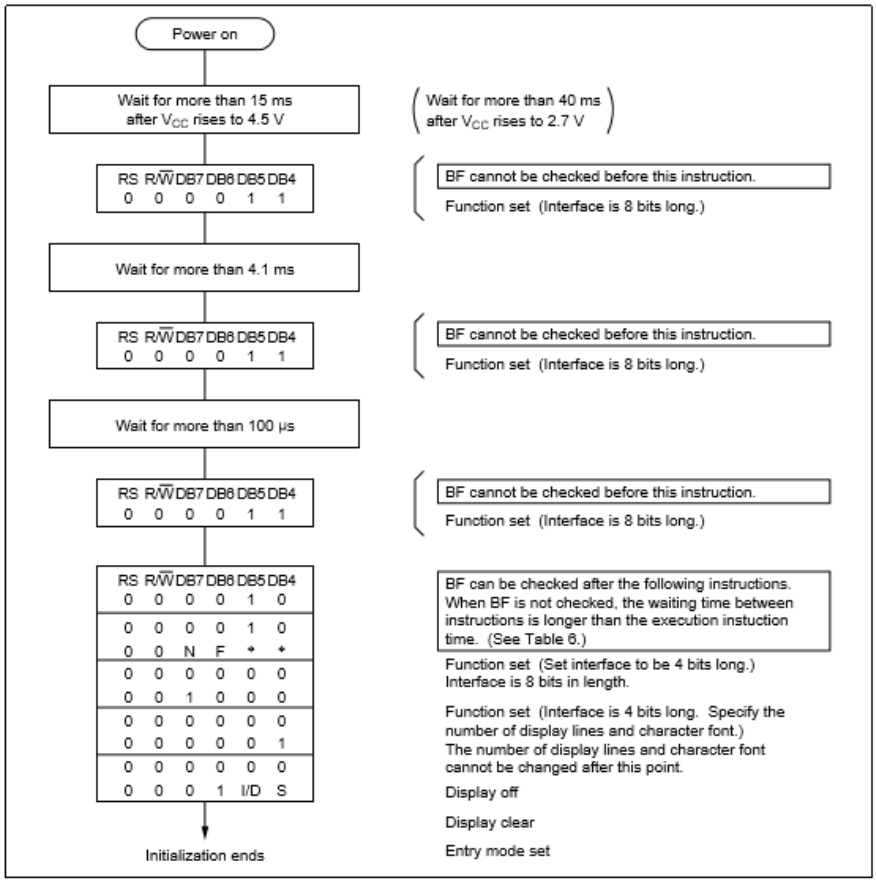


Figura 4.4.b Configuración inicial de la LCD, datasheet.

Comenzamos la configuración a través de un retardo para permitir la alimentación del display y nos disponemos a enviar información a la pantalla.

Para realizar los envíos debemos en primer lugar setear el bit 6 del puerto D que es el encargado de habilitar o deshabilitar el el modo línea de la LCD. Una vez se encuentra activo este pin, enviamos la información a través del puerto C y desactivamos el bit para visualizar los cambios.

```
SREnviarInfo:

    sbi PORTD, 6
    out PORTC, regEnvioInfor
    cbi PORTD, 6
    ret
```

Figura 4.4.c Subrutina envío de información.

```
ConfiguracionLCD:

    rcall Loop50ms
    ldi regEnvioInfor, 0b00110000
    rcall SREnviarInfo
    ldi regEnvioInfor, 0b00110000
    rcall SREnviarInfo
    rcall Loop100ms
    ldi regEnvioInfor, 0b00110000
    rcall SREnviarInfo
    rcall Loop100ms
    ldi regEnvioInfor, 0b00100000
    rcall SREnviarInfo
    ldi regEnvioInfor, 0b11000000
    rcall SREnviarInfo
    rcall Loop100ms
    ldi regEnvioInfor, 0b00000000
    rcall SREnviarInfo
    ldi regEnvioInfor, 0b00010000
    rcall SREnviarInfo
    rcall Loop100ms
    ldi regEnvioInfor, 0b00000000
    rcall SREnviarInfo
    ldi regEnvioInfor, 0b01100000
    rcall SREnviarInfo
    rcall Loop100ms
    ldi regEnvioInfor, 0b00000000
    rcall SREnviarInfo
    ldi regEnvioInfor, 0b11000000
    rcall SREnviarInfo
    rcall Loop100ms
```

Figura 4.4.d Envío de información a la LCD.

Ahora activando el pìn Rs de la LCD (bit 2 del puerto A del microcontrolador), estamos listos para poder enviar mensajes. Nuevamente con la hoja de características, buscamos los distintos símbolos que queremos representar y los enviamos a la pantalla en dos tandas. Por ejemplo el espacio en blanco después de la G.

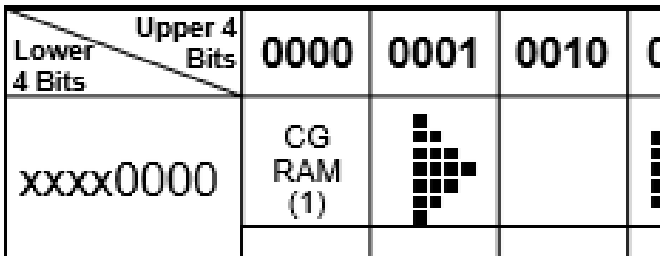


Figura 4.4.e. Tabla para la visualización de un espacio en blanco.

Primero enviaremos los 4 bits más significativos y luego los otros 4:

```
/*Espacio*/
ldi regEnvioInfor,0b00100000
rcall SREnviarInfo
ldi regEnvioInfor,0b00000000
rcall SREnviarInfo
rcall Loop100ms
```

Figura 4.4.f. Ejemplo envío de información a la LCD.

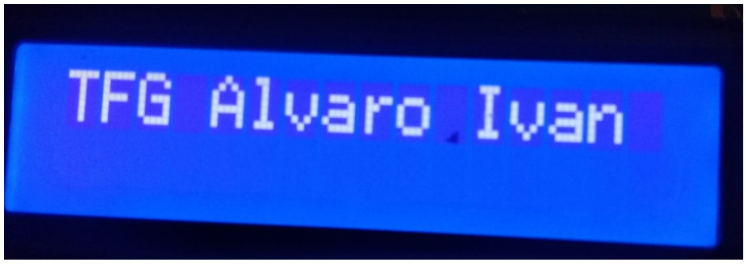


Figura 4.4.g. Mensaje a través de una pantalla LCD.

4.5. Programando la pantalla GLCD.

Continuamos en nuestro aprendizaje esta vez a través de la programación de la pantalla Gráfica de Cristal Líquido, previamente comenzaremos explicando su funcionamiento. La GLCD así como su hermana la LCD deberá ser previamente configurada al iniciar el programa. En las especificaciones del componente vienen los pasos a seguir para ello. A diferencia del componente anterior, este no solo poseerá una subrutina de escritura sino una también de configuración. A través de esta subrutina podemos configurar varias opciones pero será principalmente utilizada para inicializar la pantalla y para elegir la posición inicial donde queremos pintar los píxeles. Para ello, guiándonos por la hoja de especificaciones, debemos enviar en modo configuración la especificación del pixel del eje vertical que querremos pintar con 11XXXXXX y 10111YYY. Siendo las X el valor en binario del bit en el que empezaremos en el eje horizontal (desde el 0 al 64) y las Y el número en binario de la página que queremos pintar (de la 0 a la 7).

```
; Subrutina utilizada para pasar a modo configuracion
SRModoConfiguracion:

    ; Desactivamos los bits 3 y 2
    ; del puerto A
    cbi PORTA,3
    cbi PORTA,2

    ; Activamos el pin E
    sbi PORTD,6
    rcall Retardo

    ; Desactivamos el pin E
    cbi PORTD,6
    rcall Retardo
    ret
```

```
; Subrutina encargada de enviar información a la
; pantalla. A diferencia del anterior activamos
; el bit 2 del puerto A, en vez de desactivarlo.
SREnviarInfor:

    ; Activamos el bit 2 del puerto A
    sbi PORTA,2
    cbi PORTA,2

    ; Activamos el pin E
    sbi PORTD,6
    rcall Retardo

    ; Desactivamos el pin E
    cbi PORTD,6
    rcall Retardo
    ret
```

Figura 4.5.a Subrutinas para la configuración y el envío de información.

```
; Configuramos para comenzar en el primer pixel izquierdo de la pantalla
ldi regUsoGeneral,0b01000000
out PORTC, regUsoGeneral
rcall SRModoConfiguracion
; Configuramos para comenzar en la primera página
ldi regUsoGeneral,0b10111000
out PORTC, regUsoGeneral
rcall SRModoConfiguracion
```

Figura 4.5.b. Ejemplo de configuración para empezar en el pixel (0,0) de la matriz.

Como se mencionó en el capítulo de **Hardware**, la pantalla posee dos controladores, uno para activar el lado izquierdo de la pantalla y otro para activar el lado derecho. Nosotros debemos seleccionar qué pantalla deseamos dibujar. Si ambos controladores están activos a la vez, lo que enviemos por el puerto C se imprimirá en ambas caras.

```
; Activar controlador izquierdo
SRControladorIzquierdo:
    cbi PORTB,0
    sbi PORTB,1
    ret
```

```
; Activar controlador derecho
SRControladorDerecho:
    sbi PORTB,0
    cbi PORTB,1
    ret
```

Figura 4.5.c. Subrutinas para la activación de los controladores.

El funcionamiento para escribir en la pantalla es el siguiente: especificamos tanto la página como el píxel donde deseamos comenzar, enviamos por el puerto C el byte completo de la página a través de la subrutina de envío. Siendo los 1 encender el pixel y los 0 apagarlos. La GLCD mueve automáticamente la posición del eje Y en 1, por lo que no tenemos que volver a especificar la ubicación de escritura y continuamos escribiendo el resto de posiciones hasta hacerlo con toda la pantalla.

```
; Subrutina utilizada para pasar a modo configuracion
SRModoConfiguracion:

    ; Desactivamos los bits 3 y 2
    ; del puerto A
    cbi PORTA,3
    cbi PORTA,2

    ; Activamos el pin E
    sbi PORTD,6
    rcall Retardo

    ; Desactivamos el pin E
    cbi PORTD,6
    rcall Retardo
    ret
```

```
; Subrutina encargada de enviar información a la
; pantalla. A diferencia del anterior activamos
; el bit 2 del puerto A, en vez de desactivarlo.
SREnviarInfor:

    ; Activamos el bit 2 del puerto A
    sbi PORTA,2
    cbi PORTA,2

    ; Activamos el pin E
    sbi PORTD,6
    rcall Retardo

    ; Desactivamos el pin E
    cbi PORTD,6
    rcall Retardo
    ret
```

Figura 4.5.d. Subrutinas para la configuración y el envío de información.

4.5.1 Uso de Tablas para mostrar imágenes.

Esto a la hora de mostrar imágenes resulta algo bastante tedioso ya que tenemos que configurar 8192 píxeles en total. Por ello se planteó buscar una forma más rápida a la hora de realizar esta acción. Se encontró la opción de la utilización de tablas de búsqueda. Estas tablas son una forma de compactar información para usarlas de una forma más cómoda.

Para el dibujo de una imagen en la GLCD, transformamos dicha imagen en bytes con LCDAssistants, agrupamos los datos añadiendo una etiqueta y definimos los datos con la directiva "**.db**". Luego leeremos estas tablas con la utilización de punteros, que son pares de registros predefinidos en ensamblador.

```
ULL:

    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0xC0, 0xFC, 0xFF, 0xFF, 0xFF, 0xFF, 0x3F, 0x03, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x80, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF
    .db 0xFF, 0xFF, 0x1F, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
```

Figura 4.5.1. Ejemplo de tabla.

La tabla tiene una medida de 16bytesx64líneas, haciendo un equivalente en píxeles de 128x64. El primer paso para esta labor es inicializar los punteros con los valores de las tablas, dado que los punteros son de 16 bits hay que multiplicar por 2 su valor.

```
; Apuntamos con el puntero a la dirección
; de la etiqueta ULL
ldi ZH,HIGH(ULL*2)
ldi ZL,LOW(ULL*2)
```

Figura 4.5.1.a Apuntando con el puntero a la dirección de la tabla.

Luego realizaremos la llamada a la subrutina de pintado. Esta inicializa los valores de los registros encargados de los ejes horizontal y vertical. Llamamos a las rutinas encargadas de pintar cada parte de la pantalla (controlador derecho y controlador izquierdo). Incrementamos en uno el valor del puntero vertical para pasar de páginas, una vez cubiertas las 8 páginas habremos acabado.

```
SRPintar:

    push regPunteroVertical
    push regPunteroHorizontal

    ldi regPunteroHorizontal, 0
    ldi regPunteroVertical, 0

SiguienteLinea:

    rcall SRPintarIzquierda
    rcall SRDibujar

    rcall SRPintarDerecha
    rcall SRDibujar

    inc regPunteroVertical

    cpi regPunteroVertical,8
    brne SiguienteLinea

    pop regPunteroHorizontal
    pop regPunteroVertical
    ret
```

Figura 4.5.1.b Subrutina para pintar tablas en la GLCD.

Las subrutinas SRPintarIzquierda y SRPintarDerecha realizan la misma función solo que para los distintos controladores. Con ellas primero llamamos al controlador deseado, configuramos la pantalla para comenzar en el pixel 1 de la posición horizontal. Sumamos la dirección del puntero a un registro para configurar la página que deseamos pintar y retornamos de la subrutina.

```
SRPintarDerecha:

    push regUsoGeneral
    rcall SRControladorDerecho
    ldi regUsoGeneral,0b01000000
    out PORTC, regUsoGeneral
    rcall SRModoConfiguracion
    ldi regUsoGeneral,0b10111000
    or regUsoGeneral,regPunteroVertical
    out PORTC, regUsoGeneral
    rcall SRModoConfiguracion
    pop regUsoGeneral
    ret
```

Figura 4.5.1.c Subrutina para seleccionar la página y la línea a pintar de la pantalla derecha.

Por último la subrutina SRDibujar, lo que hace es, cargar a una constante de la memoria del programa la dirección apuntada por Z. Enviamos por la salida el byte correspondiente (corresponderá a la página deseada y en la línea deseada ya que se configura con anterioridad). Aumentamos la dirección del puntero para dibujar la siguiente línea. Y así consecutivamente hasta llegar a las 64 líneas correspondientes del controlador de la pantalla. Luego se hará lo mismo para el controlador derecho.

```
SRDibujar:
    ; Cargamos los registros en la pila
    push r0    .
    push regPunteroHorizontal

SRDibujar1:
    lpm
    out PORTC, r0
    rcall SREnviarInfor
    adiw ZL,1
    inc regPunteroHorizontal
    cpi regPunteroHorizontal, 64
    brne SRDibujar1
    pop regPunteroHorizontal
    pop r0

    ret
```

Figura 4.5.1.d Subrutina para seleccionar la página y la línea a pintar de la pantalla derecha.

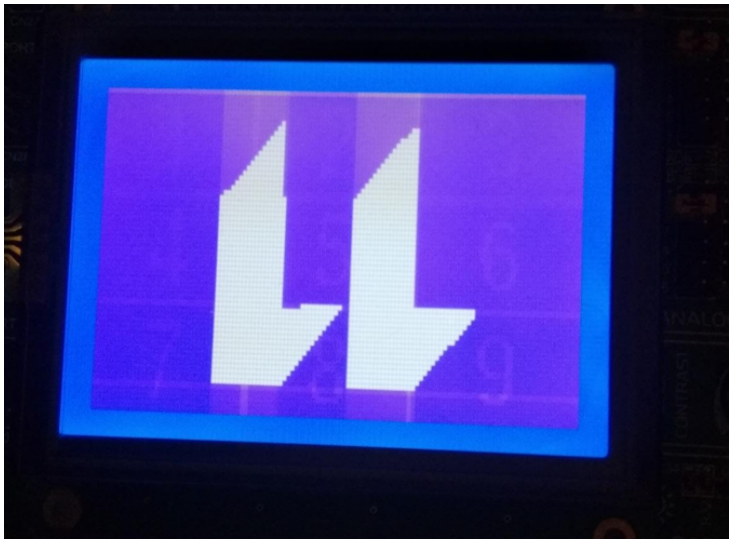


Figura 4.5.1.e Imagen Logo ULL en la GLCD.

4.6. Programando el panel táctil.

Procedemos con la programación del panel táctil y el programa final propuesto para el proyecto. Sabemos que este panel envía cierto voltaje por los pines 0 y 1 del puerto A según la pulsación en la componente la vertical y horizontal respectivamente.

Es decir, activando el bit 2 del puerto A leeremos un voltaje analógico a través del pin 1 según dónde se pulse la pantalla. Si pulsamos en la página superior enviará un voltaje mayor y si pulsamos en la inferior enviará un voltaje menor. En caso opuesto, si activamos el pin 3, leeremos un voltaje menor cuanto más a la izquierda se pulse y un voltaje superior cuanto más a la derecha. Para conseguir que se lea correctamente, mientras está activado uno de los bits debe estar desactivado el siguiente.

Los inconvenientes surgidos eran en primer lugar que desconocemos los voltajes de entrada y en segundo lugar no sabíamos cómo tratar las entradas analógicas en los pines.

4.6.1 Datos de entrada a los pines 0 y 1 del microcontrolador.

Para resolver este problema realizamos dos programas muy sencillos que activasen las distintas lecturas y con un multímetro leíamos los datos del bit correspondiente. Realizamos la configuración inicial de la pantalla previamente.

```
; Lectura Voltajes Eje Horizontal
cbi PORTA,2
sbi PORTA,3
Lectura:
    rjmp lectura


 ; Lectura Voltajes Eje Vertical
 cbi PORTA,3
 sbi PORTA,2
 Lectura:
     rjmp lectura
```

Figura 4.6.1. Programa para la lectura de los voltajes.

Lectura Eje Horizontal	Lectura Eje Vertical
0.35 – 4.01 V	0.68 – 4.04 V

La calidad del multímetro utilizado no era muy buena, esto provocaba que el valor variase. Se intentó utilizar una media de valores

4.6.2 Traducción de los voltajes analógicos.

Resuelto la primera duda nos disponemos a resolver la segunda. Indagando en el tema descubrimos que existen dos registros que nos dan la posibilidad de realizar una conversión analógica-digital. Estos registros son los llamados ADMUX y ADCSRA. El primero tiene la labor de realizar la configuración de la conversión, debemos elegir un voltaje de referencia además del pin a través del cual haremos la conversión, en nuestro caso serán el 0 y el 1 del puerto A. El segundo registro es el responsable de realizar la conversión, este registro necesita un tiempo para realizar la conversión y avisa cuando esta haya acabado a través de su bit 4 por lo que deberemos generar un bucle que espere el resultado de la operación. El convertidor guarda este valor en dos registros especiales llamados ADCH, que posee los dos bits más significativos y ADCL, que contiene los 8 bits menos significativos. El voltaje de referencia utilizado en nuestro proyecto será de 5 voltios ya que es la opción que más se acerca a nuestras necesidades.

4.7. Programa final – Desbloqueo de pantalla táctil a través de una contraseña numérica.

El programa final consistirá en los siguientes pasos:

- Mostramos el logo de la universidad y esperamos unos segundos.
- Mostramos el teclado de los números para poder introducir la contraseña.
- Al pulsar en un número, mostraremos que el número ha sido pulsado.
- Después de cuatro pulsaciones mostramos un mensaje.
- Si la contraseña ha sido incorrecta mostraremos la imagen de FAILED.
- Si la contraseña ha sido correcta mostraremos la imagen de UNLOCKED.
- Volvemos al inicio.

En primer lugar inicializamos la GLCD, tal y como hemos explicado en capítulos anteriores y enviamos por pantalla el logo de la ULL a través del retardo esperamos un par y de segundos, limpiamos la pantalla y mostramos la pantalla de los números.

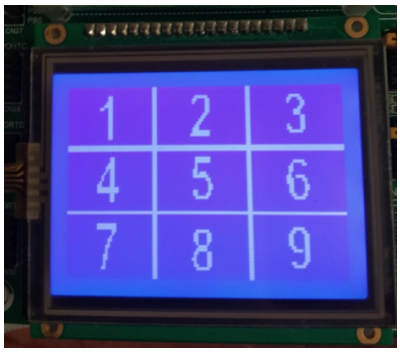


Figura 4.7.a Teclado numérico.

Para limpiar la pantalla y que no se superpusieran una imagen sobre la otra se creó una subrutina que básicamente lo que hace es imprimir por pantalla una tabla de una imagen completamente en blanco. De esta forma apagaremos todos los píxeles evitando problemas.

El siguiente paso a realizar es la detección de pulsaciones a través de la pantalla táctil. Para ello debemos tener en cuenta varios aspectos referentes al convertidor analógico digital.

Debemos configurar la conversión para la utilización de un voltaje de referencia de 5 v y que nos lea el pin deseado. En este caso sería el pin 1 del puerto A.

```
admuxConfig:

    ldi regUsoGeneral,0b01000001
    out admux,regUsoGeneral
    clr regAdchVertical
    cbi PORTA,2
    sbi PORTA,3
    sbi adcsra,6
```

Figura 4.7.b Programa para la lectura de los voltajes

Una vez configurado la medida procedemos a realizar un bucle que esté constantemente a la espera de que acabe la conversión. Se sabe que a través del bit 4 del convertidor adcsra se determina cuando finaliza la conversión por lo que comprobamos dicho bits en un bucle. Una vez finalizada la conversión, guardamos los datos en los registros y volvemos a realizar la medición para comprobar que ha sido correcta comparando ambas mediciones. Si coinciden se continuará con el programa, de lo contrario se volverá a medir. El paso de comprobación se realizó debido a que a veces la conversión daba problemas y pintaba pixeles de forma aleatoria debido probablemente a un voltaje de fuga o a una mala pulsación.

```
bucleComprobacionVertical:

    sbis adcsra,4
    rjmp bucleComprobacionVertical
    rcall Retardo4ms
    in regUsoGeneral, adcl
    in regAdchVertical, adch
    mov r18,regUsoGeneral
    sbi adcsra,6
```

```
bucleComprobacionVertical2:

    sbis adcsra,4
    rjmp bucleComprobacionVertical2
    in regUsoGeneral, adcl
    in regAdchVertical, adch

    cp regUsoGeneral,r18
    brne bucleConfig
```

Figura 4.7.c Bucles medida de voltaje según la vertical y comprobación.

Una vez tenemos los datos de ADCL y ADCH, tenemos que trabajar con ellos para descubrir en donde se ha pulsado. Sabiendo que el máximo voltaje proporcionado por el convertidor admux es de 5 voltios y este es representado por 11 ADCH y 11111111 ADCL, siendo este número 1023 en decimal, debemos descubrir los valores de los límites de la pantalla para los rangos de voltaje que teníamos. A partir de tres, averiguamos los valores proporcionados por la pantalla según la posición.

Lectura Eje Horizontal	Lectura Eje Vertical
0.35 – 4.01 V	0.68 – 4.04 V
70 – 820 valor ADC	139 – 827 valor ADC
00 (ADCH) 01000110 (ADCL) a 11(ADCH) 00110100 (ADCL)	00 (ADCH) 10001011 (ADCL) a 11(ADCH) 00111011 (ADCL)

Ya sabemos los distintos límites de la pantalla, ahora necesitamos saber los rangos de valores de cada columna y cada fila para poder determinar qué número se ha pulsado.

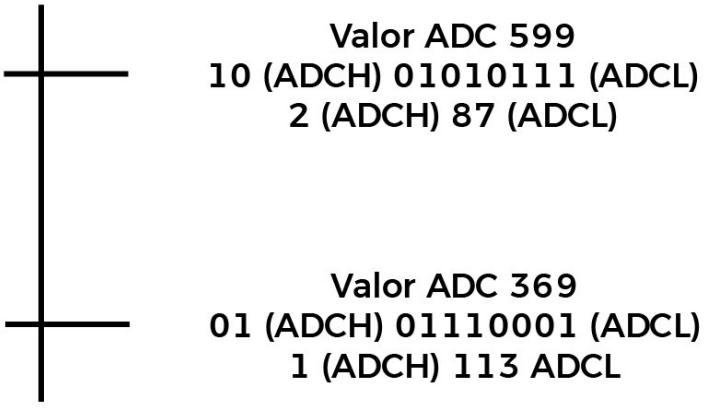


Figura 4.7.d Valores límites de filas.

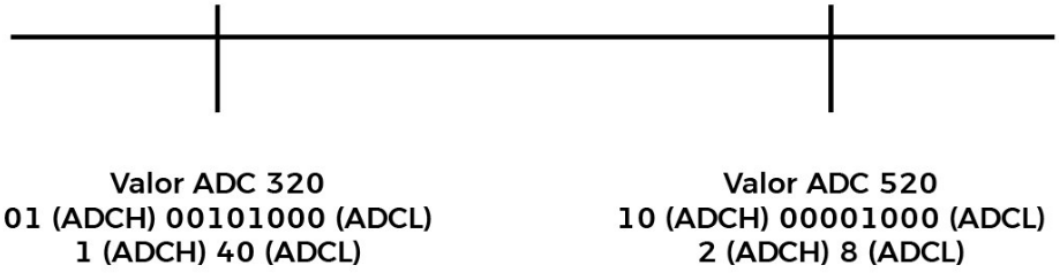


Figura 4.7.e Valores límites de columnas.

Conociendo los límites de cada fila procedemos a realizar un filtro por filas, comprobando el valor primero de ADCH.

```
; Realizamos un primer filtro a través de los valores de adch

cpi regAdchVertical,1
breq posibleFila23
cpi regAdchVertical,2
breq posibleFila12
cpi regAdchVertical,3
breq primeraFila
rjmp posibleFila3
```

Figura 4.7.f. Filtrado de filas por ADCH.

Debido a que para los valores 1 y 2 de ADCH podemos tener la posibilidad de haber pulsado en dos filas distintas, entramos en una etiqueta que comprobará el valor de ADCL para determinar exactamente cuál ha sido pulsado. Utilizando los valores límites anteriormente calculados realizamos este filtro.

```
; Una vez comparado el valor de adch compararemos el valor de
; adcl para saber qué fila ha sido presionada y proceder a leer
; el eje vertical de la pulsación

posibleFila3:
    cpi regUsoGeneral, 139
    brlo admuxConfig
    rjmp terceraFila

posibleFila23:

    cpi regUsoGeneral,113
    brlo terceraFila
    rjmp segundaFila

posibleFila12:

    cpi regUsoGeneral, 87
    brlo segundaFila
    rjmp primeraFila
```

Figura 4.7.g. Filtrado de filas por ADCL.

Una vez detectada la fila que se ha pulsada se procederá a guardar su valor en un registro, realizaremos el mismo filtrado anterior pero está vez leyendo la componente horizontal de la pantalla.

```
; Asignamos el valor correspondiente a cada fila al registro
; encargado

primeraFila:
    ldi regFila,1
    rjmp bucleHorizontal

segundaFila:
    ldi regFila,2
    rjmp bucleHorizontal

terceraFila:
    ldi regFila,3
    rjmp bucleHorizontal
```

Figura 4.7.h. Almacenando en un registro la fila pulsada.

Ya conseguimos saber la fila y la columna que se ha pulsado, solo queda encender el número correspondiente a esa fila

```
primeraColumna:

    ; Comparación número 1

    cpi regFila, 1
    breq encenderNumero01

    ; Comparación numero 4

    cpi regFila, 2
    breq encenderNumero04

    ; Comparación numero 7
    cpi regFila, 3
    breq encenderNumero07
```

Figura 4.7.i. Selección de número pulsado según fila y columna.

Posteriormente a leer la tabla asociada a encender cada número debemos realizar dos funciones. La primera función es referente al desbloqueo, se coloca en cada número que deseamos que forme parte del pin la subrutina de desbloqueo, para el primer número asignamos SRDesbloqueo, para el segundo número SRDesbloqueo1, etc. Este paso no se realizará en caso de que no queramos asociar un número al desbloqueo. El segundo paso es llamar a la subrutina encargada de contar el número de veces que se ha pulsado. Esta subrutina aumentará el registro de pulsaciones y comprobará si ha llegado a 4. En caso de llegar a dicho número se compara que el registro de desbloqueo también se encuentra en 4. Si se cumple la condición desbloquea la pantalla de lo contrario muestra la imagen de pin fallido.

```
SREncenderNumero4:

    ldi ZH,HIGH(Numero4*2)
    ldi ZL,LOW(Numero4*2)
    rcall SRPintar
    rcall SRDesbloqueo1
    rcall SRContador
    ret
```

Figura 4.7.j. Encendido de números.

El último aspecto a tener en cuenta es que debemos establecer un orden para el pin, es decir que la pantalla se desbloquee si los números se han pulsado en orden y no solo si se han pulsado los números correctos. Para ello se utiliza distintas subrutinas de desbloqueo.

```
SRDesbloqueo:
    cpi regDesbloqueo,0
    breq SRDesbloqueoAumento
    ret

SRDesbloqueo1:
    cpi regDesbloqueo,1
    breq SRDesbloqueoAumento
    ret

SRDesbloqueo2:
    cpi regDesbloqueo,2
    breq SRDesbloqueoAumento
    ret
SRDesbloqueo3:
    cpi regDesbloqueo,3
    breq SRDesbloqueoAumento
    ret

SRDesbloqueoAumento:
    inc regDesbloqueo
    ret
```

Figura 4.7.k. Subrutina de desbloqueo.

El funcionamiento es el siguiente, al pulsar un número cualquiera dentro del pin, este llamará a su subrutina de desbloqueo correspondiente, esta subrutina primero comprobará que el registro de desbloqueo, si tiene el valor correspondiente al que debería aumentará el valor en 1, de lo contrario no hará nada. De esta forma se establece un orden en las pulsaciones. Por último el contador llamará a cada subrutina correspondiente una vez pulsados los 4 números.

```
SRContador:
    inc regVecesPulsado

    ; Comprueba el valor del registro de desbloqueo, si es correcto
    ; llama a la etiqueta desbloquear.

    cpi regDesbloqueo,4
    breq SRDesbloquear

    ; Comprueba el valor del registro de pulsaciones, en caso de haber
    ; pulsado 4 veces saltará a la etiqueta Fallo

    cpi regVecesPulsado,4
    breq SRFallo
    ret
```

Figura 4.7.l. Subrutina SRContador.

Capítulo 5.
Problemas y soluciones durante el desarrollo

Capítulo 5. Problemas y soluciones durante el desarrollo

Durante el desarrollo del proyecto se cometieron una serie de errores ya sea por falta de experiencia o fallos de compresión. Estos se fueron solucionando y se mencionan a continuación los más relevantes.

5.1. Switchs.

A la hora de trabajar con la placa debemos tener en cuenta que para utilizar los distintos periféricos debemos configurar los Switchs de la misma. Teniendo una mala configuración de los mismos puede que la placa no se comporte como realmente debería. Este problema provocó gran pérdida de tiempo durante la programación del display a 7 segmentos ya que provocaba que el programa hiciera acciones aleatorias distintas de la esperada y se supuso que era un error de programación cuando verdaderamente era de configuración de la placa.

Figura 5.1. Error en los switchs.

5.2. Errores en el envío de archivos HEX.

Un error que cabe destacar es que a la hora del envío del archivo HEX, tuve un problema que me ocupó una gran cantidad de tiempo perdido. Este problema se debe a que no tenía correctamente seleccionadas las opciones del AVRFlash. Debemos fijarnos que esté desmarcada la casilla de **JTAGEN** en la pantalla principal del programa, ya que esta opción a la hora de cargar el archivo lo carga pero el programa se comporta de una forma distinta a lo que debería. Esto genera una gran pérdida de tiempo ya que al cargarse el archivo aparentemente

de forma correcta, nos induce a pensar que ha habido un fallo en la programación de nuestro proyecto. Cuando son proyectos cortos no supone un problema, pero en caso de que este sea largo y complicado puede llegar a producir grandes pérdidas de tiempo. Esta opción viene marcada por defecto cuando se instala el programa por primera vez.

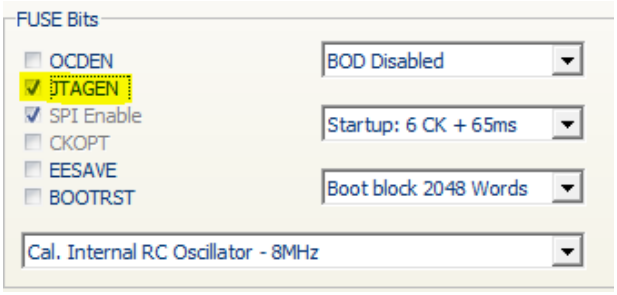


Figura 5.2. Errores selección de archivos con AVRFlash.

5.3. PUSH y POP.

A la hora de entrar en las subrutinas, cuando modificamos los datos debemos utilizar las instrucciones PUSH y POP para almacenar los datos en la pila y más tarde recuperarlos, en caso de que no queramos que se produzcan los cambios, tan solo de forma temporal. El inconveniente de esto es que se debe tener claro en qué orden entran a la pila para saber en qué orden debemos sacarlos de la misma.

```
Loop1s:
    push 25
    push 26
    push 27
    .
    .
    pop 25
    pop 26
    pop 27
    ret
```

```
Loop1s:
    push 25
    push 26
    push 27
    .
    .
    pop 27
    pop 26
    pop 25
    ret
```

Figura 5.3. Errores push y pop.

La imagen izquierda es la forma incorrecta de hacerla y la de la derecha la correcta. Debemos sacar los registros en orden inversos al introducido.

5.4. Retorno de subrutinas y etiquetas.

A la hora de llamar a una subrutina debemos realizar la instrucción **ret** para salir de la misma. En caso de no ejecutarse tendremos problemas a la hora de que el programa siga el orden que deseamos. Otro fallo es que si usamos **ret** desde una etiqueta, en vez de desde una subrutina comenzaremos en la primera línea del programa en vez de donde debe realizarse. Diferenciar bien entre lo que llamamos subrutina y lo que llamamos etiquetas.

5.5. Instrucciones BRXX.

Las instrucciones de tipo BRXX son aquellas que realizan saltos relativos en función a una condición. Se cometieron ciertos errores debido a que a veces estos saltos eran demasiado largos. Esta instrucción nos permite hacer un salto como máximo de 64 instrucciones. La forma de solucionarlo fue crear una etiqueta cerca de la instrucción que realizara otra instrucción llamada RJMP que no tiene tantas limitaciones.

```
terceraColumna:
    cpi regFila, 1
    breq encenderNumero03

    cpi regFila, 2
    breq encenderNumero06

    cpi regFila, 3
    breq encenderNumero09

    ldi regFila,0
    rjmp inicio
```

```
encenderNumero03:
    rcall SREncenderNumero3
    rjmp inicio
encenderNumero06:
    rcall SREncenderNumero6
    rjmp inicio
encenderNumero09:
    rcall SREncenderNumero9
    rjmp inicio
```

Figura 5.5. Ejemplo solución de errores por salto relativo.

Capítulo 6.
Presupuesto

Capítulo 6. Presupuesto

Para calcular el presupuesto del proyecto se buscó entre las páginas principales de componentes electrónicos el mejor precio y se encontró una oferta en la página llamada "MIKROE" (https://www.mikroe.com/easyavr). El precio total de la placa con todos sus componentes supone un precio de 107.28 euros.

Además se ha de tener en cuenta el sueldo del programador. Según un estudio de la página compurterhoy.com, un programador europeo cobra una media de 48.000 euros al año, siendo esto 21 euros la hora aproximadamente. Suponiendo un tiempo aproximado 200 horas para la realización de este proyecto, llegamos al siguiente presupuesto:

	Precio
Placa EasyAVRv7	107.28 €
Programador	4200 €
Total	4307.28 €

Capítulo 7.
Conclusiones

Capítulo 7. Conclusiones

Se ha finalizado el proyecto correctamente y conseguido los objetivos propuestos, aprendiendo en el proceso a programar los distintos periféricos de la placa así como la resolución de problemas con el lenguaje de programación Ensamblador. Durante el proceso se ha aprendido distintas formas de solucionar problemas de forma ingeniosa, siendo abordados desde un punto de vista tanto lógico como técnico.

El proyecto se podría utilizar de distintas maneras, por ejemplo para el acceso a una puerta. Conectando el microcontrolador a una cerradura eléctrica, abriéndose en caso de conseguir insertar correctamente la contraseña o bloqueandola de lo contrario.

El proceso ha sido bastante satisfactorio ya que se resolvían los problemas que iban surgiendo y la placa gracias a su precio resulta bastante fácilmente accesible. Además de que existe una gran cantidad de documentación en internet la cuál puede ser consultada fácilmente.

Capítulo 8.
Referencias

Capítulo 8. Referencias

- Instrucciones AVR :

Guía rápida del ensamblador de los microprocesadores ATMEL-AVR. Evelio J. González González, Grupo de Computadores y Control. Universidad de la Laguna

https://microchip.com/webdoc/avrassembler/avrassembler.wb_instruction_list.html

http://www.exa.unicen.edu.ar/catedras/tmicrocon/Material/5_ASM_C.pdf

http://www.dte.us.es/personal/pparra/EdC-T4-AVR-v0-1a-parte

http://www.avrbeginners.net/

- ADMUX:

http://openenergymonitor.blogspot.com/2012/08/low-level-adc-control-admux.html

- Tutoriales ensamblador:

https://www.askix.com/tutorial-de-ensamblador-avr-4.html

- Manejos de tablas:

http://www.todopic.com.ar/foros/index.php?topic=33910.0

https://www.askix.com/tutorial-de-ensamblador-avr-4_7.html

- Manual AVR:

https://download.mikroe.com/documents/full-featured-boards/easy/easyavr-v7/easyavr-v7-manual-v101.pdf

- Definiciones generales:

https://es.wikipedia.org/

Capítulo 9.
Anexos

Capítulo 9. Anexos

9.1 Códigos

Programación de Leds

```
/*
 *  Alvaro Ivan Arcia Gonzalez
 *  Codigo Leds
 */

 .include "m32def.inc"

.CSEG
.ORG $0000
.def regRetardo1 = r17
.def regRetardo2 = r18
.def regRetardo3 = r19
.def registroUsoGeneral = r16


Start:

    ; Inicializamos la pila

    ldi registroUsoGeneral, high(RAMEND)
    out SPH, registroUsoGeneral
    ldi registroUsoGeneral, low(RAMEND)
    out SPL, registroUsoGeneral

    ; Cargamos en el registro r16 el valor 0x11111111

    ldi registroUsoGeneral, 255

    ; Configuramos todos los puertos como pines de salida

    out DDRA, registroUsoGeneral
    out DDRB, registroUsoGeneral
    out DDRC, registroUsoGeneral
    out DDRD, registroUsoGeneral
```

```
bucle0:

    ; Encendemos los leds asociados a los pines del PORTA y apagamos
el resto
    ldi registroUsoGeneral, 0b11111111
    out PORTA, registroUsoGeneral
    ldi r16, 0b00000000
    out PORTB, registroUsoGeneral
    out PORTC, registroUsoGeneral
    out PORTD, registroUsoGeneral
    rcall Retardo

    ; Encendemos los leds asociados a los pines del PORTB y apagamos
el resto
    ldi registroUsoGeneral, 0b11111111
    out PORTB, registroUsoGeneral
    ldi registroUsoGeneral, 0b00000000
    out PORTA, registroUsoGeneral
    out PORTC, registroUsoGeneral
    out PORTD, registroUsoGeneral
    rcall Retardo

    ; Encendemos los leds asociados a los pines del PORTC y apagamos
el resto
    ldi registroUsoGeneral, 0b11111111
    out PORTC, registroUsoGeneral
    ldi registroUsoGeneral, 0b00000000
    out PORTA, registroUsoGeneral
    out PORTB, registroUsoGeneral
    out PORTD, registroUsoGeneral
    rcall Retardo

    ; Encendemos los leds asociados a los pines del PORTD y apagamos
el resto
    ldi registroUsoGeneral, 0b11111111
    out PORTD, registroUsoGeneral
    ldi registroUsoGeneral, 0b00000000
    out PORTA, registroUsoGeneral
    out PORTB, registroUsoGeneral
    out PORTC, registroUsoGeneral
    rcall Retardo
```

```
    ; Apagamos todos los leds
    ldi registroUsoGeneral, 0b00000000
    out PORTA, registroUsoGeneral
    out PORTB, registroUsoGeneral
    out PORTC, registroUsoGeneral
    out PORTD, registroUsoGeneral

    ; Creamos un bucle infinito.
    rjmp bucle0

Retardo:

            ldi regRetardo3, 3

        bucle1:

            ldi regRetardo2, 255

        bucle2:

            ldi regRetardo1, 255

        bucle3:

            dec regRetardo1
            BRNE bucle3
            dec regRetardo2
            BRNE bucle2
            dec regRetardo3
            BRNE bucle1
            ret
```

Programación del Display a 7 segmentos

```
/*
 *  Alvaro Ivan Arcia Gonzalez
 *  Codigo Leds
 */


.include "m32def.inc"

.CSEG
.ORG $0000



 rjmp Start

Start:
    ldi r16,high(RAMEND)     ;Creación de la pila
    out SPH, r16
    ldi r16, low(RAMEND)
    out SPL, r16

    ldi r16, 0b11111111
    out DDRC, r16
    ldi r16, 0b00000000
    out DDRB, r16
    ldi r16, 0b00001111
    out DDRA, r16

    ldi r16, 0b00000001
    out PORTA, r16

Programa:

    sbis PINB, 1

    rjmp Programa

    ldi r16, 0b00000001
    out PORTA, r16

    rcall Numero9
```

```
rcall Retardo

ldi r16, 0b00000010
out PORTA, r16

rcall Numero8
rcall Retardo

ldi r16, 0b00000100
out PORTA, r16

rcall Numero7
rcall Retardo

ldi r16, 0b00001000
out PORTA, r16

rcall Numero6
rcall Retardo

ldi r16, 0b00000100
out PORTA, r16

rcall Numero5
rcall Retardo

ldi r16, 0b00000010
out PORTA, r16

rcall Numero4
rcall Retardo

ldi r16, 0b00000001
out PORTA, r16

rcall Numero3
rcall Retardo

ldi r16, 0b00000010
out PORTA, r16

rcall Numero2
rcall Retardo
```

```
    ldi r16, 0b00000100
    out PORTA, r16

    rcall Numero1
    rcall Retardo

    ldi r16, 0b00001111
    out PORTA, r16

    rcall Numero0
    rcall Retardo

    rcall NoNumb

    rjmp Programa


NoNumb:

    ldi r16, 0b00000000
    out PORTC, r16
    ret


Numero0:

    ldi r16, 0b00111111
    out PORTC, r16
    ret

Numero1:

    ldi r16, 0b00000110
    out PORTC, r16
    ret

Numero2:

    ldi r16, 0b01011011
    out PORTC, r16
    ret
```

```
Numero3:

    ldi r16, 0b01001111
    out PORTC, r16
    ret
Numero4:

    ldi r16, 0b01100110
    out PORTC, r16
    ret

Numero5:

    ldi r16, 0b01101101
    out PORTC, r16
    ret

Numero6:

    ldi r16, 0b01111101
    out PORTC, r16
    ret

Numero7:

    ldi r16, 0b00000111
    out PORTC, r16
    ret

Numero8:

    ldi r16, 0b01111111
    out PORTC, r16
    ret

Numero9:

    ldi r16, 0b01100111
    out PORTC, r16
    ret

Retardo:
          ldi r20, 10
```

```
bucle1:

      ldi r19, 100

bucle2:

      ldi r18, 255

bucle3:

      dec r18
      BRNE bucle3
      dec r19
      BRNE bucle2
      dec r20
      BRNE bucle1
      ret
```

Programación del LCD

```
/*
 * Álvaro Iván Arcia González
 * Código pantalla LCD
 */


.include "m32def.inc"

.CSEG
.ORG $0000

.def regEnvioInfor = r16
.def regUsoGeneral = r16
.def regDelay1 = r18
.def regDelay2 = r19
.def regDelay3 = r20

Start:

    ldi regUsoGeneral,high(RAMEND)
    out SPH, regUsoGeneral
    ldi regUsoGeneral, low(RAMEND)
    out SPL, regUsoGeneral


    ldi regEnvioInfor, 0b11111111
    out DDRA, regEnvioInfor
    ; Establecemos el puerto C como salida
    ; aunque solo será utilizado los 4 bits más
    ; significativos
    out DDRC, regUsoGeneral
    out DDRD, regUsoGeneral




ConfiguracionLCD:

    rcall Loop50ms
    ldi regEnvioInfor, 0b00110000
    rcall SREnviarInfo
```

```
    ldi regEnvioInfor, 0b00110000
    rcall SREnviarInfo
    rcall Loop100ms
    ldi regEnvioInfor, 0b00110000
    rcall SREnviarInfo
    rcall Loop100ms
    ldi regEnvioInfor, 0b00100000
    rcall SREnviarInfo
    ldi regEnvioInfor, 0b11000000
    rcall SREnviarInfo
    rcall Loop100ms
    ldi regEnvioInfor, 0b00000000
    rcall SREnviarInfo
    ldi regEnvioInfor, 0b10000000
    rcall SREnviarInfo
     rcall Loop100ms
    ldi regEnvioInfor, 0b00000000
    rcall SREnviarInfo
    ldi regEnvioInfor, 0b00010000
    rcall SREnviarInfo
    rcall Loop100ms
    ldi regEnvioInfor, 0b00000000
    rcall SREnviarInfo
    ldi regEnvioInfor, 0b01100000
    rcall SREnviarInfo
    rcall Loop100ms
    ldi regEnvioInfor, 0b00000000
    rcall SREnviarInfo
    ldi regEnvioInfor, 0b11000000
    rcall SREnviarInfo
    rcall Loop100ms


     sbi PORTA,2

Mensaje:


    /*T*/
    ldi regEnvioInfor,0b01010000
    rcall SREnviarInfo
    ldi regEnvioInfor,0b01000000
    rcall SREnviarInfo
```

```
rcall Loop100ms

/*F*/
ldi regEnvioInfor,0b01000000
rcall SREnviarInfo
ldi regEnvioInfor,0b01100000
rcall SREnviarInfo
rcall Loop100ms

/*G*/
ldi regEnvioInfor,0b01000000
rcall SREnviarInfo
ldi regEnvioInfor,0b01110000
rcall SREnviarInfo
rcall Loop100ms

/*Espacio*/
ldi regEnvioInfor,0b00100000
rcall SREnviarInfo
ldi regEnvioInfor,0b00000000
rcall SREnviarInfo
rcall Loop100ms

/*A*/
ldi regEnvioInfor,0b01000000
rcall SREnviarInfo
ldi regEnvioInfor,0b00010000
rcall SREnviarInfo
rcall Loop100ms

/*l*/
ldi regEnvioInfor,0b01100000
rcall SREnviarInfo
ldi regEnvioInfor,0b11000000
rcall SREnviarInfo
rcall Loop100ms

/*v*/
ldi regEnvioInfor,0b01110000
rcall SREnviarInfo
ldi regEnvioInfor,0b01100000
rcall SREnviarInfo
rcall Loop100ms
```

```
/*a*/
ldi regEnvioInfor,0b01100000
rcall SREnviarInfo
ldi regEnvioInfor,0b00010000
rcall SREnviarInfo
rcall Loop100ms

/*r*/
ldi regEnvioInfor,0b01110000
rcall SREnviarInfo
ldi regEnvioInfor,0b00100000
rcall SREnviarInfo
rcall Loop100ms

/*0*/
ldi regEnvioInfor,0b01100000
rcall SREnviarInfo
ldi regEnvioInfor,0b11110000  ;
rcall SREnviarInfo
rcall Loop100ms

/*Espacio*/
ldi regEnvioInfor,0b00100000
rcall SREnviarInfo
ldi regEnvioInfor,0b00000000
rcall SREnviarInfo
rcall Loop100ms

/*I*/
ldi regEnvioInfor,0b01000000
rcall SREnviarInfo
ldi regEnvioInfor,0b10010000
rcall SREnviarInfo
rcall Loop100ms

/*v*/
ldi regEnvioInfor,0b01110000
rcall SREnviarInfo
ldi regEnvioInfor,0b01100000
rcall SREnviarInfo
rcall Loop100ms
```

```
/*a*/
ldi regEnvioInfor,0b01100000
rcall SREnviarInfo
ldi regEnvioInfor,0b00010000
rcall SREnviarInfo
rcall Loop100ms

/*n*/
ldi regEnvioInfor,0b01100000
rcall SREnviarInfo
ldi regEnvioInfor,0b11100000
rcall SREnviarInfo
rcall Loop100ms


/****Limpiar pantalla****/

cbi PORTA,2

ldi regEnvioInfor,0b00000000
rcall SREnviarInfo
ldi regEnvioInfor,0b00010000
rcall SREnviarInfo
rcall Loop100ms

ldi regEnvioInfor,0b00000000
rcall SREnviarInfo
ldi regEnvioInfor,0b00110000
rcall SREnviarInfo
rcall Loop100ms

sbi PORTA,2

/*Mensaje de nuevo*/

/*T*/
ldi regEnvioInfor,0b01010000
rcall SREnviarInfo
ldi regEnvioInfor,0b01000000
rcall SREnviarInfo
rcall Loop500ms

/*F*/
```

```
ldi regEnvioInfor,0b01000000
rcall SREnviarInfo
ldi regEnvioInfor,0b01100000
rcall SREnviarInfo
rcall Loop500ms

/*G*/
ldi regEnvioInfor,0b01000000
rcall SREnviarInfo
ldi regEnvioInfor,0b01110000
rcall SREnviarInfo
rcall Loop500ms

/*Espacio*/
ldi regEnvioInfor,0b00100000
rcall SREnviarInfo
ldi regEnvioInfor,0b00000000
rcall SREnviarInfo
rcall Loop500ms

/*A*/
ldi regEnvioInfor,0b01000000
rcall SREnviarInfo
ldi regEnvioInfor,0b00010000
rcall SREnviarInfo
rcall Loop500ms

/*l*/
ldi regEnvioInfor,0b01100000
rcall SREnviarInfo
ldi regEnvioInfor,0b11000000
rcall SREnviarInfo
rcall Loop500ms

/*v*/
ldi regEnvioInfor,0b01110000
rcall SREnviarInfo
ldi regEnvioInfor,0b01100000
rcall SREnviarInfo
rcall Loop500ms

/*a*/
ldi regEnvioInfor,0b01100000
```

```
rcall SREnviarInfo
ldi regEnvioInfor,0b00010000
rcall SREnviarInfo
rcall Loop500ms

/*r*/
ldi regEnvioInfor,0b01110000
rcall SREnviarInfo
ldi regEnvioInfor,0b00100000
rcall SREnviarInfo
rcall Loop500ms

/*0*/
ldi regEnvioInfor,0b01100000
rcall SREnviarInfo
ldi regEnvioInfor,0b11110000
rcall SREnviarInfo
rcall Loop500ms

/*Espacio*/
ldi regEnvioInfor,0b00100000
rcall SREnviarInfo
ldi regEnvioInfor,0b00000000
rcall SREnviarInfo
rcall Loop500ms

/*I*/
ldi regEnvioInfor,0b01000000
rcall SREnviarInfo
ldi regEnvioInfor,0b10010000
rcall SREnviarInfo
rcall Loop500ms

/*v*/
ldi regEnvioInfor,0b01110000
rcall SREnviarInfo
ldi regEnvioInfor,0b01100000
rcall SREnviarInfo
rcall Loop500ms

/*a*/
ldi regEnvioInfor,0b01100000
rcall SREnviarInfo
```

```
    ldi regEnvioInfor,0b00010000
    rcall SREnviarInfo
    rcall Loop500ms

    /*n*/
    ldi regEnvioInfor,0b01100000
    rcall SREnviarInfo
    ldi regEnvioInfor,0b11100000
    rcall SREnviarInfo
    rcall Loop500ms

    rcall Loop1s
    /****Limpiar pantalla****/

    cbi PORTA,2

    ldi regEnvioInfor,0b00000000
    rcall SREnviarInfo
    ldi regEnvioInfor,0b00010000
    rcall SREnviarInfo
    rcall Loop100ms

    ldi regEnvioInfor,0b00000000
    rcall SREnviarInfo
    ldi regEnvioInfor,0b00110000
    rcall SREnviarInfo
    rcall Loop100ms

    sbi PORTA,2

    rjmp Mensaje   ;Bucle infinito


SREnviarInfo:

    sbi PORTD, 6
    out PORTC, regEnvioInfor
    cbi PORTD, 6
    ret
```

```
Loop100ms:

          ldi  regDelay1, 130
          ldi  regDelay2, 222
    L1: dec  regDelay2
          brne L1
          dec  regDelay1
          brne L1
          nop
          ret


Loop100us:

          ldi  regDelay1, 33
    L2: dec  regDelay1
          brne L2
          nop
          ret



Loop500ms:

    ldi  regDelay1, 3
    ldi  regDelay2, 8
    ldi  regDelay3, 120
L3: dec  regDelay3
    brne L3
    dec  regDelay2
    brne L3
    dec  regDelay1
    brne L3
    ret

Loop1s:

    ldi  regDelay1, 6
    ldi  regDelay2, 19
    ldi  regDelay3, 174
L4: dec  regDelay3
    brne L4
```

```
dec  regDelay2
brne L4
dec  regDelay1
brne L4
rjmp PC+1
ret
```

Programación del GLCD Tablas

```
/*
 *  Álvaro Iván Arcia González
 *  TFG Uso_De_Tablas
 */


.include "m32def.inc"

.CSEG
.ORG $0000

.def regUsoGeneral = r16
.def regPunteroVertical = r20
.def regPunteroHorizontal = r21

Start:

    ; Inicializamos la pila
    ldi regUsoGeneral,high(RAMEND)
    out SPH, regUsoGeneral
    ldi regUsoGeneral, low(RAMEND)
    out SPL, regUsoGeneral

    ; Declaramos todos los pines como salida
    ; ya que no trabajaremos con ningún dato de entrada
    ldi regUsoGeneral,255
    out DDRA, regUsoGeneral
    out DDRB, regUsoGeneral
    out DDRC, regUsoGeneral
    out DDRD, regUsoGeneral


; Comienzo de la inicialización de la pantalla
ConfiguracionInicial:

    ; Desactivación del modo reset
    sbi PORTD,7

    ; Activamos la pantalla
    ldi regUsoGeneral, 0b00111111
    out PORTC, regUsoGeneral
```

```
    ; Inicialmos la configuración de ambos controladores
    rcall SRControladorIzquierdo
    rcall SRModoConfiguracion
    rcall SRControladorDerecho
    rcall SRModoConfiguracion


loop:
    ; Apuntamos con el puntero a la dirección
    ; de la etiqueta ULL
    ldi ZH,HIGH(ULL*2)
    ldi ZL,LOW(ULL*2)


    ; Pintamos la pantalla
    rcall SRPintar

    ; Realizamos un bucle infinito
    rjmp loop



; Subrutina utilizada para pasar a modo configuracion
SRModoConfiguracion:

    ; Desactivamos los bits 3 y 2
    ; del puerto A
    cbi PORTA,3
    cbi PORTA,2

    ; Activamos el pin E
    sbi PORTD,6
    rcall Retardo

    ; Desactivamos el pin E
    cbi PORTD,6
    rcall Retardo
    ret

; Subrutina encargada de enviar información a la
; pantalla. A diferencia del anterior activamos
```

```
; el bit 2 del puerto A, en vez de desactivarlo.
SREnviarInfor:

    ; Activamos el bit 2 del puerto A
    sbi PORTA,2
    cbi PORTA,2

    ; Activamos el pin E
    sbi PORTD,6
    rcall Retardo

    ; Desactivamos el pin E
    cbi PORTD,6
    rcall Retardo
    ret

; Activar controlador izquierdo
SRControladorIzquierdo:
    cbi PORTB,0
    sbi PORTB,1
    ret

; Activar controlador derecho
SRControladorDerecho:
    sbi PORTB,0
    cbi PORTB,1
    ret


; Subrutina de dibujo, recorreremos la tabla y la
; GLCD para ir pintando la imagen
SRPintar:
    ; Introducimos los registros utilizados en la pila
    push regPunteroVertical
    push regPunteroHorizontal

    ; Asignamos el valor de cada dirección horizontal y vertical.
    ldi regPunteroHorizontal, 0
    ldi regPunteroVertical, 0

SiguienteLinea:
```

```
    ; Llamamos a la subrutina encargada de pintar la pantalla
izquierda
    rcall SRPintarIzquierda
    rcall SRDibujar
    ; Llamamos a la subrutina encargada de pintar la pantalla
derecha
    rcall SRPintarDerecha
    rcall SRDibujar

    ; Incrementamos el valor del punntero vertical
    inc regPunteroVertical

    ; Si llegamos a 8 hemos pintado todas las páginas por
    ; Comparamos el registro del puntero vertical, en caso de que
    ; se cumpla la condición habremos recorrido toda la pantalla y
    ; salimos de SRPintar
    cpi regPunteroVertical,8

    ; Si no se cumple, pintamos la siguiente componente verical.
    brne SiguienteLinea

    ; Recuperamos los valores de los registros
    pop regPunteroHorizontal
    pop regPunteroVertical
    ret


SRPintarIzquierda:

    ; Introducimos los valores a la pila
    push regUsoGeneral

    ; Encendemos el controlador de la pantalla izquierda
    rcall SRControladorIzquierdo

    ; Configuramos para comenzar en el primer pixel izquierdo de la
pantalla
    ldi regUsoGeneral,0b01000000
    out PORTC, regUsoGeneral
    rcall SRModoConfiguracion

    ; Sumamos la dirección del puntero a la configuración de las
paginas
```

```
    ; para determinar qué páginas vamos a pintar
    ldi regUsoGeneral,0b10111000
    or regUsoGeneral,regPunteroVertical
    out PORTC, regUsoGeneral
    rcall SRModoConfiguracion

    ; Recuperamos los registros
    pop regUsoGeneral
    ret


; La subrutina será igual a la anterior a diferencia que será con
el
; controlador derecho en vez del izquierdo
SRPintarDerecha:

    push regUsoGeneral
    rcall SRControladorDerecho
    ; Configuramos para comenzar en el primer pixel izquierdo de la
pantalla
    ldi regUsoGeneral,0b01000000
    out PORTC, regUsoGeneral
    rcall SRModoConfiguracion

    ; Sumamos la dirección del puntero a la configuración de las
paginas
    ; para determinar qué páginas vamos a pintar
    ldi regUsoGeneral,0b10111000
    or regUsoGeneral,regPunteroVertical
    out PORTC, regUsoGeneral
    rcall SRModoConfiguracion

    ; Recuperamos los registros
    pop regUsoGeneral
    ret

SRDibujar:
    ; Cargamos los registros en la pila
    push r0    .
    push regPunteroHorizontal

SRDibujar1:
    ; Instrucción utilizada para cargar una constante de la
```

```
    ; memoria de programa apuntada por Z
    lpm
    out PORTC, r0
    ; Lo enviamos por la salida
    rcall SREnviarInfor  ;Y se muestra por pantalla
    ; aumentamos la dirección del puntero,
    adiw ZL,1

    ; Aumentamos la dirección del Puntero Horizontal
    ; y comprobamos que no haya superado los 64 pixeles del
controlador.
    inc regPunteroHorizontal
    cpi regPunteroHorizontal, 64

    ; Si no se ha superado vamos con el siguiente
    brne SRDibujar1

    ; Recuperamos los registros
    pop regPunteroHorizontal
    pop r0

    ret



Retardo:
            ldi r19, 1
      bucle02:
            ldi r18, 40
      bucle03:
            dec r18
            BRNE bucle03
            dec r19
            BRNE bucle02
            ret


ULL:

    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0xC0, 0xFC, 0xFF, 0xFF, 0xFF,
0xFF, 0xFF, 0x3F, 0x03, 0x00, 0x00
```

```
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x80, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF
    .db 0xFF, 0xFF, 0x1F, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0xFE, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
0x40, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
0xFF, 0xFF, 0xFF, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0xC0, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
0xFF, 0x1F, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x80, 0xFC,
0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF
    .db 0x1F, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
0xFF, 0xFF, 0xFF, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x80, 0xFC, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0x3F,
0x01, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x80, 0xFF, 0xFF,
0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0x03
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
```

```
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
0xFF, 0xFF, 0xFF, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x80, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0x1F, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0xC0, 0xFC, 0xFF, 0xFF, 0xFF,
0xFF, 0xFF, 0xFF, 0x7F, 0x07, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
0xFF, 0xFF, 0x01, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0xC0, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
0xFF, 0xFF, 0x07, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x7F, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xF8, 0xC0,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x80
    .db 0xC0, 0xE0, 0xFC, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
0xFF, 0x1F, 0x00, 0x00, 0x00, 0x00
```

```
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
0x00, 0x00, 0x00, 0x80, 0x80, 0xC0
    .db 0xC0, 0xC0, 0xE0, 0xE0, 0xF0, 0xF0, 0xF8, 0xF8, 0xC0, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
0xFF, 0xFF, 0x80, 0x00, 0x80, 0x80
    .db 0x80, 0x80, 0xC0, 0xE0, 0xE0, 0xF0, 0xF0, 0xF0, 0xF8, 0xF8,
0xE0, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x07, 0x0F, 0x1F, 0x7F, 0xFF, 0xFF, 0xFF, 0xFF,
0xFF, 0xFE, 0xFC, 0xF8, 0xF0, 0xF0
    .db 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xF0, 0xF0,
0xF0, 0xF8, 0xFC, 0xFC, 0xFF, 0xFF
    .db 0xFF, 0x7F, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0x1F,
0x01, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0x7F
    .db 0x7F, 0x7F, 0x3F, 0x3F, 0x3F, 0x1F, 0x1F, 0x1F, 0x0F, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF
    .db 0xFF, 0x7F, 0x7F, 0x7F, 0x3F, 0x3F, 0x3F, 0x1F, 0x1F, 0x0F,
0x0F, 0x0E, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x03,
0x03, 0x07, 0x07, 0x0F, 0x0F, 0x0F
    .db 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x07,
0x07, 0x07, 0x03, 0x01, 0x01, 0x00
    .db 0x00, 0x00, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x0F, 0x0F, 0x0F, 0x07, 0x07, 0x03, 0x03, 0x03,
0x01, 0x01, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x0F, 0x07, 0x07, 0x07, 0x03,
0x03, 0x03, 0x01, 0x01, 0x01, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
```

Programación del Código final

```
/*
 *  Código Final
 *  TFG Programa de desbloqueo
 */


 .CSEG
.ORG $0000


.def regUsoGeneral = r16
.def regVecesPulsado = r29
.def regDesbloqueo = r19
.def regAdchVertical = r17
.def regAdchHorizontal = r24
.def regVerificacion = r18
.def regFila = r31



; Inicio del código

start:

    ; Creación de la pila

    ldi regUsoGeneral,high(RAMEND)
    out SPH, regUsoGeneral
    ldi regUsoGeneral, low(RAMEND)
    out SPL, regUsoGeneral

    ; Establecemos como entrada los pines 0 y 1 del Puerto A
    ; El resto de pines serán de salida.

    ldi regUsoGeneral, 0b01111100
    out DDRA, regUsoGeneral

    ; Establecemos como salida los pines del puerto B
    ; Refinarlo
```

```
    ldi regUsoGeneral,255
    out DDRB, regUsoGeneral

    ; Establecemos como salida los pines del puerto C
    ; a través de estos pines le enviaremos información a la
pantalla

    ldi regUsoGeneral,255
    out DDRC, regUsoGeneral

    ; Establecemos como salida los pines del puerto D

    ldi regUsoGeneral,255
    out DDRD, regUsoGeneral

    ; Cargamos la dirección del puntero con la tabla Blanco ya que
    ; limpiaremos la pantalla al iniciar el código

    ldi ZH,HIGH(Blanco*2)
    ldi ZL,LOW(Blanco*2)

    ; Inicializamos los registros encargados de la comprobación del
pin
    ; y la comprobación de la cantidad de veces pulsada

    ldi regVecesPulsado, 0
    ldi regDesbloqueo,0


programa:

    ; Iniciamos la configuración de la pantalla para empezar a
trabajar con ella

    sbi PORTD,7
    ldi regUsoGeneral, 0b00111111
    out PORTC, regUsoGeneral


    rcall SRPantallaIzquierdaOn
    rcall SRModoConfig
    rcall SRPantallaDerechaOn
    rcall SRModoConfig
```

```
    cbi PORTB,0
    cbi PORTB,1


    ; A través del modo configuración establecemos que comenzaremos
a
    ; dibujar la pantalla en el pixel 1 del eje horizontal

    ldi regUsoGeneral,0b01000000
    out PORTC, regUsoGeneral
    rcall SRModoConfig

    ; A través del modo configuración establecemos que comenzaremos
a
    ; dibujar la pantalla en el pixel 1 del eje vertical

    ldi regUsoGeneral,0b10111000
    out PORTC, regUsoGeneral
    rcall SRModoConfig


    ;Imprimimos por pantalla el logo de la ULL

    ldi ZH,HIGH(ULL*2)
    ldi ZL,LOW(ULL*2)
    rcall SRPintar
    rcall Loop1s

inicio:
    ; Limpiamos la pantalla

    rcall SRLimpiar

    ; Dibujamos los números

    rcall SRNumeros


bucle:
    ; Configuramos admux para que nos lea el pin 1 del puerto A

    ldi regUsoGeneral,0b01000001
```

```
    out admux,regUsoGeneral

    ; Activamos el panel para que nos proporcione voltajes
    ; en función de la vertical

    cbi PORTA,2
    sbi PORTA,3
    ldi regUsoGeneral, 0b10000000
    out adcsra, regUsoGeneral


; Bucle intermedio para reiniciar la medida

admuxConfig:

    ldi regUsoGeneral,0b01000001
    out admux,regUsoGeneral
    clr regAdchVertical
    cbi PORTA,2
    sbi PORTA,3
    sbi adcsra,6




bucleComprobacionVertical:

    ; Creamos un bucle que compruebe constantemente el pin 4 del
    ; convertidor adcsra ya que este se activa cuando acaba la
conversión

    sbis adcsra,4
    rjmp bucleComprobacionVertical
    rcall Delay4ms

    ; Introducimos los valores de la conversión en los registros

    in regUsoGeneral, adcl
    in regAdchVertical, adch

    ; Guardamos el valor del adcl en el r18 para más tarde verificar
que
```

```
    ; la medida ha sido correcta
    mov regVerificacion,regUsoGeneral

    ; Volvemos a realizar la medición y su bucle

    sbi adcsra,6

bucleComprobacionVertical2:

    sbis adcsra,4
    rjmp bucleComprobacionVertical2
    in regUsoGeneral, adcl
    in regAdchVertical, adch

    cp regUsoGeneral,regVerificacion
    brne admuxConfig

    ; Realizamos un primer filtro a través de los valores de adch

    cpi regAdchVertical,1
    breq posibleFila23
    cpi regAdchVertical,2
    breq posibleFila12
    cpi regAdchVertical,3
    breq primeraFila
    rjmp posibleFila3


; Una vez comparado el valor de adch compararemos el valor de
; adcl para saber qué fila ha sido presionada y proceder a leer
; el eje vertical de la pulsación

posibleFila3:
    cpi regUsoGeneral, 139
    brlo admuxConfig
    rjmp terceraFila

posibleFila23:

    cpi regUsoGeneral,113
    brlo terceraFila
    rjmp segundaFila
```

```
posibleFila12:

    cpi regUsoGeneral, 87
    brlo segundaFila
    rjmp primeraFila

; Asignamos el valor correspondiente a cada fila al registro
; encargado

primeraFila:
    ldi regFila,1
    rjmp bucleHorizontal

segundaFila:
    ldi regFila,2
    rjmp bucleHorizontal

terceraFila:
    ldi regFila,3
    rjmp bucleHorizontal

;Leemos el eje horizontal de la pulsación a través del mismo
proceso
; anteriormente hecho

bucleHorizontal:

    ldi regUsoGeneral,0b01000000
    out admux,regUsoGeneral

    sbi PORTA,2
    cbi PORTA,3
    rcall Delay4ms

    sbi adcsra,6

bucleComprobacionHorizontal:

    sbis adcsra,4
    rjmp bucleComprobacionHorizontal
    ldi r23,37

    rcall SRPantallaIzquierdaOn
```

```
    in regUsoGeneral, adcl
    in regAdchHorizontal, adch

    ; Una vez obtenido los valores de la conversión procedemos a
filtrar por
    ; el valor del adch

    cpi regAdchHorizontal,3
    breq terceraColumna
    cpi regAdchHorizontal,2
    breq posibleColumna23
    cpi regAdchHorizontal,1
    breq posibleColumna12
    rjmp primeraColumna

; Filtramos por los valores de adcl para saber qué columna ha sido
seleccionada

posibleColumna12:
    cpi regUsoGeneral,105
    brlo primeraColumna
    rjmp segundaColumna

posibleColumna23:
    cpi regUsoGeneral, 43
    brlo segundaColumna
    rjmp terceraColumna

; Comparamos el valor del registro encargado de almacenar el numero
; de la fila y encendemos el número correspondiente

primeraColumna:

    ; Comparación número 1

    cpi regFila, 1
    breq encenderNumero01

    ; Comparación numero 4

    cpi regFila, 2
    breq encenderNumero04
```

```
    ; Comparación numero 7
    cpi regFila, 3
    breq encenderNumero07

    ldi regFila,0
    rjmp inicio

;Acordarse de nombre el problema de las llamadas a las breq
; Parte del código encargadas para la llamada a las subrutinas de
encendido del número
encenderNumero01:
    rcall SREncenderNumero1
    rjmp inicio
encenderNumero04:
    rcall SREncenderNumero4
    rjmp inicio
encenderNumero07:
    rcall SREncenderNumero7
    rjmp inicio


segundaColumna:

    cpi regFila, 1
    breq encenderNumero02

    cpi regFila, 2
    breq encenderNumero05

    cpi regFila, 3
    breq encenderNumero08

    ldi regFila,0
    rjmp inicio

encenderNumero02:
    rcall SREncenderNumero2
    rjmp inicio
encenderNumero05:
    rcall SREncenderNumero5
    rjmp inicio
encenderNumero08:
    rcall SREncenderNumero8
```

```
    rjmp inicio

terceraColumna:
    cpi regFila, 1
    breq encenderNumero03

    cpi regFila, 2
    breq encenderNumero06

    cpi regFila, 3
    breq encenderNumero09

    ldi regFila,0
    rjmp inicio

encenderNumero03:
    rcall SREncenderNumero3
    rjmp inicio
encenderNumero06:
    rcall SREncenderNumero6
    rjmp inicio
encenderNumero09:
    rcall SREncenderNumero9
    rjmp inicio



; Subrutinas de encendido de número
SREncenderNumero1:

    ldi ZH,HIGH(Numero1*2)
    ldi ZL,LOW(Numero1*2)
    rcall SRPintar

    ; Llamada a la secuencia de desbloqueo3

    rcall SRDesbloqueo3

    ; Llamada al contador
    rcall SRContador
    ret
```

```
SREncenderNumero2:

    ldi ZH, HIGH(Numero2*2)
    ldi ZL, LOW(Numero2*2)
    rcall SRPintar
    rcall SRContador
    ret

SREncenderNumero3:

    ldi ZH,HIGH(Numero3*2)
    ldi ZL,LOW(Numero3*2)
    rcall SRPintar
    rcall SRContador
    ret

SREncenderNumero4:

    ldi ZH,HIGH(Numero4*2)
    ldi ZL,LOW(Numero4*2)
    rcall SRPintar
    rcall SRDesbloqueo1
    rcall SRContador
    ret

SREncenderNumero5:
    ldi ZH,HIGH(Numero5*2)
    ldi ZL,LOW(Numero5*2)
    rcall SRPintar
    rcall SRDesbloqueo
    rcall SRContador
    ret

SREncenderNumero6:
    ldi ZH,HIGH(Numero6*2)
    ldi ZL,LOW(Numero6*2)
    rcall SRPintar
    rcall SRContador
    ret

SREncenderNumero7:
    ldi ZH,HIGH(Numero7*2)
    ldi ZL,LOW(Numero7*2)
```

```
    rcall SRDesbloqueo2
    rcall SRPintar
    rcall SRContador
    ret

SREncenderNumero8:
    ldi ZH,HIGH(Numero8*2)
    ldi ZL,LOW(Numero8*2)
    rcall SRPintar
    rcall SRContador
    ret

SREncenderNumero9:
    ldi ZH,HIGH(Numero9*2)
    ldi ZL,LOW(Numero9*2)
    rcall SRPintar
    rcall SRContador
    ret

; Subrutina encargada del desbloqueo, se irán llamando a las
secuencias en orden.
; estas comparan el registro encargado y en caso de estar con el
valor correcto
; se aumentará el valor y saldrá de la subrutina, de lo contrario
saldrá sin realizar
; ningún cambio
SRDesbloqueo:
    cpi regDesbloqueo,0
    breq SRDesbloqueoAumento
    ret

SRDesbloqueo1:
    cpi regDesbloqueo,1
    breq SRDesbloqueoAumento
    ret

SRDesbloqueo2:
    cpi regDesbloqueo,2
    breq SRDesbloqueoAumento
    ret
SRDesbloqueo3:
    cpi regDesbloqueo,3
    breq SRDesbloqueoAumento
```

```
    ret

SRDesbloqueoAumento:
    inc regDesbloqueo
    ret

; Subrutina encargada de leer el número de veces que se ha pulsado
; un número en la pantalla.

SRContador:
    inc regVecesPulsado

    ; Comprueba el valor del registro de desbloqueo, si es correcto
    ; llama a la etiqueta desbloquear.

    cpi regDesbloqueo,4
    breq SRDesbloquear

    ; Comprueba el valor del registro de pulsaciones, en caso de
haber
    ; pulsado 4 veces saltará a la etiqueta Fallo

    cpi regVecesPulsado,4
    breq SRFallo
    ret

; Subrutina de desbloqueo

SRDesbloquear:

    ; Pinta en la GLCD la imagen de desbloqueo

    rcall SRDesbloqueado
    rcall Loop1s

    ; Setea los valores de las cuentas a 0

    ldi regDesbloqueo,0
    ldi regVecesPulsado,0
    ret

; Subrutina de fallo
```

```
SRFallo:
    ; Pinta en la GLCD la imagen de failed

    rcall Fallado
    rcall Loop1s

    ; Setea los valores de las cuentas a 0

    ldi regDesbloqueo,0
    ldi regVecesPulsado,0
    ret

; Comentar aquí

SRNumeros:
    ldi ZH,HIGH(Numeros*2)
    ldi ZL,LOW(Numeros*2)
    rcall SRPintar
    ret

SRLimpiar:

    ldi ZH,HIGH(Blanco*2)
    ldi ZL,LOW(Blanco*2)
    rcall SRPintar
    ret

Fallado:
    ldi ZH,HIGH(Failed*2)
    ldi ZL,LOW(Failed*2)
    rcall SRPintar
    ret

SRDesbloqueado:

    ldi ZH,HIGH(Unlocked*2)
    ldi ZL,LOW(Unlocked*2)
    rcall SRPintar
    ret


; Subrutina utilizada para pasar a modo configuracion
SRModoConfiguracion:
```

```
    ; Desactivamos los bits 3 y 2
    ; del puerto A
    cbi PORTA,3
    cbi PORTA,2

    ; Activamos el pin E
    sbi PORTD,6
    rcall Delay

    ; Desactivamos el pin E
    cbi PORTD,6
    rcall Delay
    ret

; Subrutina encargada de enviar información a la
; pantalla. A diferencia del anterior activamos
; el bit 2 del puerto A, en vez de desactivarlo.
SREnviarInfor:

    ; Activamos el bit 2 del puerto A
    sbi PORTA,2
    cbi PORTA,2

    ; Activamos el pin E
    sbi PORTD,6
    rcall Delay

    ; Desactivamos el pin E
    cbi PORTD,6
    rcall Delay
    ret

; Activar controlador izquierdo
SRControladorIzquierdo:
    cbi PORTB,0
    sbi PORTB,1
    ret

; Activar controlador derecho
SRControladorDerecho:
    sbi PORTB,0
    cbi PORTB,1
```

```
    ret


; Subrutina de dibujo, recorreremos la tabla y la
; GLCD para ir pintando la imagen
SRPintar:
    ; Introducimos los registros utilizados en la pila
    push regPunteroVertical
    push regPunteroHorizontal

    ; Asignamos el valor de cada dirección horizontal y vertical.
    ldi regPunteroHorizontal, 0
    ldi regPunteroVertical, 0

SiguienteLinea:

    ; Llamamos a la subrutina encargada de pintar la pantalla
izquierda
    rcall SRPintarIzquierda
    rcall SRDibujar
    ; Llamamos a la subrutina encargada de pintar la pantalla
derecha
    rcall SRPintarDerecha
    rcall SRDibujar

    ; Incrementamos el valor del punntero vertical
    inc regPunteroVertical

    ; Si llegamos a 8 hemos pintado todas las páginas por
    ; Comparamos el registro del puntero vertical, en caso de que
    ; se cumpla la condición habremos recorrido toda la pantalla y
    ; salimos de SRPintar
    cpi regPunteroVertical,8

    ; Si no se cumple, pintamos la siguiente componente verical.
    brne SiguienteLinea

    ; Recuperamos los valores de los registros
    pop regPunteroHorizontal
    pop regPunteroVertical
    ret
```

```
SRPintarIzquierda:

    ; Introducimos los valores a la pila
    push regUsoGeneral

    ; Encendemos el controlador de la pantalla izquierda
    rcall SRControladorIzquierdo

    ; Configuramos para comenzar en el primer pixel izquierdo de la
pantalla
    ldi regUsoGeneral,0b01000000
    out PORTC, regUsoGeneral
    rcall SRModoConfiguracion

    ; Sumamos la dirección del puntero a la configuración de las
paginas
    ; para determinar qué páginas vamos a pintar
    ldi regUsoGeneral,0b10111000
    or regUsoGeneral,regPunteroVertical
    out PORTC, regUsoGeneral
    rcall SRModoConfiguracion

    ; Recuperamos los registros
    pop regUsoGeneral
    ret


; La subrutina será igual a la anterior a diferencia que será con
el
; controlador derecho en vez del izquierdo
SRPintarDerecha:

    push regUsoGeneral
    rcall SRControladorDerecho
    ; Configuramos para comenzar en el primer pixel izquierdo de la
pantalla
    ldi regUsoGeneral,0b01000000
    out PORTC, regUsoGeneral
    rcall SRModoConfiguracion

    ; Sumamos la dirección del puntero a la configuración de las
paginas
    ; para determinar qué páginas vamos a pintar
```

```
    ldi regUsoGeneral,0b10111000
    or regUsoGeneral,regPunteroVertical
    out PORTC, regUsoGeneral
    rcall SRModoConfiguracion

    ; Recuperamos los registros
    pop regUsoGeneral
    ret

SRDibujar:
    ; Cargamos los registros en la pila
    push r0    .
    push regPunteroHorizontal

SRDibujar1:
    ; Instrucción utilizada para cargar una constante de la
    ; memoria de programa apuntada por Z
    lpm
    out PORTC, r0
    ; Lo enviamos por la salida
    rcall SREnviarInfor  ;Y se muestra por pantalla
    ; aumentamos la dirección del puntero,
    adiw ZL,1

    ; Aumentamos la dirección del Puntero Horizontal
    ; y comprobamos que no haya superado los 64 pixeles del
controlador.
    inc regPunteroHorizontal
    cpi regPunteroHorizontal, 64

    ; Si no se ha superado vamos con el siguiente
    brne SRDibujar1

    ; Recuperamos los registros
    pop regPunteroHorizontal
    pop r0

    ret



SRPantallaIzquierdaOn:
```

```
    cbi PORTB,0;
    sbi PORTB,1
    ret

SRPantallaDerechaOn:

    sbi PORTB,0
    cbi PORTB,1
    ret

SRModoConfig:

    cbi PORTA,2
    cbi PORTA,3
    sbi PORTD,6
    rcall Delay100us
    cbi PORTD,6
    rcall Delay100us

    ret



Leer:
    push regUsoGeneral

    ldi regUsoGeneral,0     ;PC todo salida
    out DDRC, regUsoGeneral
    sbi PORTA,2
    sbi PORTA,3
    sbi PORTD,6
    rcall Delay100us
    cbi PORTD,6
    rcall Delay100us
    sbi PORTD,6
    rcall Delay100us
    in r28, PINC
    cbi PORTD,6
    rcall Delay100us
    ldi regUsoGeneral,255   ;PC todo salida
    out DDRC, regUsoGeneral

    pop regUsoGeneral
```

```
    ret



Delay:
    push r25
    push r26
    ldi r25, 1
bucle02:
    ldi r26, 40
bucle03:
    dec r25
    BRNE bucle03
    dec r26
    BRNE bucle02
    pop r26
    pop r25
    ret


Loop1s:
    push 25
    push 26
    push 27
    ldi  r25, 82
    ldi  r26, 43
    ldi  r27, 0
L1: dec  r27
    brne L1
    dec  r26
    brne L1
    dec  r25
    brne L1
    lpm
    nop
    pop 27
    pop 26
    pop 25
    ret
```

```
Delay100us:
    ldi  r25, 33
L2: dec  r25
    brne L2
    nop
    ret

Delay4ms:
    ldi  r25, 6
    ldi  r26, 49
L3: dec  r25
    brne L3
    dec  r25
    brne L3
    ret




Failed:

    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
```

```
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xC0, 0xC0,
0xC0, 0xC0, 0xC0, 0xC0, 0xC0, 0xC0
    .db 0xC0, 0xC0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x80, 0xC0, 0xC0,
0x80, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0xC0, 0xC0, 0xC0, 0x00, 0x00, 0x00, 0x00, 0x00,
0xC0, 0xC0, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0xC0, 0xC0, 0xC0, 0xC0, 0xC0,
0xC0, 0xC0, 0xC0, 0xC0, 0xC0, 0x00
    .db 0x00, 0x00, 0xC0, 0xC0, 0xC0, 0xC0, 0xC0, 0xC0, 0xC0, 0x80,
0x80, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0xC0, 0xC0, 0xC0, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0xFF, 0xC1, 0xC1, 0xC1, 0xC1, 0xC1
    .db 0x01, 0x01, 0x00, 0x00, 0x00, 0xE0, 0x7C, 0x07, 0x03, 0x07,
0x7F, 0xFC, 0xC0, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00,
0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0xE1, 0xE1,
0xE1, 0xE1, 0xE1, 0x01, 0x01, 0x00
    .db 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0x01, 0x01, 0x01, 0x01, 0x03,
0x0F, 0xFF, 0xFC, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x0F, 0xFF, 0x1F, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
```

```
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0xFF, 0x01, 0x01, 0x01, 0x01, 0x01
    .db 0x00, 0x00, 0x80, 0xF0, 0x7F, 0x0F, 0x0E, 0x0E, 0x0E, 0x0E,
0x0E, 0x0F, 0x7F, 0xFE, 0xE0, 0x00
    .db 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00,
0xFF, 0xFF, 0xE0, 0xE0, 0xE0, 0xE0
    .db 0xE0, 0xE0, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0xE0, 0xE0,
0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0x00
    .db 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0xE0, 0xE0, 0xE0, 0xE0, 0x70,
0x7C, 0x3F, 0x0F, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0xF0, 0xF7, 0xF0, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
```

```
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00



Unlocked:

    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
```

```
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x80, 0x80, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x80
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x80, 0x80, 0x80, 0x80, 0x00,
0x00, 0x00, 0x00, 0x00, 0x80, 0x80
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x80, 0x80, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x80, 0x80, 0x80, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x80, 0x80, 0x80, 0x80, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x80, 0x80
    .db 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x80, 0x80, 0x80, 0x00,
0x00, 0x00, 0x80, 0x80, 0x80, 0x80
    .db 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x00, 0x00, 0x00, 0x80,
0x80, 0x80, 0x80, 0x80, 0x80, 0x80
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x80,
0x80, 0x80, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0xFF
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x0F, 0x0F, 0x3C,
0xE0, 0x80, 0x00, 0x00, 0xFF, 0xFF
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0xF8, 0xFE, 0x3F, 0x03, 0x03, 0x03, 0x03, 0x03, 0x3F, 0xFE,
0xF8, 0x00, 0x00, 0x00, 0x00, 0xF0
```

```
    .db 0xFC, 0xFF, 0x07, 0x03, 0x03, 0x03, 0x03, 0x07, 0x3F, 0x3C,
0x20, 0x00, 0x00, 0x00, 0xFF, 0xFF
    .db 0xFF, 0xC0, 0xC0, 0xF0, 0xF8, 0x0E, 0x07, 0x01, 0x00, 0x00,
0x00, 0x00, 0xFF, 0xFF, 0xFF, 0xC3
    .db 0xC3, 0xC3, 0xC3, 0xC3, 0x03, 0x03, 0x00, 0x00, 0x00, 0xFF,
0xFF, 0xFF, 0x03, 0x03, 0x03, 0x03
    .db 0x07, 0x1F, 0xFE, 0xF8, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1F,
0xFF, 0x3F, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x7F, 0xFF, 0xE0, 0xC0,
0xC0, 0xC0, 0xC0, 0xC0, 0xE0, 0x7F
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00,
0x01, 0x07, 0x3C, 0xFC, 0xFF, 0xFF
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0xC0, 0xC0, 0xC0,
0xC0, 0xC0, 0xC0, 0x00, 0x00, 0x00
    .db 0x1F, 0x7F, 0xFC, 0xC0, 0xC0, 0xC0, 0xC0, 0xC0, 0xFC, 0x7F,
0x1F, 0x00, 0x00, 0x00, 0x00, 0x0F
    .db 0x3F, 0xFF, 0xE0, 0xC0, 0xC0, 0xC0, 0xC0, 0xE0, 0x7C, 0x3C,
0x04, 0x00, 0x00, 0x00, 0xFF, 0xFF
    .db 0xFF, 0x03, 0x01, 0x01, 0x07, 0x1F, 0xFC, 0xF0, 0x80, 0x00,
0x00, 0x00, 0xFF, 0xFF, 0xFF, 0xC1
    .db 0xC1, 0xC1, 0xC1, 0xC1, 0xC0, 0xC0, 0x00, 0x00, 0x00, 0xFF,
0xFF, 0xFF, 0xC0, 0xC0, 0xC0, 0xC0
    .db 0xE0, 0xF8, 0x7F, 0x1F, 0x00, 0x00, 0x00, 0x00, 0x00, 0xE0,
0xEF, 0xE0, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01,
0x01, 0x01, 0x01, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x01, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x01, 0x01, 0x01
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x01, 0x01, 0x01, 0x01,
0x01, 0x01, 0x01, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x01, 0x01, 0x01, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x01, 0x01, 0x01, 0x01, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x01, 0x01
    .db 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x01, 0x00,
0x00, 0x00, 0x01, 0x01, 0x01, 0x01
    .db 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x00, 0x00, 0x00, 0x01,
0x01, 0x01, 0x01, 0x01, 0x01, 0x01
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01,
0x01, 0x01, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
```

```
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00

Numeros:

    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x80
    .db 0x80, 0x40, 0x60, 0xF8, 0xF8, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x30, 0x18, 0x08, 0x08, 0x08
    .db 0x08, 0xF8, 0xF0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
```

```
    .db 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x30, 0x18, 0x08,
0x08, 0x08, 0x98, 0xF0, 0x60, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0xC0, 0xE0, 0xF0, 0x98, 0x8C
    .db 0x86, 0x83, 0x81, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x60, 0xC0, 0x80,
0x83, 0x83, 0x83, 0xFC, 0x7C, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30,
0x30, 0x30, 0x30, 0x30, 0x30, 0x30
    .db 0x30, 0x30, 0x30, 0x33, 0x33, 0x30, 0x30, 0x30, 0x30, 0x30,
0x30, 0x30, 0x30, 0x30, 0x30, 0x30
    .db 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0xFF, 0xFF,
0x30, 0x30, 0x30, 0x30, 0x30, 0x30
    .db 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30,
0x31, 0x31, 0x31, 0x31, 0x31, 0x31
    .db 0x31, 0x31, 0x31, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30,
0x30, 0x30, 0x30, 0x30, 0x30, 0x30
    .db 0x30, 0x30, 0x30, 0x30, 0xFF, 0xFF, 0x30, 0x30, 0x30, 0x30,
0x30, 0x30, 0x30, 0x30, 0x30, 0x30
    .db 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30,
0x30, 0x30, 0x30, 0x30, 0x30, 0x30
    .db 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30,
0x30, 0x30, 0x30, 0x30, 0x30, 0x30
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0xC0, 0x20, 0x18, 0x0C, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
```

```
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x70, 0x7F, 0x43, 0x63, 0x63
    .db 0x63, 0xC3, 0x83, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xC0, 0xF8, 0x9C, 0x02,
0x42, 0x42, 0x42, 0xC6, 0x84, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x0E, 0x0F
    .db   0x0C, 0x0C, 0x0C, 0x0C, 0x7F, 0x7F, 0x0C, 0x0C, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x18, 0x38, 0x20, 0x20, 0x20
    .db 0x20, 0x30, 0x1F, 0x06, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x0F, 0x1F, 0x30,
0x20, 0x20, 0x20, 0x30, 0x1F, 0x06
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04,
0x04, 0x04, 0x04, 0x04, 0xC4, 0xC4
    .db 0xC4, 0xC4, 0xC4, 0xC4, 0xC4, 0xC4, 0xC4, 0x04, 0x04, 0x04,
0x04, 0x04, 0x04, 0x04, 0x04, 0x04
    .db 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0xFF, 0xFF,
0x04, 0x04, 0x04, 0x04, 0x04, 0x04
    .db 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04,
0x04, 0x04, 0x04, 0x04, 0x04, 0x04
    .db 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04,
0x04, 0x04, 0x04, 0x04, 0x04, 0x04
    .db 0x04, 0x04, 0x04, 0x04, 0xFF, 0xFF, 0x04, 0x04, 0x04, 0x04,
0x04, 0x04, 0x04, 0x04, 0x04, 0x04
    .db 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x84, 0x84,
0x84, 0x84, 0x84, 0x04, 0x04, 0x04
```

```
    .db 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04,
0x04, 0x04, 0x04, 0x04, 0x04, 0x04
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0xE0, 0x38, 0x0C, 0x03, 0x01, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x8E, 0xDF, 0x61, 0x61, 0x61
    .db 0x61, 0x9F, 0x8E, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1E, 0x3F, 0x61, 0x40,
0x40, 0x40, 0x01, 0xFF, 0xFE, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x10, 0x1F, 0x03, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x02, 0x0F, 0x18, 0x10, 0x10, 0x10
    .db 0x10, 0x1F, 0x0F, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x04, 0x0C, 0x08,
0x08, 0x08, 0x0C, 0x07, 0x03, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00

Blanco:

    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
```

```
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
```

```
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
```

```
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00


Numero1:
```

```
    .db 0xFE, 0xFE, 0xFE, 0xFE, 0xFE, 0xFE, 0xFE, 0xFE, 0xFE, 0xFE,
0xFE, 0xFE, 0xFE, 0xFE, 0xFE, 0xFE
    .db 0xFE, 0xFE, 0xFE, 0xFE, 0xFE, 0xFE, 0xFE, 0xFE, 0xFE, 0xFE,
0xFE, 0xFE, 0xFE, 0xFE, 0xFE, 0xFE
    .db 0xFE, 0xFE, 0xFE, 0xFE, 0xFE, 0xFE, 0xFE, 0xFE, 0xFF, 0xFF,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x30, 0x18, 0x08, 0x08, 0x08
    .db 0x08, 0xF8, 0xF0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x30, 0x18, 0x08,
0x08, 0x08, 0x98, 0xF0, 0x60, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF
    .db 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF
    .db 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0xC0, 0xE0, 0xF0, 0x98, 0x8C
    .db 0x86, 0x83, 0x81, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x60, 0xC0, 0x80,
0x83, 0x83, 0x83, 0xFC, 0x7C, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F,
0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F
    .db 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F,
0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F
    .db 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0xFF, 0xFF,
0x30, 0x30, 0x30, 0x30, 0x30, 0x30
    .db 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30,
0x31, 0x31, 0x31, 0x31, 0x31, 0x31
    .db 0x31, 0x31, 0x31, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30,
0x30, 0x30, 0x30, 0x30, 0x30, 0x30
```

```
    .db 0x30, 0x30, 0x30, 0x30, 0xFF, 0xFF, 0x30, 0x30, 0x30, 0x30,
0x30, 0x30, 0x30, 0x30, 0x30, 0x30
    .db 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30,
0x30, 0x30, 0x30, 0x30, 0x30, 0x30
    .db 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30,
0x30, 0x30, 0x30, 0x30, 0x30, 0x30
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0xC0, 0x20, 0x18, 0x0C, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x70, 0x7F, 0x43, 0x63, 0x63
    .db 0x63, 0xC3, 0x83, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xC0, 0xF8, 0x9C, 0x02,
0x42, 0x42, 0x42, 0xC6, 0x84, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x0E, 0x0F
    .db 0x0C, 0x0C, 0x0C, 0x0C, 0x7F, 0x7F, 0x0C, 0x0C, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x18, 0x38, 0x20, 0x20, 0x20
    .db 0x20, 0x30, 0x1F, 0x06, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x0F, 0x1F, 0x30,
0x20, 0x20, 0x20, 0x30, 0x1F, 0x06
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04,
0x04, 0x04, 0x04, 0x04, 0xC4, 0xC4
    .db 0xC4, 0xC4, 0xC4, 0xC4, 0xC4, 0xC4, 0xC4, 0x04, 0x04, 0x04,
0x04, 0x04, 0x04, 0x04, 0x04, 0x04
```

```
    .db 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0xFF, 0xFF,
0x04, 0x04, 0x04, 0x04, 0x04, 0x04
    .db 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04,
0x04, 0x04, 0x04, 0x04, 0x04, 0x04
    .db 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04,
0x04, 0x04, 0x04, 0x04, 0x04, 0x04
    .db 0x04, 0x04, 0x04, 0x04, 0xFF, 0xFF, 0x04, 0x04, 0x04, 0x04,
0x04, 0x04, 0x04, 0x04, 0x04, 0x04
    .db 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x84, 0x84,
0x84, 0x84, 0x84, 0x04, 0x04, 0x04
    .db 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04,
0x04, 0x04, 0x04, 0x04, 0x04, 0x04
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0xE0, 0x38, 0x0C, 0x03, 0x01, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x8E, 0xDF, 0x61, 0x61, 0x61
    .db 0x61, 0x9F, 0x8E, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1E, 0x3F, 0x61, 0x40,
0x40, 0x40, 0x01, 0xFF, 0xFE, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x10, 0x1F, 0x03, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x02, 0x0F, 0x18, 0x10, 0x10, 0x10
    .db 0x10, 0x1F, 0x0F, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x04, 0x0C, 0x08,
0x08, 0x08, 0x0C, 0x07, 0x03, 0x00
```

```
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00

Numero2:

    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x80
    .db 0x80, 0x40, 0x60, 0xF8, 0xF8, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0xFE, 0xFE, 0xFE, 0xFE, 0xFE, 0xFE
    .db 0xFE, 0xFE, 0xFE, 0xFE, 0xFE, 0xFE, 0xFE, 0xFE, 0xFE, 0xFE,
0xFE, 0xFE, 0xFE, 0xFE, 0xFE, 0xFE
    .db 0xFE, 0xFE, 0xFE, 0xFE, 0xFE, 0xFE, 0xFE, 0xFE, 0xFE, 0xFE,
0xFE, 0xFE, 0xFE, 0xFE, 0xFE, 0xFE
    .db 0xFE, 0xFE, 0xFE, 0xFE, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x30, 0x18, 0x08,
0x08, 0x08, 0x98, 0xF0, 0x60, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF
    .db 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF
    .db 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF
    .db 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x60, 0xC0, 0x80,
0x83, 0x83, 0x83, 0xFC, 0x7C, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30,
0x30, 0x30, 0x30, 0x30, 0x30, 0x30
    .db 0x30, 0x30, 0x30, 0x33, 0x33, 0x30, 0x30, 0x30, 0x30, 0x30,
0x30, 0x30, 0x30, 0x30, 0x30, 0x30
    .db 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0xFF, 0xFF,
0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F
```

```
    .db 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F,
0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F
    .db 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F,
0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F
    .db 0x3F, 0x3F, 0x3F, 0x3F, 0xFF, 0xFF, 0x30, 0x30, 0x30, 0x30,
0x30, 0x30, 0x30, 0x30, 0x30, 0x30
    .db 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30,
0x30, 0x30, 0x30, 0x30, 0x30, 0x30
    .db 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30,
0x30, 0x30, 0x30, 0x30, 0x30, 0x30
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0xC0, 0x20, 0x18, 0x0C, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x70, 0x7F, 0x43, 0x63, 0x63
    .db 0x63, 0xC3, 0x83, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xC0, 0xF8, 0x9C, 0x02,
0x42, 0x42, 0x42, 0xC6, 0x84, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x0E, 0x0F
    .db 0x0C, 0x0C, 0x0C, 0x0C, 0x7F, 0x7F, 0x0C, 0x0C, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x18, 0x38, 0x20, 0x20, 0x20
    .db 0x20, 0x30, 0x1F, 0x06, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x0F, 0x1F, 0x30,
0x20, 0x20, 0x20, 0x30, 0x1F, 0x06
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
```

```
    .db 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04,
0x04, 0x04, 0x04, 0x04, 0xC4, 0xC4
    .db 0xC4, 0xC4, 0xC4, 0xC4, 0xC4, 0xC4, 0xC4, 0x04, 0x04, 0x04,
0x04, 0x04, 0x04, 0x04, 0x04, 0x04
    .db 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0xFF, 0xFF,
0x04, 0x04, 0x04, 0x04, 0x04, 0x04
    .db 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04,
0x04, 0x04, 0x04, 0x04, 0x04, 0x04
    .db 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04,
0x04, 0x04, 0x04, 0x04, 0x04, 0x04
    .db 0x04, 0x04, 0x04, 0x04, 0xFF, 0xFF, 0x04, 0x04, 0x04, 0x04,
0x04, 0x04, 0x04, 0x04, 0x04, 0x04
    .db 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x84, 0x84,
0x84, 0x84, 0x84, 0x04, 0x04, 0x04
    .db 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04,
0x04, 0x04, 0x04, 0x04, 0x04, 0x04
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0xE0, 0x38, 0x0C, 0x03, 0x01, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x8E, 0xDF, 0x61, 0x61, 0x61
    .db 0x61, 0x9F, 0x8E, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1E, 0x3F, 0x61, 0x40,
0x40, 0x40, 0x01, 0xFF, 0xFE, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x10, 0x1F, 0x03, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x02, 0x0F, 0x18, 0x10, 0x10, 0x10
    .db 0x10, 0x1F, 0x0F, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
```

```
    .db 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x04, 0x0C, 0x08,
0x08, 0x08, 0x0C, 0x07, 0x03, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00

Numero3:

    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x80
    .db 0x80, 0x40, 0x60, 0xF8, 0xF8, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x30, 0x18, 0x08, 0x08, 0x08
    .db 0x08, 0xF8, 0xF0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0xFE, 0xFE, 0xFE, 0xFE,
0xFE, 0xFE, 0xFE, 0xFE, 0xFE, 0xFE
    .db 0xFE, 0xFE, 0xFE, 0xFE, 0xFE, 0xFE, 0xFE, 0xFE, 0xFE, 0xFE,
0xFE, 0xFE, 0xFE, 0xFE, 0xFE, 0xFE
    .db 0xFE, 0xFE, 0xFE, 0xFE, 0xFE, 0xFE, 0xFE, 0xFE, 0xFE, 0xFE,
0xFE, 0xFE, 0xFE, 0xFE, 0xFE, 0xFE
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0xC0, 0xE0, 0xF0, 0x98, 0x8C
    .db 0x86, 0x83, 0x81, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF
    .db 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF
    .db 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF
    .db 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30,
0x30, 0x30, 0x30, 0x30, 0x30, 0x30
```

```
    .db 0x30, 0x30, 0x30, 0x33, 0x33, 0x30, 0x30, 0x30, 0x30, 0x30,
0x30, 0x30, 0x30, 0x30, 0x30, 0x30
    .db 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0xFF, 0xFF,
0x30, 0x30, 0x30, 0x30, 0x30, 0x30
    .db 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30,
0x31, 0x31, 0x31, 0x31, 0x31, 0x31
    .db 0x31, 0x31, 0x31, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30,
0x30, 0x30, 0x30, 0x30, 0x30, 0x30
    .db 0x30, 0x30, 0x30, 0x30, 0xFF, 0xFF, 0x3F, 0x3F, 0x3F, 0x3F,
0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F
    .db 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F,
0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F
    .db 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F,
0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0xC0, 0x20, 0x18, 0x0C, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x70, 0x7F, 0x43, 0x63, 0x63
    .db 0x63, 0xC3, 0x83, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xC0, 0xF8, 0x9C, 0x02,
0x42, 0x42, 0x42, 0xC6, 0x84, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x0E, 0x0F
    .db 0x0C, 0x0C, 0x0C, 0x0C, 0x7F, 0x7F, 0x0C, 0x0C, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x18, 0x38, 0x20, 0x20, 0x20
    .db 0x20, 0x30, 0x1F, 0x06, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
```

```
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x0F, 0x1F, 0x30,
0x20, 0x20, 0x20, 0x30, 0x1F, 0x06
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04,
0x04, 0x04, 0x04, 0x04, 0xC4, 0xC4
    .db 0xC4, 0xC4, 0xC4, 0xC4, 0xC4, 0xC4, 0xC4, 0x04, 0x04, 0x04,
0x04, 0x04, 0x04, 0x04, 0x04, 0x04
    .db 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0xFF, 0xFF,
0x04, 0x04, 0x04, 0x04, 0x04, 0x04
    .db 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04,
0x04, 0x04, 0x04, 0x04, 0x04, 0x04
    .db 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04,
0x04, 0x04, 0x04, 0x04, 0x04, 0x04
    .db 0x04, 0x04, 0x04, 0x04, 0xFF, 0xFF, 0x04, 0x04, 0x04, 0x04,
0x04, 0x04, 0x04, 0x04, 0x04, 0x04
    .db 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x84, 0x84,
0x84, 0x84, 0x84, 0x04, 0x04, 0x04
    .db 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04,
0x04, 0x04, 0x04, 0x04, 0x04, 0x04
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0xE0, 0x38, 0x0C, 0x03, 0x01, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x8E, 0xDF, 0x61, 0x61, 0x61
    .db 0x61, 0x9F, 0x8E, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1E, 0x3F, 0x61, 0x40,
0x40, 0x40, 0x01, 0xFF, 0xFE, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x10, 0x1F, 0x03, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
```

```
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x02, 0x0F, 0x18, 0x10, 0x10, 0x10
    .db 0x10, 0x1F, 0x0F, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x04, 0x0C, 0x08,
0x08, 0x08, 0x0C, 0x07, 0x03, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00

Numero4:

    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x80
    .db 0x80, 0x40, 0x60, 0xF8, 0xF8, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x30, 0x18, 0x08, 0x08, 0x08
    .db 0x08, 0xF8, 0xF0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x30, 0x18, 0x08,
0x08, 0x08, 0x98, 0xF0, 0x60, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0xC0, 0xE0, 0xF0, 0x98, 0x8C
    .db 0x86, 0x83, 0x81, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x60, 0xC0, 0x80,
0x83, 0x83, 0x83, 0xFC, 0x7C, 0x00
```

```
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0,
0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0
    .db 0xF0, 0xF0, 0xF0, 0xF3, 0xF3, 0xF0, 0xF0, 0xF0, 0xF0,
0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0
    .db 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xFF,
0x30, 0x30, 0x30, 0x30, 0x30, 0x30
    .db 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30,
0x31, 0x31, 0x31, 0x31, 0x31, 0x31
    .db 0x31, 0x31, 0x31, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30,
0x30, 0x30, 0x30, 0x30, 0x30, 0x30
    .db 0x30, 0x30, 0x30, 0x30, 0xFF, 0xFF, 0x30, 0x30, 0x30,
0x30, 0x30, 0x30, 0x30, 0x30, 0x30
    .db 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30,
0x30, 0x30, 0x30, 0x30, 0x30, 0x30
    .db 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30,
0x30, 0x30, 0x30, 0x30, 0x30, 0x30
    .db 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF
    .db 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF
    .db 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x70, 0x7F, 0x43, 0x63, 0x63
    .db 0x63, 0xC3, 0x83, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xC0, 0xF8, 0x9C, 0x02,
0x42, 0x42, 0x42, 0xC6, 0x84, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF
    .db 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF
    .db 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x18, 0x38, 0x20, 0x20, 0x20
```

```
    .db 0x20, 0x30, 0x1F, 0x06, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x0F, 0x1F, 0x30,
0x20, 0x20, 0x20, 0x30, 0x1F, 0x06
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07,
0x07, 0x07, 0x07, 0x07, 0xC7, 0xC7
    .db 0xC7, 0xC7, 0xC7, 0xC7, 0xC7, 0xC7, 0xC7, 0x07, 0x07, 0x07,
0x07, 0x07, 0x07, 0x07, 0x07, 0x07
    .db 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0xFF, 0xFF,
0x04, 0x04, 0x04, 0x04, 0x04, 0x04
    .db 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04,
0x04, 0x04, 0x04, 0x04, 0x04, 0x04
    .db 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04,
0x04, 0x04, 0x04, 0x04, 0x04, 0x04
    .db 0x04, 0x04, 0x04, 0x04, 0xFF, 0xFF, 0x04, 0x04, 0x04, 0x04,
0x04, 0x04, 0x04, 0x04, 0x04, 0x04
    .db 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x84, 0x84,
0x84, 0x84, 0x84, 0x04, 0x04, 0x04
    .db 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04,
0x04, 0x04, 0x04, 0x04, 0x04, 0x04
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0xE0, 0x38, 0x0C, 0x03, 0x01, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x8E, 0xDF, 0x61, 0x61, 0x61
    .db 0x61, 0x9F, 0x8E, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1E, 0x3F, 0x61, 0x40,
0x40, 0x40, 0x01, 0xFF, 0xFE, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
```

```
    .db 0x10, 0x1F, 0x03, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x02, 0x0F, 0x18, 0x10, 0x10, 0x10
    .db 0x10, 0x1F, 0x0F, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x04, 0x0C, 0x08,
0x08, 0x08, 0x0C, 0x07, 0x03, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00


Numero5:

    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x80
    .db 0x80, 0x40, 0x60, 0xF8, 0xF8, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x30, 0x18, 0x08, 0x08, 0x08
    .db 0x08, 0xF8, 0xF0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x30, 0x18, 0x08,
0x08, 0x08, 0x98, 0xF0, 0x60, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0xC0, 0xE0, 0xF0, 0x98, 0x8C
```

```
    .db 0x86, 0x83, 0x81, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x60, 0xC0, 0x80,
0x83, 0x83, 0x83, 0xFC, 0x7C, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30,
0x30, 0x30, 0x30, 0x30, 0x30, 0x30
    .db 0x30, 0x30, 0x30, 0x33, 0x33, 0x30, 0x30, 0x30, 0x30, 0x30,
0x30, 0x30, 0x30, 0x30, 0x30, 0x30
    .db 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0xFF, 0xFF,
0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0
    .db 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0,
0xF1, 0xF1, 0xF1, 0xF1, 0xF1, 0xF1
    .db 0xF1, 0xF1, 0xF1, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0,
0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0
    .db 0xF0, 0xF0, 0xF0, 0xF0, 0xFF, 0xFF, 0x30, 0x30, 0x30, 0x30,
0x30, 0x30, 0x30, 0x30, 0x30, 0x30
    .db 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30,
0x30, 0x30, 0x30, 0x30, 0x30, 0x30
    .db 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30,
0x30, 0x30, 0x30, 0x30, 0x30, 0x30
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0xC0, 0x20, 0x18, 0x0C, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF
    .db 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF
    .db 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF
    .db 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xC0, 0xF8, 0x9C, 0x02,
0x42, 0x42, 0x42, 0xC6, 0x84, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x0E, 0x0F
```

```
    .db 0x0C, 0x0C, 0x0C, 0x0C, 0x7F, 0x7F, 0x0C, 0x0C, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF
    .db 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF
    .db 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF
    .db 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x0F, 0x1F, 0x30,
0x20, 0x20, 0x20, 0x30, 0x1F, 0x06
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04,
0x04, 0x04, 0x04, 0x04, 0xC4, 0xC4
    .db 0xC4, 0xC4, 0xC4, 0xC4, 0xC4, 0xC4, 0xC4, 0x04, 0x04, 0x04,
0x04, 0x04, 0x04, 0x04, 0x04, 0x04
    .db 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0xFF, 0xFF,
0x07, 0x07, 0x07, 0x07, 0x07, 0x07
    .db 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07,
0x07, 0x07, 0x07, 0x07, 0x07, 0x07
    .db 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07,
0x07, 0x07, 0x07, 0x07, 0x07, 0x07
    .db 0x07, 0x07, 0x07, 0x07, 0xFF, 0xFF, 0x04, 0x04, 0x04, 0x04,
0x04, 0x04, 0x04, 0x04, 0x04, 0x04
    .db 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x84, 0x84,
0x84, 0x84, 0x84, 0x04, 0x04, 0x04
    .db 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04,
0x04, 0x04, 0x04, 0x04, 0x04, 0x04
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0xE0, 0x38, 0x0C, 0x03, 0x01, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x8E, 0xDF, 0x61, 0x61, 0x61
    .db 0x61, 0x9F, 0x8E, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
```

```
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1E, 0x3F, 0x61, 0x40,
0x40, 0x40, 0x01, 0xFF, 0xFE, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x10, 0x1F, 0x03, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x02, 0x0F, 0x18, 0x10, 0x10, 0x10
    .db 0x10, 0x1F, 0x0F, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x04, 0x0C, 0x08,
0x08, 0x08, 0x0C, 0x07, 0x03, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00

Numero6:

    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x80
    .db 0x80, 0x40, 0x60, 0xF8, 0xF8, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x30, 0x18, 0x08, 0x08, 0x08
    .db 0x08, 0xF8, 0xF0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x30, 0x18, 0x08,
0x08, 0x08, 0x98, 0xF0, 0x60, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
```

```
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0xC0, 0xE0, 0xF0, 0x98, 0x8C
    .db 0x86, 0x83, 0x81, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x60, 0xC0, 0x80,
0x83, 0x83, 0x83, 0xFC, 0x7C, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30,
0x30, 0x30, 0x30, 0x30, 0x30, 0x30
    .db 0x30, 0x30, 0x30, 0x33, 0x33, 0x30, 0x30, 0x30, 0x30, 0x30,
0x30, 0x30, 0x30, 0x30, 0x30, 0x30
    .db 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0xFF, 0xFF,
0x30, 0x30, 0x30, 0x30, 0x30, 0x30
    .db 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30,
0x31, 0x31, 0x31, 0x31, 0x31, 0x31
    .db 0x31, 0x31, 0x31, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30,
0x30, 0x30, 0x30, 0x30, 0x30, 0x30
    .db 0x30, 0x30, 0x30, 0x30, 0xFF, 0xFF, 0xF0, 0xF0, 0xF0, 0xF0,
0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0
    .db 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0,
0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0
    .db 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0,
0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0xC0, 0x20, 0x18, 0x0C, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x70, 0x7F, 0x43, 0x63, 0x63
    .db 0x63, 0xC3, 0x83, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF
    .db 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF
```

```
    .db 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x0E, 0x0F
    .db 0x0C, 0x0C, 0x0C, 0x0C, 0x7F, 0x7F, 0x0C, 0x0C, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x18, 0x38, 0x20, 0x20, 0x20
    .db 0x20, 0x30, 0x1F, 0x06, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF
    .db 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF
    .db 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF
    .db 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04,
0x04, 0x04, 0x04, 0x04, 0xC4, 0xC4
    .db 0xC4, 0xC4, 0xC4, 0xC4, 0xC4, 0xC4, 0xC4, 0x04, 0x04, 0x04,
0x04, 0x04, 0x04, 0x04, 0x04, 0x04
    .db 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0xFF, 0xFF,
0x04, 0x04, 0x04, 0x04, 0x04, 0x04
    .db 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04,
0x04, 0x04, 0x04, 0x04, 0x04, 0x04
    .db 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04,
0x04, 0x04, 0x04, 0x04, 0x04, 0x04
    .db 0x04, 0x04, 0x04, 0x04, 0xFF, 0xFF, 0x07, 0x07, 0x07, 0x07,
0x07, 0x07, 0x07, 0x07, 0x07, 0x07
    .db 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x87, 0x87,
0x87, 0x87, 0x87, 0x07, 0x07, 0x07
    .db 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07,
0x07, 0x07, 0x07, 0x07, 0x07, 0x07
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0xE0, 0x38, 0x0C, 0x03, 0x01, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x8E, 0xDF, 0x61, 0x61, 0x61
```

```
    .db 0x61, 0x9F, 0x8E, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1E, 0x3F, 0x61, 0x40,
0x40, 0x40, 0x01, 0xFF, 0xFE, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x10, 0x1F, 0x03, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x02, 0x0F, 0x18, 0x10, 0x10, 0x10
    .db 0x10, 0x1F, 0x0F, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x04, 0x0C, 0x08,
0x08, 0x08, 0x0C, 0x07, 0x03, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00

Numero7:

    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x80
    .db 0x80, 0x40, 0x60, 0xF8, 0xF8, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x30, 0x18, 0x08, 0x08, 0x08
    .db 0x08, 0xF8, 0xF0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x30, 0x18, 0x08,
0x08, 0x08, 0x98, 0xF0, 0x60, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
```

```
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0xC0, 0xE0, 0xF0, 0x98, 0x8C
    .db 0x86, 0x83, 0x81, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x60, 0xC0, 0x80,
0x83, 0x83, 0x83, 0xFC, 0x7C, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30,
0x30, 0x30, 0x30, 0x30, 0x30, 0x30
    .db 0x30, 0x30, 0x30, 0x33, 0x33, 0x30, 0x30, 0x30, 0x30, 0x30,
0x30, 0x30, 0x30, 0x30, 0x30, 0x30
    .db 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0xFF, 0xFF,
0x30, 0x30, 0x30, 0x30, 0x30, 0x30
    .db 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30,
0x31, 0x31, 0x31, 0x31, 0x31, 0x31
    .db 0x31, 0x31, 0x31, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30,
0x30, 0x30, 0x30, 0x30, 0x30, 0x30
    .db 0x30, 0x30, 0x30, 0x30, 0xFF, 0xFF, 0x30, 0x30, 0x30, 0x30,
0x30, 0x30, 0x30, 0x30, 0x30, 0x30
    .db 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30,
0x30, 0x30, 0x30, 0x30, 0x30, 0x30
    .db 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30,
0x30, 0x30, 0x30, 0x30, 0x30, 0x30
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0xC0, 0x20, 0x18, 0x0C, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x70, 0x7F, 0x43, 0x63, 0x63
    .db 0x63, 0xC3, 0x83, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
```

```
    .db 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xC0, 0xF8, 0x9C, 0x02,
0x42, 0x42, 0x42, 0xC6, 0x84, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x0E, 0x0F
    .db 0x0C, 0x0C, 0x0C, 0x0C, 0x7F, 0x7F, 0x0C, 0x0C, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x18, 0x38, 0x20, 0x20, 0x20
    .db 0x20, 0x30, 0x1F, 0x06, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x0F, 0x1F, 0x30,
0x20, 0x20, 0x20, 0x30, 0x1F, 0x06
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0xFC, 0xFC, 0xFC, 0xFC, 0xFC, 0xFC, 0xFC, 0xFC, 0xFC, 0xFC,
0xFC, 0xFC, 0xFC, 0xFC, 0xFC, 0xFC
    .db 0xFC, 0xFC, 0xFC, 0xFC, 0xFC, 0xFC, 0xFC, 0xFC, 0xFC, 0xFC,
0xFC, 0xFC, 0xFC, 0xFC, 0xFC, 0xFC
    .db 0xFC, 0xFC, 0xFC, 0xFC, 0xFC, 0xFC, 0xFC, 0xFC, 0xFF, 0xFF,
0x04, 0x04, 0x04, 0x04, 0x04, 0x04
    .db 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04,
0x04, 0x04, 0x04, 0x04, 0x04, 0x04
    .db 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04,
0x04, 0x04, 0x04, 0x04, 0x04, 0x04
    .db 0x04, 0x04, 0x04, 0x04, 0xFF, 0xFF, 0x04, 0x04, 0x04, 0x04,
0x04, 0x04, 0x04, 0x04, 0x04, 0x04
    .db 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x84, 0x84,
0x84, 0x84, 0x84, 0x04, 0x04, 0x04
    .db 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04,
0x04, 0x04, 0x04, 0x04, 0x04, 0x04
    .db 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF
    .db 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF
```

```
    .db 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x8E, 0xDF, 0x61, 0x61, 0x61
    .db 0x61, 0x9F, 0x8E, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1E, 0x3F, 0x61, 0x40,
0x40, 0x40, 0x01, 0xFF, 0xFE, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF
    .db 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF
    .db 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x02, 0x0F, 0x18, 0x10, 0x10, 0x10
    .db 0x10, 0x1F, 0x0F, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x04, 0x0C, 0x08,
0x08, 0x08, 0x0C, 0x07, 0x03, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00


Numero8:

    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x80
    .db 0x80, 0x40, 0x60, 0xF8, 0xF8, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x30, 0x18, 0x08, 0x08, 0x08
    .db 0x08, 0xF8, 0xF0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
```

```
    .db 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x30, 0x18, 0x08,
0x08, 0x08, 0x98, 0xF0, 0x60, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0xC0, 0xE0, 0xF0, 0x98, 0x8C
    .db 0x86, 0x83, 0x81, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x60, 0xC0, 0x80,
0x83, 0x83, 0x83, 0xFC, 0x7C, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30,
0x30, 0x30, 0x30, 0x30, 0x30, 0x30
    .db 0x30, 0x30, 0x30, 0x33, 0x33, 0x30, 0x30, 0x30, 0x30, 0x30,
0x30, 0x30, 0x30, 0x30, 0x30, 0x30
    .db 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0xFF, 0xFF,
0x30, 0x30, 0x30, 0x30, 0x30, 0x30
    .db 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30,
0x31, 0x31, 0x31, 0x31, 0x31, 0x31
    .db 0x31, 0x31, 0x31, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30,
0x30, 0x30, 0x30, 0x30, 0x30, 0x30
    .db 0x30, 0x30, 0x30, 0x30, 0xFF, 0xFF, 0x30, 0x30, 0x30, 0x30,
0x30, 0x30, 0x30, 0x30, 0x30, 0x30
    .db 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30,
0x30, 0x30, 0x30, 0x30, 0x30, 0x30
    .db 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30,
0x30, 0x30, 0x30, 0x30, 0x30, 0x30
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0xC0, 0x20, 0x18, 0x0C, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
```

```
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x70, 0x7F, 0x43, 0x63, 0x63
    .db 0x63, 0xC3, 0x83, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xC0, 0xF8, 0x9C, 0x02,
0x42, 0x42, 0x42, 0xC6, 0x84, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x0E, 0x0F
    .db 0x0C, 0x0C, 0x0C, 0x0C, 0x7F, 0x7F, 0x0C, 0x0C, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x18, 0x38, 0x20, 0x20, 0x20
    .db 0x20, 0x30, 0x1F, 0x06, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x0F, 0x1F, 0x30,
0x20, 0x20, 0x20, 0x30, 0x1F, 0x06
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04,
0x04, 0x04, 0x04, 0x04, 0xC4, 0xC4
    .db 0xC4, 0xC4, 0xC4, 0xC4, 0xC4, 0xC4, 0xC4, 0x04, 0x04, 0x04,
0x04, 0x04, 0x04, 0x04, 0x04, 0x04
    .db 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0xFF, 0xFF,
0xF4, 0xF4, 0xF4, 0xF4, 0xF4, 0xF4
    .db 0xF4, 0xF4, 0xF4, 0xF4, 0xF4, 0xF4, 0xF4, 0xF4, 0xF4, 0xF4,
0xF4, 0xF4, 0xF4, 0xF4, 0xF4, 0xF4
    .db 0xF4, 0xF4, 0xF4, 0xF4, 0xF4, 0xF4, 0xF4, 0xF4, 0xF4, 0xF4,
0xF4, 0xF4, 0xF4, 0xF4, 0xF4, 0xF4
    .db 0xF4, 0xF4, 0xF4, 0xF4, 0xFF, 0xFF, 0x04, 0x04, 0x04, 0x04,
0x04, 0x04, 0x04, 0x04, 0x04, 0x04
    .db 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x84, 0x84,
0x84, 0x84, 0x84, 0x04, 0x04, 0x04
```

```
    .db 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04,
0x04, 0x04, 0x04, 0x04, 0x04, 0x04
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0xE0, 0x38, 0x0C, 0x03, 0x01, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF
    .db 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF
    .db 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF
    .db 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1E, 0x3F, 0x61, 0x40,
0x40, 0x40, 0x01, 0xFF, 0xFE, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x10, 0x1F, 0x03, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF
    .db 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF
    .db 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF
    .db 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x04, 0x0C, 0x08,
0x08, 0x08, 0x0C, 0x07, 0x03, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00


Numero9:

    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x80
    .db 0x80, 0x40, 0x60, 0xF8, 0xF8, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
```

```
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x30, 0x18, 0x08, 0x08, 0x08
    .db 0x08, 0xF8, 0xF0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x30, 0x18, 0x08,
0x08, 0x08, 0x98, 0xF0, 0x60, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0xC0, 0xE0, 0xF0, 0x98, 0x8C
    .db 0x86, 0x83, 0x81, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x60, 0xC0, 0x80,
0x83, 0x83, 0x83, 0xFC, 0x7C, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30,
0x30, 0x30, 0x30, 0x30, 0x30, 0x30
    .db 0x30, 0x30, 0x30, 0x33, 0x33, 0x30, 0x30, 0x30, 0x30, 0x30,
0x30, 0x30, 0x30, 0x30, 0x30, 0x30
    .db 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0xFF, 0xFF,
0x30, 0x30, 0x30, 0x30, 0x30, 0x30
    .db 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30,
0x31, 0x31, 0x31, 0x31, 0x31, 0x31
    .db 0x31, 0x31, 0x31, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30,
0x30, 0x30, 0x30, 0x30, 0x30, 0x30
    .db 0x30, 0x30, 0x30, 0x30, 0xFF, 0xFF, 0x30, 0x30, 0x30, 0x30,
0x30, 0x30, 0x30, 0x30, 0x30, 0x30
    .db 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30,
0x30, 0x30, 0x30, 0x30, 0x30, 0x30
```

```
    .db 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30,
0x30, 0x30, 0x30, 0x30, 0x30, 0x30
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0xC0, 0x20, 0x18, 0x0C, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x70, 0x7F, 0x43, 0x63, 0x63
    .db 0x63, 0xC3, 0x83, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xC0, 0xF8, 0x9C, 0x02,
0x42, 0x42, 0x42, 0xC6, 0x84, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x0E, 0x0F
    .db 0x0C, 0x0C, 0x0C, 0x0C, 0x7F, 0x7F, 0x0C, 0x0C, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x18, 0x38, 0x20, 0x20, 0x20
    .db 0x20, 0x30, 0x1F, 0x06, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x0F, 0x1F, 0x30,
0x20, 0x20, 0x20, 0x30, 0x1F, 0x06
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04,
0x04, 0x04, 0x04, 0x04, 0xC4, 0xC4
    .db 0xC4, 0xC4, 0xC4, 0xC4, 0xC4, 0xC4, 0xC4, 0x04, 0x04, 0x04,
0x04, 0x04, 0x04, 0x04, 0x04, 0x04
    .db 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0xFF, 0xFF,
0x04, 0x04, 0x04, 0x04, 0x04, 0x04
    .db 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04,
0x04, 0x04, 0x04, 0x04, 0x04, 0x04
```

```
    .db 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04,
0x04, 0x04, 0x04, 0x04, 0x04, 0x04
    .db 0x04, 0x04, 0x04, 0x04, 0xFF, 0xFF, 0xFC, 0xFC, 0xFC, 0xFC,
0xFC, 0xFC, 0xFC, 0xFC, 0xFC, 0xFC
    .db 0xFC, 0xFC, 0xFC, 0xFC, 0xFC, 0xFC, 0xFC, 0xFC, 0xFC, 0xFC,
0xFC, 0xFC, 0xFC, 0xFC, 0xFC, 0xFC
    .db 0xFC, 0xFC, 0xFC, 0xFC, 0xFC, 0xFC, 0xFC, 0xFC, 0xFC, 0xFC,
0xFC, 0xFC, 0xFC, 0xFC, 0xFC, 0xFC
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0xE0, 0x38, 0x0C, 0x03, 0x01, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x8E, 0xDF, 0x61, 0x61, 0x61
    .db 0x61, 0x9F, 0x8E, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF
    .db 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF
    .db 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x10, 0x1F, 0x03, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x02, 0x0F, 0x18, 0x10, 0x10, 0x10
    .db 0x10, 0x1F, 0x0F, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF
    .db 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF
    .db 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF

ULL:
```

```
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x80
    .db 0xC0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x80, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x80, 0xC0, 0xE0,
0xF0, 0xF8, 0xFC, 0xFE, 0xFF, 0xFF
    .db 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x80, 0xC0, 0xE0, 0xF0, 0xF8, 0xFC
    .db 0xFE, 0xFE, 0xFF, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0xE0, 0xF0, 0xF0, 0xF8, 0xFC, 0xFE, 0xFF, 0xFF, 0xFF, 0xFF,
0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF
    .db 0xFF, 0xE0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0xE0, 0xF0, 0xF8, 0xFC, 0xFE, 0xFF,
0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF
```

```
    .db 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF
    .db 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF
    .db 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF
    .db 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF
    .db 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
```

```
    .db 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF
    .db 0xFF, 0xFF, 0xF8, 0xF8, 0xF8, 0xF8, 0xF8, 0xFC, 0xFC, 0xFC,
0xFC, 0xFC, 0xFC, 0x7C, 0x3C, 0x1C
    .db 0x0C, 0x04, 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF
    .db 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFC, 0xFC, 0xFC, 0xFC, 0xFC,
0xFC, 0xFC, 0xFC, 0xFC, 0xFC, 0xFC
    .db 0xFC, 0x7C, 0x3C, 0x1C, 0x0C, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF
    .db 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0x7F, 0x3F, 0x1F, 0x0F, 0x07,
0x03, 0x01, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF
    .db 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0x7F, 0x3F,
0x1F, 0x0F, 0x07, 0x03, 0x01, 0x01
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03,
0x03, 0x03, 0x03, 0x03, 0x03, 0x03
    .db 0x03, 0x03, 0x03, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03,
0x03, 0x03, 0x03, 0x03, 0x03, 0x03
    .db 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x01, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
    .db 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00
```

9.2 Datasheets

A continuación se mostrarán las partes más relevantes de los datasheets utilizados ya que si se incluyese todas las páginas sería demasiado extenso el documento.

HD44780U (LCD-II)

(Dot Matrix Liquid Crystal Display Controller/Driver)

HITACHI

Description

The HD44780U dot-matrix liquid crystal display controller and driver LSI displays alphanumerics, Japanese kana characters, and symbols. It can be configured to drive a dot-matrix liquid crystal display under the control of a 4- or 8-bit microprocessor. Since all the functions such as display RAM, character generator, and liquid crystal driver, required for driving a dot-matrix liquid crystal display are internally provided on one chip, a minimal system can be interfaced with this controller/driver.

A single HD44780U can display up to one 8-character line or two 8-character lines.

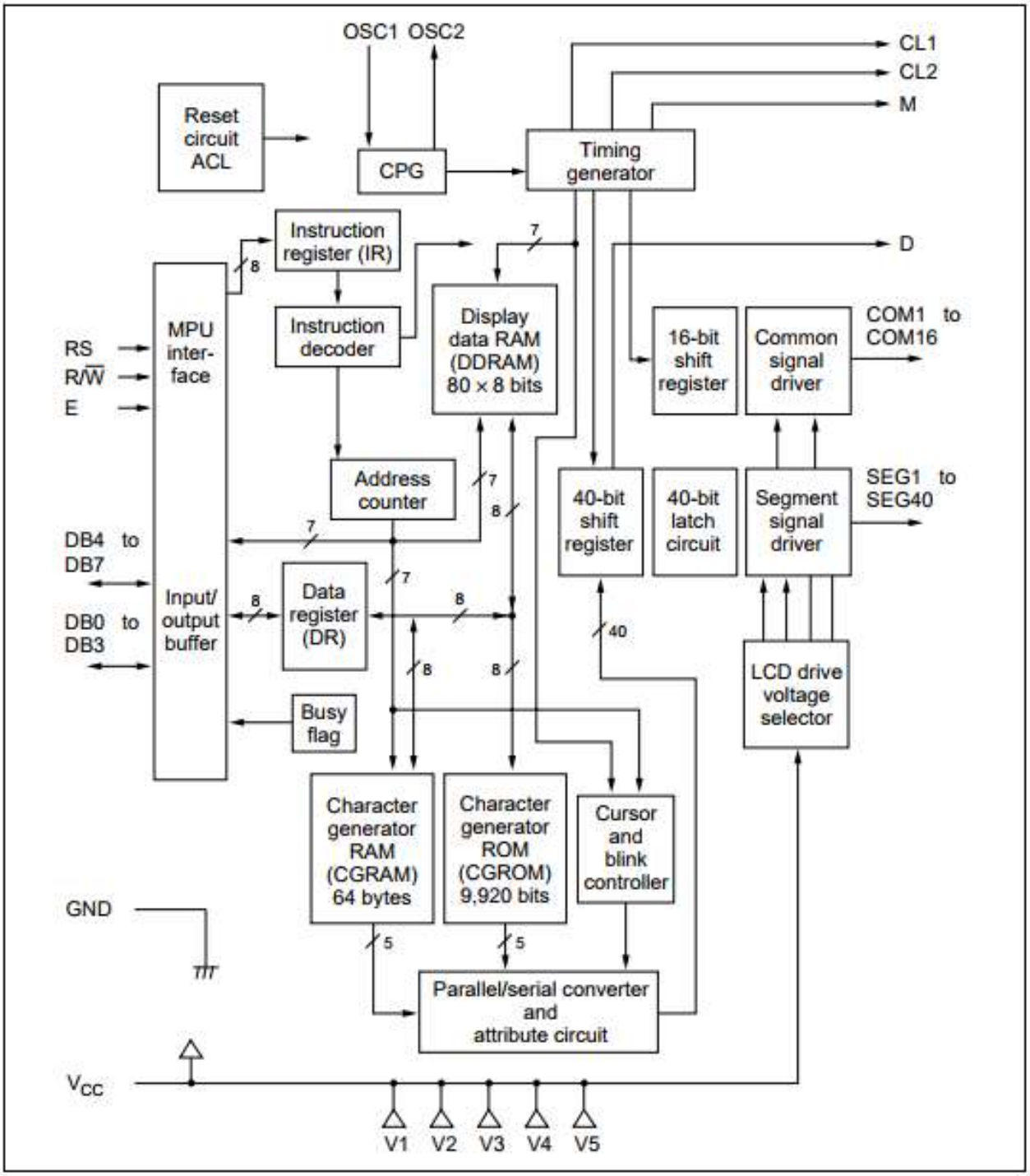
The HD44780U has pin function compatibility with the HD44780S which allows the user to easily replace an LCD-II with an HD44780U. The HD44780U character generator ROM is extended to generate 208 5 × 8 dot character fonts and 32 5 × 10 dot character fonts for a total of 240 different character fonts.

The low power supply (2.7V to 5.5V) of the HD44780U is suitable for any portable battery-driven product requiring low power dissipation.

Features

- 5 × 8 and 5 × 10 dot matrix possible
- Low power operation support:
  - 2.7 to 5.5V
- Wide range of liquid crystal display driver power
  - 3.0 to 11V
- Liquid crystal drive waveform
  - A (One line frequency AC waveform)
- Correspond to high speed MPU bus interface
  - 2 MHz (when V_{CC} = 5V)
- 4-bit or 8-bit MPU interface enabled
- 80 × 8-bit display RAM (80 characters max.)
- 9,920-bit character generator ROM for a total of 240 character fonts
  - 208 character fonts (5 × 8 dot)
  - 32 character fonts (5 × 10 dot)

HD44780U Block Diagram



Pin Functions

Signal	No. of Lines	I/O	Device Interfaced with	Function
RS	1	I	MPU	Selects registers. 0: Instruction register (for write) Busy flag: address counter (for read) 1: Data register (for write and read)
R/$\overline{W}$	1	I	MPU	Selects read or write. 0: Write 1: Read
E	1	I	MPU	Starts data read/write.
DB4 to DB7	4	I/O	MPU	Four high order bidirectional tristate data bus pins. Used for data transfer and receive between the MPU and the HD44780U. DB7 can be used as a busy flag.
DB0 to DB3	4	I/O	MPU	Four low order bidirectional tristate data bus pins. Used for data transfer and receive between the MPU and the HD44780U. These pins are not used during 4-bit operation.
CL1	1	O	Extension driver	Clock to latch serial data D sent to the extension driver
CL2	1	O	Extension driver	Clock to shift serial data D
M	1	O	Extension driver	Switch signal for converting the liquid crystal drive waveform to AC
D	1	O	Extension driver	Character pattern data corresponding to each segment signal
COM1 to COM16	16	O	LCD	Common signals that are not used are changed to non-selection waveforms. COM9 to COM16 are non-selection waveforms at 1/8 duty factor and COM12 to COM16 are non-selection waveforms at 1/11 duty factor.
SEG1 to SEG40	40	O	LCD	Segment signals
V1 to V5	5	—	Power supply	Power supply for LCD drive V_{cc} –V5 = 11 V (max)
V_{cc}, GND	2	—	Power supply	V_{cc}: 2.7V to 5.5V, GND: 0V
OSC1, OSC2	2	—	Oscillation resistor clock	When crystal oscillation is performed, a resistor must be connected externally. When the pin input is an external clock, it must be input to OSC1.

Function Description

Registers

The HD44780U has two 8-bit registers, an instruction register (IR) and a data register (DR).

The IR stores instruction codes, such as display clear and cursor shift, and address information for display data RAM (DDRAM) and character generator RAM (CGRAM). The IR can only be written from the MPU.

The DR temporarily stores data to be written into DDRAM or CGRAM and temporarily stores data to be read from DDRAM or CGRAM. Data written into the DR from the MPU is automatically written into DDRAM or CGRAM by an internal operation. The DR is also used for data storage when reading data from DDRAM or CGRAM. When address information is written into the IR, data is read and then stored into the DR from DDRAM or CGRAM by an internal operation. Data transfer between the MPU is then completed when the MPU reads the DR. After the read, data in DDRAM or CGRAM at the next address is sent to the DR for the next read from the MPU. By the register selector (RS) signal, these two registers can be selected (Table 1).

Busy Flag (BF)

When the busy flag is 1, the HD44780U is in the internal operation mode, and the next instruction will not be accepted. When RS = 0 and R/$\overline{W}$ = 1 (Table 1), the busy flag is output to DB7. The next instruction must be written after ensuring that the busy flag is 0.

Address Counter (AC)

The address counter (AC) assigns addresses to both DDRAM and CGRAM. When an address of an instruction is written into the IR, the address information is sent from the IR to the AC. Selection of either DDRAM or CGRAM is also determined concurrently by the instruction.

After writing into (reading from) DDRAM or CGRAM, the AC is automatically incremented by 1 (decremented by 1). The AC contents are then output to DB0 to DB6 when RS = 0 and R/$\overline{W}$ = 1 (Table 1).

Table 1 Register Selection

RS	R/$\overline{W}$	Operation
0	0	IR write as an internal operation (display clear, etc.)
0	1	Read busy flag (DB7) and address counter (DB0 to DB6)
1	0	DR write as an internal operation (DR to DDRAM or CGRAM)
1	1	DR read as an internal operation (DDRAM or CGRAM to DR)

Table 4 Correspondence between Character Codes and Character Patterns (ROM Code: A00)

Upper 4 Bits / Lower 4 Bits	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
xxxx0000	CG RAM (1)															
xxxx0001	(2)															
xxxx0010	(3)															
xxxx0011	(4)															
xxxx0100	(5)															
xxxx0101	(6)															
xxxx0110	(7)															
xxxx0111	(8)															
xxxx1000	(1)															
xxxx1001	(2)															
xxxx1010	(3)															
xxxx1011	(4)															
xxxx1100	(5)															
xxxx1101	(6)															
xxxx1110	(7)															
xxxx1111	(8)															

Note: The user can specify any pattern for character-generator RAM.

Table 4 Correspondence between Character Codes and Character Patterns (ROM Code: A02)

Lower 4 Bits \ Upper 4 Bits	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
xxxx0000	CG RAM (1)															
xxxx0001	(2)															
xxxx0010	(3)															
xxxx0011	(4)															
xxxx0100	(5)															
xxxx0101	(6)															
xxxx0110	(7)															
xxxx0111	(8)															
xxxx1000	(1)															
xxxx1001	(2)															
xxxx1010	(3)															
xxxx1011	(4)															
xxxx1100	(5)															
xxxx1101	(6)															
xxxx1110	(7)															
xxxx1111	(8)															

Reset Function

Initializing by Internal Reset Circuit

An internal reset circuit automatically initializes the HD44780U when the power is turned on. The following instructions are executed during the initialization. The busy flag (BF) is kept in the busy state until the initialization ends (BF = 1). The busy state lasts for 10 ms after V_{CC} rises to 4.5 V.

1. Display clear
2. Function set:
   DL = 1; 8-bit interface data
   N = 0; 1-line display
   F = 0; 5 × 8 dot character font
3. Display on/off control:
   D = 0; Display off
   C = 0; Cursor off
   B = 0; Blinking off
4. Entry mode set:
   I/D = 1; Increment by 1
   S = 0; No shift

Note: If the electrical characteristics conditions listed under the table Power Supply Conditions Using Internal Reset Circuit are not met, the internal reset circuit will not operate normally and will fail to initialize the HD44780U. For such a case, initial-ization must be performed by the MPU as explained in the section, Initializing by Instruction.

Instructions

Outline

Only the instruction register (IR) and the data register (DR) of the HD44780U can be controlled by the MPU. Before starting the internal operation of the HD44780U, control information is temporarily stored into these registers to allow interfacing with various MPUs, which operate at different speeds, or various peripheral control devices. The internal operation of the HD44780U is determined by signals sent from the MPU. These signals, which include register selection signal (RS), read/

write signal (R/$\overline{W}$), and the data bus (DB0 to DB7), make up the HD44780U instructions (Table 6). There are four categories of instructions that:

- Designate HD44780U functions, such as display format, data length, etc.
- Set internal RAM addresses
- Perform data transfer with internal RAM
- Perform miscellaneous functions

Normally, instructions that perform data transfer with internal RAM are used the most. However, auto-incrementation by 1 (or auto-decrementation by 1) of internal HD44780U RAM addresses after each data write can lighten the program load of the MPU. Since the display shift instruction (Table 11) can perform concurrently with display data write, the user can minimize system development time with maximum programming efficiency.

When an instruction is being executed for internal operation, no instruction other than the busy flag/address read instruction can be executed.

Because the busy flag is set to 1 while an instruction is being executed, check it to make sure it is 0 before sending another instruction from the MPU.

Note: Be sure the HD44780U is not in the busy state (BF = 0) before sending an instruction from the MPU to the HD44780U. If an instruction is sent without checking the busy flag, the time between the first instruction and next instruction will take much longer than the instruction time itself. Refer to Table 6 for the list of each instruc-tion execution time.

Table 6 Instructions

	Code											Execution Time (max) (when f_{cp} or f_{osc} is 270 kHz)
Instruction	RS	R/$\overline{W}$	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Description	
Clear display	0	0	0	0	0	0	0	0	0	1	Clears entire display and sets DDRAM address 0 in address counter.	
Return home	0	0	0	0	0	0	0	0	1	—	Sets DDRAM address 0 in address counter. Also returns display from being shifted to original position. DDRAM contents remain unchanged.	1.52 ms
Entry mode set	0	0	0	0	0	0	0	1	I/D	S	Sets cursor move direction and specifies display shift. These operations are performed during data write and read.	37 µs
Display on/off control	0	0	0	0	0	0	1	D	C	B	Sets entire display (D) on/off, cursor on/off (C), and blinking of cursor position character (B).	37 µs
Cursor or display shift	0	0	0	0	0	1	S/C	R/L	—	—	Moves cursor and shifts display without changing DDRAM contents.	37 µs
Function set	0	0	0	0	1	DL	N	F	—	—	Sets interface data length (DL), number of display lines (N), and character font (F).	37 µs
Set CGRAM address	0	0	0	1	ACG	ACG	ACG	ACG	ACG	ACG	Sets CGRAM address. CGRAM data is sent and received after this setting.	37 µs
Set DDRAM address	0	0	1	ADD	ADD	ADD	ADD	ADD	ADD	ADD	Sets DDRAM address. DDRAM data is sent and received after this setting.	37 µs
Read busy flag & address	0	1	BF	AC	AC	AC	AC	AC	AC	AC	Reads busy flag (BF) indicating internal operation is being performed and reads address counter contents.	0 µs

Table 6 Instructions (cont)

Instruction	Code RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Description	Execution Time (max) (when f_{cp} or f_{OSC} is 270 kHz)
Write data to CG or DDRAM	1	0	Write data								Writes data into DDRAM or CGRAM.	37 µs t_{ADD} = 4 µs*
Read data from CG or DDRAM	1	1	Read data								Reads data from DDRAM or CGRAM.	37 µs t_{ADD} = 4 µs*
	I/D = 1: Increment I/D = 0: Decrement S = 1: Accompanies display shift S/C = 1: Display shift S/C = 0: Cursor move R/L = 1: Shift to the right R/L = 0: Shift to the left DL = 1: 8 bits, DL = 0: 4 bits N = 1: 2 lines, N = 0: 1 line F = 1: 5 × 10 dots, F = 0: 5 × 8 dots BF = 1: Internally operating BF = 0: Instructions acceptable										DDRAM: Display data RAM CGRAM: Character generator RAM ACG: CGRAM address ADD: DDRAM address (corresponds to cursor address) AC: Address counter used for both DD and CGRAM addresses	Execution time changes when frequency changes Example: When f_{cp} or f_{OSC} is 250 kHz, $37\ \mu s \times \frac{270}{250} = 40\ \mu s$

Note: — indicates no effect.

* After execution of the CGRAM/DDRAM data write or read instruction, the RAM address counter is incremented or decremented by 1. The RAM address counter is updated after the busy flag turns off. In Figure 10, t_{ADD} is the time elapsed after the busy flag turns off until the address counter is updated.

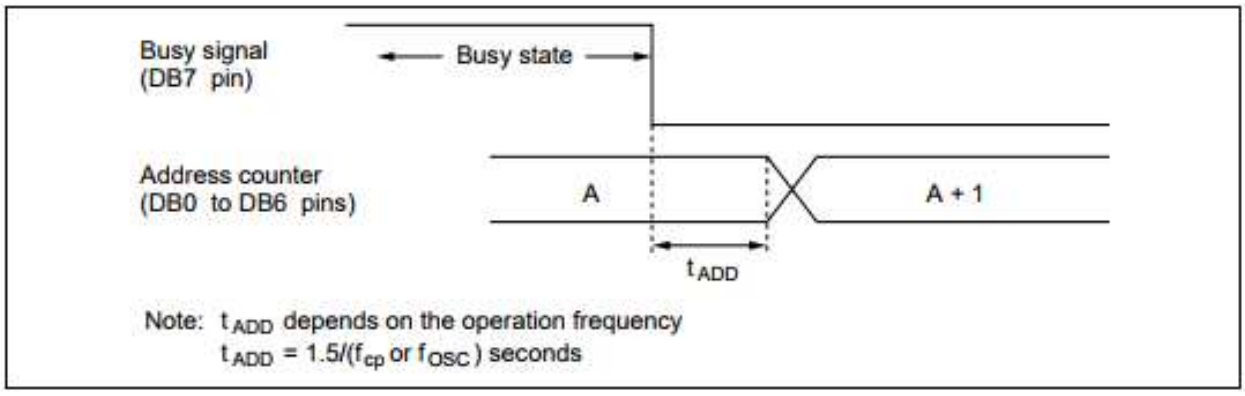


Figure 10 Address Counter Update

Instruction Description

Clear Display

Clear display writes space code 20H (character pattern for character code 20H must be a blank pattern) into all DDRAM addresses. It then sets DDRAM address 0 into the address counter, and returns the display to its original status if it was shifted. In other words, the display disappears and the cursor or blinking goes to the left edge of the display (in the first line if 2 lines are displayed). It also sets I/D to 1 (increment mode) in entry mode. S of entry mode does not change.

Return Home

Return home sets DDRAM address 0 into the address counter, and returns the display to its original status if it was shifted. The DDRAM contents do not change.

The cursor or blinking go to the left edge of the display (in the first line if 2 lines are displayed).

Entry Mode Set

I/D: Increments (I/D = 1) or decrements (I/D = 0) the DDRAM address by 1 when a character code is written into or read from DDRAM.

The cursor or blinking moves to the right when incremented by 1 and to the left when decremented by 1. The same applies to writing and reading of CGRAM.

S: Shifts the entire display either to the right (I/D = 0) or to the left (I/D = 1) when S is 1. The display does not shift if S is 0.

If S is 1, it will seem as if the cursor does not move but the display does. The display does not shift when reading from DDRAM. Also, writing into or reading out from CGRAM does not shift the display.

Display On/Off Control

D: The display is on when D is 1 and off when D is 0. When off, the display data remains in DDRAM, but can be displayed instantly by setting D to 1.

C: The cursor is displayed when C is 1 and not displayed when C is 0. Even if the cursor disappears, the function of I/D or other specifications will not change during display data write. The cursor is displayed using 5 dots in the 8th line for 5 × 8 dot character font selection and in the 11th line for the 5 × 10 dot character font selection (Figure 13).

B: The character indicated by the cursor blinks when B is 1 (Figure 13). The blinking is displayed as switching between all blank dots and displayed characters at a speed of 409.6-ms intervals when f_{cp} or f_{OSC} is 250 kHz. The cursor and blinking can be set to display simultaneously. (The blinking frequency changes according to f_{OSC} or the reciprocal of f_{cp}. For example, when f_{cp} is 270 kHz, 409.6 × 250/270 = 379.2 ms.)

Cursor or Display Shift

Cursor or display shift shifts the cursor position or display to the right or left without writing or reading display data (Table 7). This function is used to correct or search the display. In a 2-line display, the cursor moves to the second line when it passes the 40th digit of the first line. Note that the first and second line displays will shift at the same time.

When the displayed data is shifted repeatedly each line moves only horizontally. The second line display does not shift into the first line position.

The address counter (AC) contents will not change if the only action performed is a display shift.

Function Set

DL: Sets the interface data length. Data is sent or received in 8-bit lengths (DB7 to DB0) when DL is 1, and in 4-bit lengths (DB7 to DB4) when DL is 0.When 4-bit length is selected, data must be sent or received twice.

N: Sets the number of display lines.

F: Sets the character font.

Note: Perform the function at the head of the program before executing any instructions (except for the read busy flag and address instruction). From this point, the function set instruction cannot be executed unless the interface data length is changed.

Set CGRAM Address

Set CGRAM address sets the CGRAM address binary AAAAAA into the address counter.

Data is then written to or read from the MPU for CGRAM.

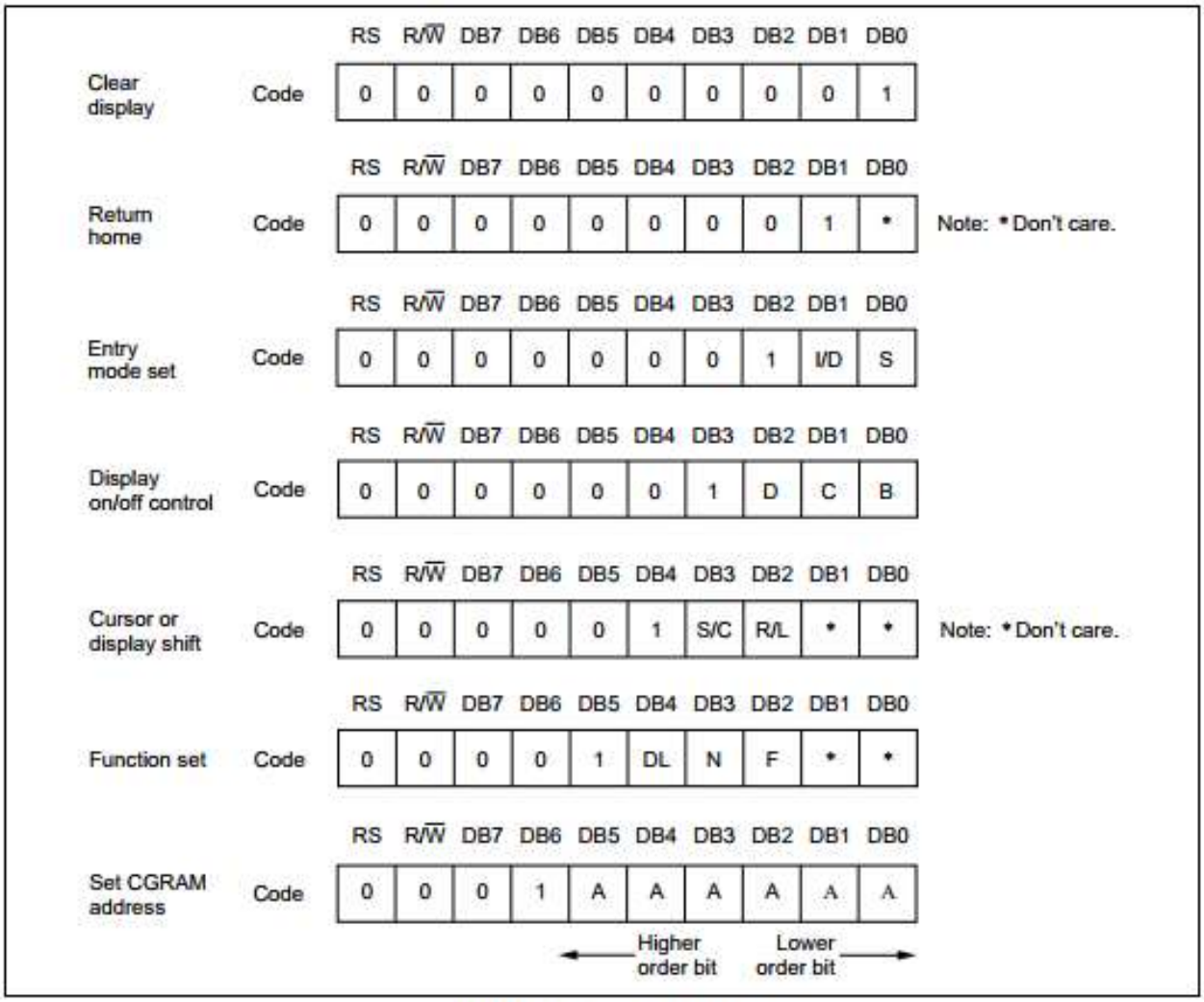


Figure 11 Instruction (1)

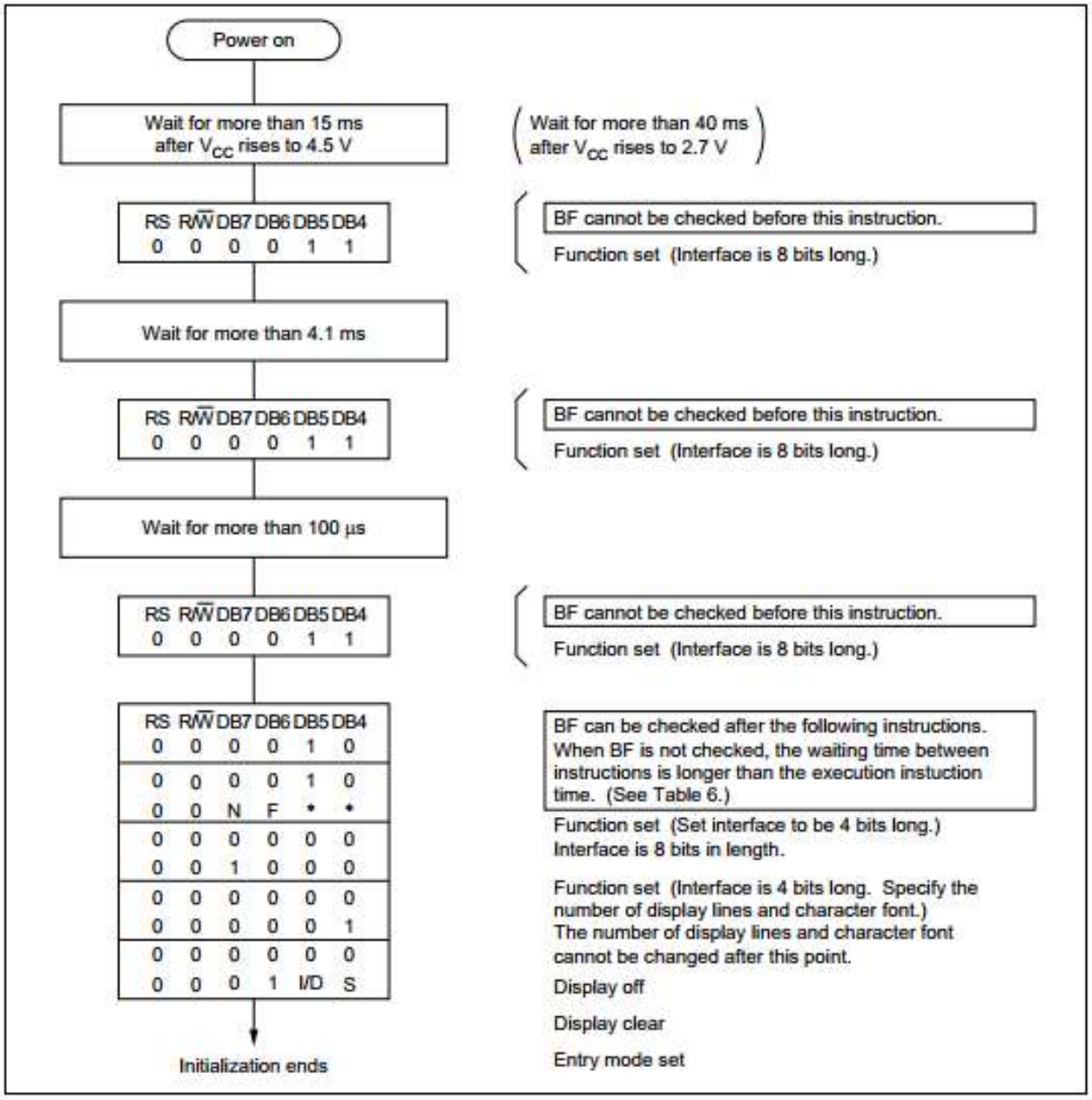


Figure 24 4-Bit Interface

WiRobot™

The Heart of Robot

MGL5128 128x64 Graphic LCD Module

User Manual

DrRobot®

Version: 1.0.0

January 2005

VI. Interface Description

Pin No.	Symbol	Level	Description
1	VDD	5.0V	Supply voltage for logic
2	V_{ss}	0V	Ground
3	V_o	(Variable)	Operating voltage for LCD
4	DB0	H/L	Data bit 0
5	DB1	H/L	Data bit 1
6	DB2	H/L	Data bit 2
7	DB3	H/L	Data bit 3
8	DB4	H/L	Data bit 4
9	DB5	H/L	Data bit 5
10	DB6	H/L	Data bit 6
11	DB7	H/L	Data bit 7
12	CS1	L	Select Column 1~ Column 64
13	CS2	L	Select Column 65~ Column 128
14	RST	L	Reset signal
15	R/W	H/L	H: Read (MPU•Module) , L: Write (MPU•Module)
16	D/I	H/L	H: Data , L : Instruction
17	E	H	Enable signal
18	Vee	—	Negative Voltage output
19	A	—	Power Supply for LED backlight (+)
20	K	—	Power Supply for LED backlight (-)

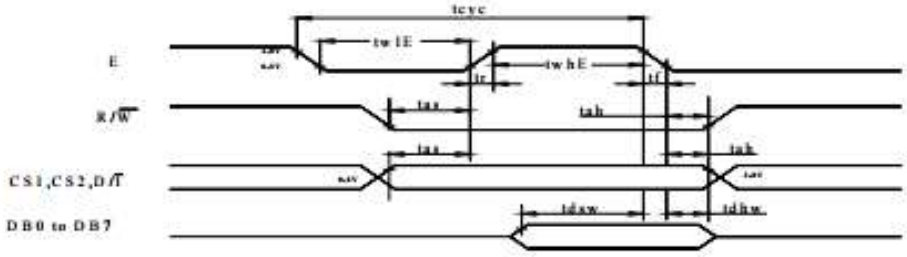
Dr Robot®
25 Valleywood Dr., Unit 20, Markham, ON, L3R 5L9 Canada Tel: (905) 943-9573 Fax: (905) 943-9187

VIII. Timing Characteristics

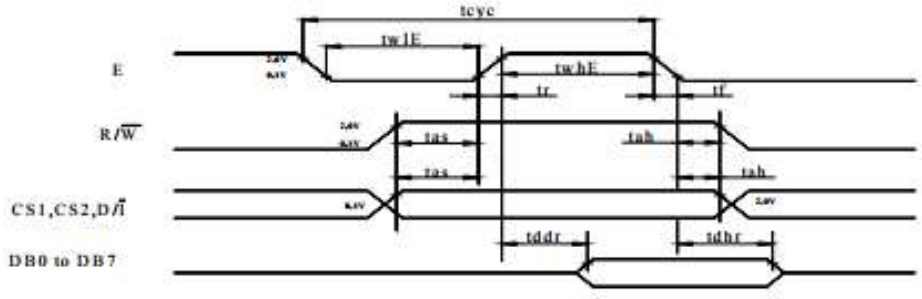
MPU Interface (T=25°, VDD=+5.0V±10.5)

Characteristic	Symbol	Min	Typ	Max	Unit
E cycle	tcyc	1000	—	—	ns
E high level width	twhE	450	—	—	ns
E low level width	twlE	450	—	—	ns
E rise time	tr	—	—	25	ns
E tall time	tf	—	—	25	ns
Address set-up time	tas	140	—	—	ns
Address hold time	tah	10	—	—	ns
Data set-up time	tdsw	200	—	—	ns
Data delay time	tddr	—	—	320	ns
Data hold time (write)	tdhw	10	—	—	ns
Data hold time (read)	tdhr	20	—	—	ns

MPU Read Timing



MPU Write Timing



Dr.Robot® www.DrRobot.com
25 Valleywood Dr., Unit 20, Markham, ON, L3R 5L9 Canada Tel: (905) 943-9572 Fax: (905) 943-9197

IX. Display Control Instruction

The display control instructions control the internal state of the KS0108B. Instruction is received from MPU to KS0108B for the display control. The following table shows various instructions

Instruction	D/I	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Function
Display ON/OFF	0	0	0	0	1	1	1	1	1	0/1	Controls the display on or off. Internal status and display RAM data are not affected. 0:OFF, 1:ON
Set Address	0	0	0	1	Y address (0~63)						Sets the Y address in the Y address counter.
Set Page (X address)	0	0	1	0	1	1	1	Page (0 ~7)			Sets the X address at the X address register.
Display Start Line	0	0	1	1	Display start line(0~63)						Indicates the display data RAM displayed at the top of the screen.
Status Read	0	1	BUSY	0	ON/OFF	RESET	0	0	0	0	Read status. BUSY 0:Ready 1:In operation ON/OFF 0:Display ON 1:Display OFF RESET 0:Normal 1:Reset
Write Display Data	1	0	Display Data								Writes data (DB0:7)into display data RAM. After writing instruction, Y address is increased by 1 automatically.
Read Display Data	1	1	Display Data								Reads data (DB0:7) from display data RAM to the data bus.

Dr Robot®
25 Valleywood Dr., Unit 20, Markham, ON, L3R 5L9 Canada Tel: (905) 943-9572 Fax: (905) 943-9197

X. Detailed Explanation

Display On/Off

R/W	D/I	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	1	1	1	1	1	D

The display data appears when D is and disappears when D is 0. Though the data is not on the screen with D = 0, it remains in the display data RAM. Therefore, you can make it appear by changing D = 0 into D = 1.

Display Start Line

R/W	D/I	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	1	1	A	A	A	A	A	A

Z address AAAAAA (binary) of the display data RAM is set in the display start line register and displayed at the top of the screen. Figure 2. shows examples of display (1/64 duty cycle) when the start line = 0-3. When the display duty cycle is 1/64 or more (ex. 1/32, 1/24 etc.), the data of total line number of LCD screen, from the line specified by display start line instruction, is displayed

Set Page (X Address)

R/W	D/I	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	1	0	1	1	1	A	A	A

X address AAA (binary) of the display data RAM is set in the X address register. After that, writing or reading to or from MPU is executed in this specified page until the next page is set. See Figure 1.

Set Y Address

R/W	D/I	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	1	A	A	A	A	A	A

Y address AAAAAA (binary) of the display data RAM is set in the Y address counter. After that, Y address counter is increased by 1 every time the data is written or read to or from MPU.

Status Read

R/W	D/I	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	Busy	0	On/Off	RESET	0	0	0	0

Dr.Robot www.DrRobot.com
33 Valleywood Dr., Unit 20, Markham, ON, L3R 5L9 Canada Tel: (905) 943-9573 Fax: (905) 943-9197

Busy

When busy is 1, the LSI is executing internal operations. No instruction are accepted while busy is 1, so you should make sure that busy is 0 before writing the next instruction.

ON/OFF

Shows the liquid crystal display condition: on condition or off condition.
When on/off is 1, the display is in off condition.

When on/off is 0, the display is in on condition

RESET

RESET = 1 shows that the system is being initialized. In this condition, no instructions except status read can be accepted.

RESET = 0 shows that initializing has system is in the usual operation condition.

Write Display Data

R/W	D/I	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
1	0	D	D	D	D	D	D	D	D

Writes 8-bit data DDDDDDDD (binary) into the display data RAM. The Y address is increased by 1 automatically.

Read Display Data

R/W	D/I	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
1	1	D	D	D	D	1	D	D	D

Reads out 8-bit data DDDDDDDD (binary) from the display data RAM. Then Y address is increased by 1 automatically.

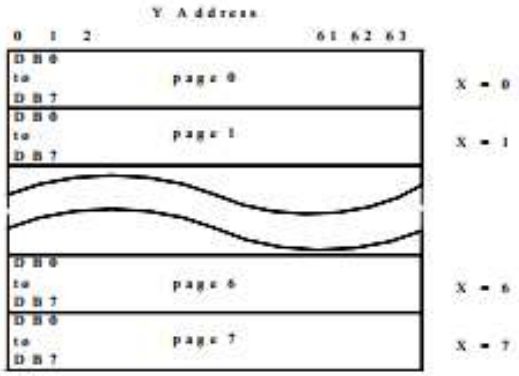
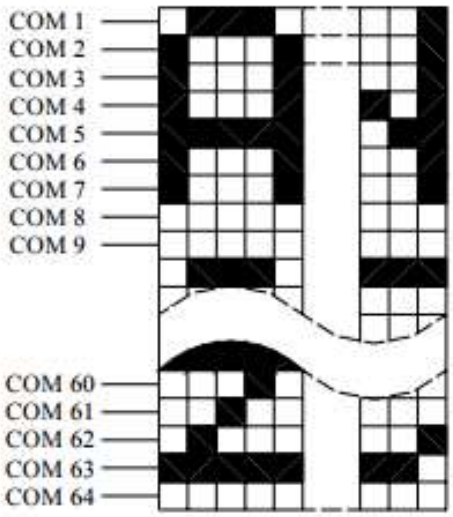


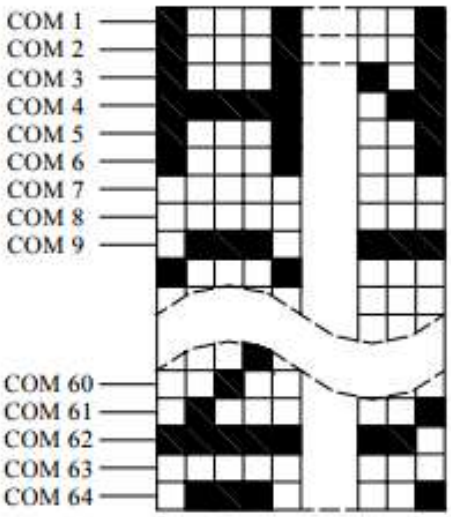
Figure 1.

Dr Robot®
www.DrRobot.com
25 Valleywood Dr., Unit 20, Markham, ON, L3R 5L9 Canada Tel: (905) 943-9572 Fax: (905) 943-9197

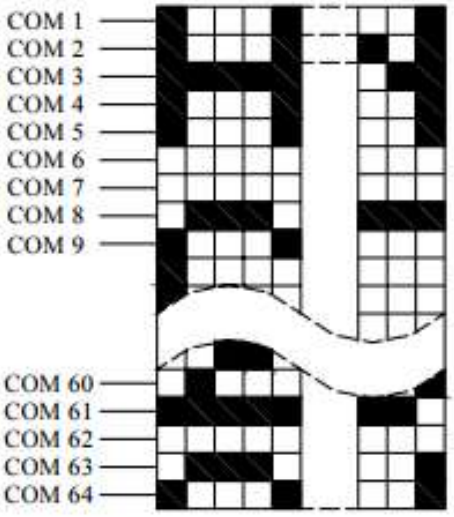
One dummy read is necessary right after the address setting. For details, refer to the explanation of output register in "Function of Each Block".



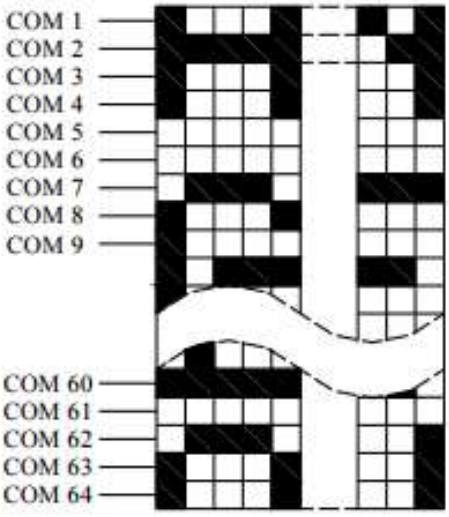
Start line = 0



Start line = 1



Start line = 3



Start line = 4

Dr Robot®
25 Valleywood Dr., Unit 20, Markham, ON, L3R 5L9 Canada Tel: (905) 943-8572 Fax: (905) 943-9197

Features

- High-performance, Low-power Atmel®AVR® 8-bit Microcontroller
- Advanced RISC Architecture
  - 131 Powerful Instructions – Most Single-clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 16 MIPS Throughput at 16MHz
  - On-chip 2-cycle Multiplier
- High Endurance Non-volatile Memory segments
  - 32Kbytes of In-System Self-programmable Flash program memory
  - 1024Bytes 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
- JTAG (IEEE std. 1149.1 Compliant) Interface
  - Boundary-scan Capabilities According to the JTAG Standard
  - Extensive On-chip Debug Support
  - Programming of Flash, EEPROM, Fuses, and Lock Bits through the JTAG Interface
- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescalers and Compare Modes
  - One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Four PWM Channels
  - 8-channel, 10-bit ADC
    8 Single-ended Channels
    7 Differential Channels in TQFP Package Only
    2 Differential Channels with Programmable Gain at 1x, 10x, or 200x
  - Byte-oriented Two-wire Serial Interface
  - Programmable Serial USART
  - Master/Slave SPI Serial Interface
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - On-chip Analog Comparator
- Special Microcontroller Features
  - Power-on Reset and Programmable Brown-out Detection
  - Internal Calibrated RC 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
  - 32 Programmable I/O Lines
  - 40-pin PDIP, 44-lead TQFP, and 44-pad QFN/MLF
- Operating Voltages
  - 2.7V - 5.5V for ATmega32L
  - 4.5V - 5.5V for ATmega32
- Speed Grades
  - 0 - 8MHz for ATmega32L
  - 0 - 16MHz for ATmega32
- Power Consumption at 1MHz, 3V, 25°C
  - Active: 1.1mA
  - Idle Mode: 0.35mA
  - Power-down Mode: < 1µA

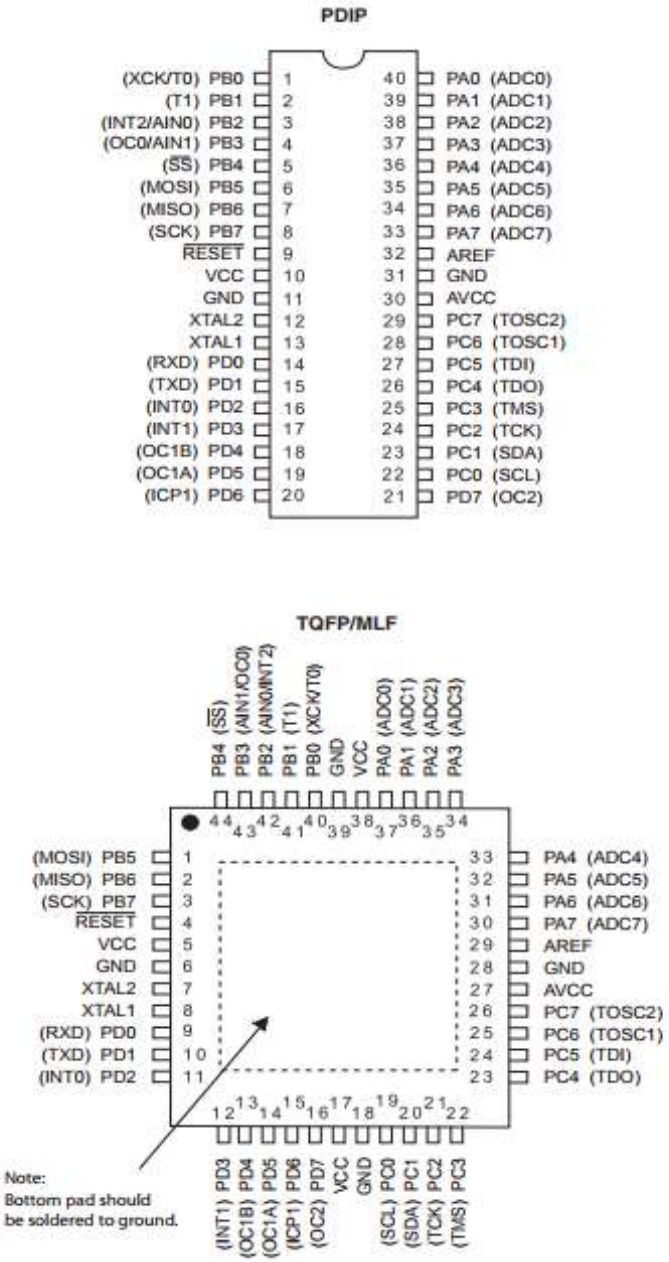
8-bit AVR® Microcontroller with 32KBytes In-System Programmable Flash

ATmega32
ATmega32L

2503Q–AVR–02/11

Pin Configurations

Figure 1. Pinout ATmega32



The Atmel®AVR®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 one 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 ATmega32 provides the following features: 32Kbytes of In-System Programmable Flash Program memory with Read-While-Write capabilities, 1024bytes EEPROM, 2Kbyte SRAM, 32 general purpose I/O lines, 32 general purpose working registers, a JTAG interface for Boundary-scan, On-chip Debugging support and programming, three flexible Timer/Counters with compare modes, Internal and External Interrupts, a serial programmable USART, a byte oriented Two-wire Serial Interface, an 8-channel, 10-bit ADC with optional differential input stage with programmable gain (TQFP package only), 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 USART, Two-wire interface, A/D Converter, 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 External 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.

The device is manufactured using Atmel's high density nonvolatile 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 ATmega32 is a powerful microcontroller that provides a highly-flexible and cost-effective solution to many embedded control applications.

The Atmel AVR ATmega32 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.

Pin Descriptions

VCC Digital supply voltage.

GND Ground.

Port A (PA7..PA0) Port A serves as the analog inputs to the A/D Converter.

Port A also serves as an 8-bit bi-directional I/O port, if the A/D Converter is not used. Port pins can provide internal pull-up resistors (selected for each bit). The Port A output buffers have symmetrical drive characteristics with both high sink and source capability. When pins PA0 to PA7 are used as inputs and are externally pulled low, they will source current if the internal pull-up resistors are activated. The Port A pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port B (PB7..PB0)

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.

Port B also serves the functions of various special features of the ATmega32 as listed on page 57.

Port C (PC7..PC0)

Port C is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port C 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. If the JTAG interface is enabled, the pull-up resistors on pins PC5(TDI), PC3(TMS) and PC2(TCK) will be activated even if a reset occurs.

The TD0 pin is tri-stated unless TAP states that shift out data are entered.

Port C also serves the functions of the JTAG interface and other special features of the ATmega32 as listed on page 60.

Port D (PD7..PD0)

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.

Port D also serves the functions of various special features of the ATmega32 as listed on page 62.

$\overline{\text{RESET}}$

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. The minimum pulse length is given in Table 15 on page 37. Shorter pulses are not guaranteed to generate a reset.

XTAL1

Input to the inverting Oscillator amplifier and input to the internal clock operating circuit.

XTAL2

Output from the inverting Oscillator amplifier.

AVCC

AVCC is the supply voltage pin for Port A and the A/D Converter. 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.

AREF

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

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. Note that 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.

The AVR Status Register – SREG – is defined as:

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

- **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: Bit 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 is useful in BCD arithmetic. See the "Instruction Set Description" for detailed information.

- **Bit 4 – S: Sign Bit, 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 arithmetics. 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.

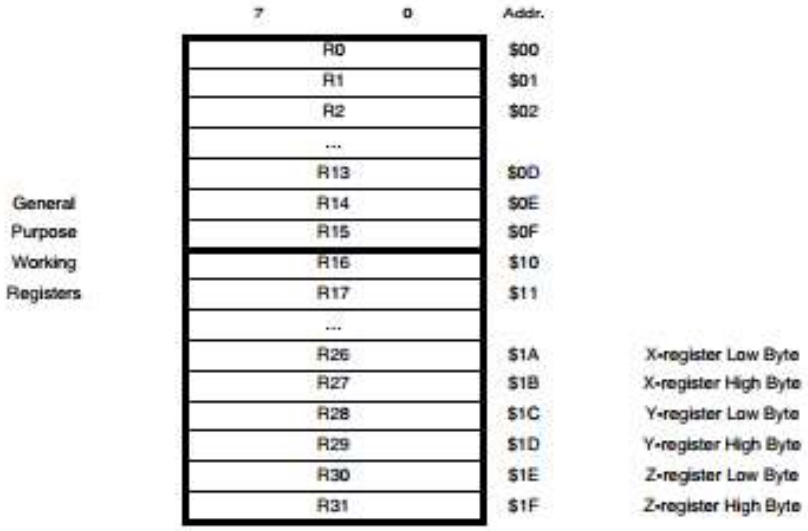
General Purpose Register File

The Register File is optimized for the Atmel®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 4 shows the structure of the 32 general purpose working registers in the CPU.

Figure 4. AVR CPU General Purpose Working Registers



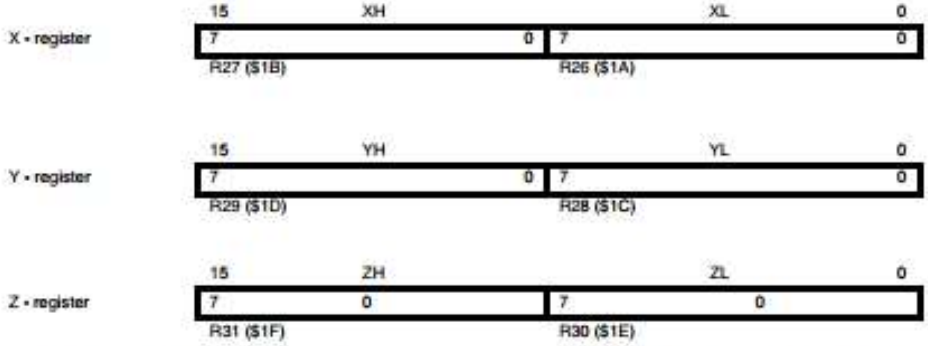
Most of the instructions operating on the Register File have direct access to all registers, and most of them are single cycle instructions.

As shown in Figure 4, 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.

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 Figure 5.

Figure 5. 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).

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 Pointer Register always points to the top of the Stack. Note that the Stack is implemented as growing from higher memory locations to lower memory locations. This implies that a Stack PUSH command decreases the Stack Pointer.

The Stack Pointer points to the data SRAM Stack area where the Subroutine and Interrupt Stacks are located. This Stack space in the data SRAM must be defined by the program before any subroutine calls are executed or interrupts are enabled. The Stack Pointer must be set to point above $60. The Stack Pointer is decremented by one when data is pushed onto the Stack with the PUSH instruction, and it is decremented by two when the return address is pushed onto the Stack with subroutine call or interrupt. The Stack Pointer is incremented by one when data is popped from the Stack with the POP instruction, and it is incremented by two when data is popped from the Stack with return from subroutine RET or return from interrupt RETI.

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.

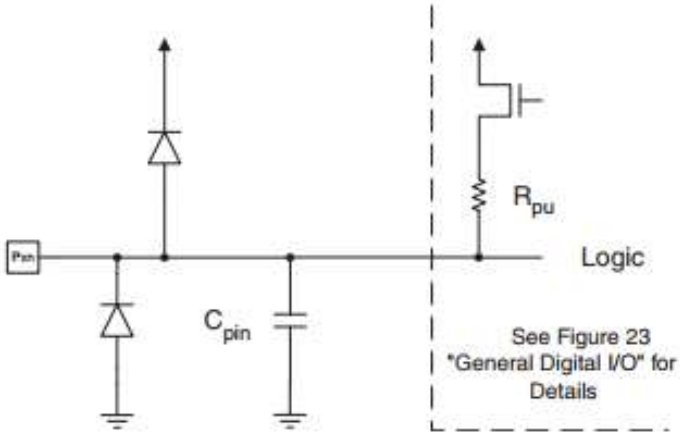
Bit	15	14	13	12	11	10	9	8	
	SP15	SP14	SP13	SP12	SP11	SP10	SP9	SP8	SPH
	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	SPL
	7	6	5	4	3	2	1	0	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

I/O Ports

Introduction

All AVR ports have true Read-Modify-Write functionality when used as general digital I/O ports. This means that the direction of one port pin can be changed without unintentionally changing the direction of any other pin with the SBI and CBI instructions. The same applies when changing drive value (if configured as output) or enabling/disabling of pull-up resistors (if configured as input). Each output buffer has symmetrical drive characteristics with both high sink and source capability. The pin driver is strong enough to drive LED displays directly. All port pins have individually selectable pull-up resistors with a supply-voltage invariant resistance. All I/O pins have protection diodes to both V_{CC} and Ground as indicated in Figure 22. Refer to "Electrical Characteristics" on page 287 for a complete list of parameters.

Figure 22. I/O Pin Equivalent Schematic



All registers and bit references in this section are written in general form. A lower case "x" represents the numbering letter for the port, and a lower case "n" represents the bit number. However, when using the register or bit defines in a program, the precise form must be used. that is, PORTB3 for bit no. 3 in Port B, here documented generally as PORTxn. The physical I/O Registers and bit locations are listed in "Register Description for I/O Ports" on page 64.

Three I/O memory address locations are allocated for each port, one each for the Data Register – PORTx, Data Direction Register – DDRx, and the Port Input Pins – PINx. The Port Input Pins I/O location is read only, while the Data Register and the Data Direction Register are read/write. In addition, the Pull-up Disable – PUD bit in SFIOR disables the pull-up function for all pins in all ports when set.

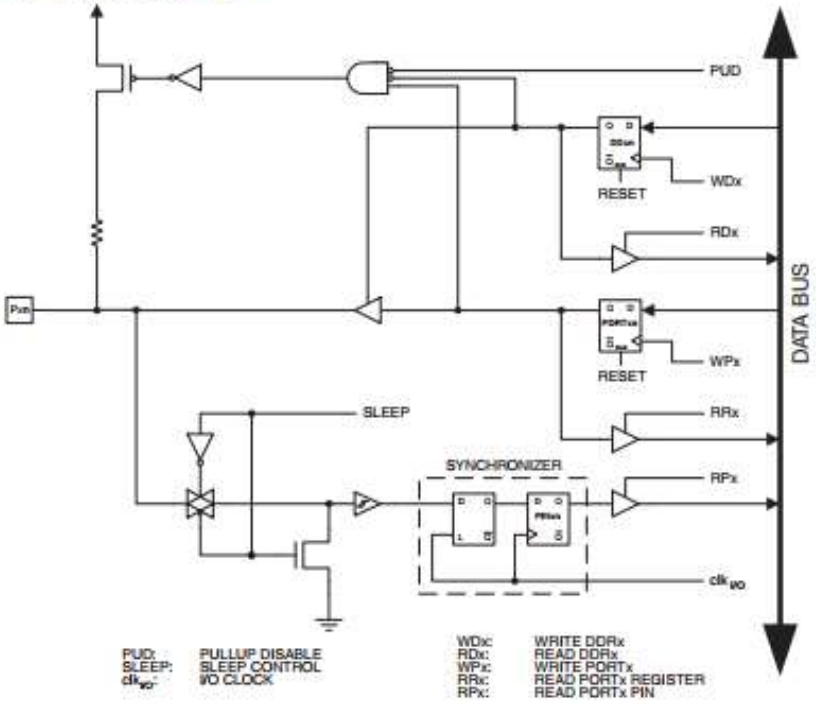
Using the I/O port as General Digital I/O is described in "Ports as General Digital I/O" on page 50. Most port pins are multiplexed with alternate functions for the peripheral features on the device. How each alternate function interferes with the port pin is described in "Alternate Port Functions" on page 54. Refer to the individual module sections for a full description of the alternate functions.

Note that enabling the alternate function of some of the port pins does not affect the use of the other pins in the port as general digital I/O.

Ports as General Digital I/O

The ports are bi-directional I/O ports with optional internal pull-ups. Figure 23 shows a functional description of one I/O-port pin, here generically called Pxn.

Figure 23. General Digital I/O[1]



Note: 1. WPx, WDx, RRx, RPx, and RDx are common to all pins within the same port. $clk_{I/O}$, SLEEP, and PUD are common to all ports.

Configuring the Pin

Each port pin consists of three register bits: DDxn, PORTxn, and PINxn. As shown in "Register Description for I/O Ports" on page 64, the DDxn bits are accessed at the DDRx I/O address, the PORTxn bits at the PORTx I/O address, and the PINxn bits at the PINx I/O address.

The DDxn bit in the DDRx Register selects the direction of this pin. If DDxn is written logic one, Pxn is configured as an output pin. If DDxn is written logic zero, Pxn is configured as an input pin.

If PORTxn is written logic one when the pin is configured as an input pin, the pull-up resistor is activated. To switch the pull-up resistor off, PORTxn has to be written logic zero or the pin has to be configured as an output pin. The port pins are tri-stated when a reset condition becomes active, even if no clocks are running.

If PORTxn is written logic one when the pin is configured as an output pin, the port pin is driven high (one). If PORTxn is written logic zero when the pin is configured as an output pin, the port pin is driven low (zero).

When switching between tri-state ({DDxn, PORTxn} = 0b00) and output high ({DDxn, PORTxn} = 0b11), an intermediate state with either pull-up enabled ({DDxn, PORTxn} = 0b01) or output low ({DDxn, PORTxn} = 0b10) must occur. Normally, the pull-up enabled state is fully acceptable, as a high-impedant environment will not notice the difference between a strong high driver

Table 33. Overriding Signals for Alternate Functions in PD3..PD0

Signal Name	PD3/INT1	PD2/INT0	PD1/TXD	PD0/RXD
PUOE	0	0	TXEN	RXEN
PUOV	0	0	0	PORTD0 • $\overline{PUD}$
DDOE	0	0	TXEN	RXEN
DDOV	0	0	1	0
PVOE	0	0	TXEN	0
PVOV	0	0	TXD	0
DIEOE	INT1 ENABLE	INT0 ENABLE	0	0
DIEOV	1	1	0	0
DI	INT1 INPUT	INT0 INPUT	–	RXD
AIO	–	–	–	–

Register Description for I/O Ports

Port A Data Register – PORTA

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

Port A Data Direction Register – DDRA

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

Port A Input Pins Address – PINA

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

Port B Data Register – PORTB

Bit	7	6	5	4	3	2	1	0	
	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	PORTB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Port B Data Direction Register – DDRB

Bit	7	6	5	4	3	2	1	0	
	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	DDRB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Port B Input Pins Address – PINB

Bit	7	6	5	4	3	2	1	0	
	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	PINB
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

Port C Data Register – PORTC

Bit	7	6	5	4	3	2	1	0	
	PORTC7	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0	PORTC
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Port C Data Direction Register – DDRC

Bit	7	6	5	4	3	2	1	0	
	DDC7	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0	DDRC
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Port C Input Pins Address – PINC

Bit	7	6	5	4	3	2	1	0	
	PINC7	PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0	PINC
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

Port D Data Register – PORTD

Bit	7	6	5	4	3	2	1	0	
	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	PORTD
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Port D Data Direction Register – DDRD

Bit	7	6	5	4	3	2	1	0	
	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	DDRD
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Port D Input Pins Address – PIND

Bit	7	6	5	4	3	2	1	0	
	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	PIND
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

Analog to Digital Converter

Features

- 10-bit Resolution
- 0.5 LSB Integral Non-linearity
- ±2 LSB Absolute Accuracy
- 13 µs - 260 µs Conversion Time
- Up to 15 kSPS at Maximum Resolution
- 8 Multiplexed Single Ended Input Channels
- 7 Differential Input Channels
- 2 Differential Input Channels with Optional Gain of 10x and 200x
- Optional Left adjustment for ADC Result Readout
- 0 - V_{CC} ADC Input Voltage Range
- Selectable 2.56V ADC Reference Voltage
- Free Running or Single Conversion Mode
- ADC Start Conversion by Auto Triggering on Interrupt Sources
- Interrupt on ADC Conversion Complete
- Sleep Mode Noise Canceler

The ATmega32 features a 10-bit successive approximation ADC. The ADC is connected to an 8-channel Analog Multiplexer which allows 8 single-ended voltage inputs constructed from the pins of Port A. The single-ended voltage inputs refer to 0V (GND).

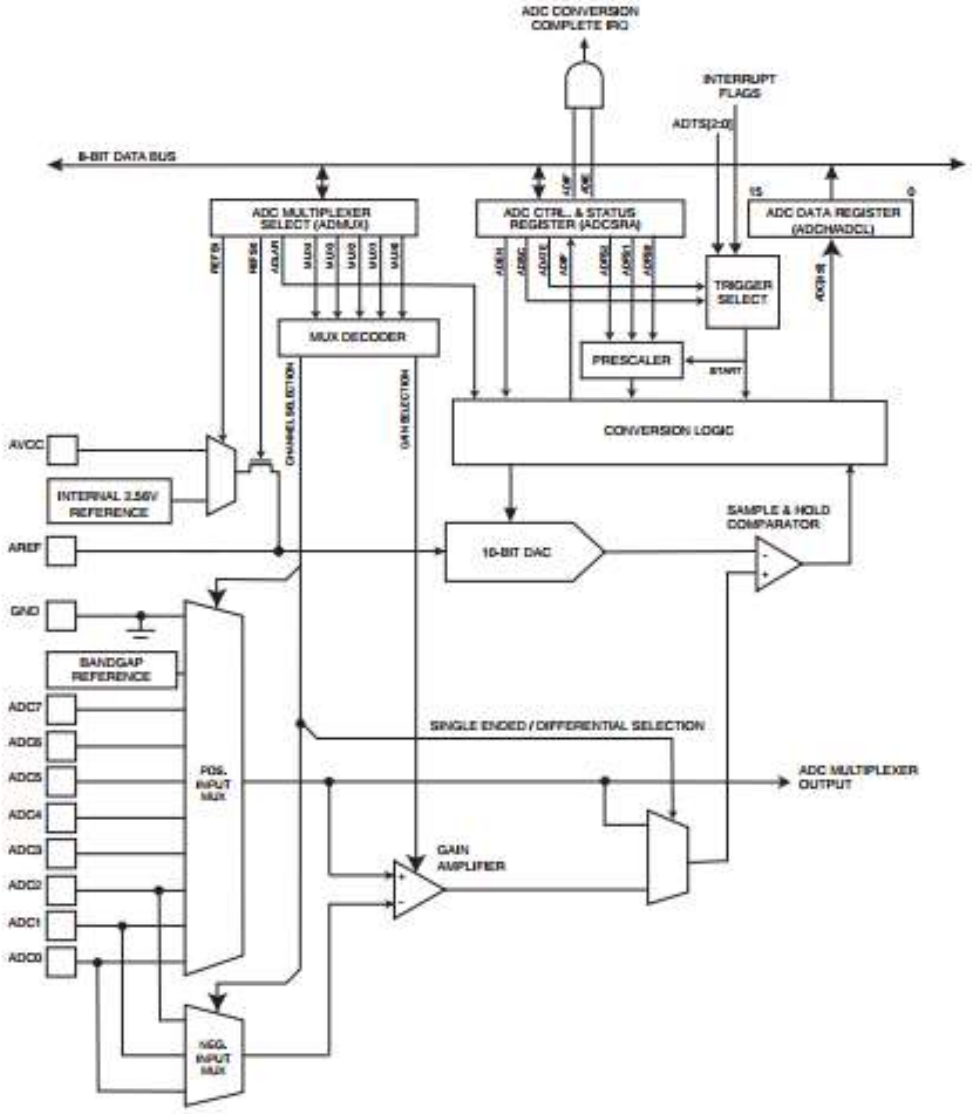
The device also supports 16 differential voltage input combinations. Two of the differential inputs (ADC1, ADC0 and ADC3, ADC2) are equipped with a programmable gain stage, providing amplification steps of 0 dB (1x), 20 dB (10x), or 46 dB (200x) on the differential input voltage before the A/D conversion. Seven differential analog input channels share a common negative terminal (ADC1), while any other ADC input can be selected as the positive input terminal. If 1x or 10x gain is used, 8-bit resolution can be expected. If 200x gain is used, 7-bit resolution can be expected.

The ADC contains a Sample and Hold circuit which ensures that the input voltage to the ADC is held at a constant level during conversion. A block diagram of the ADC is shown in Figure 98.

The ADC has a separate analog supply voltage pin, AVCC. AVCC must not differ more than ±0.3V from V_{CC}. See the paragraph "ADC Noise Canceler" on page 208 on how to connect this pin.

Internal reference voltages of nominally 2.56V or AVCC are provided On-chip. The voltage reference may be externally decoupled at the AREF pin by a capacitor for better noise performance.

Figure 98. Analog to Digital Converter Block Schematic



Operation

The ADC converts an analog input voltage to a 10-bit digital value through successive approximation. The minimum value represents GND and the maximum value represents the voltage on the AREF pin minus 1 LSB. Optionally, AVCC or an internal 2.56V reference voltage may be connected to the AREF pin by writing to the REFSn bits in the ADMUX Register. The internal voltage reference may thus be decoupled by an external capacitor at the AREF pin to improve noise immunity.

The analog input channel and differential gain are selected by writing to the MUX bits in ADMUX. Any of the ADC input pins, as well as GND and a fixed bandgap voltage reference, can be selected as single ended inputs to the ADC. A selection of ADC input pins can be selected as positive and negative inputs to the differential gain amplifier.

If differential channels are selected, the differential gain stage amplifies the voltage difference between the selected input channel pair by the selected gain factor. This amplified value then

becomes the analog input to the ADC. If single ended channels are used, the gain amplifier is bypassed altogether.

The ADC is enabled by setting the ADC Enable bit, ADEN in ADCSRA. Voltage reference and input channel selections will not go into effect until ADEN is set. The ADC does not consume power when ADEN is cleared, so it is recommended to switch off the ADC before entering power saving sleep modes.

The ADC generates a 10-bit result which is presented in the ADC Data Registers, ADCH and ADCL. By default, the result is presented right adjusted, but can optionally be presented left adjusted by setting the ADLAR bit in ADMUX.

If the result is left adjusted and no more than 8-bit precision is required, it is sufficient to read ADCH. Otherwise, ADCL must be read first, then ADCH, to ensure that the content of the Data Registers belongs to the same conversion. Once ADCL is read, ADC access to Data Registers is blocked. This means that if ADCL has been read, and a conversion completes before ADCH is read, neither register is updated and the result from the conversion is lost. When ADCH is read, ADC access to the ADCH and ADCL Registers is re-enabled.

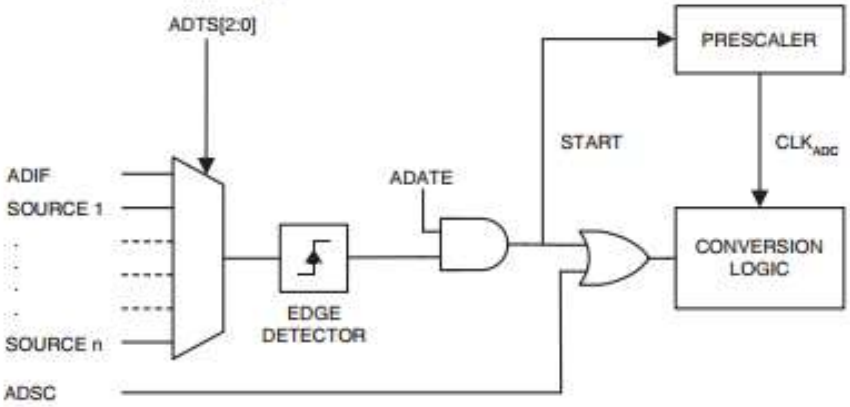
The ADC has its own interrupt which can be triggered when a conversion completes. When ADC access to the Data Registers is prohibited between reading of ADCH and ADCL, the interrupt will trigger even if the result is lost.

Starting a Conversion

A single conversion is started by writing a logical one to the ADC Start Conversion bit, ADSC. This bit stays high as long as the conversion is in progress and will be cleared by hardware when the conversion is completed. If a different data channel is selected while a conversion is in progress, the ADC will finish the current conversion before performing the channel change.

Alternatively, a conversion can be triggered automatically by various sources. Auto Triggering is enabled by setting the ADC Auto Trigger Enable bit, ADATE in ADCSRA. The trigger source is selected by setting the ADC Trigger Select bits, ADTS in SFIOR (see description of the ADTS bits for a list of the trigger sources). When a positive edge occurs on the selected trigger signal, the ADC prescaler is reset and a conversion is started. This provides a method of starting conversions at fixed intervals. If the trigger signal still is set when the conversion completes, a new conversion will not be started. If another positive edge occurs on the trigger signal during conversion, the edge will be ignored. Note that an Interrupt Flag will be set even if the specific interrupt is disabled or the global interrupt enable bit in SREG is cleared. A conversion can thus be triggered without causing an interrupt. However, the Interrupt Flag must be cleared in order to trigger a new conversion at the next interrupt event.

Figure 99. ADC Auto Trigger Logic

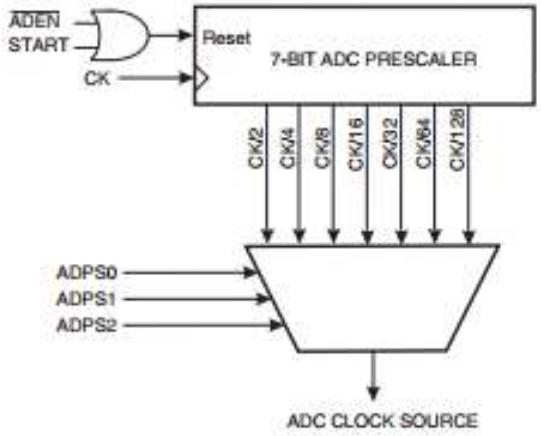


Using the ADC Interrupt Flag as a trigger source makes the ADC start a new conversion as soon as the ongoing conversion has finished. The ADC then operates in Free Running mode, constantly sampling and updating the ADC Data Register. The first conversion must be started by writing a logical one to the ADSC bit in ADCSRA. In this mode the ADC will perform successive conversions independently of whether the ADC Interrupt Flag, ADIF is cleared or not.

If Auto Triggering is enabled, single conversions can be started by writing ADSC in ADCSRA to one. ADSC can also be used to determine if a conversion is in progress. The ADSC bit will be read as one during a conversion, independently of how the conversion was started.

Prescaling and Conversion Timing

Figure 100. ADC Prescaler



By default, the successive approximation circuitry requires an input clock frequency between 50kHz and 200kHz to get maximum resolution. If a lower resolution than 10 bits is needed, the input clock frequency to the ADC can be higher than 200kHz to get a higher sample rate.

The ADC module contains a prescaler, which generates an acceptable ADC clock frequency from any CPU frequency above 100kHz. The prescaling is set by the ADPS bits in ADCSRA. The prescaler starts counting from the moment the ADC is switched on by setting the ADEN bit in ADCSRA. The prescaler keeps running for as long as the ADEN bit is set, and is continuously reset when ADEN is low.

When initiating a single ended conversion by setting the ADSC bit in ADCSRA, the conversion starts at the following rising edge of the ADC clock cycle. See "Differential Gain Channels" on page 206 for details on differential conversion timing.

A normal conversion takes 13 ADC clock cycles. The first conversion after the ADC is switched on (ADEN in ADCSRA is set) takes 25 ADC clock cycles in order to initialize the analog circuitry.

The actual sample-and-hold takes place 1.5 ADC clock cycles after the start of a normal conversion and 13.5 ADC clock cycles after the start of a first conversion. When a conversion is complete, the result is written to the ADC Data Registers, and ADIF is set. In single conversion mode, ADSC is cleared simultaneously. The software may then set ADSC again, and a new conversion will be initiated on the first rising ADC clock edge.

When Auto Triggering is used, the prescaler is reset when the trigger event occurs. This assures a fixed delay from the trigger event to the start of conversion. In this mode, the sample-and-hold takes place 2 ADC clock cycles after the rising edge on the trigger source signal. Three additional CPU clock cycles are used for synchronization logic.

When using Differential mode, along with Auto Trigging from a source other than the ADC Conversion Complete, each conversion will require 25 ADC clocks. This is because the ADC must be disabled and re-enabled after every conversion.

In Free Running mode, a new conversion will be started immediately after the conversion completes, while ADSC remains high. For a summary of conversion times, see Table 81.

Figure 101. ADC Timing Diagram, First Conversion (Single Conversion Mode)

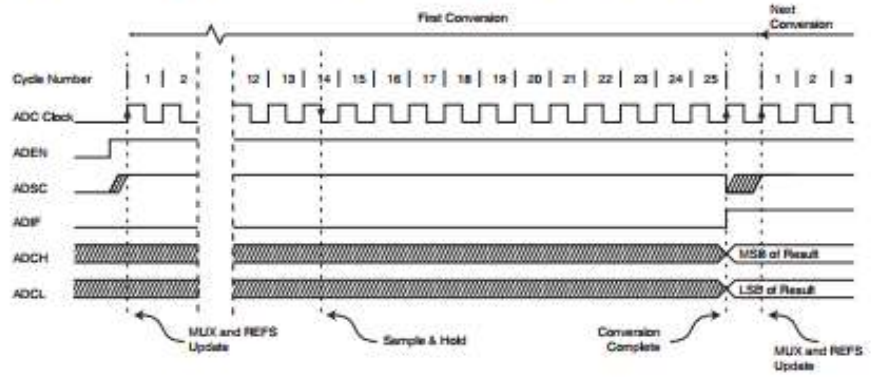


Figure 102. ADC Timing Diagram, Single Conversion

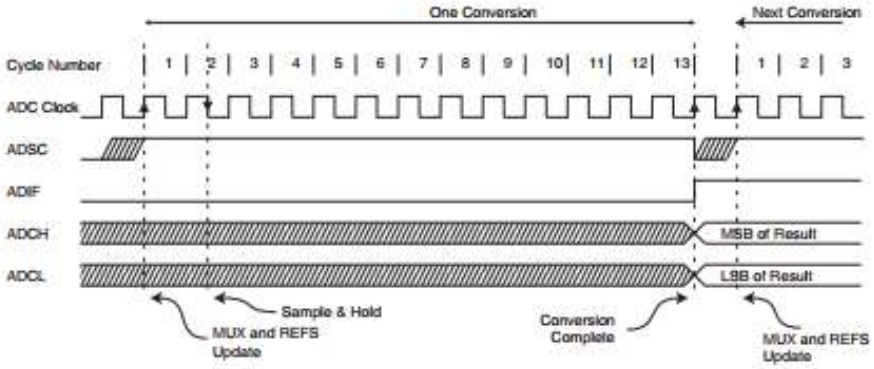


Figure 103. ADC Timing Diagram, Auto Triggered Conversion

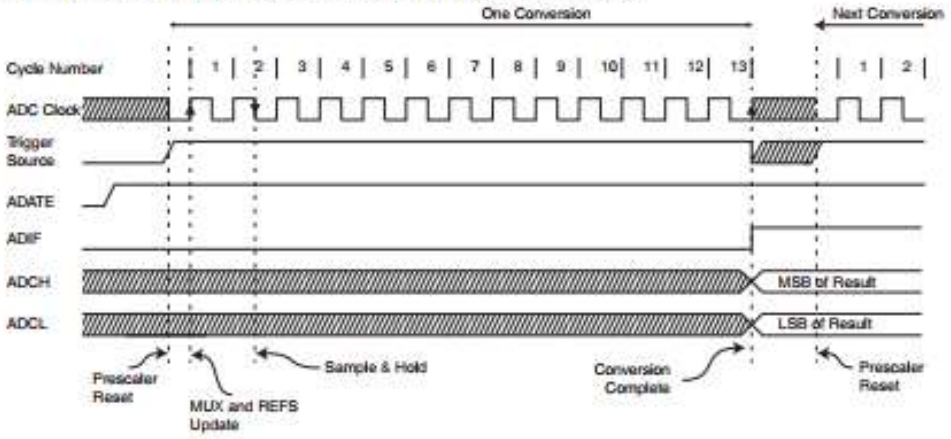


Figure 104. ADC Timing Diagram, Free Running Conversion

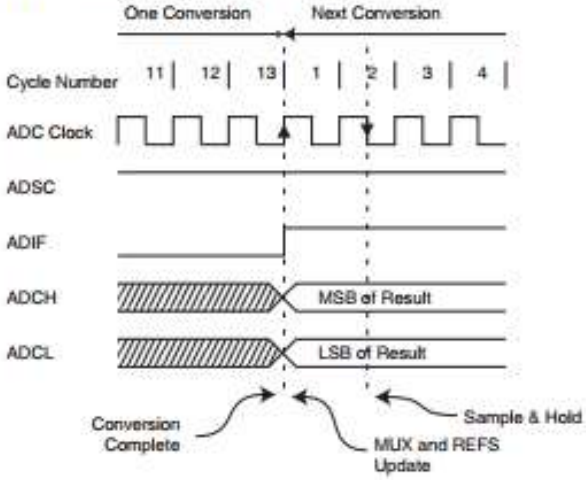


Table 81. ADC Conversion Time

Condition	Sample & Hold (Cycles from Start of Conversion)	Conversion Time (Cycles)
First conversion	13.5	25
Normal conversions, single ended	1.5	13
Auto Triggered conversions	2	13.5
Normal conversions, differential	1.5/2.5	13/14

Differential Gain Channels

When using differential gain channels, certain aspects of the conversion need to be taken into consideration.

Differential conversions are synchronized to the internal clock CK_{ADC2} equal to half the ADC clock. This synchronization is done automatically by the ADC interface in such a way that the sample-and-hold occurs at a specific phase of CK_{ADC2}. A conversion initiated by the user (that is,

all single conversions, and the first free running conversion) when CK_{ADC2} is low will take the same amount of time as a single ended conversion (13 ADC clock cycles from the next prescaled clock cycle). A conversion initiated by the user when CK_{ADC2} is high will take 14 ADC clock cycles due to the synchronization mechanism. In Free Running mode, a new conversion is initiated immediately after the previous conversion completes, and since CK_{ADC2} is high at this time, all automatically started (that is, all but the first) free running conversions will take 14 ADC clock cycles.

The gain stage is optimized for a bandwidth of 4kHz at all gain settings. Higher frequencies may be subjected to non-linear amplification. An external low-pass filter should be used if the input signal contains higher frequency components than the gain stage bandwidth. Note that the ADC clock frequency is independent of the gain stage bandwidth limitation. For example, the ADC clock period may be 6 µs, allowing a channel to be sampled at 12 kSPS, regardless of the bandwidth of this channel.

If differential gain channels are used and conversions are started by Auto Triggering, the ADC must be switched off between conversions. When Auto Triggering is used, the ADC prescaler is reset before the conversion is started. Since the gain stage is dependent of a stable ADC clock prior to the conversion, this conversion will not be valid. By disabling and then re-enabling the ADC between each conversion (writing ADEN in ADCSRA to "0" then to "1"), only extended conversions are performed. The result from the extended conversions will be valid. See "Prescaling and Conversion Timing" on page 204 for timing details.

Changing Channel or Reference Selection

The MUXn and REFS1:0 bits in the ADMUX Register are single buffered through a temporary register to which the CPU has random access. This ensures that the channels and reference selection only takes place at a safe point during the conversion. The channel and reference selection is continuously updated until a conversion is started. Once the conversion starts, the channel and reference selection is locked to ensure a sufficient sampling time for the ADC. Continuous updating resumes in the last ADC clock cycle before the conversion completes (ADIF in ADCSRA is set). Note that the conversion starts on the following rising ADC clock edge after ADSC is written. The user is thus advised not to write new channel or reference selection values to ADMUX until one ADC clock cycle after ADSC is written.

If Auto Triggering is used, the exact time of the triggering event can be indeterministic. Special care must be taken when updating the ADMUX Register, in order to control which conversion will be affected by the new settings.

If both ADATE and ADEN is written to one, an interrupt event can occur at any time. If the ADMUX Register is changed in this period, the user cannot tell if the next conversion is based on the old or the new settings. ADMUX can be safely updated in the following ways:

1. When ADATE or ADEN is cleared.
2. During conversion, minimum one ADC clock cycle after the trigger event.
3. After a conversion, before the Interrupt Flag used as trigger source is cleared.

When updating ADMUX in one of these conditions, the new settings will affect the next ADC conversion.

Special care should be taken when changing differential channels. Once a differential channel has been selected, the gain stage may take as much as 125 µs to stabilize to the new value. Thus conversions should not be started within the first 125 µs after selecting a new differential channel. Alternatively, conversion results obtained within this period should be discarded.

The same settling time should be observed for the first differential conversion after changing ADC reference (by changing the REFS1:0 bits in ADMUX).

ADC Input Channels

When changing channel selections, the user should observe the following guidelines to ensure that the correct channel is selected:

In Single Conversion mode, always select the channel before starting the conversion. The channel selection may be changed one ADC clock cycle after writing one to ADSC. However, the simplest method is to wait for the conversion to complete before changing the channel selection.

In Free Running mode, always select the channel before starting the first conversion. The channel selection may be changed one ADC clock cycle after writing one to ADSC. However, the simplest method is to wait for the first conversion to complete, and then change the channel selection. Since the next conversion has already started automatically, the next result will reflect the previous channel selection. Subsequent conversions will reflect the new channel selection.

When switching to a differential gain channel, the first conversion result may have a poor accuracy due to the required settling time for the automatic offset cancellation circuitry. The user should preferably disregard the first conversion result.

ADC Voltage Reference

The reference voltage for the ADC (V_{REF}) indicates the conversion range for the ADC. Single ended channels that exceed V_{REF} will result in codes close to 0x3FF. V_{REF} can be selected as either AVCC, internal 2.56V reference, or external AREF pin.

AVCC is connected to the ADC through a passive switch. The internal 2.56V reference is generated from the internal bandgap reference (V_{BG}) through an internal amplifier. In either case, the external AREF pin is directly connected to the ADC, and the reference voltage can be made more immune to noise by connecting a capacitor between the AREF pin and ground. V_{REF} can also be measured at the AREF pin with a high impedant voltmeter. Note that V_{REF} is a high impedant source, and only a capacitive load should be connected in a system.

If the user has a fixed voltage source connected to the AREF pin, the user may not use the other reference voltage options in the application, as they will be shorted to the external voltage. If no external voltage is applied to the AREF pin, the user may switch between AVCC and 2.56V as reference selection. The first ADC conversion result after switching reference voltage source may be inaccurate, and the user is advised to discard this result.

If differential channels are used, the selected reference should not be closer to AVCC than indicated in Table 121 on page 293.

ADC Noise Canceler

The ADC features a noise canceler that enables conversion during sleep mode to reduce noise induced from the CPU core and other I/O peripherals. The noise canceler can be used with ADC Noise Reduction and Idle mode. To make use of this feature, the following procedure should be used:

1. Make sure that the ADC is enabled and is not busy converting. Single Conversion Mode must be selected and the ADC conversion complete interrupt must be enabled.
2. Enter ADC Noise Reduction mode (or Idle mode). The ADC will start a conversion once the CPU has been halted.
3. If no other interrupts occur before the ADC conversion completes, the ADC interrupt will wake up the CPU and execute the ADC Conversion Complete interrupt routine. If another interrupt wakes up the CPU before the ADC conversion is complete, that interrupt will be executed, and an ADC Conversion Complete interrupt request will be generated when the ADC conversion completes. The CPU will remain in active mode until a new sleep command is executed.

Note that the ADC will not be automatically turned off when entering other sleep modes than Idle mode and ADC Noise Reduction mode. The user is advised to write zero to ADEN before entering such sleep modes to avoid excessive power consumption. If the ADC is enabled in such

sleep modes and the user wants to perform differential conversions, the user is advised to switch the ADC off and on after waking up from sleep to prompt an extended conversion to get a valid result.

Analog Input Circuitry

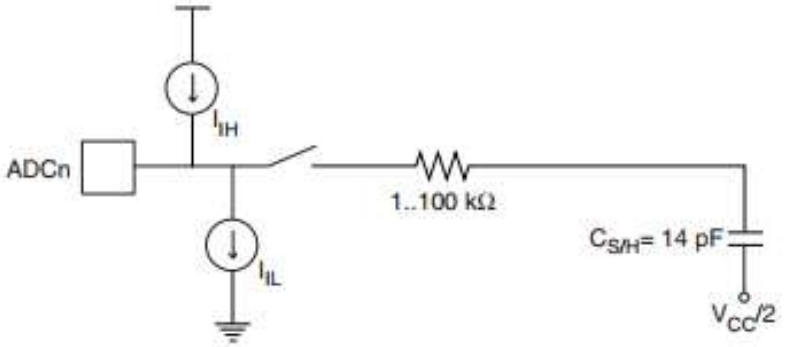
The Analog Input Circuitry for single ended channels is illustrated in Figure 105. An analog source applied to ADCn is subjected to the pin capacitance and input leakage of that pin, regardless of whether that channel is selected as input for the ADC. When the channel is selected, the source must drive the S/H capacitor through the series resistance (combined resistance in the input path).

The ADC is optimized for analog signals with an output impedance of approximately 10 kΩ or less. If such a source is used, the sampling time will be negligible. If a source with higher impedance is used, the sampling time will depend on how long time the source needs to charge the S/H capacitor, with can vary widely. The user is recommended to only use low impedant sources with slowly varying signals, since this minimizes the required charge transfer to the S/H capacitor.

If differential gain channels are used, the input circuitry looks somewhat different, although source impedances of a few hundred kΩ or less is recommended.

Signal components higher than the Nyquist frequency ($f_{ADC}/2$) should not be present for either kind of channels, to avoid distortion from unpredictable signal convolution. The user is advised to remove high frequency components with a low-pass filter before applying the signals as inputs to the ADC.

Figure 105. Analog Input Circuitry



Analog Noise Canceling Techniques

Digital circuitry inside and outside the device generates EMI which might affect the accuracy of analog measurements. If conversion accuracy is critical, the noise level can be reduced by applying the following techniques:

1. Keep analog signal paths as short as possible. Make sure analog tracks run over the analog ground plane, and keep them well away from high-speed switching digital tracks.
2. The AVCC pin on the device should be connected to the digital V_{CC} supply voltage via an LC network as shown in Figure 106.
3. Use the ADC noise canceler function to reduce induced noise from the CPU.
4. If any ADC port pins are used as digital outputs, it is essential that these do not switch while a conversion is in progress.

Table 82. Correlation between Input Voltage and Output Codes

V_{ADCn}	Read code	Corresponding Decimal Value
V_{ADCm} + V_{REF}/GAIN	0x1FF	511
V_{ADCm} + 511/512 V_{REF}/GAIN	0x1FF	511
V_{ADCm} + 510/512 V_{REF}/GAIN	0x1FE	510
...	...	...
V_{ADCm} + 1/512 V_{REF}/GAIN	0x001	1
V_{ADCm}	0x000	0
V_{ADCm} - 1/512 V_{REF}/GAIN	0x3FF	-1
...	...	...
V_{ADCm} - 511/512 V_{REF}/GAIN	0x201	-511
V_{ADCm} - V_{REF}/GAIN	0x200	-512

Example:

ADMUX = 0xED (ADC3 - ADC2, 10x gain, 2.56V reference, left adjusted result)

Voltage on ADC3 is 300 mV, voltage on ADC2 is 500 mV.

ADCR = 512 × 10 × (300 - 500) / 2560 = -400 = 0x270

ADCL will thus read 0x00, and ADCH will read 0x9C. Writing zero to ADLAR right adjusts the result: ADCL = 0x70, ADCH = 0x02.

ADC Multiplexer Selection Register – ADMUX

Bit	7	6	5	4	3	2	1	0	
	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0	ADMUX
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:6 – REFS1:0: Reference Selection Bits**

These bits select the voltage reference for the ADC, as shown in Table 83. If these bits are changed during a conversion, the change will not go in effect until this conversion is complete (ADIF in ADCSRA is set). The internal voltage reference options may not be used if an external reference voltage is being applied to the AREF pin.

Table 83. Voltage Reference Selections for ADC

REFS1	REFS0	Voltage Reference Selection
0	0	AREF, Internal Vref turned off
0	1	AVCC with external capacitor at AREF pin
1	0	Reserved
1	1	Internal 2.56V Voltage Reference with external capacitor at AREF pin

- **Bit 5 – ADLAR: ADC Left Adjust Result**

The ADLAR bit affects the presentation of the ADC conversion result in the ADC Data Register. Write one to ADLAR to left adjust the result. Otherwise, the result is right adjusted. Changing the ADLAR bit will affect the ADC Data Register immediately, regardless of any ongoing conver-

sions. For a complete description of this bit, see "The ADC Data Register – ADCL and ADCH" on page 217.

- **Bits 4:0 – MUX4:0: Analog Channel and Gain Selection Bits**

The value of these bits selects which combination of analog inputs are connected to the ADC. These bits also select the gain for the differential channels. See Table 84 for details. If these bits are changed during a conversion, the change will not go in effect until this conversion is complete (ADIF in ADCSRA is set).

Table 84. Input Channel and Gain Selections

MUX4..0	Single Ended Input	Positive Differential Input	Negative Differential Input	Gain
00000	ADC0	N/A		
00001	ADC1			
00010	ADC2			
00011	ADC3			
00100	ADC4			
00101	ADC5			
00110	ADC6			
00111	ADC7			
01000	N/A	ADC0	ADC0	10x
01001		ADC1	ADC0	
01010		ADC0	ADC0	200x
01011		ADC1	ADC0	
01100		ADC2	ADC2	10x
01101		ADC3	ADC2	
01110		ADC2	ADC2	200x
01111		ADC3	ADC2	
10000		ADC0	ADC1	1x
10001		ADC1	ADC1	
10010		ADC2	ADC1	
10011		ADC3	ADC1	
10100		ADC4	ADC1	
10101		ADC5	ADC1	
10110		ADC6	ADC1	
10111		ADC7	ADC1	
11000		ADC0	ADC2	
11001		ADC1	ADC2	
11010		ADC2	ADC2	
11011		ADC3	ADC2	
11100		ADC4	ADC2	

Table 84. Input Channel and Gain Selections (Continued)

MUX4..0	Single Ended Input	Positive Differential Input	Negative Differential Input	Gain
11101		ADC5	ADC2	1x
11110	1.22V (V_{BG})	N/A		
11111	0V (GND)			

ADC Control and Status Register A – ADCSRA

Bit	7	6	5	4	3	2	1	0	
	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – ADEN: ADC Enable**

Writing this bit to one enables the ADC. By writing it to zero, the ADC is turned off. Turning the ADC off while a conversion is in progress, will terminate this conversion.

- **Bit 6 – ADSC: ADC Start Conversion**

In Single Conversion mode, write this bit to one to start each conversion. In Free Running Mode, write this bit to one to start the first conversion. The first conversion after ADSC has been written after the ADC has been enabled, or if ADSC is written at the same time as the ADC is enabled, will take 25 ADC clock cycles instead of the normal 13. This first conversion performs initialization of the ADC.

ADSC will read as one as long as a conversion is in progress. When the conversion is complete, it returns to zero. Writing zero to this bit has no effect.

- **Bit 5 – ADATE: ADC Auto Trigger Enable**

When this bit is written to one, Auto Triggering of the ADC is enabled. The ADC will start a conversion on a positive edge of the selected trigger signal. The trigger source is selected by setting the ADC Trigger Select bits, ADTS in SFIOR.

- **Bit 4 – ADIF: ADC Interrupt Flag**

This bit is set when an ADC conversion completes and the Data Registers are updated. The ADC Conversion Complete Interrupt is executed if the ADIE bit and the I-bit in SREG are set. ADIF is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, ADIF is cleared by writing a logical one to the flag. Beware that if doing a Read-Modify-Write on ADCSRA, a pending interrupt can be disabled. This also applies if the SBI and CBI instructions are used.

- **Bit 3 – ADIE: ADC Interrupt Enable**

When this bit is written to one and the I-bit in SREG is set, the ADC Conversion Complete Interrupt is activated.

- **Bits 2:0 – ADPS2:0: ADC Prescaler Select Bits**

These bits determine the division factor between the XTAL frequency and the input clock to the ADC.

Table 85. ADC Prescaler Selections

ADPS2	ADPS1	ADPS0	Division Factor
0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

The ADC Data Register – ADCL and ADCH

ADLAR = 0

Bit	15	14	13	12	11	10	9	8	
	–	–	–	–	–	–	ADC9	ADC8	ADCH
	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0	ADCL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

ADLAR = 1

Bit	15	14	13	12	11	10	9	8	
	ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADCH
	ADC1	ADC0	–	–	–	–	–	–	ADCL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

When an ADC conversion is complete, the result is found in these two registers. If differential channels are used, the result is presented in two's complement form.

When ADCL is read, the ADC Data Register is not updated until ADCH is read. Consequently, if the result is left adjusted and no more than 8-bit precision is required, it is sufficient to read ADCH. Otherwise, ADCL must be read first, then ADCH.

The ADLAR bit in ADMUX, and the MUXn bits in ADMUX affect the way the result is read from the registers. If ADLAR is set, the result is left adjusted. If ADLAR is cleared (default), the result is right adjusted.

- **ADC9:0: ADC Conversion Result**

These bits represent the result from the conversion, as detailed in "ADC Conversion Result" on page 213.

Special FunctionIO Register – SFIOR

Bit	7	6	5	4	3	2	1	0	
	ADTS2	ADTS1	ADTS0	–	ACME	PUD	PSR2	PSR10	SFIOR
Read/Write	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:5 – ADTS2:0: ADC Auto Trigger Source**

If ADATE in ADCSRA is written to one, the value of these bits selects which source will trigger an ADC conversion. If ADATE is cleared, the ADTS2:0 settings will have no effect. A conversion will be triggered by the rising edge of the selected Interrupt Flag. Note that switching from a trigger source that is cleared to a trigger source that is set, will generate a positive edge on the trigger signal. If ADEN in ADCSRA is set, this will start a conversion. Switching to Free Running mode (ADTS[2:0]=0) will not cause a trigger event, even if the ADC Interrupt Flag is set.

Table 86. ADC Auto Trigger Source Selections

ADTS2	ADTS1	ADTS0	Trigger Source
0	0	0	Free Running mode
0	0	1	Analog Comparator
0	1	0	External Interrupt Request 0
0	1	1	Timer/Counter0 Compare Match
1	0	0	Timer/Counter0 Overflow
1	0	1	Timer/Counter1 Compare Match B
1	1	0	Timer/Counter1 Overflow
1	1	1	Timer/Counter1 Capture Event

- **Bit 4 – Reserved Bit**

This bit is reserved for future use in the ATmega32. For ensuring compability with future devices, this bit must be written zero when SFIOR is written.

Register Summary

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
$3F ($5F)	SREG	I	T	H	S	V	N	Z	C	10
$3E ($5E)	SPH	–	–	–	–	SP11	SP10	SP9	SP8	12
$3D ($5D)	SPL	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	12
$3C ($5C)	OCR0	Timer/Counter0 Output Compare Register								82
$3B ($5B)	GICR	INT1	INT0	INT2	–	–	–	IVSEL	IVCE	47, 67
$3A ($5A)	GIFR	INTF1	INTF0	INTF2	–	–	–	–	–	68
$39 ($59)	TIMSK	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0	82, 112, 130
$38 ($58)	TIFR	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0	83, 112, 130
$37 ($57)	SPMCR	SPMIE	RWWSB	–	RWWSRE	BLBSET	PGWRT	PGERS	SPMEN	248
$36 ($56)	TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE	177
$35 ($55)	MCUCR	SE	SM2	SM1	SM0	ISC11	ISC10	ISC01	ISC00	32, 66
$34 ($54)	MCUCSR	JTD	ISC2	–	JTRF	WDRF	BORF	EXTRF	PORF	40, 67, 228
$33 ($53)	TCCR0	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00	80
$32 ($52)	TCNT0	Timer/Counter0 (8 Bits)								82
$31[1] ($51)[1]	OSCCAL	Oscillator Calibration Register								30
	OCDR	On-Chip Debug Register								224
$30 ($50)	SFIOR	ADTS2	ADTS1	ADTS0	–	ACME	PUD	PSR2	PSR10	56,85,131,198,218
$2F ($4F)	TCCR1A	COM1A1	COM1A0	COM1B1	COM1B0	FOC1A	FOC1B	WGM11	WGM10	107
$2E ($4E)	TCCR1B	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	110
$2D ($4D)	TCNT1H	Timer/Counter1 – Counter Register High Byte								111
$2C ($4C)	TCNT1L	Timer/Counter1 – Counter Register Low Byte								111
$2B ($4B)	OCR1AH	Timer/Counter1 – Output Compare Register A High Byte								111
$2A ($4A)	OCR1AL	Timer/Counter1 – Output Compare Register A Low Byte								111
$29 ($49)	OCR1BH	Timer/Counter1 – Output Compare Register B High Byte								111
$28 ($48)	OCR1BL	Timer/Counter1 – Output Compare Register B Low Byte								111
$27 ($47)	ICR1H	Timer/Counter1 – Input Capture Register High Byte								111
$26 ($46)	ICR1L	Timer/Counter1 – Input Capture Register Low Byte								111
$25 ($45)	TCCR2	FOC2	WGM20	COM21	COM20	WGM21	CS22	CS21	CS20	125
$24 ($44)	TCNT2	Timer/Counter2 (8 Bits)								127
$23 ($43)	OCR2	Timer/Counter2 Output Compare Register								127
$22 ($42)	ASSR	–	–	–	–	AS2	TCN2UB	OCR2UB	TCR2UB	128
$21 ($41)	WDTCR	–	–	–	WDTOE	WDE	WDP2	WDP1	WDP0	42
$20[2] ($40)[2]	UBRRH	URSEL	–	–	–	UBRR[11:8]				164
	UCSRC	URSEL	UMSEL	UPM1	UPM0	USBS	UCSZ1	UCSZ0	UCPOL	162
$1F ($3F)	EEARH	–	–	–	–	–	–	EEAR9	EEAR8	19
$1E ($3E)	EEARL	EEPROM Address Register Low Byte								19
$1D ($3D)	EEDR	EEPROM Data Register								19
$1C ($3C)	EECR	–	–	–	–	EERIE	EEMWE	EEWE	EERE	19
$1B ($3B)	PORTA	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0	64
$1A ($3A)	DDRA	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0	64
$19 ($39)	PINA	PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0	64
$18 ($38)	PORTB	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	64
$17 ($37)	DDRB	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	64
$16 ($36)	PINB	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	65
$15 ($35)	PORTC	PORTC7	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0	65
$14 ($34)	DDRC	DDC7	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0	65
$13 ($33)	PINC	PINC7	PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0	65
$12 ($32)	PORTD	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	65
$11 ($31)	DDRD	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	65
$10 ($30)	PIND	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	65
$0F ($2F)	SPDR	SPI Data Register								138
$0E ($2E)	SPSR	SPIF	WCOL	–	–	–	–	–	SPI2X	138
$0D ($2D)	SPCR	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	136
$0C ($2C)	UDR	USART I/O Data Register								159
$0B ($2B)	UCSRA	RXC	TXC	UDRE	FE	DOR	PE	U2X	MPCM	160
$0A ($2A)	UCSRB	RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8	161
$09 ($29)	UBRRL	USART Baud Rate Register Low Byte								164
$08 ($28)	ACSR	ACD	ACBG	ACO	ACI	ACIE	ACIC	ACIS1	ACIS0	199
$07 ($27)	ADMUX	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0	214
$06 ($26)	ADCSRA	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	216
$05 ($25)	ADCH	ADC Data Register High Byte								217
$04 ($24)	ADCL	ADC Data Register Low Byte								217
$03 ($23)	TWDR	Two-wire Serial Interface Data Register								179
$02 ($22)	TWAR	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE	179

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
$01 ($21)	TWSR	TWS7	TWS6	TWS5	TWS4	TWS3	–	TWPS1	TWPS0	178
$00 ($20)	TWBR	Two-wire Serial Interface Bit Rate Register								177

Notes:
1. When the OCDEN Fuse is unprogrammed, the OSCCAL Register is always accessed on this address. Refer to the debugger specific documentation for details on how to use the OCDR Register.
2. Refer to the USART description for details on how to access UBRRH and UCSRC.
3. For compatibility with future devices, reserved bits should be written to zero if accessed. Reserved I/O memory addresses should never be written.
4. Some of the Status Flags are cleared by writing a logical one to them. Note that the CBI and SBI instructions will operate on all bits in the I/O Register, writing a one back into any flag read as set, thus clearing the flag. The CBI and SBI instructions work with registers $00 to $1F only.

Instruction Set Summary

Mnemonics	Operands	Description	Operation	Flags	#Clocks
ARITHMETIC AND LOGIC INSTRUCTIONS					
ADD	Rd, Rr	Add two Registers	Rd ← Rd + Rr	Z,C,N,V,H	1
ADC	Rd, Rr	Add with Carry two Registers	Rd ← Rd + Rr + C	Z,C,N,V,H	1
ADIW	Rdl,K	Add Immediate to Word	Rdh:Rdl ← Rdh:Rdl + K	Z,C,N,V,S	2
SUB	Rd, Rr	Subtract two Registers	Rd ← Rd - Rr	Z,C,N,V,H	1
SUBI	Rd, K	Subtract Constant from Register	Rd ← Rd - K	Z,C,N,V,H	1
SBC	Rd, Rr	Subtract with Carry two Registers	Rd ← Rd - Rr - C	Z,C,N,V,H	1
SBCI	Rd, K	Subtract with Carry Constant from Reg.	Rd ← Rd - K - C	Z,C,N,V,H	1
SBIW	Rdl,K	Subtract Immediate from Word	Rdh:Rdl ← Rdh:Rdl - K	Z,C,N,V,S	2
AND	Rd, Rr	Logical AND Registers	Rd ← Rd • Rr	Z,N,V	1
ANDI	Rd, K	Logical AND Register and Constant	Rd ← Rd • K	Z,N,V	1
OR	Rd, Rr	Logical OR Registers	Rd ← Rd v Rr	Z,N,V	1
ORI	Rd, K	Logical OR Register and Constant	Rd ← Rd v K	Z,N,V	1
EOR	Rd, Rr	Exclusive OR Registers	Rd ← Rd ⊕ Rr	Z,N,V	1
COM	Rd	One's Complement	Rd ← $FF – Rd	Z,C,N,V	1
NEG	Rd	Two's Complement	Rd ← $00 – Rd	Z,C,N,V,H	1
SBR	Rd,K	Set Bit(s) in Register	Rd ← Rd v K	Z,N,V	1
CBR	Rd,K	Clear Bit(s) in Register	Rd ← Rd • ($FF - K)	Z,N,V	1
INC	Rd	Increment	Rd ← Rd + 1	Z,N,V	1
DEC	Rd	Decrement	Rd ← Rd – 1	Z,N,V	1
TST	Rd	Test for Zero or Minus	Rd ← Rd • Rd	Z,N,V	1
CLR	Rd	Clear Register	Rd ← Rd ⊕ Rd	Z,N,V	1
SER	Rd	Set Register	Rd ← $FF	None	1
MUL	Rd, Rr	Multiply Unsigned	R1:R0 ← Rd x Rr	Z,C	2
MULS	Rd, Rr	Multiply Signed	R1:R0 ← Rd x Rr	Z,C	2
MULSU	Rd, Rr	Multiply Signed with Unsigned	R1:R0 ← Rd x Rr	Z,C	2
FMUL	Rd, Rr	Fractional Multiply Unsigned	R1:R0 ← (Rd x Rr) << 1	Z,C	2
FMULS	Rd, Rr	Fractional Multiply Signed	R1:R0 ← (Rd x Rr) << 1	Z,C	2
FMULSU	Rd, Rr	Fractional Multiply Signed with Unsigned	R1:R0 ← (Rd x Rr) << 1	Z,C	2
BRANCH INSTRUCTIONS					
RJMP	k	Relative Jump	PC ← PC + k + 1	None	2
IJMP		Indirect Jump to (Z)	PC ← Z	None	2
JMP	k	Direct Jump	PC ← k	None	3
RCALL	k	Relative Subroutine Call	PC ← PC + k + 1	None	3
ICALL		Indirect Call to (Z)	PC ← Z	None	3
CALL	k	Direct Subroutine Call	PC ← k	None	4
RET		Subroutine Return	PC ← Stack	None	4
RETI		Interrupt Return	PC ← Stack	I	4
CPSE	Rd,Rr	Compare, Skip if Equal	if (Rd = Rr) PC ← PC + 2 or 3	None	1 / 2 / 3
CP	Rd,Rr	Compare	Rd – Rr	Z, N,V,C,H	1
CPC	Rd,Rr	Compare with Carry	Rd – Rr – C	Z, N,V,C,H	1
CPI	Rd,K	Compare Register with Immediate	Rd – K	Z, N,V,C,H	1
SBRC	Rr, b	Skip if Bit in Register Cleared	if (Rr(b)=0) PC ← PC + 2 or 3	None	1 / 2 / 3
SBRS	Rr, b	Skip if Bit in Register is Set	if (Rr(b)=1) PC ← PC + 2 or 3	None	1 / 2 / 3
SBIC	P, b	Skip if Bit in I/O Register Cleared	if (P(b)=0) PC ← PC + 2 or 3	None	1 / 2 / 3
SBIS	P, b	Skip if Bit in I/O Register is Set	if (P(b)=1) PC ← PC + 2 or 3	None	1 / 2 / 3
BRBS	s, k	Branch if Status Flag Set	if (SREG(s) = 1) then PC←PC+k + 1	None	1 / 2
BRBC	s, k	Branch if Status Flag Cleared	if (SREG(s) = 0) then PC←PC+k + 1	None	1 / 2
BREQ	k	Branch if Equal	if (Z = 1) then PC ← PC + k + 1	None	1 / 2
BRNE	k	Branch if Not Equal	if (Z = 0) then PC ← PC + k + 1	None	1 / 2
BRCS	k	Branch if Carry Set	if (C = 1) then PC ← PC + k + 1	None	1 / 2
BRCC	k	Branch if Carry Cleared	if (C = 0) then PC ← PC + k + 1	None	1 / 2
BRSH	k	Branch if Same or Higher	if (C = 0) then PC ← PC + k + 1	None	1 / 2
BRLO	k	Branch if Lower	if (C = 1) then PC ← PC + k + 1	None	1 / 2
BRMI	k	Branch if Minus	if (N = 1) then PC ← PC + k + 1	None	1 / 2
BRPL	k	Branch if Plus	if (N = 0) then PC ← PC + k + 1	None	1 / 2
BRGE	k	Branch if Greater or Equal, Signed	if (N ⊕ V= 0) then PC ← PC + k + 1	None	1 / 2
BRLT	k	Branch if Less Than Zero, Signed	if (N ⊕ V= 1) then PC ← PC + k + 1	None	1 / 2
BRHS	k	Branch if Half Carry Flag Set	if (H = 1) then PC ← PC + k + 1	None	1 / 2
BRHC	k	Branch if Half Carry Flag Cleared	if (H = 0) then PC ← PC + k + 1	None	1 / 2
BRTS	k	Branch if T Flag Set	if (T = 1) then PC ← PC + k + 1	None	1 / 2
BRTC	k	Branch if T Flag Cleared	if (T = 0) then PC ← PC + k + 1	None	1 / 2
BRVS	k	Branch if Overflow Flag is Set	if (V = 1) then PC ← PC + k + 1	None	1 / 2
BRVC	k	Branch if Overflow Flag is Cleared	if (V = 0) then PC ← PC + k + 1	None	1 / 2

Mnemonics	Operands	Description	Operation	Flags	#Clocks
BRIE	k	Branch if Interrupt Enabled	if (I = 1) then PC ← PC + k + 1	None	1 / 2
BRID	k	Branch if Interrupt Disabled	if (I = 0) then PC ← PC + k + 1	None	1 / 2
DATA TRANSFER INSTRUCTIONS					
MOV	Rd, Rr	Move Between Registers	Rd ← Rr	None	1
MOVW	Rd, Rr	Copy Register Word	Rd+1:Rd ← Rr+1:Rr	None	1
LDI	Rd, K	Load Immediate	Rd ← K	None	1
LD	Rd, X	Load Indirect	Rd ← (X)	None	2
LD	Rd, X+	Load Indirect and Post-Inc.	Rd ← (X), X ← X + 1	None	2
LD	Rd, - X	Load Indirect and Pre-Dec.	X ← X - 1, Rd ← (X)	None	2
LD	Rd, Y	Load Indirect	Rd ← (Y)	None	2
LD	Rd, Y+	Load Indirect and Post-Inc.	Rd ← (Y), Y ← Y + 1	None	2
LD	Rd, - Y	Load Indirect and Pre-Dec.	Y ← Y - 1, Rd ← (Y)	None	2
LDD	Rd,Y+q	Load Indirect with Displacement	Rd ← (Y + q)	None	2
LD	Rd, Z	Load Indirect	Rd ← (Z)	None	2
LD	Rd, Z+	Load Indirect and Post-Inc.	Rd ← (Z), Z ← Z+1	None	2
LD	Rd, -Z	Load Indirect and Pre-Dec.	Z ← Z - 1, Rd ← (Z)	None	2
LDD	Rd, Z+q	Load Indirect with Displacement	Rd ← (Z + q)	None	2
LDS	Rd, k	Load Direct from SRAM	Rd ← (k)	None	2
ST	X, Rr	Store Indirect	(X) ← Rr	None	2
ST	X+, Rr	Store Indirect and Post-Inc.	(X) ← Rr, X ← X + 1	None	2
ST	- X, Rr	Store Indirect and Pre-Dec.	X ← X - 1, (X) ← Rr	None	2
ST	Y, Rr	Store Indirect	(Y) ← Rr	None	2
ST	Y+, Rr	Store Indirect and Post-Inc.	(Y) ← Rr, Y ← Y + 1	None	2
ST	- Y, Rr	Store Indirect and Pre-Dec.	Y ← Y - 1, (Y) ← Rr	None	2
STD	Y+q,Rr	Store Indirect with Displacement	(Y + q) ← Rr	None	2
ST	Z, Rr	Store Indirect	(Z) ← Rr	None	2
ST	Z+, Rr	Store Indirect and Post-Inc.	(Z) ← Rr, Z ← Z + 1	None	2
ST	-Z, Rr	Store Indirect and Pre-Dec.	Z ← Z - 1, (Z) ← Rr	None	2
STD	Z+q,Rr	Store Indirect with Displacement	(Z + q) ← Rr	None	2
STS	k, Rr	Store Direct to SRAM	(k) ← Rr	None	2
LPM		Load Program Memory	R0 ← (Z)	None	3
LPM	Rd, Z	Load Program Memory	Rd ← (Z)	None	3
LPM	Rd, Z+	Load Program Memory and Post-Inc	Rd ← (Z), Z ← Z+1	None	3
SPM		Store Program Memory	(Z) ← R1:R0	None	-
IN	Rd, P	In Port	Rd ← P	None	1
OUT	P, Rr	Out Port	P ← Rr	None	1
PUSH	Rr	Push Register on Stack	Stack ← Rr	None	2
POP	Rd	Pop Register from Stack	Rd ← Stack	None	2
BIT AND BIT-TEST INSTRUCTIONS					
SBI	P,b	Set Bit in I/O Register	I/O(P,b) ← 1	None	2
CBI	P,b	Clear Bit in I/O Register	I/O(P,b) ← 0	None	2
LSL	Rd	Logical Shift Left	Rd(n+1) ← Rd(n), Rd(0) ← 0	Z,C,N,V	1
LSR	Rd	Logical Shift Right	Rd(n) ← Rd(n+1), Rd(7) ← 0	Z,C,N,V	1
ROL	Rd	Rotate Left Through Carry	Rd(0)←C,Rd(n+1)← Rd(n),C←Rd(7)	Z,C,N,V	1
ROR	Rd	Rotate Right Through Carry	Rd(7)←C,Rd(n)← Rd(n+1),C←Rd(0)	Z,C,N,V	1
ASR	Rd	Arithmetic Shift Right	Rd(n) ← Rd(n+1), n=0..6	Z,C,N,V	1
SWAP	Rd	Swap Nibbles	Rd(3..0)←Rd(7..4),Rd(7..4)←Rd(3..0)	None	1
BSET	s	Flag Set	SREG(s) ← 1	SREG(s)	1
BCLR	s	Flag Clear	SREG(s) ← 0	SREG(s)	1
BST	Rr, b	Bit Store from Register to T	T ← Rr(b)	T	1
BLD	Rd, b	Bit load from T to Register	Rd(b) ← T	None	1
SEC		Set Carry	C ← 1	C	1
CLC		Clear Carry	C ← 0	C	1
SEN		Set Negative Flag	N ← 1	N	1
CLN		Clear Negative Flag	N ← 0	N	1
SEZ		Set Zero Flag	Z ← 1	Z	1
CLZ		Clear Zero Flag	Z ← 0	Z	1
SEI		Global Interrupt Enable	I ← 1	I	1
CLI		Global Interrupt Disable	I ← 0	I	1
SES		Set Signed Test Flag	S ← 1	S	1
CLS		Clear Signed Test Flag	S ← 0	S	1
SEV		Set Twos Complement Overflow.	V ← 1	V	1
CLV		Clear Twos Complement Overflow	V ← 0	V	1
SET		Set T in SREG	T ← 1	T	1
CLT		Clear T in SREG	T ← 0	T	1
SEH		Set Half Carry Flag in SREG	H ← 1	H	1

Mnemonics	Operands	Description	Operation	Flags	#Clocks
CLH		Clear Half Carry Flag in SREG	H ← 0	H	1
MCU CONTROL INSTRUCTIONS					
NOP		No Operation		None	1
SLEEP		Sleep	(see specific descr. for Sleep function)	None	1
WDR		Watchdog Reset	(see specific descr. for WDR/timer)	None	1
BREAK		Break	For On-Chip Debug Only	None	N/A