Estación de monitorización y control de baterías auxiliares en pequeñas instalaciones fotovoltaicas destinadas a vehículos camperizados

Sergio González De Paz

Tutor: Evelio José González González

Departmento de Ingeniería Informática y de Sistemas
Universidad de La Laguna

Memoria para la obtención del grado de
Ingeniería Electrónica Industrial y Automática

septiembre 2022

Agradecimientos

Quisiera agradecer a mis padres y mi hermano la paciencia y el apoyo durante todos estos años.

A mis amigos que me han aguantado y guiado desde que tengo memoria, a todos con los que he compartido este tiempo en la universidad y me han ayudado incondicionalmente a superar cada bache hasta llegar a aquí.

Y por supuesto, a todos aquellos profesores que han logrado captar mi atención y motivarme a seguir desde que comencé, curso tras curso, a terminar este proyecto.

Resumen

The aim of this project is to design and prototype a system aimed to help in the implementation of a photovoltaic, off-grid electric system on a camper van. This System will collect voltage and current data that will be stored in a microcontroller, and project it into a small monochromatic screen. That data will also be used to decide which source should be chosen to charge our system, either the solar panels or the car's own alternator.

The data obtained will be taken from various sources, either voltage dividers or Hall effect current sensors that will need treatment to be shown in the correct manner. This readings have values that we'll analyze in order to stablish limits that will not damage our microcontroller.

Índice general

Índice de figuras

Nomenclatura

Acrónimos / Abreviaturas

ADC *Analog to Digital Converter*, conversor analógico digital

ARM *Advanced RISC Machines*, máquina RISC avanzada

AVR *Advanced Virtual RISC*, RISC avanzado virtual

DVD *Digital Versatile Disc*, disco versátil digital

$EEPROM$ *Electrically Erasable Programmable Read-Only Memory*, memoria de sólo lectura programable y borrable eléctricamente

$GIMP$ *GNU Image Manipulation Program*, programa de manipulación de imágenes GNU

$GLIB$ *General Language for Instrument Behavior*, lenguaje general para el comportamiento de los instrumentos

IDE *Integrated Development Enviroment*, entorno de desarrollo integrado

$MPPT$ *Maximum Power Point Tracking*, seguimiento del punto de máxima potencia

PCB *Printed Circuit Board*, placa de Circuito Impreso

PIC *peripheral interface controller*, controlador de interfaz periférica

PWM *Pulse Width Modulation*, modulación por ancho de pulso

$RISC$ *Reduced Instruction Set Computer*, ordenador con set de instrucciones reducidos

RTC *Real Time Counter*, contador de tiempo real

$SRAM$ *Static Random Access Memory*, memoria de acceso aleatorio estática

USB *Universal Shared Bus*, bus universal en serie

VHS *Video Home System*, sistema de vídeo doméstico

Capítulo 1

Introducción

1.1. La electrónica de consumo y su evolución

En los 70, los aparatos electrónicos se encontraban ya, en mayor o menor medida presentes en el mundo. Aquellos dispositivos capaces de grabar en VHS lo que la televisión mostraba en aquella época cumplían, salvando las diferencias en calidad y funcionalidad, con el mismo objetivo que cualquiera de los últimos modelos fabricados en 2010. Sin embargo, basta con abrir ambos dispositivos para darnos cuenta de la principal diferencia; aquellas placas de circuitos, llenas hasta el último centímetro de componentes electrónicos, dan paso a placas que, incluso introduciendo funcionalidades adicionales tan complejas como las de reproducir un DVD, se encuentran mayoritariamente desiertas. El único impedimento por el que mantienen el tamaño es que la tecnología del VHS y sus dimensiones no ha variado desde la puesta a la venta del primer video-grabador en 1976 [1].

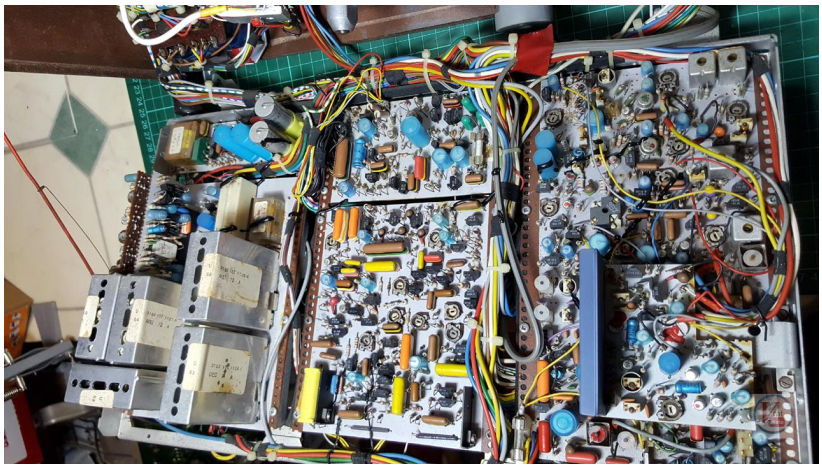


Figura 1.1 Electrónica de un grabador de vídeo Phillips N1500 de los 70

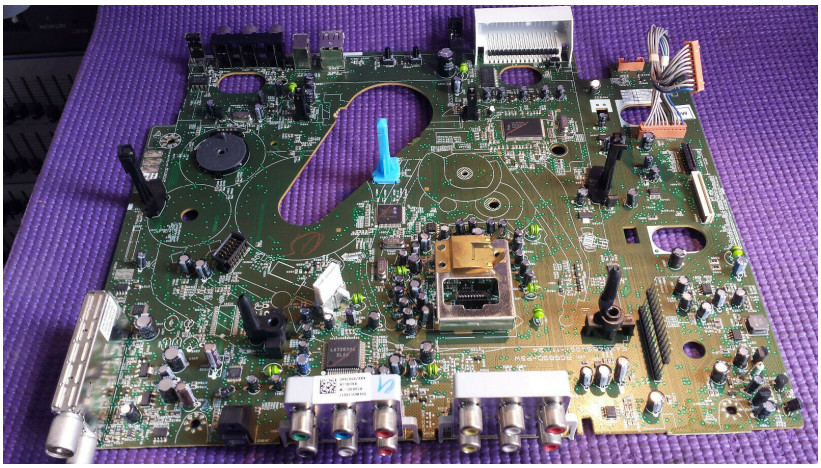


Figura 1.2 Placa base de un combo VHS/DVD LG RC689D de 2011

La mejora de la tecnología no sólo trae consigo la desaparición de la mayoría de elementos analógicos; al analizar una placa base, llama la atención que donde antes existían cientos de circuitos integrados, ahora la mayoría a desaparecido, dejando en su lugar un microchip con muchas más conexiones, y proporcionalmente más pequeño. La principal responsable de esto es la miniaturización de los transistores. En 1971, Intel fabricaba y vendía el Intel 4004, con transistores fabricados a 10 µm [2] por el equivalente actual de 430 USD. a día de hoy, AMD produce y vende procesadores fabricados a 5 nm a partir de 299 USD. [3]

La mejora de las capacidades de miniaturización de los transistores trae consigo un cambio en la forma de construir circuitos. A mediados de 1980, algunos fabricantes comienzan a implementar microprocesadores en sus diseños, reduciendo así la complejidad de las placas, el conteo de microchips, y, por tanto, el coste general del producto, tanto para el fabricante, como para el usuario final. Estos circuitos adoptaban microprocesadores como el Intel 8080, un microprocesador de 8-Bit desarrollado a finales de 1970.

Pronto desarrolladores e ingenieros se dieron cuenta de que no sólo se mejoraban y simplificaban las funcionalidades, el diseño y los procesos de fabricación. La integración facilitaba la reparación a posteriori, lo que antes requería un equipo técnico especializado y muchas horas de trabajo solucionando problemas detectados durante una fase avanzada de la fabricación o venta, ahora, en la mayoría de los casos un equipo técnico mucho menos especializado podía, simplemente, sustituir el firmware almacenado en una memoria ROM.

Este proceso continuó, la mejora en los procesos de fabricación de silicio en los 90 llevaron a los fabricantes a incluir en los propios microprocesadores más y más circuitería. Uno de los caminos tomados fue el de mejorar el microchip con funcionalidades

y características para los que antes se requerían otros microchips a su alrededor. Este cambio es responsable de crear la variante que estudiamos, el microcontrolador.

A día de hoy, los microcontroladores forman parte de todo, desde un sistema para monitorizar la radiación de fondo [4] hasta el mando que controla la televisión, incluirá un microcontrolador que llevará a cabo varias tareas a la vez con un índice de fallos cercano a cero.

1.2. Furgonetas camper y su autonomía

Desde que en 1915 la familia Conklin partió de Nueva York en su vehículo apodado «Gypsy Van» hasta San Francisco, con lo que se considera la primera autocaravana que existe [5], la idea de poder llevar tu propia casa sobre cuatro ruedas no ha hecho más que ganar atractivo. Esta tendencia resulta en un notable incremento de popularidad durante la pandemia, momento en el que la población, respondiendo a las medidas del distanciamiento social, una bajada generalizada de los ahorros y el cierre de un gran porcentaje de hoteles, se vuelca en la idea de la «camperización»

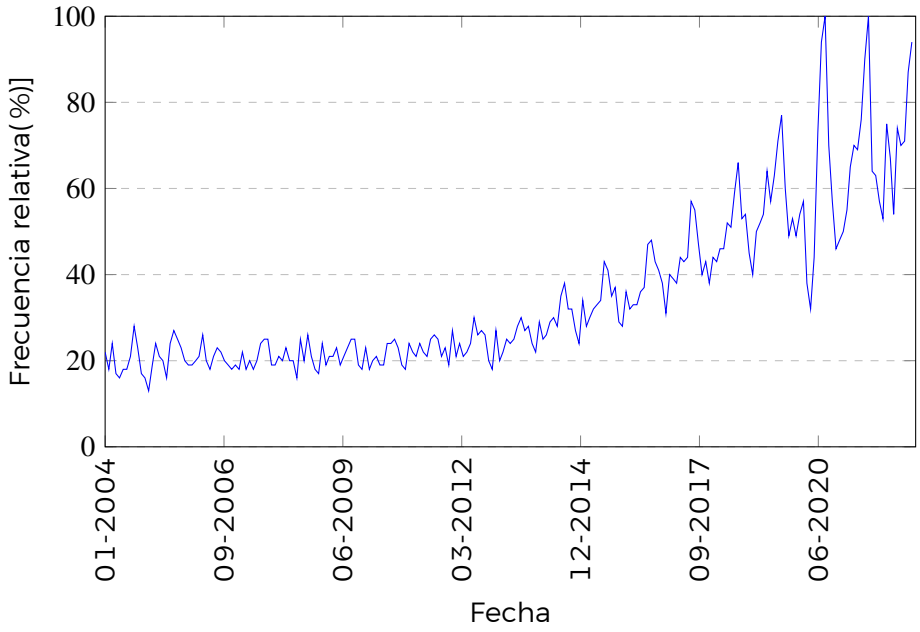


Figura 1.3 Resultados sobre la popularidad de la búsqueda «camper» fuente: Google Trends

1.2.1. ¿Que significa «Camperizar»?

La idea de camperizar un vehículo, principalmente furgonetas ligeras y furgones, es la de convertir un vehículo, ya sea temporal o definitivamente, en un pequeño hogar

en el que pasar una noche, varios días e incluso meses viviendo en ella. Para esto se realizan todo tipo de modificaciones, por ejemplo:

- Instalación de muebles de cocina.
- Adaptar el habitáculo para poder dormir en él.
- Crear zonas de almacenaje y organización extras.
- Añadir sistemas de agua y almacenaje temporal de aguas grises y negras.
- instalar baterías auxiliares y sistemas de carga e iluminación extras.

Este movimiento, sumado a la mejora de los sistemas fotovoltaicos, hace que muchos se hayan propuesto instalar pequeños sistemas solares con los que evitar el uso de la batería principal del coche, a la vez que mantienen una autonomía mayor sin tener que encender el motor de combustión interna, ahorrando gasolina, y aprovechando los recursos naturales.

1.3. Objetivo

En este punto nos encontramos con lo siguiente, si bien existen sistemas muy simples y fáciles de instalar, si buscamos crear un sistema híbrido, capaz de utilizar los paneles solares cuando estén disponibles, y cambiar automáticamente a la carga mediante el alternador del vehículo una vez este arranque, las opciones pasan de ser soluciones comunes y sencillas de encontrar, a consistir en una serie de sistemas separados dónde debemos conectar y desconectar uno u otro manualmente para poder utilizar ambas fuentes.

Con la idea de solucionar esto, nos planteamos un sistema, controlado mediante un microcontrolador, capaz de leer el estado actual del circuito, y conectar la fuente más eficiente, a la vez que nos muestra de manera sencilla un estado general sobre la carga/descarga de la batería. Este documento resumirá el proceso de desarrollo tanto del programa como de la mesa de pruebas que se ha realizado para poder verificar su correcto funcionamiento.

1.4. Estructura de la memoria

- **Capítulo 2:** Circuito fotovoltaico. Recopilación de datos necesarios para la producción del prototipo y presentación de los componentes que forman el circuito solar.
- **Capítulo 3:** Hardware. Se hablará de todo lo necesario para crear el prototipo, así como sus especificaciones y conectividad

- **Capítulo 4:** Visualización de datos. En este capítulo planteamos las ideas para mostrar los datos de la manera más simple y efectiva posible.
- **Capítulo 5:** Software. En él se muestran todas las aplicaciones necesarias para el diseño del programa.
- **Capítulo 6:** El programa. Se trata de un resumen de las funcionalidades implementadas y la organización que se sigue en su ejecución.
- **Capítulo 7:** Problemas y soluciones. Durante el proceso para finalizar nuestro proyecto hemos tenido varios problemas. En este capítulo trataremos de explicar el proceso seguido para solucionarlos.
- **Capítulo 8:** Resultados. Una visión general de la funcionalidad mediante imágenes del resultado final.
- **Capítulo 9:** Presupuesto. Resumen de costes generales, tanto de la instalación fotovoltaica como de la fabricación y desarrollo del circuito de control.
- **Capítulo 10:** Conclusión. Se trata de un balance sobre si los resultados obtenidos cumplen lo esperado.

Capítulo 2

Circuito fotovoltaico

Para crear un programa como el que planteamos, debemos tener en cuenta qué parámetros queremos recoger y en qué rango se mueven dichos parámetros. Para ello, hemos creado un circuito completo, similar al que se utilizaría en una pequeña furgoneta camper que posea potencia suficiente para cumplir con las tareas requeridas, por ejemplo, cargar equipos como móviles y linternas, mover una pequeña bomba de agua para un grifo... Además de autonomía nocturna para poder mantener un circuito de iluminación.

2.1. Cálculo de consumos y equipamiento

Nuestra idea de instalación cumpliría lo necesario para que el usuario, aparte de cubrir sus necesidades diarias, pueda realizar tareas relacionadas con trabajos de informática por unas 4 o 5 horas sin necesitar de la batería del mismo. El equipamiento conectado a la red es el siguiente:

- Bomba de agua para el fregadero.
- Nevera de compresor.
- Iluminación interior.
- 3 cargadores USB.
- 1 cargador estándar para portátil.

Esta serie de electrodomésticos y cargadores nos ayudarán a escalar el circuito. Por un lado, tenemos el consumo diario que debemos suministrar a la red. Por otro lado, debemos tener en cuenta el pico de corriente que puede surgir si todos los electrodomésticos se encuentran funcionando a la vez.

Consumos por equipo			
Producto	Porcentaje de uso diario (24 h	Consumo	Consumo diario
Bomba de agua	8,3 %	50 W	100 W
Nevera Adventurer 50 Litre	15 W h, 24 h	60 W	360 W
4x bombilla led, 12V	25 %	12 W	72 W
Baseus Cargador Coche USB C de 3 salidas	20 %	65 W	312 W
QYD 65W USB-C	20 %	65 W	312 W
Totales:		252 W Pico	1156 W

Sabiendo entonces que debemos proporcionar hasta 250 W como máximo en un sólo momento, y, planteando una situación dónde estemos 24 h sin recargar la batería, tendremos que instalar un circuito fotovoltaico que soporte $\frac{252W}{12V} = 21A$ pico y $\frac{1,156kWh}{12V} = 96,6Ah$. Cabe destacar, que la posibilidad de que el circuito consuma 21 A es muy remota, deberíamos comenzar a cargar 3 equipos USB, un portátil, encender toda la luminaria, el agua y que el compresor de la nevera se encienda en este momento. Sin embargo, como ya veremos, nuestro sensor de corriente, que es el único elemento en contacto directo con el circuito solar, debe de ser capaz de soportar tal carga para evitar problemas.

2.2. Equipo Solar

2.2.1. Instalación presupuestada

Con el fin de cumplir con las especificaciones establecidas, gracias a estos cálculos y la aplicación de la unión europea « Photovoltaic Geographical Information System » [6] para una instalación con una autonomía de 100 A h, 480 W de producción solar, con un consumo diario de 1,156 kW y 12 V se instalarán los siguientes elementos:

- 3 Paneles solares fotovoltaico XUNZEL de 120W.
- Batería solar de gel U-POWER 12V 100Ah.
- Regulador SmartSolar VICTRON MPPT 100/30.

La instalación se realizará en un string con 3 paneles en serie, la corriente máxima que se puede suministrar con los paneles al 100 % es de 360 W, que coincide con el amperaje máximo que es capaz de suministrar el regulador solar $\frac{360W}{12V} = 30A$.

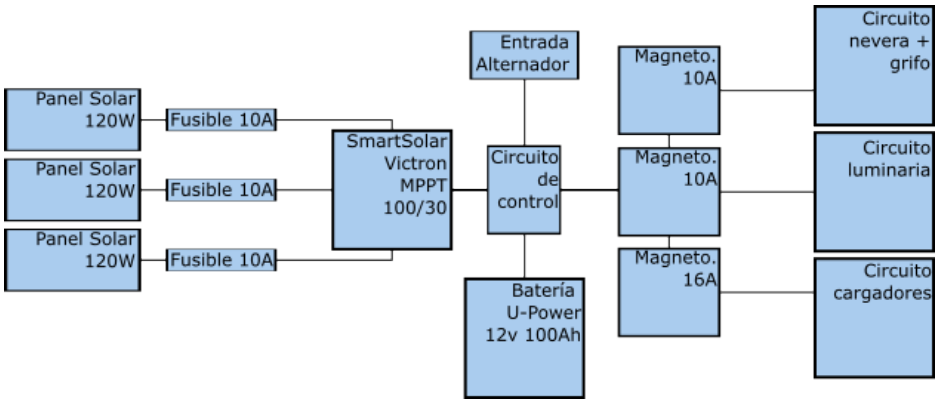


Figura 2.1 Esquema de la instalación solar, incluyendo protecciones y el equipo de control que implementaremos más adelante.

Tenemos que tener en cuenta que, debido a la disposición de las placas solares, colocadas horizontalmente en el techo del vehículo, la eficiencia es menor que planteando una instalación que se despliegue *in situ*, optimizando inclinación y azimut.

Los resultados de estos cálculos pueden consultarse en el apéndice B.3.1 y B.3.2. Con ellos podemos comprobar que, con el ritmo de consumo estipulado, la instalación en el techo requerirá de un mayor apoyo de alternativas como el alternador en los meses de invierno, mientras que el sistema con placas instalables en el exterior, colocadas a 25° de inclinación y -7° de azimut, pasan menos de un 6 % del tiempo descargadas.

Por supuesto, la decisión de una u otra tendrá que ser tomada por el interesado final, ya que no se trata de una vivienda, sino de un vehículo de ocio dónde, dependiendo de la situación, se pueden reducir los consumos, siendo el único consumo esencial el uso de la nevera. Influirá, de la misma manera, el emplazamiento del vehículo, empeorando la situación en mayor o menor medida dependiendo de dónde nos situemos.

Nuestros cálculos se basan en unas condiciones meteorológicas altamente favorables, en la isla de Tenerife, pero variarán dependiendo de que parte de la isla, o incluso si se viaja a otros lugares.

Protecciones

Para que la instalación sea completamente segura, se requieren protecciones eléctricas que garanticen que no se sobrepasan los límites establecidos, estas son:

- **Fusibles de protección DC ZPV25:** Nos aseguran que el panel solar no sobrepase su intensidad máxima (7 A) y cortan el circuito en caso de hacerlo.
- **Interruptor automático Schneider Electric A9N61528 C60H:** Se trata de un interruptor magnetotérmico de 10 A que cortará la circulación si el consumo en el circuito es excesivo. También nos permitirá compartimentar la instalación en caso de querer bajar el consumo.

- **Interruptor automático Schneider Electric A9N61531 C60H:** Cumple la misma función que el anterior, pero con una corriente superior, debido a que la suma de los dos cargadores funcionando supera los 120 W. Este interruptor es de 16 A.

2.2.2. Instalación de pruebas

Debido a que se trata de un circuito de pruebas, y no tiene ningún fin real, salvo el de proporcionar los datos necesarios, el equipo consta de piezas ya usadas que, si bien no tienen por qué proporcionar una eficiencia del 100 %, mantienen el funcionamiento mínimo para lo que buscamos, y lo sobrepasan en algunas medidas.

- **2 Paneles solares Aide Solar AD185M5-Aa:** Se trata de dos paneles de 185 W capaz de dar 5,1 A a 37,1 V. Ambos paneles son obtenidos tras ser retirados de una nave por motivos ajenos al estado o eficiencia del panel en sí y presentan un correcto funcionamiento.
- **Cargador solar PWM ES-HP2430:** Este cargador solar se desmonta de una cruz de farmacia solar que fue convertida a corriente alterna debido a la poca exposición al sol del lugar, es capaz de proporcionar hasta 30A pico de corriente, además de identificar y convertir el voltaje a 12 V o 24 V.
- **Batería de GEL LivEN LEVG50-12:** Se trata de una batería retirada también de otra cruz de farmacia solar, debido a la perdida notable de capacidad. Que se estimó en su momento en menos del 40 %. Su voltaje de funcionamiento es de 12 V y su capacidad máxima original de 50 A h.

El resto de características se pueden encontrar en el anexo B.2.

Figura 2.2 Cargador solar

Figura 2.3 Batería LivEN

Figura 2.4 Panel solar Aide Solar

2.3. Intrumentos de medida

Para realizar las medidas experimentales necesarias, y comprobar el funcionamiento, nos hemos provisto de dos instrumentos:

- **Multímetro digital KONSHI DT850L:** Multímetro digital de 3½ dígitos. Capaz de realizar lecturas de hasta 10A siempre y cuando sean realizadas por un periodo no mayor de 10 s.
- **Osciloscopio digital Hantek DSO5102P:** osciloscopio de dos canales, totalmente digital, capaz de leer frecuencias de hasta 100 MHz y exportar las tomas de datos mediante un dispositivo USB y una frecuencia de muestreo de 1Gsa/S.

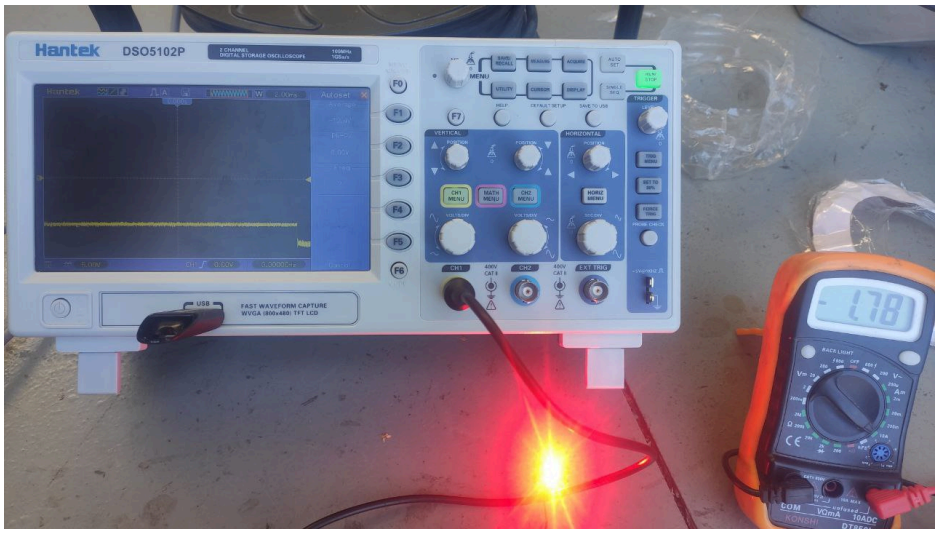


Figura 2.5 Toma de datos mediante ambos dispositivos simultáneamente

2.4. Lectura de valores de trabajo de la instalación

Con el circuito montado, hemos realizado varias medidas para comprobar cuál es el voltaje máximo, cual es el mínimo, tanto del circuito solar, como de la salida de la batería. Además, hemos realizado lecturas de intensidad para comprobar que todo funciona correctamente.

2.4.1. Lecturas de voltaje

Mediante el multímetro y el osciloscopio se han realizado las siguientes mediciones de voltaje:

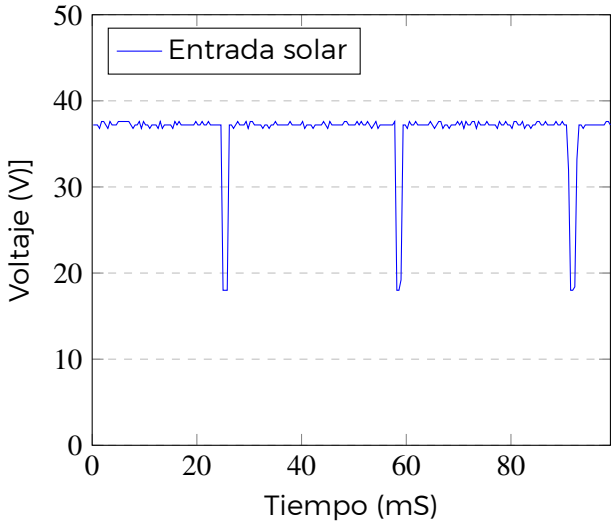


Figura 2.6 Voltaje a la entrada del cargador solar

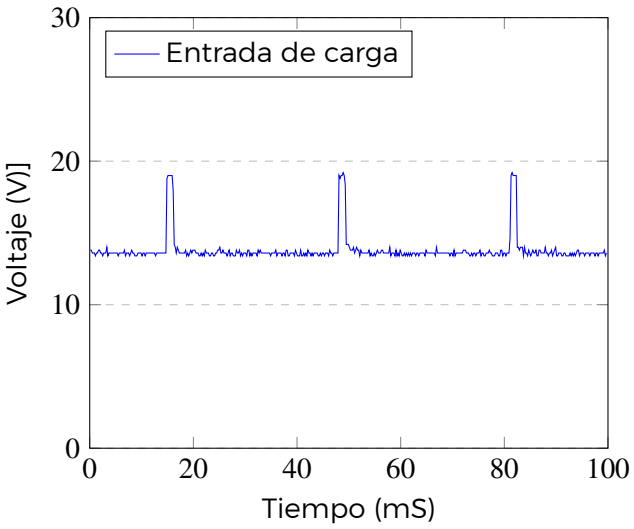


Figura 2.7 Voltaje a la entrada de la batería

Estas gráficas son los valores recabados y exportados directamente de nuestro osciloscopio. las figuras 2.8 y 2.9 reflejan las medidas tomadas in situ por el osciloscopio.

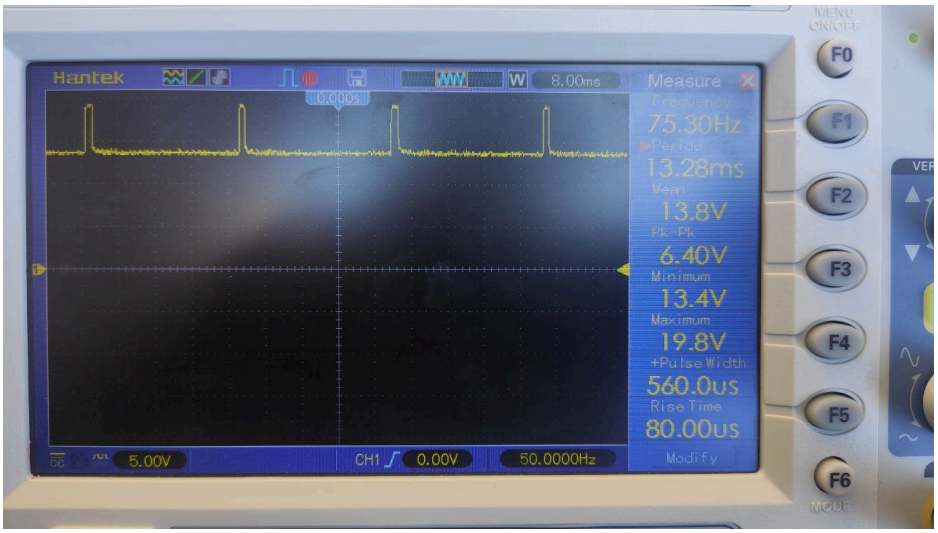


Figura 2.8 Datos recabados por el osciloscopio a la entrada de la batería

2.4.2. Lecturas de intensidad

La intensidad será tomada directamente por el multímetro y por el cargador solar, completando la instalación con una resistencia en la zona de consumos para forzar una descarga y así poder recabar dichos datos.

Como vemos, todos los elementos del sistema funcionan correctamente.

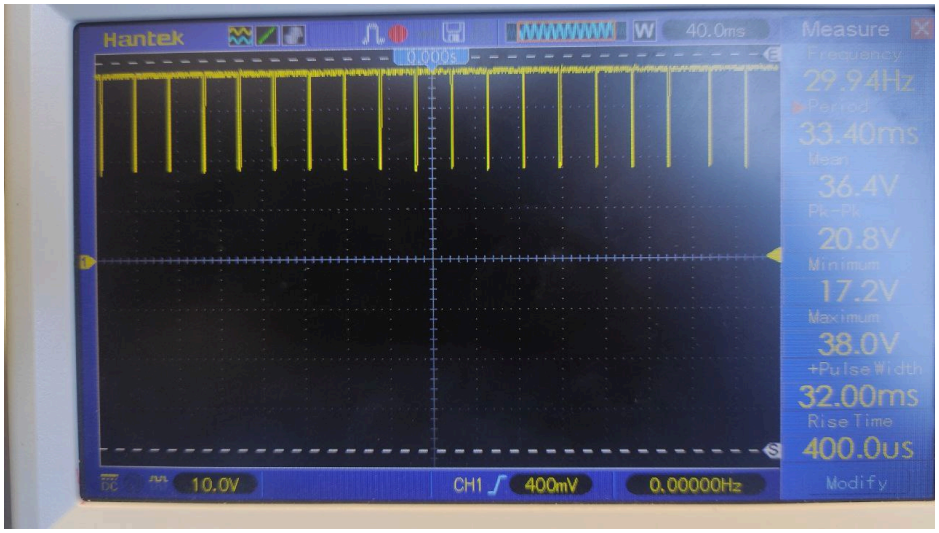


Figura 2.9 Datos recabados por el osciloscopio a la salida de los paneles

Figura 2.10 Lectura de intensidad por los diferentes equipos durante la carga de la batería

Figura 2.11 Intensidad consumida durante un ciclo de descarga

2.4.3. Interpretación de los datos

Con todos estos datos, podemos sacar en claro lo siguiente.

- El voltaje de operación de los paneles es casi el original, a pleno rendimiento, ofrecen 36,4 V.
- El voltaje de trabajo de la batería llega a un máximo de 19,8 V, siendo su voltaje medio de 13,8 V durante la operación de carga.
- Aunque sólo conseguimos una cara de 3 A, nuestros cálculos indican que nuestro medidor de corriente debería leer hasta un máximo de 21 A.

Estos parámetros serán utilizados más adelante para poder establecer que equipo necesitaremos y los pasos para poder leer tales vo.ltajes en un microcontrolador que trabaja sólo a 5 V

Capítulo 3

Hardware

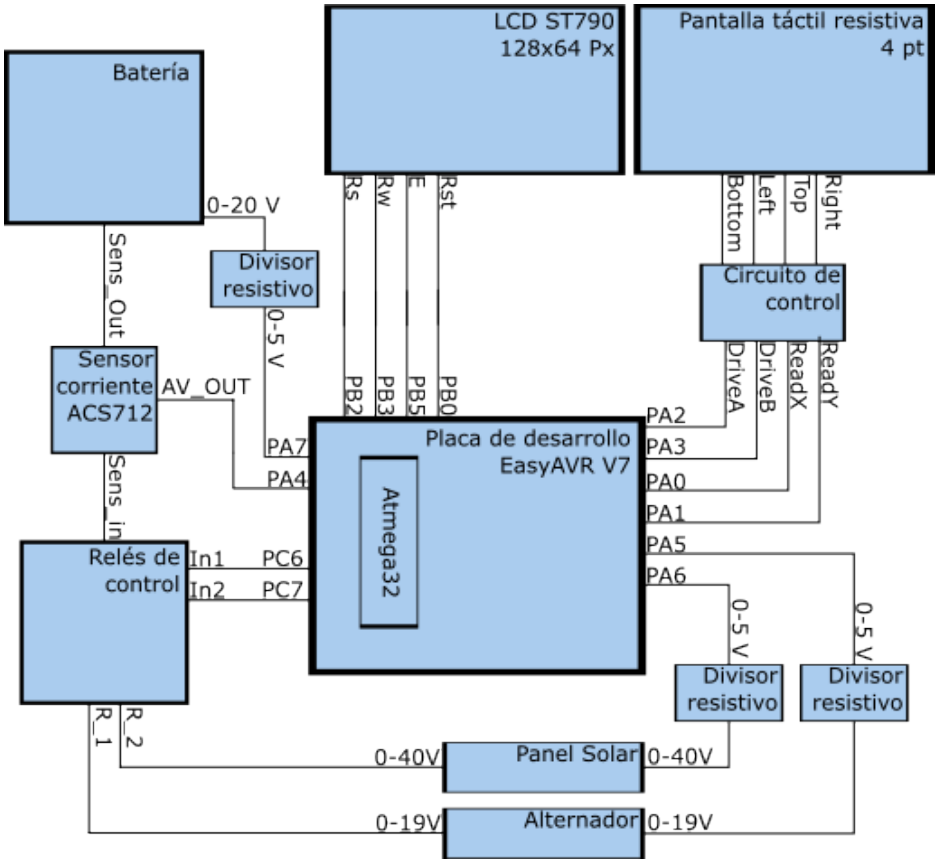


Figura 3.1 Esquema del montaje completo

Tanto para realizar las lecturas de los valores, como para poder realizar la programación del microcontrolador, y mostrar dichas lecturas, se requiere de una serie de elementos y programas. En este capítulo mostraremos los sistemas y periféricos princi-

pales necesarios para poder realizar pruebas y controles sobre el programa que vamos a realizar.

3.1. Montaje

en la figura 3.1 podemos observar el montaje completo del sistema.

Para el montaje de prueba y exposición, el panel solar y el alternador se han sustituido por una fuente de alimentación de 24 V y un circuito de conversión de voltaje variable. Para crear flujos de corriente, la batería se sustituye por un voltaje variable y una resistencia de 56 Ω y 11 W que creará flujos de hasta $21V/56\Omega = 0{,}375A$.

Aparte del sistema fotovoltaico previamente explicado, a continuación, se explicará el resto de sistemas principales del que está formado el circuito.

3.2. Placa de desarollo

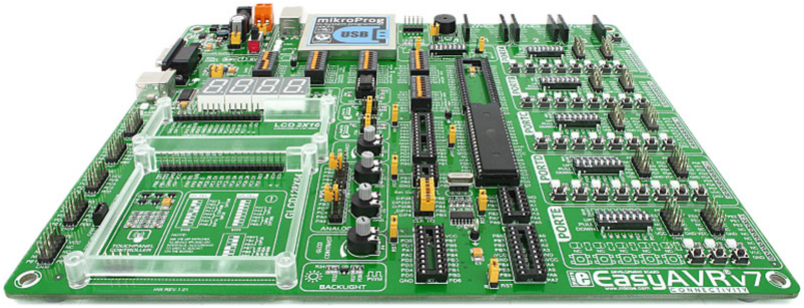


Figura 3.2 Placa de desarrollo EasyAVR V7

Si bien existe la posibilidad de crear un circuito simple en una protoboard para poder programar y utilizar un microcontrolador, el uso de una placa de desarrollo facilita muchísimo las cosas, no sólo por la capacidad de poder utilizar diferentes tipos de microcontroladores sin necesidad de cambiar el circuito, si no por las facilidades en cuanto a elección de salidas, simulación de inputs, disponibilidad directa e inmediata de un programador y disponibilidad de varios periféricos.

La placa seleccionada ha sido la placa *Easy AVR v7* desarrollado por el fabricante MIKROe. en el apéndice B.1.3

3.2.1. Características

Se trata de una placa con cabeceras para todos los puertos, compatibilidad para cualquier microcontrolador de la serie AVR desde 8 pines hasta 64. Posee un circuito de alimentación propio que permite utilizar el mismo USB que utilizamos para programar para alimentar toda la circuitería, por supuesto, esto no es obligatorio y cuenta

con alimentación externa. A continuación, se listan los periféricos y características que vamos a utilizar para nuestro programa.

3.2.2. Principales circuitos y periféricos utilizados

- **Pantalla ST7920:** Se trata de una pantalla de 128x64 píxeles monocromática, capaz de recibir en serie y en paralelo, con un voltaje de alimentación de –2,7 V a 5,5 V. En nuestro caso recurriremos a la transmisión en serie.
- **Pantalla táctil HST035003-A:** Se trata de una pantalla táctil resistiva genérica, constituida por dos capas transparentes de igual resistencia, su funcionamiento se basa en que, al pulsar sobre la superficie ambas láminas entran en contacto. Con el circuito correcto podemos utilizar este contacto entre las dos láminas, y su consiguiente cambio en la resistencia, para averiguar exactamente dónde estamos pulsando.
- **Circuito de control táctil:** Cada uno de los cables del panel táctil (Top, Bottom, Left & Right) cumple un funcionamiento; si queremos leer el eje X, alimentamos Left con voltaje, conectamos Right a GND y leemos desde el pin Bottom. En el caso de querer leer el eje Y, conectamos Top a voltaje, Bottom a GND y leemos desde el pin Left. Para simplificar todo este trabajo, el circuito de control nos da 4 cables, ReadX, ReadY, DriveA y DriveB. Para poder hacer las lecturas simplemente Alimentaremos DriveA y leeremos ReadX, esto nos dará el resultado del eje X. Para el eje Y, conectaremos DriveB y leeremos ReadY.

3.3. Microcontrolador

Figura 3.3 Microcontrolador ATmega32

3.3.1. Selección del microcontrolador

A la hora de elegir un microcontrolador, se deben de tener en cuenta varias características propias del integrado, Existen microcontroladores de diferentes fabricantes y modelos, cada uno con su propia arquitectura y capacidades.

3.3.2. AVR vs PIC

A la hora de programar este proyecto nos planteamos dos tipos de microcontroladores, los microcontroladores PIC y los AVR. Cada uno de estos ofrece diferentes anchos de banda, set de instrucciones, memorias... Cada uno debe de tener en cuenta las capacidades del microchip para poder elegir uno u otro. En nuestro caso, el hecho de que los microcontroladores AVR tengan integrados en ellos una memoria EEPROM facilita mucho el trabajo ya que, cuando hablamos de un sistema, que en esencia posee Kbytes de memoria, el uso de imágenes para mostrar en pantalla puede provocar que la memoria del dispositivo quede completamente llena.

3.3.3. ATmega3216-PU

Se trata del actor principal de este proyecto, un microcontrolador fabricado en un encapsulado de 40 patas por el fabricante Microchip. Sus principales características son las siguientes:

- **Memoria:**
  - 32 Kbytes de memoria flash programable.
  - 1024 bytes de memoria EEPROM.
  - 2 Kbytes de memoria SRAM.
  - Vida útil: 10000 ciclos para la memoria Flash, 100000 para la memoria EEPROM.
- **Set de instrucciones:**
  - Arquitectura RISC.
  - 131 instrucciones capaces de ejecutarse en un sólo ciclo de reloj.
  - 32x8 registros de propósito general.
- **Periféricos:**
  - Dos contadores de 8 Bits con preescaladores y comparadores separados.
  - Un contador de 16 Bits con preescaladores y comparadores separados, además de un modo extra de captura.
  - un contador RTC.
  - Cuatro Canales generadores de PWM.
  - ADC de 8 canales y 10 Bits.
  - Watchdog programable.
  - Interfaz serial.

- Comparador analógico integrado.

- **Voltajes de operación:** de 4,5 V a 5,5 V.
- **Velocidad de operación:** de 0 MHz a 16 MHz.
- **Salidas y entradas:** 32 Líneas programables como salidas o entradas.

El resto de características puede encontrarse en el apéndice B.2.1.

3.4. Otros periféricos utilizados

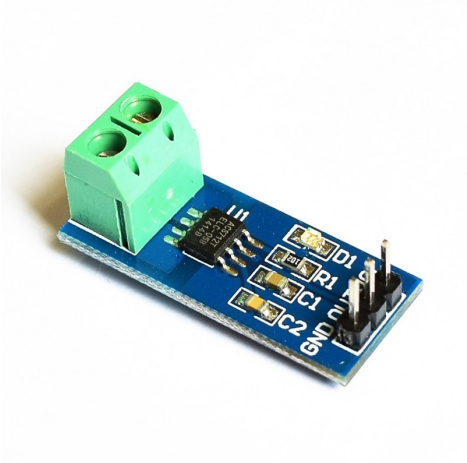


Figura 3.4 Sensor de corriente de efecto Hall ACS712

Tanto para poder leer los diferentes parámetros, como para poder crear un circuito que nos permita su simulación, independientemente de la hora del día, se han recurrido a los siguientes circuitos y accesorios:

- **Sensor de efecto hall ACS712:** Sensor de corriente capaz de detectar, mediante el efecto Hall, la corriente que circula a través de una parte de su circuito. Por la otra parte devolverá, de 0 a Vcc, un voltaje proporcional a la corriente que circula. Este modelo ofrece una lectura de −30 A a 30 A, con un paso de 66 mV por cada amperio. Esto nos permitirá saber también, si nuestra batería se encuentra cargando o descargando. Estos valores se ajustan a los necesarios para nuestros cálculos teóricos.
- **Conversor de voltaje variable LM317:** Se trata de un circuito integrado capaz de convertir un voltajes de 0 V a 40 V a cualquier voltaje hasta 3-Vcc. Tras las pruebas de campo, será utilizado para crear un esquema de funcionamiento capaz de testear el comportamiento de nuestro programa.

- **Fuente de alimentación Sunpower SPS-075-24:** Al igual que el LM317, se trata de una fuente de alimentación que utilizaremos para emular tanto la entrada del panel solar como la entrada del alternador, se trata de una fuente de 24 V y 3,2 A. Para nuestra simulación planteamos el uso de menos de 400 mA a, como máximo 21 V, por lo que la fuente será más que suficiente para cumplir este cometido.

Capítulo 4

Visualización de datos

La parte más importante de este proyecto consiste en poder mostrar los datos sin que se requiera de una experiencia anterior, o un manual para poder entender dónde estamos, y a dónde queremos ir. El arte de la usabilidad es un tema increíblemente amplio que nos ayuda en el día a día a facilitar el acceso a estudios, encuestas, estadísticas... En nuestro caso, debido a las limitaciones de resolución (128x64 píxeles) y datos, nuestro objetivo se centra en conseguir que los datos puedan ser interpretados sin mostrar ningún texto, ya que éste ocuparía demasiado.

4.1. Consideraciones previas

Con los datos recogidos podemos mostrar infinidad de variables: potencia de entrada, salida, voltajes, intensidades... Sin embargo, no todo el mundo necesita tener un centro de información con todos los parámetros que un sistema como este puede ofrecer. El objetivo de muchos es simplemente saber que está pasando; ¿estamos cargando o descargando la batería? ¿de dónde viene la energía actualmente? El motor está encendido, ¿se está utilizando para cargar?

Por otro lado, está la importancia de cada dato. tenemos menús que representan información, mediante gráficos o datos en crudo, pero también existen otros menús, como el de configuración. Estos menús no son utilizados la mayoría del tiempo, y, por tanto, no requieren de tener una presencia tan importante como los primeros.

4.2. Las pantallas

Desde el primer momento se plantearon 4 pantallas, con las dos primeras cambiando de forma y idea con el paso del tiempo. Durante la etapa de diseño final quedó claro que debemos mostrar:

1. **Información visual:** Se trata de una pantalla que mostrará la información sin ningún tipo de dato numérico. El objetivo es la consulta rápida del estado actual.

2. **Datos:** En esta pantalla se recogerán las variables tomadas. Valores de voltaje e intensidad quedarán reflejados aquí.
3. **Control automático:** Se decidió integrar un control automático. Por lo general, el sistema decidirá que fuente es más importante, dando prioridad siempre al panel solar si este está disponible. Si el usuario quiere, sin embargo, puede solicitar que se cargue desde el alternador, siempre que este esté disponible.
4. **Calibración:** En caso de que la respuesta del panel no sea correcta, se ha creado un menú que lanza una rutina de calibración con el objetivo de corregir estos errores.

4.3. Menú

Figura 4.1 Diferentes iteraciones de los iconos del menú

Con estos datos en mente pasamos a estructurar los elementos de la pantalla. Se tomó la decisión de cuanto tomaría de espacio el menú en la pantalla que, si bien es espacio que no puede ser ocupado por los verdaderos valores, debe de tener un tamaño suficiente para que un dedo pueda tocarlo sin equivocarse. Por ello, el menú ocupa el 25 %. Por otro lado, como decíamos, los logos de control manual y calibración llevan a menús secundarios que no suelen usarse, por eso se tomó la decisión que los menús secundarios ocuparían tanto como un menú primario.

4.4. Pantallas

4.4.1. Primera iteración

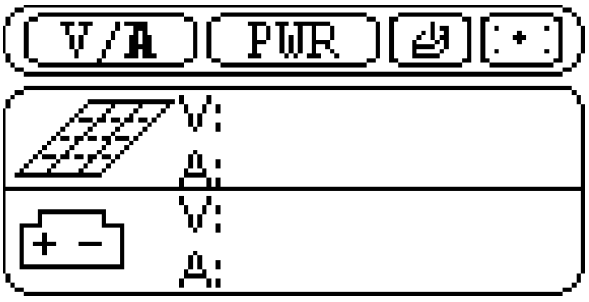


Figura 4.2 Primera idea sobre la visualización

Durante los primeros meses de vida del proyecto, se planteó una pantalla menos visual, con más letras, este sistema planteaba una visualización pura de datos, sin recoger valores del alternador, en la que se daría la información sobre pantalla y batería

en el primer menú, y una segunda con valores sobre las potencias momentáneas y acumuladas. Esta idea se muestra en la figura 4.2. Pronto nos dimos cuenta de que la visualización se hacía pesada, los menús con palabras eran menos intuitivos, y perdías la sencillez de la visualización al tener que leer todos los datos para poder interpretar la situación.

4.4.2. Pantalla 1: visualización en gráficos

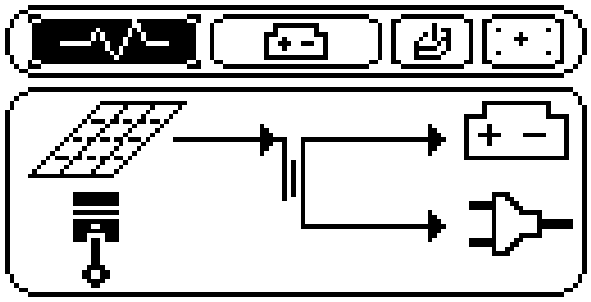


Figura 4.3 pantalla de información

Con la nueva idea, nos planteamos una pantalla muy simple, como vemos en la figura 4.3, llevaba por los datos, muestra mediante flechas y líneas hacia dónde circula la corriente, y desde dónde proviene su fuente. De esta manera seríamos capaces de saber que está haciendo el circuito en todo momento.

4.4.3. Pantalla 2: datos

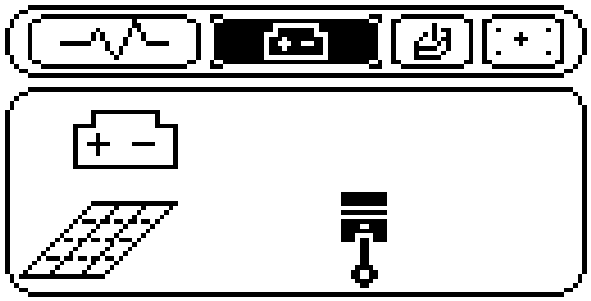


Figura 4.4 Pantalla de visualización de datos

La segunda pantalla muestra todos los datos. Como vemos en la figura 4.4, apenas hay datos, sólo logos. Esto es porque será el microcontrolador el encargado de colocar las letras dónde sea necesario. Esta decisión se tomó porque de esta manera, si cambia el número de datos, las unidades se mantienen junto a él, por ejemplo, en un dato que mostremos como $16,6V$ si la V es fija, tendremos, visualmente un espacio que no necesitamos ($9,9 \quad V$).

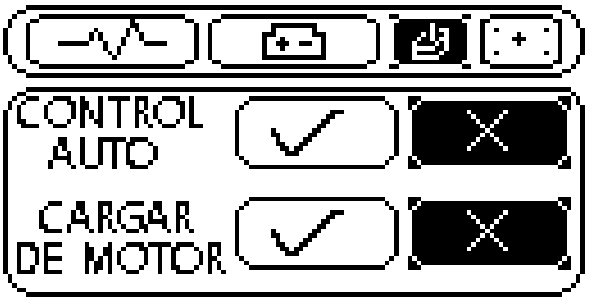


Figura 4.5 Pantalla de automatización

4.4.4. Pantalla 3: automatización

La tercera pantalla consiste en un pequeño árbol de decisión dónde tomaremos o no el control sobre cuál es la fuente de carga. Ya sea en caso de errores de lectura, o fallo del panel solar, podemos forzar que el sistema cargue del alternador directamente.

4.4.5. Pantalla 4: calibración

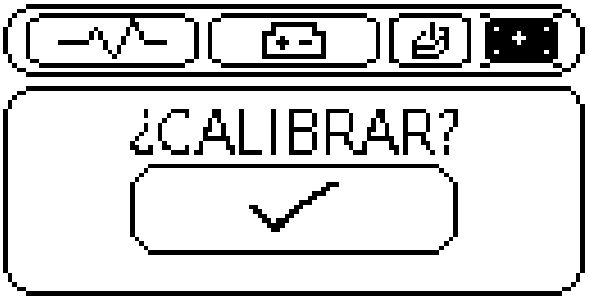


Figura 4.6 Pantalla de calibración

La última pantalla tiene una sola opción, y más grande de lo normal para poder facilitar el pulsar la opción en caso de problemas con la calibración. Esta pantalla solo queda a la espera de que pulsemos para lanzar la rutina de calibración de la misma.

Finalmente, el proceso de importar y exportar estas pantallas se explica en el punto 5.1.

Capítulo 5

Software

5.1. Generación de pantallas mediante la aplicación GIMP

> «Tanto si eres diseñador gráfico, fotógrafo, ilustrador o científico, GIMP te proporciona sofisticadas herramientas para realizar tu trabajo. Puedes mejorar aún más tu productividad con GIMP gracias a las numerosas opciones de personalización y a los plugins de terceros».[7]

GIMP es un programa de edición de imágenes completamente gratuito y con licencia pública general de GNU [8] [9], Las funcionalidades de este programa son inmensas aunque para el proyecto destaca una en concreto, la habilidad de utilizar addons de terceros nos permite recurrir a funcionalidades extras como es la capacidad de exportar nuestras imágenes a mapas de bits en un formato compatible con nuestro lenguaje de programación, incluyendo el archivo Source y el de Header.

5.1.1. Creación de pantallas

Para crear nuestras pantallas debemos instalar el plug-in «Export-GLIB-Pixmap». [10]. Después de esto crearemos una imagen con la resolución exacta de nuestra pantalla

Una vez creada, y limitando la gama de colores a blanco y negro, crearemos las imágenes y pantallas que queramos mostrar.

A continuación, cuando tengamos nuestra imagen lista, cambiaremos el modo de trabajo de RGB a indexado en blanco y negro y nos aseguraremos que no existe más de una capa en la imagen y que el canal alfa[1] ha sido eliminado.

Finalmente, la opción «Export GLIB Pixmap» pasará a estar disponible y nos creará dos archivos por cada imagen que exportemos.

[1] Canal adicional al RGB que define la transparencia de cada píxel.

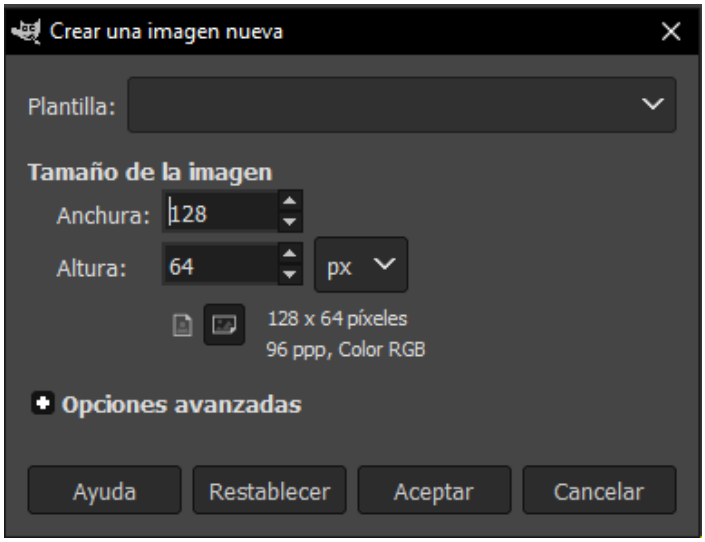
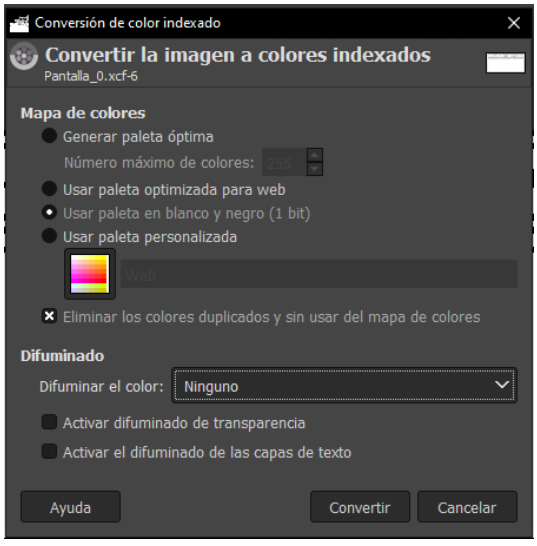
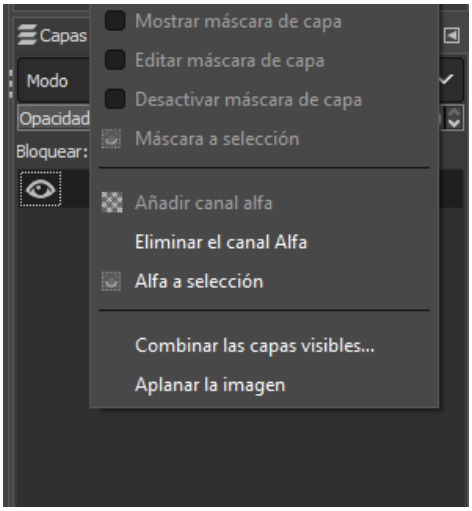


Figura 5.1 Pantalla de creación de imágenes con la resolución adecuada



(a) Indexado blanco y negro



(b) Eliminación del canal Alfa en la pestaña capas

5.2. Creación de esquemas de conexión con Multisim

> «Multisim™ Electronics Workbench™ es un software estándar en industria para diseño de circuitos y simulación SPICE para electrónica de potencia, analógica y digital en la educación y la investigación». [11]

En resumen, Multisim es la herramienta que nos ayudará a crear los circuitos que necesitaremos para acompañar nuestra simulación. En esencia es una versión más gráfica que el software LTSpice, con una interfaz mucho más visual.

5.2.1. Creación de un divisor resistivo y pruebas

Debido a que nuestra mesa de trabajo sólo dispone de una fuente de alimentación de 24V, y, como veremos en la sección 2.4 El circuito solar genera alrededor de unos 40V, crearemos el convertidor de manera teórica para su incorporación en el producto final.

Cálculos teóricos

Para calcular un divisor de tensión con una resistencia de carga se calcula, en primer lugar, la resistencia equivalente:

$$R_{eq} = \frac{R_2 * R_L}{R_2 + R_L} \tag{5.1}$$

$$V_o = V_{DC1} * (\frac{R_{eq}}{R_1 + R_{eq}}) \tag{5.2}$$

Siendo V_{DC1} = 40 V y V_o = 5 V y fijando R_1 en 100 kΩ:

$$5 = 40 * (\frac{R_{eq}}{100K + R_{eq}})$$

$$12{,}5K = R_{eq} - 0{,}125R_{eq}$$

$$R_{eq} = 14\,285{,}71\,\Omega$$

Con esta resistencia calculamos R_2 y R_L, para ello, fijamos R_2 a 100 kΩ

$$14285,71 = \frac{100K * R_L}{100K + R_L}$$

$$R_L = 16666{,}6666$$

Ajustando a la resistencia inferior, para garantizar que la lectura nunca supere los 5 V, escogeremos 15 kΩ.

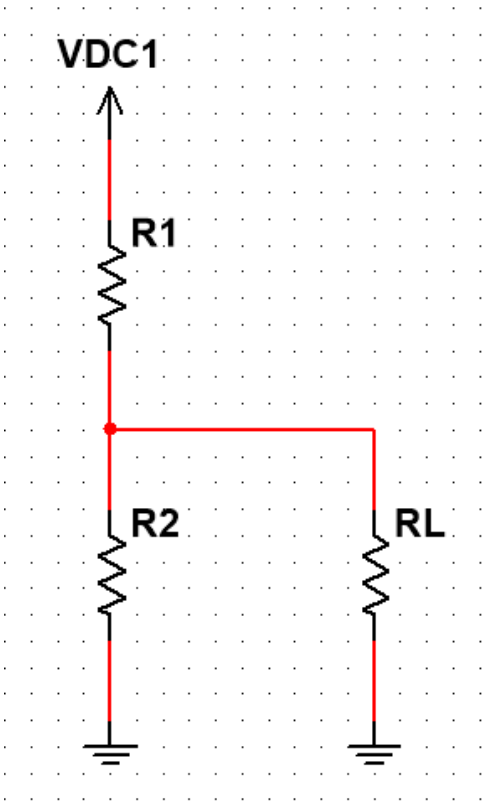


Figura 5.3 Circuito ejemplo de un divisor resistivo

Pasamos estos valores a la simulación de nuevo e incluimos un multímetro para comprobar que, efectivamente, con el valor más grande de voltaje, obtenemos 5 V o menos. la figura 5.4 nos confirma que el divisor cumple con los requisitos y con un consumo de corriente mínimo.

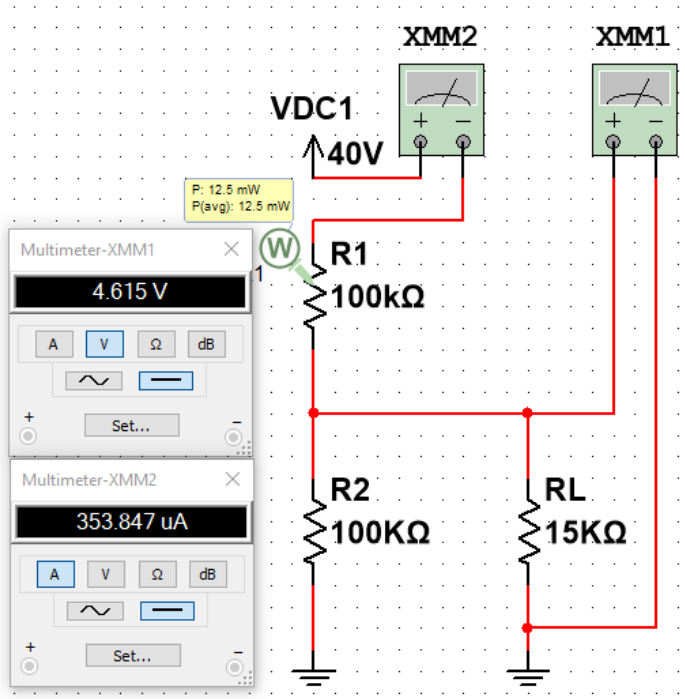


Figura 5.4 Comprobación de la simulación correcta

5.2.2. Preparar los datos para el programa

Es importante que, una vez conseguido el resultado buscado, volvamos hacia atrás y comprobemos el voltaje teórico de entrada. Este dato será necesario en el capítulo 6 a la hora de crear una referencia para el ADC. El valor de V_{DC1} teórico es de 43,333 V.

El diagrama de conexión de nuestro sistema se encuentra en el apéndice B.1.

5.3. Creación del programa mediante MicroChip Studio

> «Microchip Studio es un entorno de desarrollo integrado (IDE) para desarrollar y depurar aplicaciones de microcontroladores AVR® y SAM. Fusiona todas las grandes características y funcionalidades de Atmel Studio con la cartera de herramientas de desarrollo de Microchip, muy bien soportadas, para ofrecerle un entorno fluido y fácil de usar para escribir, construir y depurar sus aplicaciones escritas en C/C++ o código ensamblador». [12]

Microchip Studio es la base de nuestro proyecto, en esencia, es un compilador capaz de convertir, tanto C, como C++ y Ensamblador al lenguaje de nuestro Microcontrolador. Dado que está desarrollado específicamente por el propio fabricante, posee muchas soluciones dedicadas, por ejemplo, la capacidad de obtener una estimación del espacio utilizado en la memoria interna al final de cada compilación.

5.3.1. Creación de un programa

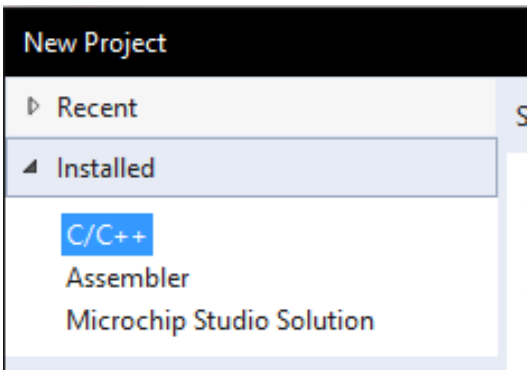
Para poner a prueba tanto nuestro microcontrolador como la aplicación y tener una primera toma de contacto, vamos a crear un pequeño código que encienda un led cada vez que pulsemos un botón. Los pasos a seguir son los siguientes:

1. Crearemos un proyecto nuevo, indicando si lo que queremos es crear un proyecto en ensamblador, código C o Código C++.

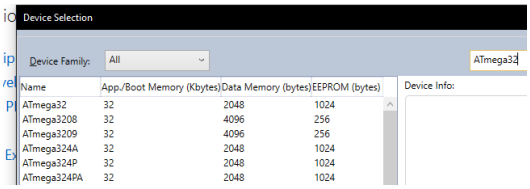
2. A continuación la ventana nos preguntará que tipo de microcontrolador vamos a utilizar, seleccionaremos entonces la familia correcta.

3. Ahora es cuestión de realizar nuestro programa, y una vez finalizado, pulsar en *build solution*, o simplemente, pulsar la tecla F7.

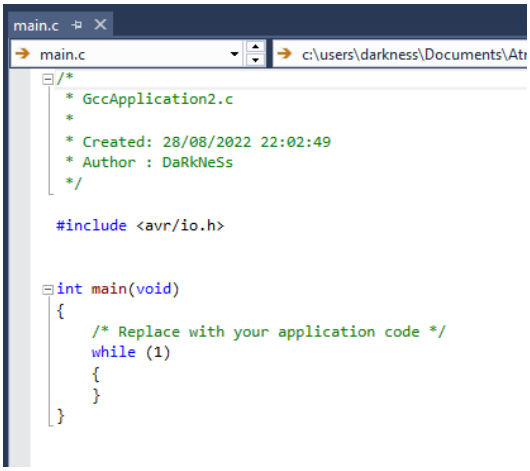
4. Si todo va bien, el programa nos devolverá un aviso de que todo está correcto, y nos dará una vista general del espacio usado por el programa en cuestión.



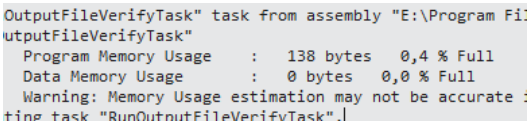
(a) selección del tipo de proyecto



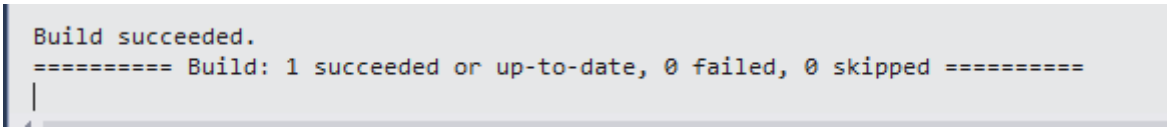
(b) Selección del modelo de microcontrolador



(c) Pantalla de programación



(d) Datos de uso de memoria



(e) Resultado correcto

Figura 5.5 Pasos para crear el programa

Una vez terminado, nos quedará un programa como este:

```
1  /*
2   * Tf_test_2.c
3   *
4   * Created: 20/04/2022 18:44:09
5   * Author : Sergio
6   * Creating the basics of my Project, a step by step how-to project about programing
       an ATmega32
7   * Chapter two, Make led in B2 switch on, every time the button in PORTB6 is pushed
8   */
9
10 #define F_CPU 1000000 /* Frequency of the clock, remember to ask if its using a 1Mhz?
       */
```

```
11
12 #include <avr/io.h>
13
14 int main (void)
15 {
16   DDRB = 0b00000100;    //PORTB I/O Configuration
17   PORTB = 0x00;   //PORTB '0'
18
19
20   while(1)
21   {
22     if (PINB & (1<<6)) //If button is pushed
23     {
24       PORTB = PORTB | (1<<2); //PORTB2 is set to high
25     }
26     else
27     {
28       PORTB = PORTB & ~(1<<2); // Keeps the led OFF
29     }
30   }
31 }
```

5.4. programar el microcontrolador mediante AVRFLASH

> «El programador AVRprog es una herramienta de alto rendimiento utilizada para programar las familias de microcontroladores AVR de Atmel. El programador AVRflash se comunica con el microcontrolador a través de un cable USB que también es utilizado para alimentar el programador AVRprog». [13]

El programa AVRFlash nos permite programar directamente en la placa de desarrollo. Con él programaremos tiempos de boot, bits de parada, fusibles internos y velocidad y origen de la CPU.

5.4.1. Cargando el programa

El proceso es simple: una vez compilado el programa en Microchip Studio, abriremos el software AVRFlash, Pulsaremos File -> Load Hex y localizaremos, dentro de la carpeta del proyecto, una carpeta llamada «debug». Dentro de ella existe un archivo con la extensión *.Hex*. Una vez tengamos el archivo cargado, y tengamos la placa de desarrollo encendida y conectada al ordenador, pulsaremos en write. Comenzará entonces un proceso de carga que tardará más o menos dependiendo del tamaño del archivo. Una vez acabe, el programa estará subido al microcontrolador para ser probado y la aplicación puede cerrarse.

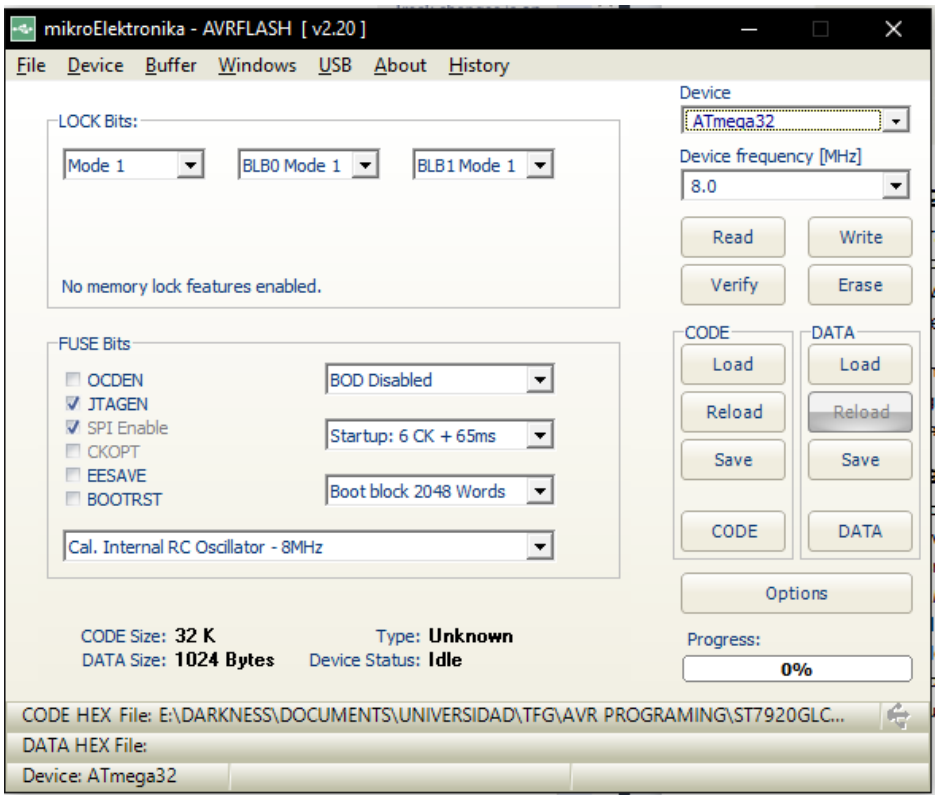


Figura 5.6 Pantalla principal del software AVRFlash con el archivo Hexadecimal y la configuración ya cargada

Capítulo 6

El programa

En este capítulo explicaremos los pasos que sigue el programa en su ejecución normal.

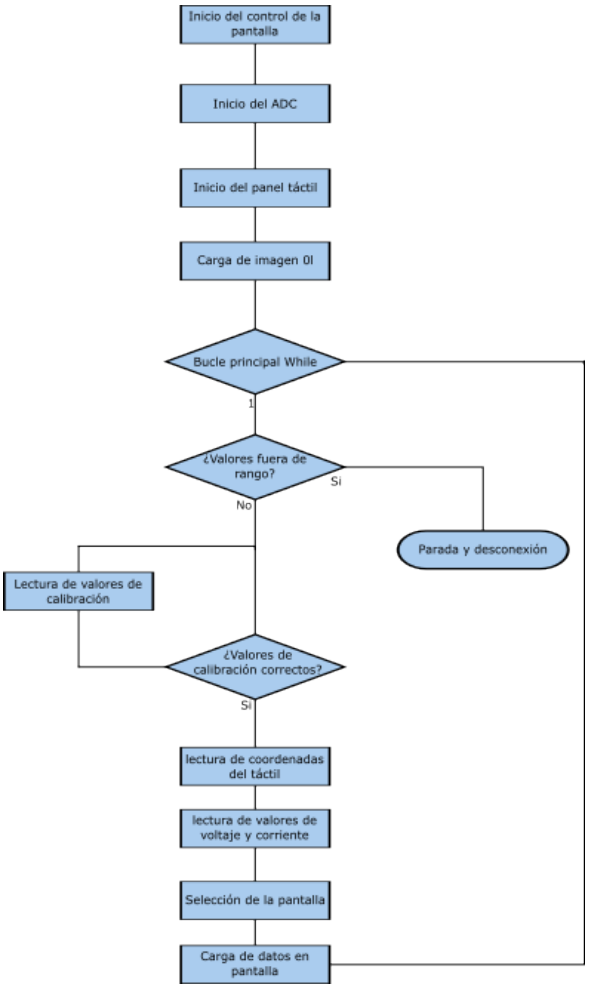


Figura 6.1 Diagrama de flujo del proceso de funcionamiento del programa

6.1. Inicio del control de la pantalla

La pantalla es el primer sistema que se inicia y el primero que fue implementado. En ella podremos mostrar los valores leídos por el ADC, y así poder validarlo, actuando de alguna manera como «debugger» de nuestro programa.

6.1.1. La librería u8g2

Este tipo de pantallas se pueden utilizar de dos maneras: control paralelo, donde cada bit es enviado a través de líneas individuales a la vez, o en serie, donde enviaremos un tren de datos que será recibido por la pantalla. En nuestro caso hemos optado por utilizar el control en serie mediante la librería u8g2[14].

Esta librería es actualmente una de las más grandes que existen, cubriendo cientos de modelos de pantalla, ya sea a color, o monocromáticas, sea de la resolución que sea. Además, la biblioteca incluye un modo tanto para controlar pantallas en C como en C++.

La primera acción consiste en cargar la configuración y ejecutar los comandos para configurar la pantalla con la comunicación en serie.

6.2. Configuración inicial del conversor

La siguientes instrucciones consisten en preparar el puerto A para leer, además de configurar los flags para que lea correctamente y la conversión no se realice a una frecuencia mayor de 200 Hz, ya que esto provocaría errores de lectura.

6.3. Inicio del panel táctil

Aquí se configurarán las salidas analógicas que se utilizan para controlar el panel táctil, configurando los puertos A2 y A3 como salidas y asegurándonos que estas están a 0.

6.4. Carga de la imagen 0

Una vez esté todo iniciado se realiza una primera carga de imagen para que la pantalla no quede en blanco.

6.5. Lectura de valores de voltaje y corriente y elementos de seguridad

Como explicamos en el hardware, se realizaron 3 divisores de tensión de acuerdo con los parámetros establecidos durante la lectura de datos. Cada uno de estos dará un intervalo de 0 a 5V para sus lecturas.

Sin embargo, ¿qué pasaría si una lectura se sale de escala? Nuestros divisores están diseñados para tener margen con respecto a la lectura de valores del circuito. Si alguno de estos supera lo establecido, y las lecturas se acercan a los 5V del microcontrolador, se inicia una rutina que desconecta los relés y muestra un mensaje de error. A partir de este momento el microcontrolador quedará bloqueado hasta que se reinicie el sistema, momento en el cual volverá a hacer las lecturas y a decidir si debe continuar o no.

Por otro lado, se lee también la corriente que circula por el circuito. Para ello, valiéndonos del ACS712, leeremos un valor de voltaje proporcional a la corriente que circule. en nuestro caso, partiendo de Vcc/2, 2,5 V detectará desde −30 A a 30 A.

6.6. Control del panel táctil

6.6.1. Calibración

Existen varias maneras de calibrar una pantalla, dependiendo de la precisión y los errores que queremos evitar. Podemos optar entre 2, 3, 5 o 9 puntos, cada uno resolverá de mejor manera los errores y ofrecerá una mayor precisión [15].

Problemas en las pantallas táctiles

- **Traslación:** Se trata de que cualquier contacto en el panel se traduce en una coordenada en la pantalla desplazada, ya sea en el eje horizontal, vertical, o en ambos.
- **Rotación:** En este caso, se genera un gradiente de ángulo en cualquier entrada, provocando que, por ejemplo, al «dibujar» una serie de puntos totalmente horizontal se interprete como una línea con un ángulo θ diferente al original.
- **Escala:** El último error se debe a un fallo de linealidad en la pantalla, que puede ocurrir también en ambos ejes. Se trata de un problema que se traduce en fallos de detección no lineales, por ejemplo, si representamos un círculo con la mano, obtenemos una elipse en la lectura.

Todos estos problemas se pueden observar en la figura 6.2, En ella podemos ver cómo, una entrada circular perfecta (dibujada en rojo) obtiene una salida rotada, deformada y trasladada (en azul).

Con estos problemas en mente, debemos seleccionar el número de puntos que queremos utilizar para resolverlos. Utilizar 9 puntos garantiza una precisión que no requiere una pantalla tan pequeña, mientras que utilizar sólo dos puntos no da solución total a los problemas de escala. después de realizar pruebas con la calibración a 3 puntos, comprobamos que no es necesario complicar el algoritmo y este produce resultados con la precisión necesaria.

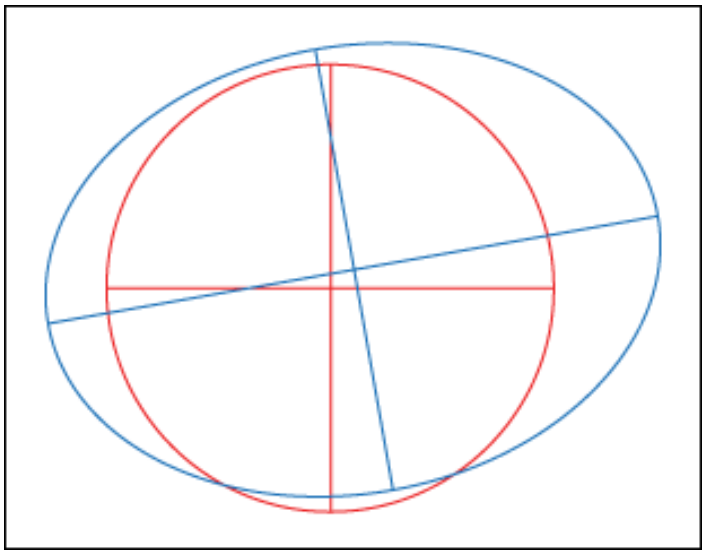


Figura 6.2 Visualización de la combinación de los problemas en pantallas táctiles

6.6.2. Diferencias cuantitativas entre los datos necesarios

Como hemos visto, debido al tamaño de la pantalla, los problemas presentes son menores y más fáciles de pasar por alto que en grandes resoluciones. Sin embargo, existe un beneficio independiente del tamaño de la pantalla que también soluciona la calibración.

Veamos un valor cualquiera en nuestra pantalla táctil tras ser leído por el ADC, En este caso, $X = 838; Y = 483$ si vemos dónde corresponde el pulso en nuestras coordenadas referenciadas a la resolución de la pantalla, tenemos, en realidad unas coordenadas aproximadas $X = 100; Y = 26$. El uso de la calibración permite que, tras realizar los cálculos, el microcontrolador traduzca los valores simples del ADC a valores similares a la resolución de la pantalla, facilitando mucho la programación de botones y menús.

6.6.3. Algoritmo para calibración de pantallas con 3 puntos

Asumiendo unas coordinadas Ideales (X_0Y_0), (X_1Y_1), (X_2Y_2), y sus correspondientes valores reales $(X'_0Y'_0)$, $(X'_1Y'_1)$, $(X'_2Y'_2)$ Las ecuaciones para los puntos obtenidos corregidos son las siguientes: Para las X:

$$\begin{cases} x_0 = KX_1x'_0 + KX_2y'_0 + KX_3 \\ X_1 = KX_1x'_1 + KX_2y'_1 + KX_3 \\ X_2 = KX_1x'_2 + KX_2y'_2 + KX_3 \end{cases} \tag{6.1}$$

Figura 6.3 Muestra de la Toma de datos por el ADC, los valores de V son, en realidad, resultados X e Y del panel táctil

Y para las Y:

$$\begin{cases} y_0 = KY_1 x'_0 + KY_2 y'_0 + KY_3 \\ y_1 = KY_1 x'_1 + KY_2 y'_1 + KY_3 \\ y_2 = KY_1 x'_2 + KY_2 y'_2 + KY_3 \end{cases} \tag{6.2}$$

Estas ecuaciones pueden ser reescritas en forma de matriz:

$$\begin{bmatrix} x'_0 & y'_0 & 1 \\ X'_1 & y'_1 & 1 \\ X'_2 & y'_2 & 1 \end{bmatrix} * \begin{bmatrix} KX'_1 \\ KX'_2 \\ KX'_3 \end{bmatrix} = \begin{bmatrix} x'_0 \\ x'_1 \\ x'_2 \end{bmatrix} \tag{6.3}$$

y

$$\begin{bmatrix} x'_0 & y'_0 & 1 \\ X'_1 & y'_1 & 1 \\ X'_2 & y'_2 & 1 \end{bmatrix} * \begin{bmatrix} KY'_1 \\ KY'_2 \\ KY'_3 \end{bmatrix} = \begin{bmatrix} y'_0 \\ y'_1 \\ y'_2 \end{bmatrix} \tag{6.4}$$

dónde las incógnitas KX_n y KY_n son parámetros de calibración que pueden ser resueltos mediante las siguientes ecuaciones:

Siendo

$$K = (x'_0 - x'_2)(y'_1 - y'_2) - (x'_1 - x'_2)(y'_0 - y'_2) \tag{6.5}$$

Entonces:

$$KX_1 = \frac{(x_0 - x_2)(y'_1 - y'_2) - (x_1 - x_2)(y'_0 - y'_2)}{K} \tag{6.6}$$

$$KX_2 = \frac{(x_1 - x_2)(x'_0 - x'_2) - (x_0 - x_2)(x'_1 - x'_2)}{K} \tag{6.7}$$

$$KX_3 = \frac{y'_0(x'_2x_1 - x'_1x_2) + y'_1(x'_0x_2 - x'_2x_0) + y'_2(x'_1x_0 - x'_0x_1)}{K} \tag{6.8}$$

$$KY_1 = \frac{(y_0 - y_2)(y'_1 - y'_2) - (y_1 - y_2)(y'_0 - y'_2)}{K} \tag{6.9}$$

$$KY_2 = \frac{(y_1 - y_2)(x'_0 - x'_2) - (y_0 - y_2)(x'_1 - x'_2)}{K} \tag{6.10}$$

$$KY_3 = \frac{y'_0(x'_2y_1 - x'_1y_2) + y'_1(x'_0y_2 - x'_2y_0) + y'_2(x'_1y_0 - x'_0y_1)}{K} \tag{6.11}$$

Calculados estos coeficientes, podemos volver a la ecuación 6.1 y 6.2 y obtener un resultado preciso y transformado a coordenadas de la pantalla de manera que sea fácil situar botones y leer las pulsaciones de pantalla sobre ellos.

6.6.4. Almacenamiento del estado de la calibración

Los parámetros obtenidos del panel táctil no pueden ser almacenados como variables normales ya que éstas se pierden al final de cada ciclo de funcionamiento, por lo que sería necesario recalibrar desde 0 en cualquier encendido. Para evitar este problema, los valores se guardarán en la memoria interna del microcontrolador.

Por otro lado, debido a que la memoria EEPROM interna del programador sólo admite números positivos, tendremos que almacenar estos parámetros en dos variables. Una variable almacenará el número en sí; la otra, será un flag que represente el si el valor es positivo o negativo.

Durante el ciclo de encendido el microcontrolador leerá las variables, comprobará si son diferentes a 0xFF, ya que es el valor con el que se almacenan de manera predeterminada, y si no coinciden con ese dato, sabrá que la calibración ya fue realizada. Existe además un flag que cambiará en este momento y que comprobará cada ciclo en caso de que el estado de la calibración haya cambiado.

Es entonces cuando se cargarán cada uno de los parámetros en variables normales, que usará el micro continuamente para verificar que valores se están utilizando.

6.6.5. Detección de las pulsaciones

Como el ADC rastrea continuamente la pantalla táctil, lo único que debemos hacer es filtrar los datos. Una vez convertidos a las coordenadas que queremos, simplemente filtraremos los botones. Por ejemplo, si nos encontramos en la pantalla de calibración y queremos detectar si se ha pulsado el botón central, sólo tenemos que poner un condicional que, si la pulsación está entre un determinado rango de X e Y, estamos tocando el botón y debemos realizar esa acción. Esto se repite para todo el menú de selección de pantallas.

6.7. Pantalla de visualización de datos

Esta pantalla realmente es una serie de imágenes que siguen un árbol de decisiones. En él se toman en cuenta los siguientes aspectos:

- Signo del amperaje que circula por el sensor de corriente.
- Estado de conexión del circuito solar y del circuito del alternador.

Con estos parámetros decidirá qué imagen enseñar, por ejemplo, si la lectura del amperaje es positiva, y el panel solar está conectado, mostrará una imagen indicando que nos encontramos cargando la batería mediante el panel solar.

6.8. Pantalla de muestra de datos

Esta es la pantalla donde se muestran todos los datos recogidos. Básicamente, mediante la librería u8g2, enviamos los parámetros leídos por el ADC a la imagen mostrándonos en tiempo real en qué situación nos encontramos.

6.9. Pantalla de control manual y automatización

La pantalla de automatización sigue un proceso muy sencillo si decidimos que queremos control manual, el circuito pasará a controlarse de otra manera, si la opción de carga desde el alternador está en «no» el microcontrolador analizará la situación, si el panel solar está disponible y funcionando, cargará del panel solar, si no, aislará la batería del resto del circuito. En caso de pulsar que cargue del alternador, ignorará el panel solar, sin embargo, no conectará el circuito a la batería principal del coche, si este no tiene el alternador funcionando ($v > 14V$).

6.10. Pantalla de calibración

El microcontrolador crea un flag de control una vez que importa los valores necesarios de calibración. Este proceso simplemente, al detectar una pulsación en el botón de calibrar en la pantalla, cambiará ese flag a 0, iniciando así el proceso de recalibrado.

Capítulo 7

Problemas y soluciones

Durante todo el proceso de diseño, implementación y producción del equipo de pruebas, surgieron una serie de problemas que tuvimos que solucionar. Tanto el problema como sus posibles soluciones se encuentran documentadas a continuación.

7.1. Hardware

7.1.1. Pantalla original dañada

Si observamos las especificaciones de nuestra placa de desarrollo, veremos que la pantalla que normalmente se utiliza es una pantalla con una resolución de 128x64 píxeles, con un chip de control Samsung KS108. Sin embargo, cuando colocamos el primer programa de visualización de imágenes, nos dimos cuenta de que la pantalla estaba dañada. Se trató de solucionar el problema, suponiendo que el fallo estaba en el programa o el microcontrolador. Finalmente, aceptando que el problema era la pantalla, se optó por comprar otra, una pantalla ST7920, cuyo funcionamiento es similar. Por otro lado, el panel táctil viene totalmente adherido a la pantalla original, por lo que se tuvo que comprar otro panel, y, debido a que el conecto no es igual, realizar algunos trabajos de soldadura para hacerlo compatible con la placa de desarrollo.

7.1.2. Sensor de efecto Hall ACS172

Como ya sabemos, nuestro sensor de efecto hall permite hasta 30A de corriente. Por nuestro estudio, siendo un consumo pico, 21 A Este sensor es correcto. Sin embargo, con una división de 66mV por amperio, sumando los problemas en la estabilidad de corriente, y que, en nuestro experimento, cargamos hasta 200 mA de corriente, la precisión con la que trabajamos se resiente. Es por ello que durante las pruebas, con amperaje 0, el circuito muestra algo más de 2 mA. La solución pasa por dos opciones, o cambiar el sensor a un ACS712ELCTR-05B-T, cuya corriente máxima es de 5 A y que ofrece 185 mV por amperio, o, sabiendo que nos encontramos con una carga

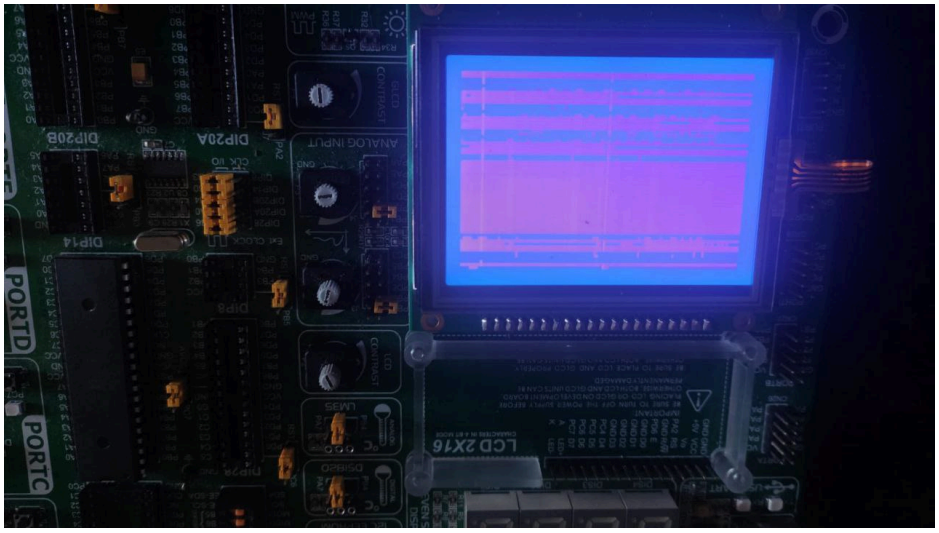


Figura 7.1 Resultados al utilizar la pantalla original

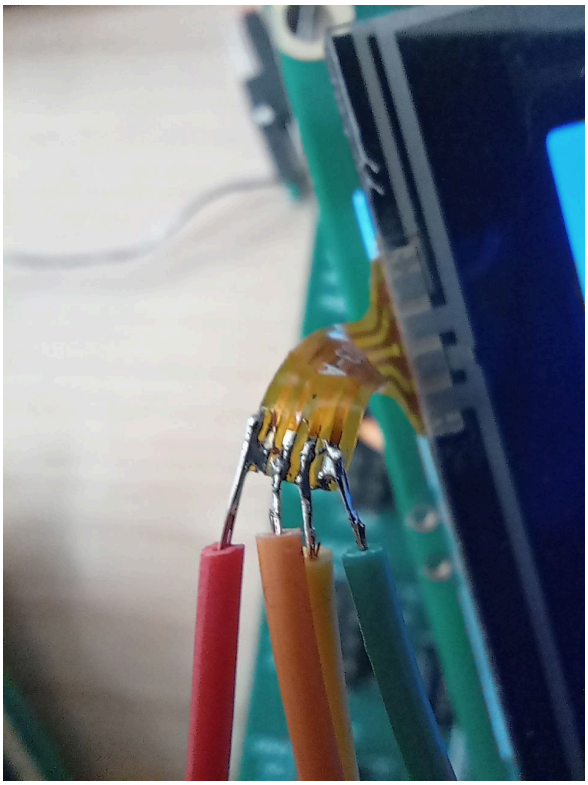


Figura 7.2 Soldadura del conector del panel táctil

tan grande, y que no es necesario tanta precisión, podemos limitar la visualización al primer decimal para evitarnos errores de lectura.

Otro problema que encontramos con este sensor es el circuito que lo acompaña. Si bien este sensor resiste hasta 30 A, el circuito en el que viene para ser conectado directamente tiene dos clavijas de conexión. Sin embargo estas clavijas no son capaces de abarcar ni 1 mm^2. Esto nos impediría realizar las pruebas con los propios paneles ya que usamos cable de 3,5 mm^2.

7.1.3. Las pantallas se encuentran invertidas

Mientras exportamos todas las pantallas a datos compatibles con el programador, nos dimos cuenta de que algunas pantallas salían invertidas al mostrarse. La solución es simple, invertir las imágenes en el editor de imágenes para que se muestren correctamente.

7.2. Software

7.2.1. Tipos de datos y su tamaño, el problema de un compilador AVR universal

Durante el proceso de desarrollo, hubo un problema que no parecía normal. En muchos cálculos, los resultados eran incorrectos. Este tema, que en principio parecía un simple tema de que se necesitaba una variable más grande para almacenar el resultado, acabó llevándonos a un problema muy arraigado en nuestro software.

Durante una conversación sobre este problema en un foro sobre compiladores y microcontroladores [16] se nos da cuenta de un fallo. En primer lugar, dos valores que caben en un unsigned integer de 8 bits, es decir, cualquier número entre 0 y 255 (00000000 - 11111111) podrían no caber en las operaciones intermedias, y volver a hacerlo en la final, es decir, (80x90)/100, cuyas variables son 80 y 90, son dos variables de 8 bits, y el resultado, 72, también lo es. Sin embargo, la operación intermedia, 80x90= 7200 no es capaz de ocupar una variable de 8 bits, y dará como resultado 255/100 = 25,5 ->25 (redondeado al no trabajar con decimales). Para esto, debemos aclarar a él compilador que los cálculos se deben hacer con una variable aún más grande. por ejemplo:

```
1  KXY01 = ((((int32_t)y_1-y_2)*((int32_t)xp_0-xp_2)-((int32_t)y_0-y_2)*((int32_t)
   xp_1-xp_2)));
2  KY2 = ((int32_t)(1000*KXY01))/k;
```

En este cálculo, aunque KY2 sea una variable pequeña, el cálculo es forzado a que se haga en un int32, para evitar que la multiplicación se salga de la capacidad de la variable.

Por otro lado, descubrimos que, si bien el programa te empuja a utilizar sus variables, int, float... Dependiendo del microcontrolador que utilices, ya sea un atmega8 o atmega32, la variable Int tomará tamaños diferentes. El estándar C solo declara cual es el tamaño mínimo de la variable, pero puede tener un tamaño mayor. Por ello, dependiendo del tipo de compilador, podemos pasar de 16 a 32 Bits usando exactamente la misma variable. Por eso, debemos siempre utilizar las variables específicas (int8_t para 8 bits, int16_t para 16...) Con esto sabremos exactamente el tamaño de la variable y que es capaz de hacer, independientemente del compilador.

Capítulo 8

Resultados obtenidos

En este capítulo mostraremos un resultado general del equipo mediante fotos tomadas en la mesa de trabajo.

8.1. Planos generales del montaje completo

Las figuras 8.1a a 8.1c muestran fotografías de cada una de las piezas que componen la mesa de pruebas.

8.2. Primera pantalla, muestra de datos visualmente

Las figuras 8.2a a 8.2e muestran las diferentes pantallas como resultado del cambio de comportamiento de las entradas; la figura 8.2f muestra el resultado de un valor excesivo en una de las entradas.

8.3. Segunda pantalla, muestra de datos numéricos

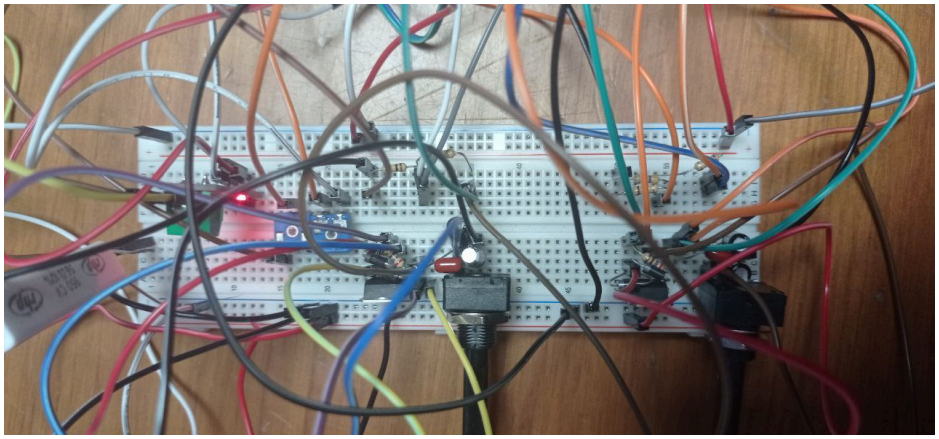
En la figura 8.3 podemos observar los datos mostrados junto a un logo identificativo para que sea más sencillo identificar cada uno de ellos.

8.4. Tercera pantalla, configuración automática

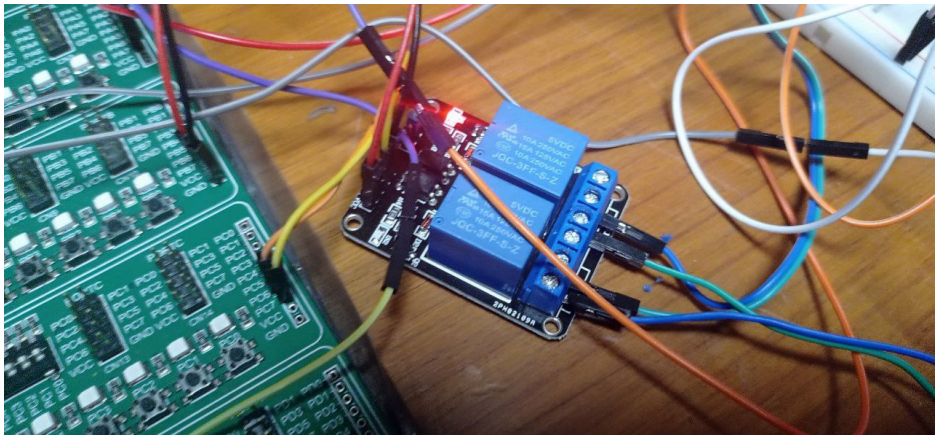
En las figuras 8.4a a 8.4c podemos observar el árbol de decisiones implementado para forzar o no la carga desde el motor manualmente.

8.5. Pantalla final de calibración

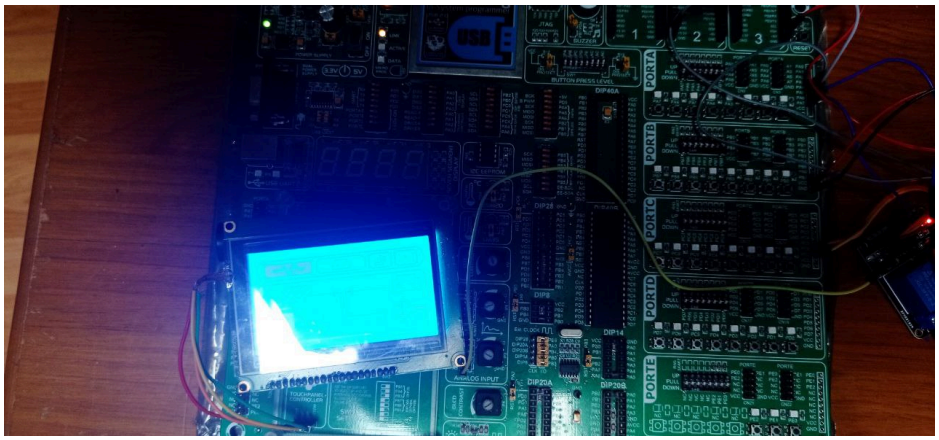
Las figuras 8.5a y 8.5b muestra la pantalla de decisión y el primer paso del proceso de calibración.



(a) Circuito de simulación de voltajes y corrientes.

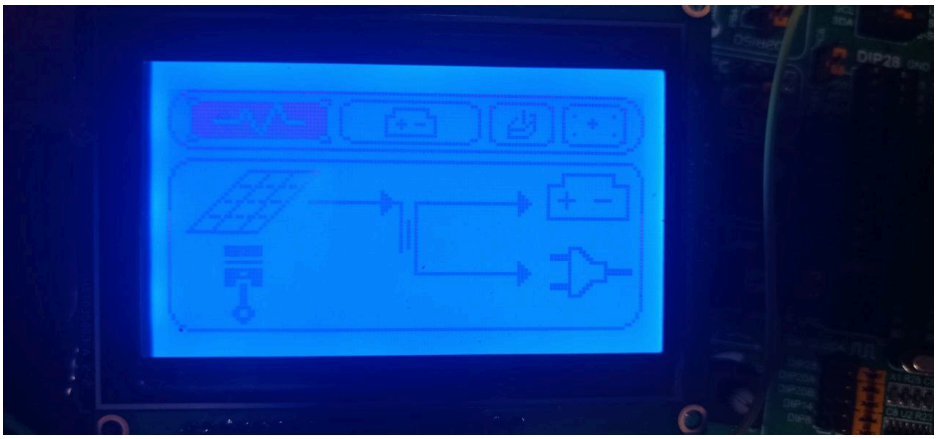


(b) Placa de control de relés.

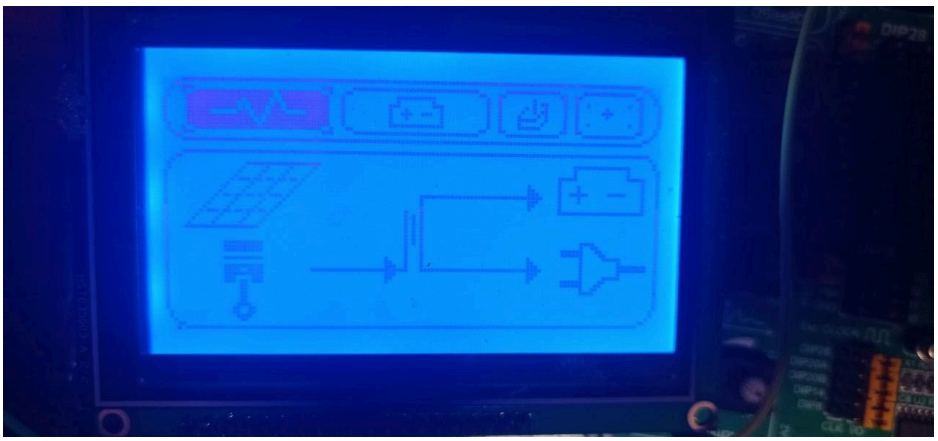


(c) Pantalla situada en la placa de desarrollo.

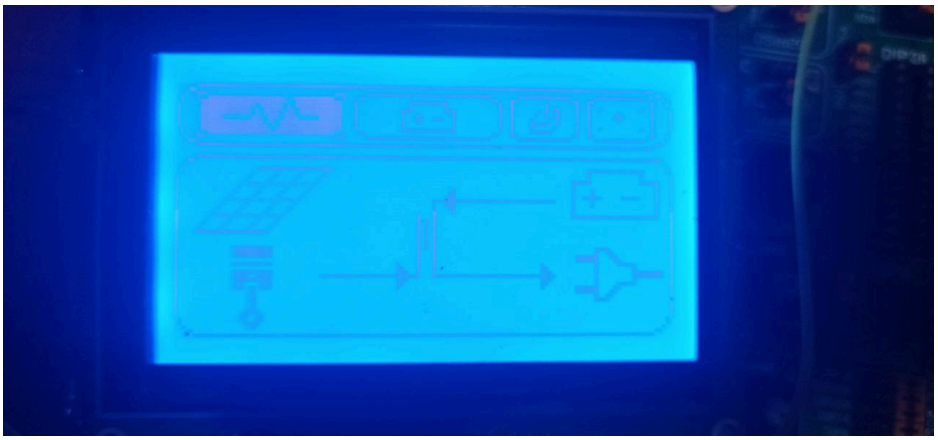
Figura 8.1 Planos generales del montaje completo



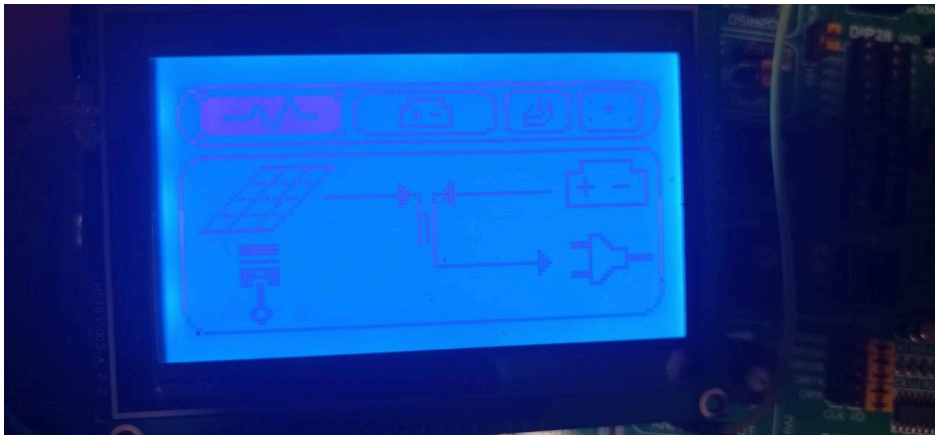
(a) Circuito cargando mediante el panel solar



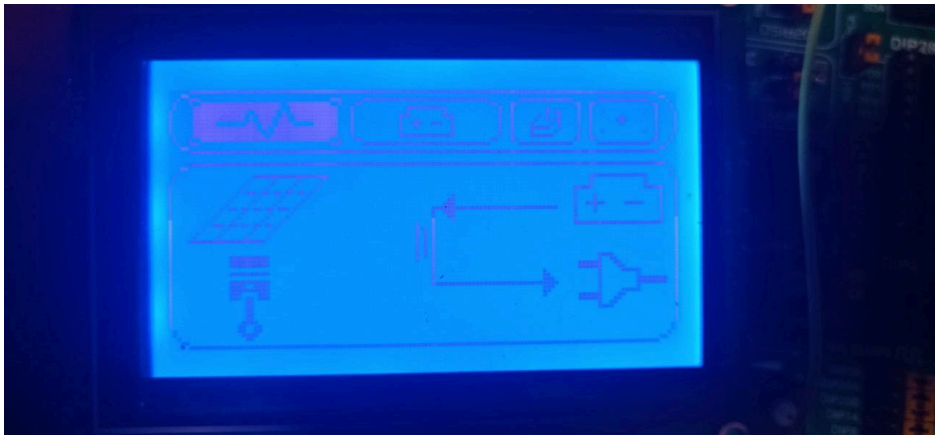
(b) Circuito cargando mediante el alternador



(c) Circuito descargando de batería y alternador



(d) Descarga mediante panel solar y batería



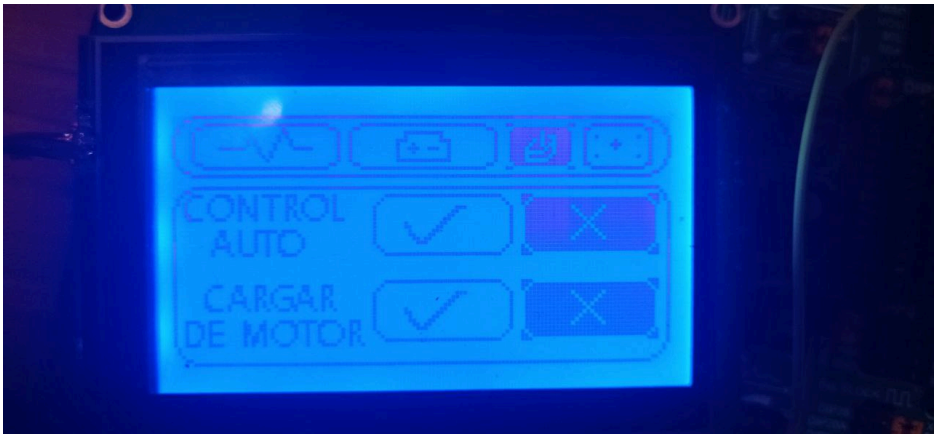
(e) Circuito aislado, trabajo mediante batería únicamente

(f) Lectura fuera de rango en uno de los valores

Figura 8.2 Primera pantalla, muestra de datos visualmente y pantalla de error de rango

Figura 8.3 Segunda pantalla, muestra de datos numéricos

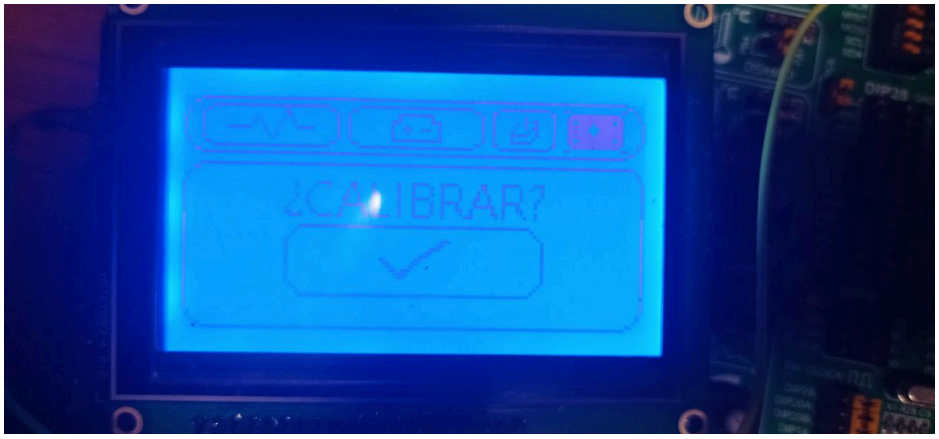
(a) Control automático On.



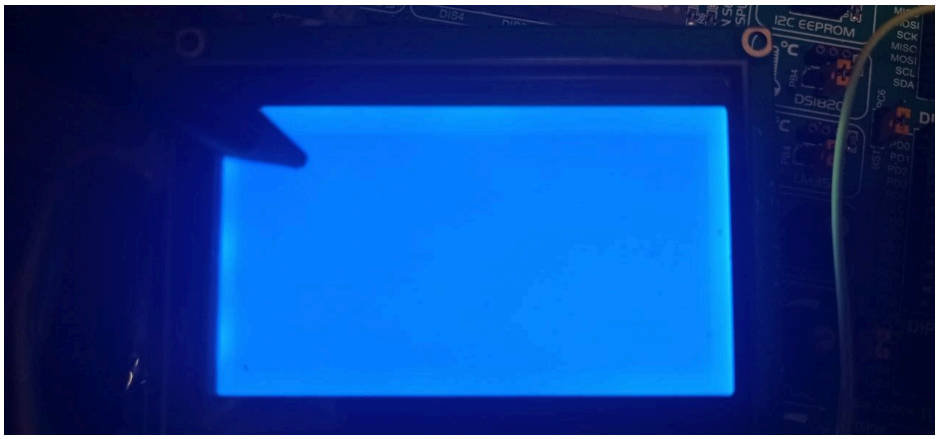
(b) Control automático Off.

(c) Forzada la carga automática

Figura 8.4 Tercera pantalla, configuración automática



(a) Esperando la entrada para calibrar



(b) Primer paso de la calibración

Figura 8.5 Pantalla final de calibración

Capítulo 9

Presupuesto

Este presupuesto se dividirá en varias partes, El precio de la instalación fotovoltaica, el precio del equipo del microcontrolador y el coste del tiempo de programación. Esta instalación incluye los electrodomésticos y accesorios calculados en su consumo. y las dos opciones de instalación, en techo o en montaje externo. El presupuesto final incluye el precio más alto de las dos opciones.

9.1. Instalación solar fotovoltaica

9.1.1. Montaje en techo

Presupuesto Para montaje en techo			
Producto	Precio	Uds.	Total
Bomba de agua 50W, 12V	41,9 €	1	41,9 €
Nevera Adventurer 50 Litre	490 €	1	490 €
Lampara con 24 Led 12V	9,9 €	4	39,6 €
Baseus Cargador de Coche USB 65W	20 €	1	20 €
QYD 65W USB-C 12V	20 €	1	20 €
Panel Fotovoltaico XUNZEL 120W	130 €	3	390 €
Batería solar de gel U-POWER 12V 100Ah	180 €	1	180 €
SmartSolar VICTRON MPPT 100/30	230€	1	230 €
Bobina de cable negro/Rojo, 20 m, 10 mm10mm	40 €	2	80 €
Bobina de cable negro/Rojo, 50 m, 2,5 mm2,5mm	25 €	1	25 €
Protecciones, Fusible 10 A	8,28 €	3	24,84 €
Protecciones, magnetotermico Schneider 500 V, 10 A	28,95 €	2	57,9 €
Protecciones, magnetotermico Schneider 500 V, 16 A	48,50 €	1	48,5 €
Herrajes y anclajes a Techo	19 €	3	57 €
	Suma:		1704,54 €
	Impuestos:	7 %	119,33 €
	Total:		1824,08 €

9.1.2. Montaje desplegable in situ

Presupuesto para sistema desplegable			
Producto	Precio	Uds.	Total
Bomba de agua 50W, 12V	41,9 €	1	41,9 €
Nevera Adventurer 50 Litre	490 €	1	490 €
Lampara con 24 Led 12V	9,9 €	4	39,6 €
Baseus Cargador de Coche USB 65W	20 €	1	20 €
QYD 65W USB-C 12V	20 €	1	20 €
Panel Fotovoltaico XUNZEL 120W	130 €	3	390 €
Batería solar de gel U-POWER 12V 100Ah	180 €	1	180 €
SmartSolar VICTRON MPPT 100/30	230€	1	230 €
Bobina de cable negro/Rojo, 20 m, 10 mm10mm	40 €	2	80 €
Bobina de cable negro/Rojo, 50 m, 2,5 mm2,5mm	25 €	1	25 €
Protecciones, Fusible 10 A	8,28 €	3	24,84 €
Protecciones, magnetotermico Schneider 500 V, 10 A	28,95 €	2	57,9 €
Protecciones, magnetotermico Schneider 500 V, 16 A	48,50 €	1	48,5 €
Soporte ajustable aluminio	49 €	3	147 €
Nivelador ajustable Acero	2,03 €	12	24,36 €
	Suma:		1819,1 €
	Impuestos:	7 %	127,34 €
	Total:		1946,44 €

9.2. Circuito de control

El circuito de control tiene dos secciones, la ocupada por el coste del producto en sí, si este fuera a terminarse instalando en una PCB con su circuitería e instalada en una furgoneta; y la que ocupa el coste de programación del producto para llegar al estado en el que se encuentra ahora mismo.

9.2.1. Coste del producto

El precio a continuación comprende 10 circuitos en total, montados y entregados además de un programador y su conector correspondiente en cada placa para programarlo.

Presupuesto Para fabricación de 10 unidades			
Producto	Precio	Uds.	Total
Componentes electrónicos PCB	57,30 €	1	57,30 €
Sensor de corriente ACS712	3,12 €	10	31,2 €
Programador USB ISP para placa ATMEL AVR	10,77 €	1	10,77 €
Pantalla ST7920	12,1 €	10	121 €
Panel resistivo	1,71 €	10	17,1 €
ATmega32	3,5 €	10	35 €
Fabricación y servicio de montaje PCB 10Ud	387 €	1	387 €
Relé automoción 12V 100A	14 €	20	280 €
	Suma:		939,37 €
	Impuestos:	7 %	65.76 €
	Total:		1005,13 €

Esto no deja un coste individual por equipo de 100,51€ sin incluir una carcasa protectora del equipo y el cableado necesario que dependerá del vehículo y la posición del sistema.

9.2.2. Programación

A continuación, se definen los costes por el desarrollo del software hasta el estado en el que se encuentra en este momento.

Presupuesto del desarrollo del programa			
	Precio/H	Horas	Total
Planteamiento y análisis	40 €	50	2000 €
Programación	50 €	160	8000 €
Implementación	35 €	20	700 €
Documentación	40 €	40	1600 €
	Suma:		12300 €
	Impuestos:	7 %	861 €
	Total:		13161 €

9.3. Instalación en Furgoneta

Finalmente, se calcula la estimación de las horas para un técnico instalando el equipo y comprobando su funcionamiento en una furgoneta, incluyendo la instalación del equipo solar.

Presupuesto Instalación en furgoneta			
	Precio/H	Horas	Total
Instalación equipo fotovoltaico	35 €	8	280 €
Instalación equipo de control	35 €	6	210 €
Pruebas de funcionamiento	35 €	2	70 €
	Suma:		560 €
	Impuestos:	7 %	39,2 €
	Total:		599,2 €

9.4. Coste total del proyecto

El coste total del proyecto, contando con instalación del equipo fotovoltaico, desarrollo de la aplicación y fabricación y montaje del mismo es el siguiente:

Presupuesto Instalación en furgoneta	
	Precio
Total costes mano de obra (MO)	13760,2 €
Total costes materiales (MT)	2951,57 €
Gastos generales: 6 % (MO+MT)	985,34 €
Suma:	17697,11 €
Impuestos: 7 %	1238,8 €
Total:	18935,91 €

Capítulo 10

Conclusion

There are some aspects of this installation that could be optimised, for example, the measurement of the ACS712 has to be done several times and averaged to make sure it is correct. This creates perceptible delays in the tactile response. Similarly, the graphical information display often takes several cycles to detect whether the battery is in a charged or discharged state.

On the other hand, our system observes and displays the raw data. In the case of a PWM converter like the one we are using, ideally we would take measurements, check the PWM cycle with interrupts and get the effective voltage and not just the peak voltage. However, the response of the device is already slow enough and the program would require an in-depth optimisation to be able to store large amounts of data in an already limited memory, as shown in Figure 10.1.

Despite this, the device detects and switches between sources effectively, and auto-protects the system as soon as it detects a failure. In general, the logos are clearly visible and recognisable as to the function they perform, and the numbers are clear even at medium distances.

We can therefore say that the system meets all the requirements of functionality and usability for the purpose for which it was created.

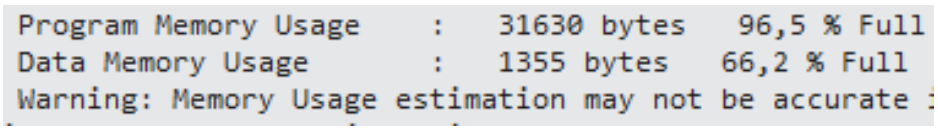


Figura 10.1 Estimated memory usage after final compilation

Bibliografía

[1] Demand.io, "The History Of The VHS Movie Industry," https://knoji.com/article/the-history-of-the-vhs-movie-industry/, February 2021, [Online; accedido 10-Agosto-2022].

[2] A. Trevennor, *A Brief History of Microcontrollers*. Berkeley, CA: Apress, 2012, pp. 3–11. [Online]. Available: https://doi.org/10.1007/978-1-4302-4447-9_1

[3] AMD, "Amd ryzen™ 5 7600x specifications," https://www.amd.com/en/products/cpu/amd-ryzen-5-7600x, August 2022, [Online; accedido 10-Septiembre-2022].

[4] A. Holovatyy, V. Teslyuk, N. Kryvinska, and A. Kazarian, "Development of microcontroller-based system for background radiation monitoring," *Sensors*, vol. 20, no. 24, p. 7322, 2020.

[5] J. E. Endelman, "Home on the road: The motor home in america," 2001.

[6] P. team, "Photovoltaic geographical information system," https://re.jrc.ec.europa.eu/pvg_tools/en/tools.html, [Recurso online, accedido: 07-Septiembre-2022].

[7] G. Software, "GIMP," https://www.gimp.org/, 1995, [Online; accedido 07-Septiembre-2022].

[8] G. G. P. License, "GNU GENERAL PUBLIC LICENSE," https://www.gimp.org/about/COPYING, June 2007, [Online; accedido 19-Agosto-2022].

[9] G. Software, "About GIMP," https://www.gimp.org/about/, 1995, [Online; accedido 13-Abril-2022].

[10] S. L. C. Software, "Creating monochrome bitmap files for LCD/GLIB using GIMP," https://community.silabs.com/s/article/creating-monochrome-bitmap-files-for-lcd-glib-using-gimp?language=en_US, 2021, [Online; accedido 1-Mayo-2022].

[11] M. Software, "¿Qué es Multisim™?" https://www.ni.com/es-es.html, 2022, [Online; accedido 07-Septiembre-2022].

[12] M. S. Software, "Microchip Studio for AVR® and SAM Devices," https://www.microchip.com/en-us/tools-resources/develop/microchip-studio#, 1998, [Online; accedido 07-Septiembre-2022].

[13] A. Software, "AVRFlash," https://download.mikroe.com/documents/full-featured-boards/easy/easyavr-v6/avrflash-manual-v101.pdf, 2022, [Online; accedido 07-Septiembre-2022].

[14] oliver olikraus, "u8g2," https://github.com/olikraus/u8g2, 2017.

[15] N. Jia, "Mmse-based multipoint calibration algorithm for touch screen applications," *Analog Devices*, 2009, [Online; accedido 10-Septiembre-2022].

[16] M. T. Inc., "Web avrfreaks," https://www.avrfreaks.net/forum/avr-arithmetics-what-im-not-getting, 2022, [Discusión online, accedido por última vez: 04-Septiembre-2022].

Apéndice A

Códigos

A.1. Código Main

```
/*
  main.c
  final Degree task project
  This Code will display a small information suit that gathers from a solar panel
    system
  Designed to be used with a 2 panels circuit + a single AGM battery, this system is
    now
  modified to read from a protoboard that 'mocks' a suit like that previously studied
    .
  current limitations in this setup is that we dont have a 40V power supply so the
    solar
  grid is limited to 22V, also, pwm reads are not getting mods so its reading average
  but direct current readings. The project uses a 4 Wire SW SPI to use the display:
  Universal 8bit Graphics Library (https://github.com/olikraus/u8g2/)
  Appart from that, all code has been produced by the author with help from its tutor
  and online help from the forums StackOverflow (https://stackoverflow.com/) and
    avrgeeks
  (https://avrgeeks.com/) and other online resources

*/

#include <VA_default.h>
#include <avr/io.h>
#include <u8g2.h>
#include <util/delay.h>
#include <stdio.h>
#include <stdlib.h>
#include <avr/eeprom.h>
#include <math.h>
#include <avr/sfr_defs.h>

```

```
27
28
29
30
31 // CPU frequency is already set on project compiler symbols
32 //Ports directions and Pins for the GLCD SPI.
33
34 #define DISPLAY_CLK_DIR DDRB
35 #define DISPLAY_CLK_PORT PORTB
36 #define DISPLAY_CLK_PIN 5
37
38 #define DISPLAY_DATA_DIR DDRB
39 #define DISPLAY_DATA_PORT PORTB
40 #define DISPLAY_DATA_PIN 3
41
42 #define DISPLAY_CS_DIR DDRB
43 #define DISPLAY_CS_PORT PORTB
44 #define DISPLAY_CS_PIN 2
45
46 #define DISPLAY_DC_DIR DDRB
47 #define DISPLAY_DC_PORT PORTB
48 #define DISPLAY_DC_PIN 1
49
50 #define DISPLAY_RESET_DIR DDRB
51 #define DISPLAY_RESET_PORT PORTB
52 #define DISPLAY_RESET_PIN 0
53
54 // Directions for Touchscreen pins
55
56 #define Drive_PORT DDRA
57 #define DriveA_dir 2
58 #define DriveB_dir 3
59 #define TCH_Read_X 2
60 #define TCH_Read_Y 3
61
62 int TCH; // to be used in the touchscreen function
63 uint8_t calibrationOK;
64 uint8_t Touchcalibration ;
65
66
67
68 int32_t k; //should be on calibration screen only, but needed for debug
69 int32_t k0;
70 int32_t k1;
71 int32_t k2;
72 int32_t k3;
73 int32_t k4;
74 int32_t k5;
```

```
75 int32_t KX2;
76 int32_t KX1;
77 int32_t KX3;
78 int32_t KY2;
79 int32_t KY1;
80 int32_t KY3;
81 //Definite Values
82 int16_t KX12 = 0;
83 int16_t KX22 = 0;
84 int16_t KX32 = 0;
85
86 int16_t KY12 = 0;
87 int16_t KY22 = 0;
88 int16_t KY32 = 0;
89 uint8_t coordupdated = 0;
90
91 uint8_t flag_pantalla;
92 uint8_t status = 99;
93 int8_t manual_control = 0;
94 uint8_t alternatorcharge = 0;
95
96
97 int x_0 = 16;
98 int x_1 = 112;
99 int x_2 = 64;
100 int y_0 = 8;
101 int y_1 = 32;
102 int y_2 = 56;
103
104 uint16_t xp_0;
105 uint16_t yp_0;
106 uint16_t xp_1;
107 uint16_t yp_1;
108 uint16_t xp_2;
109 uint16_t yp_2;
110
111 uint16_t Battery_read;
112 uint16_t Battery_units;
113 uint16_t Battery_units2;
114 uint16_t Battery_decimals;
115
116 uint16_t Solarpanel_read;
117 uint16_t Solarpanel_units;
118 uint16_t Solarpanel_units2;
119 uint16_t Solarpanel_decimals;
120
121 uint16_t alternator_read;
122 uint16_t alternator_units;
```

```
123 uint16_t alternator_units2;
124 uint16_t alternator_decimals;
125
126
127 float AcsValue=0.0,Samples=0.0,AvgAcs=0.0,AcsValueF=0.0;
128
129 #define P_CPU_NS (1000000000UL / F_CPU)
130
131 //touchscreen initialize
132
133
134 void Start_Touchscreen()
135 {
136   //Set DRIVEA and DRIVEB as outputs:
137   DDRA = 0xC;
138   PORTA = 0x00; //make sure everything is set to low
139 }
140
141
142
143 //ADC initialize
144 void Start_Adc()
145 {
146
147   DDRA=0x0;      /* Make all ADC ports as input */
148   ADCSRA = 0x86;       /* Enable ADC, fr/64  */
149   ADMUX = 0x40;      /* Vref: Avcc, ADC channel: 0 */
150
151 }
152
153 int ADC_Read(char channel)
154 {
155   int Ain,AinLow;
156   ADMUX = 0x40;            //resets the admux
157   ADMUX = ADMUX|(channel & 0x0f);    /* Set input channel to read */
158
159
160   ADCSRA |= (1<<ADSC);         /* Start conversion */
161
162   while((ADCSRA&(1<<ADIF))==0);    /* Monitor end of conversion interrupt */
163
164   _delay_us(20);
165   AinLow = (int)ADCL;          /* Read lower byte*/
166   Ain = (int)ADCH*256;         /* Read higher 2 bits and
167                     Multiply with weight */
168   Ain = Ain + AinLow;
169
170   return(Ain);             /* Return digital value*/
```

```
171 }
172
173 //Touchscreen function
174
175
176
177 int Read_touchscreen(int coord)
178 {
179
180   if (coord == 0)
181   {
182     PORTA |= (1<<DriveA_dir);
183     PORTA &= ~(1<<DriveB_dir);
184     _delay_ms(30);
185
186     TCH = ADC_Read(coord);
187
188   }
189   if (coord == 1)
190   {
191
192     PORTA &= ~(1<<DriveA_dir);
193     PORTA |= (1<<DriveB_dir);
194     _delay_ms(30);
195
196     TCH = ADC_Read(coord);
197
198   }
199
200   return(TCH);
201 }
202
203 int Receive_coordinate(uint8_t coord)
204 {
205
206   uint16_t Fcoord = 0;
207
208   uint16_t ADCCoordx = Read_touchscreen(0);
209   uint16_t ADCCoordy = Read_touchscreen(1);
210   if (ADCCoordx <= 10 && ADCCoordy <= 10)
211   {
212     return(0);
213   }
214   if (coord == 0)
215   {
216     int32_t Fcoord0 = ((int32_t)KX12*ADCCoordx);
217     Fcoord0 = ((int32_t)Fcoord0/1000);
218     int32_t Fcoord1 = ((int32_t)KX22*ADCCoordy);
```

```
219     Fcoord1 = ((int32_t)Fcoord1/10000);
220     int32_t Fcoord3 = ((int32_t)KX32);
221     Fcoord = Fcoord0 + Fcoord1 + Fcoord3;
222   }
223   else
224   {
225     Fcoord = (((int32_t)KY12*ADCCoordx)/10000) +  (((int32_t)KY22*ADCCoordy)/1000) +
       ((int32_t)KY32);
226   }
227
228   return(Fcoord);
229
230 }
231 void set_ports()
232 {
233   DDRC = 0b11111111;
234   PORTC = 0xff;
235 }
236
237
238 void coordsfromEEPROM()
239 {
240
241         uint16_t KX11 = eeprom_read_word((uint16_t*)4);
242         eeprom_busy_wait();
243         uint8_t KN = eeprom_read_byte((uint8_t*)6);
244         if (KN == 1)
245         {
246           KX12 = KX11;
247           KX12 *= -1;
248         }
249         else
250         {
251           KX12 = KX11;
252         }
253
254         uint16_t KX21 = eeprom_read_word((uint16_t*)7);
255         eeprom_busy_wait();
256         KN = eeprom_read_byte((uint8_t*)9);
257         if (KN == 1)
258         {
259           KX22 = KX21;
260           KX22 *= -1;
261         }
262         else
263         {
264           KX22 = KX21;
265         }
```

```
266
267         uint16_t KX31 = eeprom_read_word((uint16_t*)10);
268         eeprom_busy_wait();
269         KN = eeprom_read_byte((uint8_t*)12);
270         if (KN == 1)
271         {
272           KX32 = KX31;
273           KX32 *= -1;
274         }
275         else
276         {
277           KX32 = KX31;
278         }
279
280         uint16_t KY11 = eeprom_read_word((uint16_t*)13);
281         eeprom_busy_wait();
282         KN = eeprom_read_byte((uint8_t*)15);
283         if (KN == 1)
284         {
285           KY12 = KY11;
286           KY12 *= -1;
287         }
288         else
289         {
290           KY12 = KY11;
291         }
292
293         uint16_t KY21 = eeprom_read_word((uint16_t*)16);
294         eeprom_busy_wait();
295         KN = eeprom_read_byte((uint8_t*)18);
296         if (KN == 1)
297         {
298           KY22 = KY21;
299           KY22 *= -1;
300         }
301         else
302         {
303           KY22 = KY21;
304         }
305
306
307         uint16_t KY31 = eeprom_read_word((uint16_t*)19);
308         eeprom_busy_wait();
309         KN = eeprom_read_byte((uint8_t*)21);
310         if (KN == 1)
311         {
312           KY32 = KY31;
313           KY32 *= -1;
```

```
314         }
315         else
316         {
317           KY32 = KY31;
318         }
319 }
320
321
322
323
324 u8g2_t u8g2;
325
326 uint8_t u8x8_avr_delay(u8x8_t *u8x8, uint8_t msg, uint8_t arg_int, void *arg_ptr)
327 {
328   uint8_t cycles;
329
330   switch(msg)
331   {
332     case U8X8_MSG_DELAY_NANO:      // delay arg_int * 1 nano second
333       // At 20Mhz, each cycle is 50ns, the call itself is slower.
334       break;
335     case U8X8_MSG_DELAY_100NANO:     // delay arg_int * 100 nano seconds
336       // Approximate best case values...
337 #define CALL_CYCLES 26UL
338 #define CALC_CYCLES 4UL
339 #define RETURN_CYCLES 4UL
340 #define CYCLES_PER_LOOP 4UL
341
342       cycles = (100UL * arg_int) / (P_CPU_NS * CYCLES_PER_LOOP);
343
344       if(cycles > CALL_CYCLES + RETURN_CYCLES + CALC_CYCLES)
345         break;
346
347       __asm__ __volatile__ (
348       "1: sbiw %0,1" "\n\t" // 2 cycles
349       "brne 1b" : "=w" (cycles) : "0" (cycles) // 2 cycles
350       );
351       break;
352     case U8X8_MSG_DELAY_10MICRO:     // delay arg_int * 10 micro seconds
353       for(int i=0 ; i < arg_int ; i++)
354         _delay_us(10);
355       break;
356     case U8X8_MSG_DELAY_MILLI:       // delay arg_int * 1 milli second
357       for(int i=0 ; i < arg_int ; i++)
358         _delay_ms(1);
359       break;
360     default:
361       return 0;
```

```
362   }
363   return 1;
364 }
365 
366 
367 uint8_t u8x8_avr_gpio_and_delay(u8x8_t *u8x8, uint8_t msg, uint8_t arg_int, void *
      arg_ptr)
368 {
369   // Re-use library for delays
370 
371   switch(msg)
372   {
373     case U8X8_MSG_GPIO_AND_DELAY_INIT:  // called once during init phase of u8g2/u8x8
374       DISPLAY_CLK_DIR |= 1<<DISPLAY_CLK_PIN;
375       DISPLAY_DATA_DIR |= 1<<DISPLAY_DATA_PIN;
376       DISPLAY_CS_DIR |= 1<<DISPLAY_CS_PIN;
377       DISPLAY_DC_DIR |= 1<<DISPLAY_DC_PIN;
378       DISPLAY_RESET_DIR |= 1<<DISPLAY_RESET_PIN;
379       break;                // can be used to setup pins
380     case U8X8_MSG_GPIO_SPI_CLOCK:         // Clock pin: Output level in arg_int
381       if(arg_int)
382         DISPLAY_CLK_PORT |= (1<<DISPLAY_CLK_PIN);
383       else
384         DISPLAY_CLK_PORT &= ~(1<<DISPLAY_CLK_PIN);
385       break;
386     case U8X8_MSG_GPIO_SPI_DATA:          // MOSI pin: Output level in arg_int
387       if(arg_int)
388         DISPLAY_DATA_PORT |= (1<<DISPLAY_DATA_PIN);
389       else
390         DISPLAY_DATA_PORT &= ~(1<<DISPLAY_DATA_PIN);
391       break;
392     case U8X8_MSG_GPIO_CS:          // CS (chip select) pin: Output level in arg_int
393       if(arg_int)
394         DISPLAY_CS_PORT |= (1<<DISPLAY_CS_PIN);
395       else
396         DISPLAY_CS_PORT &= ~(1<<DISPLAY_CS_PIN);
397       break;
398     case U8X8_MSG_GPIO_DC:          // DC (data/cmd, A0, register select) pin: Output
      level in arg_int
399       if(arg_int)
400         DISPLAY_DC_PORT |= (1<<DISPLAY_DC_PIN);
401       else
402         DISPLAY_DC_PORT &= ~(1<<DISPLAY_DC_PIN);
403       break;
404 
405     case U8X8_MSG_GPIO_RESET:       // Reset pin: Output level in arg_int
406       if(arg_int)
407         DISPLAY_RESET_PORT |= (1<<DISPLAY_RESET_PIN);
```

```
408       else
409         DISPLAY_RESET_PORT &= ~(1<<DISPLAY_RESET_PIN);
410       break;
411     default:
412       if (u8x8_avr_delay(u8x8, msg, arg_int, arg_ptr))  // check for any delay msgs
413         return 1;
414       u8x8_SetGPIOResult(u8x8, 1);        // default return value
415       break;
416   }
417   return 1;
418 }
419 //calibration process:
420 //Values to be predefined (pixels)
421
422 // moved to general so it can be used forward
423
424
425 int Calibrate_TP()
426
427 {
428   int Fcoords = 0;
429   int Scoords = 0;
430   int Tcoords = 0;
431
432   u8g2_ClearBuffer(&u8g2);
433   _delay_ms(1000);
434   u8g2_DrawPixel(&u8g2,x_0,y_0);
435   u8g2_SendBuffer(&u8g2);
436
437   int readok = 0;
438
439   while (readok != 1)
440   {
441     _delay_ms(100);
442     xp_0 = Read_touchscreen(0);
443     _delay_ms(100);
444     yp_0 = Read_touchscreen(1);
445     _delay_ms(30);
446
447
448
449     if (xp_0 >= 30 && yp_0 >= 30)
450     {
451       readok = 1;
452       Fcoords = 1;
453     }
454   }
455
```

```
456   u8g2_ClearBuffer(&u8g2);
457   _delay_ms(1000);
458   u8g2_DrawPixel(&u8g2,x_1,y_1);
459   u8g2_SendBuffer(&u8g2);
460
461   readok = 0;
462
463   while (readok != 1)
464   {
465
466     _delay_ms(100);
467     xp_1 = Read_touchscreen(0);
468     _delay_ms(100);
469     yp_1 = Read_touchscreen(1);
470     _delay_ms(30);
471
472
473
474     if (xp_1 >= 30 && yp_1 >= 30)
475     {
476       readok = 1;
477       Scoords =1;
478     }
479   }
480
481   u8g2_ClearBuffer(&u8g2);
482   _delay_ms(1000);
483   u8g2_DrawPixel(&u8g2,x_2,y_2);
484   u8g2_SendBuffer(&u8g2);
485
486   readok = 0;
487
488   while (readok != 1)
489   {
490     _delay_ms(100);
491     xp_2 = Read_touchscreen(0);
492     _delay_ms(100);
493     yp_2 = Read_touchscreen(1);
494     _delay_ms(30);
495
496
497
498     if (xp_2 >= 30 && yp_2 >= 30)
499     {
500       readok = 1;
501       Tcoords =1;
502     }
503   }
```

```
504   if (Fcoords == 1 && Scoords == 1 && Tcoords == 1)
505   {
506     Fcoords = 0;
507     Scoords = 0;
508     Tcoords = 0;
509     readok = 1;
510 
511     k0 = ((int32_t)xp_0-xp_2);
512     k1 = yp_1-yp_2;
513     k2 = xp_1-xp_2;
514     k3 = yp_0-yp_2;
515 
516     k4 = (int32_t)((int32_t)k0*k1);
517     k5 = (uint32_t)((uint32_t)k2*k3);
518     k  = (int32_t)((int32_t)k4-k5); //((xp_0-xp_2)*(yp_1-yp_2)-(xp_1-xp_2)*(yp_0-yp_2
      ));
519 
520     // calculating KX parameters pending storing it on RAM
521 
522     int32_t KXY01 = ((((int32_t)x_0-x_2)*((int32_t)yp_1-yp_2)-((int32_t)x_1-x_2)*((
      int32_t)yp_0-yp_2)));
523     KX1 = ((int32_t)(1000*KXY01))/k;
524 
525     KXY01 = ((((int32_t)x_1-x_2)*((int32_t)xp_0-xp_2)-((int32_t)x_0-x_2)*((int32_t)
      xp_1-xp_2)));
526     KX2 = ((int32_t)(10000*KXY01))/k;
527 
528     KXY01 = ((int32_t)yp_0*(((int32_t)xp_2*x_1)-((int32_t)xp_1*x_2)));
529     KXY01 = ((int32_t)KXY01+((int32_t)yp_1*(((int32_t)xp_0*x_2)-((int32_t)xp_2*x_0)))
      );
530     KXY01 = ((int32_t)KXY01+((int32_t)yp_2*(((int32_t)xp_1*x_0)-((int32_t)xp_0*x_1)))
      );
531     KX3 = ((int32_t)KXY01/k);
532 
533     KXY01 = ((((int32_t)y_0-y_2)*((int32_t)yp_1-yp_2)-((int32_t)y_1-y_2)*((int32_t)
      yp_0-yp_2)));
534     KY1 = ((int32_t)(10000*KXY01))/k;
535 
536     KXY01 = ((((int32_t)y_1-y_2)*((int32_t)xp_0-xp_2)-((int32_t)y_0-y_2)*((int32_t)
      xp_1-xp_2)));
537     KY2 = ((int32_t)(1000*KXY01))/k;
538 
539     KXY01 = ((int32_t)yp_0*(((int32_t)xp_2*y_1)-((int32_t)xp_1*y_2)));
540     KXY01 = ((int32_t)KXY01+((int32_t)yp_1*(((int32_t)xp_0*y_2)-((int32_t)xp_2*y_0)))
      );
541     KXY01 = ((int32_t)KXY01+((int32_t)yp_2*(((int32_t)xp_1*y_0)-((int32_t)xp_0*y_1)))
      );
542     KY3 = ((int32_t)KXY01/k);
```

```
543
544     //So now we have 6 data that should be stored in EEPROM, and one Byte that will
        store a 1 if its negative
545     //are negative or positive. At the start of the program first time, it should be
        calibrated, store it on RAM
546     //and retrieve it to be used it continuosly.
547
548 uint16_t KX11;
549 uint8_t KX21;
550 uint8_t KX31;
551 uint8_t KN;
552
553     if (KX1 >= 0)
554     {
555       KN = 0;
556       eeprom_update_byte((uint8_t*)6, KN);
557       KX11 = KX1;
558     }
559     else
560     {
561       KN = 1; //1 for negative value
562       eeprom_update_byte((uint8_t*)6, KN);
563       KX11 = ((int32_t)KX1*(-1)); //i should test this
564     }
565     eeprom_update_word((uint16_t*)4, KX11);
566
567     if (KX2 >= 0)
568     {
569       uint8_t KN = 0;
570       eeprom_update_byte((uint8_t*)9, KN);
571       KX21 = KX2;
572     }
573     else
574     {
575       uint8_t KN = 1; //1 for negative value
576       eeprom_update_byte((uint8_t*)9, KN);
577       KX21 = ((int32_t)KX2*(-1)); //i should test this... Yup it works!
578     }
579     eeprom_update_word((uint16_t*)7, KX21);    //this could be a byte, but i should
        cover because i may be too close to overflow it
580
581     if (KX3 >= 0)
582     {
583       uint8_t KN = 0;
584       eeprom_update_byte((uint8_t*)12, KN);
585       KX31 = KX3;
586     }
587     else
```

```
588     {
589       uint8_t KN = 1; //1 for negative value
590       eeprom_update_byte((uint8_t*)12, KN);
591       KX31 = ((int32_t)KX3*(-1)); //i should test this... Yup it works!
592     }
593     eeprom_update_word((uint16_t*)10, KX31);     //this could be a byte, but i should
        cover because i may be too close to overflow it
594
595
596 uint16_t KY11;
597 uint8_t KY21;
598 uint8_t KY31;
599     if (KY1 >= 0)
600     {
601       KN = 0;
602       eeprom_update_byte((uint8_t*)15, KN);
603       KY11 = KY1;
604     }
605     else
606     {
607       KN = 1; //1 for negative value
608       eeprom_update_byte((uint8_t*)15, KN);
609       KY11 = ((int32_t)KY1*(-1)); //i should test this
610     }
611     eeprom_update_word((uint16_t*)13, KY11);
612
613     if (KY2 >= 0)
614     {
615       uint8_t KN = 0;
616       eeprom_update_byte((uint8_t*)18, KN);
617       KY21 = KY2;
618     }
619     else
620     {
621       uint8_t KN = 1; //1 for negative value
622       eeprom_update_byte((uint8_t*)18, KN);
623       KY21 = ((int32_t)KY2*(-1)); //i should test this... Yup it works!
624     }
625     eeprom_update_word((uint16_t*)16, KY21);     //this could be a byte, but i should
        cover because i may be too close to overflow it
626
627     if (KY3 >= 0)
628     {
629       uint8_t KN = 0;
630       eeprom_update_byte((uint8_t*)21, KN);
631       KY31 = KY3;
632     }
633     else
```

```
634     {
635       uint8_t KN = 1; //1 for negative value
636       eeprom_update_byte((uint8_t*)21, KN);
637       KY31 = ((int32_t)KY3*(-1)); //i should test this... Yup it works!
638     }
639     eeprom_update_word((uint16_t*)19, KY31);     //this could be a byte, but i should
        cover because i may be too close to overflow it
640
641
642
643     return(readok);
644   }
645   else
646   {
647     Fcoords = 0;
648     Scoords = 0;
649     Tcoords = 0;
650     Calibrate_TP();
651   }
652
653 // Calibration Eq, should be calculated and stored on EEPROM
654
655 }
656
657
658
659
660 int main(void)
661 {
662   /*
663     Select a setup procedure for your display from here: https://github.com/olikraus/
        u8g2/wiki/u8g2setupc
664     1. Arg: Address of an empty u8g2 structure
665     2. Arg: Usually U8G2_R0, others are listed here: https://github.com/olikraus/u8g2
        /wiki/u8g2reference#carduino-example
666     3. Arg: Protocol procedure (u8g2-byte), list is here: https://github.com/olikraus
        /u8g2/wiki/Porting-to-new-MCU-platform#communication-callback-eg-u8x8_byte_hw_i2c
667     4. Arg: Defined in this code itself (see above)
668   */
669   u8g2_Setup_st7920_s_128x64_f(&u8g2, U8G2_R0, u8x8_byte_4wire_sw_spi,
        u8x8_avr_gpio_and_delay);
670   u8g2_InitDisplay(&u8g2);
671   u8g2_SetPowerSave(&u8g2, 0);
672
673   /* Set up to read analog from a potentiometer */
674
675
676     Start_Adc();
```

```
677
678     Start_Touchscreen();
679
680
681     u8g2_ClearBuffer(&u8g2);
682     u8g2_DrawXBMP(&u8g2, 0, 0, VA_DEFAULT_WIDTH, VA_DEFAULT_HEIGHT, Pantalla_0_bits);
683     u8g2_SendBuffer(&u8g2);
684     set_ports();
685 //Read value from the EEPROM to check if the calibration has been done:
686
687
688
689
690
691
692   while(1){
693     if (status == 0)
694     {
695       u8g2_ClearBuffer(&u8g2);
696       u8g2_DrawXBMP(&u8g2, 0, 0, VA_DEFAULT_WIDTH, VA_DEFAULT_HEIGHT,
    out_of_range_bits);
697       u8g2_SendBuffer(&u8g2);
698       _delay_ms(30);
699
700       inline void abort(void){
701         asm("break"::);
702       }
703     }
704     else
705     {
706
707
708
709
710         //inside while so it checks every loop if we have changed the value
711         Touchcalibration = eeprom_read_byte (( uint8_t *) 3) ; //if the location has
    never been used, it has a value of FF, so we have to check if it isnt 1, to
    calibrate
712         while (Touchcalibration != 1)
713         {
714           calibrationOK = Calibrate_TP();
715           if (calibrationOK == 1)
716           {
717             Touchcalibration = 1;
718             eeprom_update_byte((uint8_t*)3, Touchcalibration);
719             u8g2_ClearBuffer(&u8g2);
720             u8g2_DrawXBMP(&u8g2, 0, 0, VA_DEFAULT_WIDTH, VA_DEFAULT_HEIGHT,
    Pantalla_0_bits);
```

```
721             u8g2_SendBuffer(&u8g2);
722           }
723           else
724           {
725             Touchcalibration = 0;
726           }
727         }
728 
729         //flag to load the parameters first time
730         if (coordupdated == 0)
731         {
732           coordsfromEEPROM();
733           coordupdated = 1;
734         }
735 
736         //Lets read coordinates and stablish in what screen we are, depending on the
    coords
737         _delay_ms(3);
738         uint16_t X_coord = Receive_coordinate(0);
739         _delay_ms(3);
740         uint16_t Y_coord = Receive_coordinate(1);
741 
742         //Reading analog parameters
743         _delay_ms(3);
744         Battery_read = ADC_Read(7);
745 
746         Battery_units = Battery_read*19; // Ratio*1000 so it keeps the value we want
    max V=21,504
747 
748         Battery_decimals = Battery_units; //Now both are equal
749         Battery_units = Battery_units/1000; //now its only the units
750         Battery_units = Battery_units*1000; //Equaling Both decimals and units
751         Battery_decimals = Battery_decimals - Battery_units; // Now this should makes
     so it keeps only the decimals
752         Battery_decimals = Battery_decimals/10;
753         Battery_units = Battery_units/1000;
754 
755         //In theory this is now units + decimals split.
756 
757 
758         _delay_ms(3);
759         Solarpanel_read = ADC_Read(6);
760 
761         Solarpanel_units = Solarpanel_read*21; // Ratio*1000 so it keeps the value we
     want, Max V=50,176
762         Solarpanel_decimals = Solarpanel_units; //Now both are equal
763         Solarpanel_units = Solarpanel_units/1000; //now its only the units
764         Solarpanel_units = Solarpanel_units*1000; //Equaling Both decimals and units
```

```
765         Solarpanel_decimals = Solarpanel_decimals - Solarpanel_units; // Now this
    should makes so it keeps only the decimals
766         Solarpanel_decimals /=10;
767         Solarpanel_units = Solarpanel_units/1000;
768         _delay_ms(3);
769         alternator_read = ADC_Read(5);
770 
771         alternator_units = alternator_read*19; // Ratio*1000 so it keeps the value we
     want mav V= 19,456
772         alternator_decimals = Solarpanel_units; //Now both are equal
773         alternator_units = alternator_units/1000; //now its only the units
774         alternator_units = alternator_units*1000; //Equaling Both decimals and units
775         alternator_decimals = alternator_decimals - alternator_units; // Now this
    should makes so it keeps only the decimals
776         alternator_decimals = alternator_decimals /1000;
777         alternator_units = alternator_units/1000;
778         _delay_ms(3);
779         if (flag_pantalla != 2)
780         {
781           for (int x = 0; x < 5; x++){ //Get 150 samples
782             AcsValue = ADC_Read(4);      //Read current sensor values
783             Samples = Samples + AcsValue;  //Add samples together
784             _delay_ms (3); // let ADC settle before next sample 3ms
785           }
786           AvgAcs=Samples/5.0;
787           AvgAcs =(0.05859375*AvgAcs)-30;
788 
789         }
790         if (manual_control == 0)
791         {
792 
793           //deciding if we should open one or another relay
794           // PC7 => solar panel
795           // PC6 => Alternator
796 
797           if (alternator_units <= 14 && Solarpanel_units >= 13)
798           {
799             //set PC7 High, PC6 low
800 
801             PORTC = 0b01111111;
802             status = 1;
803 
804           }
805           if (alternator_units > 14 && Solarpanel_units < 13)
806           {
807 
808             PORTC = 0b10111111;
809 
```

```
810             status = 2;
811           }
812           if (alternator_units > 14 && Solarpanel_units > 13)
813           {
814
815             PORTC = 0b01111111;
816             status = 1;
817           }
818           if (alternator_units <14 && Solarpanel_units <13)
819           {
820
821             PORTC = 0xff;
822             status = 3;
823           }
824           if (alternator_units >= 19)
825           {
826             PORTC = 0xff;
827             status = 0;
828           }
829           if (Solarpanel_units >= 20)
830           {
831             PORTC = 0xff;
832             flag_pantalla = 5;
833             status = 0;
834           }
835
836         }
837         if (manual_control == 1 && alternatorcharge == 1)
838         {
839           if (alternator_units >= 14)
840           {
841
842             PORTC = 0b10111111;
843
844             status = 2;
845           }
846           if (alternator_units <= 14)
847           {
848             PORTC = 0xff;
849           }
850         }
851         if (manual_control == 1 && alternatorcharge == 0)
852         {
853           if (Solarpanel_units >= 13)
854           {
855             //set PC7 High, PC6 low
856
857             PORTC = 0b01111111;
```

```
858           status = 1;
859 
860         }
861         if (Solarpanel_units <13)
862         {
863 
864           PORTC = 0xff;
865           status = 3;
866         }
867 
868       }
869 
870 
871 
872 
873       //Quick and simple menu:
874 
875       if (X_coord >= 5 && X_coord <= 43 && Y_coord <= 14 && Y_coord >= 3) //first
    screen
876       {
877         u8g2_ClearBuffer(&u8g2);
878         u8g2_DrawXBMP(&u8g2, 0, 0, VA_DEFAULT_WIDTH, VA_DEFAULT_HEIGHT,
    Pantalla_1_alternador_bateria_descarga_bits);
879         u8g2_SendBuffer(&u8g2);
880         flag_pantalla = 1;  // First screen, main
881       }
882 
883       if (X_coord >= 46 && X_coord <= 82 && Y_coord <= 14 && Y_coord >= 3)
884       {
885         u8g2_ClearBuffer(&u8g2);
886         u8g2_DrawXBMP(&u8g2, 0, 0, VA_DEFAULT_WIDTH, VA_DEFAULT_HEIGHT,
    Pantalla_2_bits);
887         u8g2_SendBuffer(&u8g2);
888         flag_pantalla = 2;  // second, data screen
889       }
890       if (X_coord >= 85 && X_coord <= 103 && Y_coord <= 14 && Y_coord >= 3)
891       {
892         u8g2_ClearBuffer(&u8g2);
893         u8g2_DrawXBMP(&u8g2, 0, 0, VA_DEFAULT_WIDTH, VA_DEFAULT_HEIGHT,
    Pantalla_3_auto_on_bits);
894         u8g2_SendBuffer(&u8g2);
895         flag_pantalla = 3;  //third, control screen
896       }
897       if (X_coord >= 105 && X_coord <= 122 && Y_coord <= 14 && Y_coord >= 3)
898       {
899 
900         u8g2_ClearBuffer(&u8g2);
```

```
901         u8g2_DrawXBMP(&u8g2, 0, 0, VA_DEFAULT_WIDTH, VA_DEFAULT_HEIGHT,
    Pantalla_4_bits);
902         u8g2_SendBuffer(&u8g2);
903         flag_pantalla = 4;  //last one, calibration screen
904       }
905       if (flag_pantalla == 4 && X_coord >= 28 && X_coord <= 100 && Y_coord <= 55 &&
     Y_coord >= 35)
906       {
907         eeprom_update_byte((uint8_t*)3, 0);
908         coordupdated = 0;
909       }
910
911       char buffer [50];
912       char buffer2 [50];
913       uint8_t width;
914       uint8_t width2;
915       uint8_t width3;
916       int16_t AvgAcsunits;
917       AvgAcsunits = AvgAcs * 100;
918       switch (flag_pantalla)
919       {
920
921
922         case(1):
923         if (status == 1 && AvgAcsunits >0)
924         {
925         u8g2_ClearBuffer(&u8g2);
926         u8g2_DrawXBMP(&u8g2, 0, 0, VA_DEFAULT_WIDTH, VA_DEFAULT_HEIGHT,
    Pantalla_1_solar_todo_bits);
927         u8g2_SendBuffer(&u8g2);
928         }
929         if (status == 1 && AvgAcsunits <0)
930         {
931           u8g2_ClearBuffer(&u8g2);
932           u8g2_DrawXBMP(&u8g2, 0, 0, VA_DEFAULT_WIDTH, VA_DEFAULT_HEIGHT,
    Pantalla_1_solar_bateria_descarga_bits);
933           u8g2_SendBuffer(&u8g2);
934         }
935         if (status == 2 && AvgAcsunits >0)
936         {
937           u8g2_ClearBuffer(&u8g2);
938           u8g2_DrawXBMP(&u8g2, 0, 0, VA_DEFAULT_WIDTH, VA_DEFAULT_HEIGHT,
    Pantalla_1_alternador_todo_bits);
939           u8g2_SendBuffer(&u8g2);
940         }
941         if (status == 2 && AvgAcsunits <0)
942         {
943           u8g2_ClearBuffer(&u8g2);
```

```
944           u8g2_DrawXBMP(&u8g2, 0, 0, VA_DEFAULT_WIDTH, VA_DEFAULT_HEIGHT,
    Pantalla_1_alternador_bateria_descarga_bits);
945           u8g2_SendBuffer(&u8g2);
946         }
947         if (status == 3)
948         {
949           u8g2_ClearBuffer(&u8g2);
950           u8g2_DrawXBMP(&u8g2, 0, 0, VA_DEFAULT_WIDTH, VA_DEFAULT_HEIGHT,
    Pantalla_1_bateria_descarga_bits);
951           u8g2_SendBuffer(&u8g2);
952         }
953
954         break;
955
956         case(2):
957           u8g2_ClearBuffer(&u8g2);
958           sprintf(buffer,"%u",Battery_units);
959           sprintf(buffer2,"%u",Battery_decimals);
960           u8g2_SetFont(&u8g2, u8g2_font_6x13_mf);
961           u8g2_DrawXBMP(&u8g2, 0, 0, VA_DEFAULT_WIDTH, VA_DEFAULT_HEIGHT,
    Pantalla_2_bits);
962           width = u8g2_DrawStr(&u8g2, 40, 36, buffer);
963           width2 = u8g2_DrawStr(&u8g2, 40+width, 36, ".");
964           width3 = u8g2_DrawStr(&u8g2, 40+width+width2, 36, buffer2);
965           u8g2_DrawStr(&u8g2, 40+width+width2+width3, 36, "V");
966
967           sprintf(buffer,"%u",Solarpanel_units);
968           sprintf(buffer2,"%u",Solarpanel_decimals);
969           width = u8g2_DrawStr(&u8g2, 32, 56, buffer);
970           width2 = u8g2_DrawStr(&u8g2, 32+width, 56, ".");
971           width3 = u8g2_DrawStr(&u8g2, 32+width+width2, 56, buffer2);
972           u8g2_DrawStr(&u8g2, 32+width+width2+width3, 56, "V");
973
974           sprintf(buffer,"%u",alternator_units);
975           sprintf(buffer2,"%u",alternator_decimals);
976           width = u8g2_DrawStr(&u8g2, 84, 56, buffer);
977           width2 = u8g2_DrawStr(&u8g2, 84+width, 56, ".");
978           width3 = u8g2_DrawStr(&u8g2, 84+width+width2, 56, buffer2);
979           u8g2_DrawStr(&u8g2, 84+width+width2+width3, 56, "V");
980
981           //Reading Amperage with means, to avoid false readings, this will lose
    responsiveness, but improve precision :
982             for (int x = 0; x < 25; x++){ //Get 150 samples
983               AcsValue = ADC_Read(4);     //Read current sensor values
984               Samples = Samples + AcsValue;  //Add samples together
985               _delay_ms (3); // let ADC settle before next sample 3ms
986             }
987             AvgAcs=Samples/25.0;
```

```
988
989
990             AvgAcs =(0.05859375*AvgAcs)-30;
991             sprintf(buffer,"%.3f",AvgAcs);
992             width = u8g2_DrawStr(&u8g2, 80, 36, buffer);
993             u8g2_DrawStr(&u8g2, 80+width, 36, "A");
994             u8g2_SendBuffer(&u8g2);
995             AvgAcs = 0;
996             Samples = 0;
997           break;
998
999           case(3):
1000            if (manual_control == 0 && alternatorcharge == 0)
1001            {
1002              u8g2_ClearBuffer(&u8g2);
1003              u8g2_DrawXBMP(&u8g2, 0, 0, VA_DEFAULT_WIDTH, VA_DEFAULT_HEIGHT,
    Pantalla_3_auto_on_bits);
1004              u8g2_SendBuffer(&u8g2);
1005            }
1006            if (manual_control == 1 && alternatorcharge == 0)
1007            {
1008              u8g2_ClearBuffer(&u8g2);
1009              u8g2_DrawXBMP(&u8g2, 0, 0, VA_DEFAULT_WIDTH, VA_DEFAULT_HEIGHT,
    Pantalla_3_auto_off_carga_off_bits);
1010              u8g2_SendBuffer(&u8g2);
1011            }
1012
1013            if (manual_control == 1 && alternatorcharge == 1)
1014            {
1015              u8g2_ClearBuffer(&u8g2);
1016              u8g2_DrawXBMP(&u8g2, 0, 0, VA_DEFAULT_WIDTH, VA_DEFAULT_HEIGHT,
    Pantalla_3_auto_off_carga_on_bits);
1017              u8g2_SendBuffer(&u8g2);
1018            }
1019
1020            if (manual_control == 0 && alternatorcharge == 1)
1021            {
1022              alternatorcharge = 0;
1023              manual_control = 0;
1024            }
1025
1026            if (X_coord >= 89 && X_coord <= 125 && Y_coord <= 31 && Y_coord >= 26)
1027            {
1028
1029              u8g2_ClearBuffer(&u8g2);
1030              u8g2_DrawXBMP(&u8g2, 0, 0, VA_DEFAULT_WIDTH, VA_DEFAULT_HEIGHT,
    Pantalla_3_auto_off_carga_off_bits);
1031              u8g2_SendBuffer(&u8g2);
```

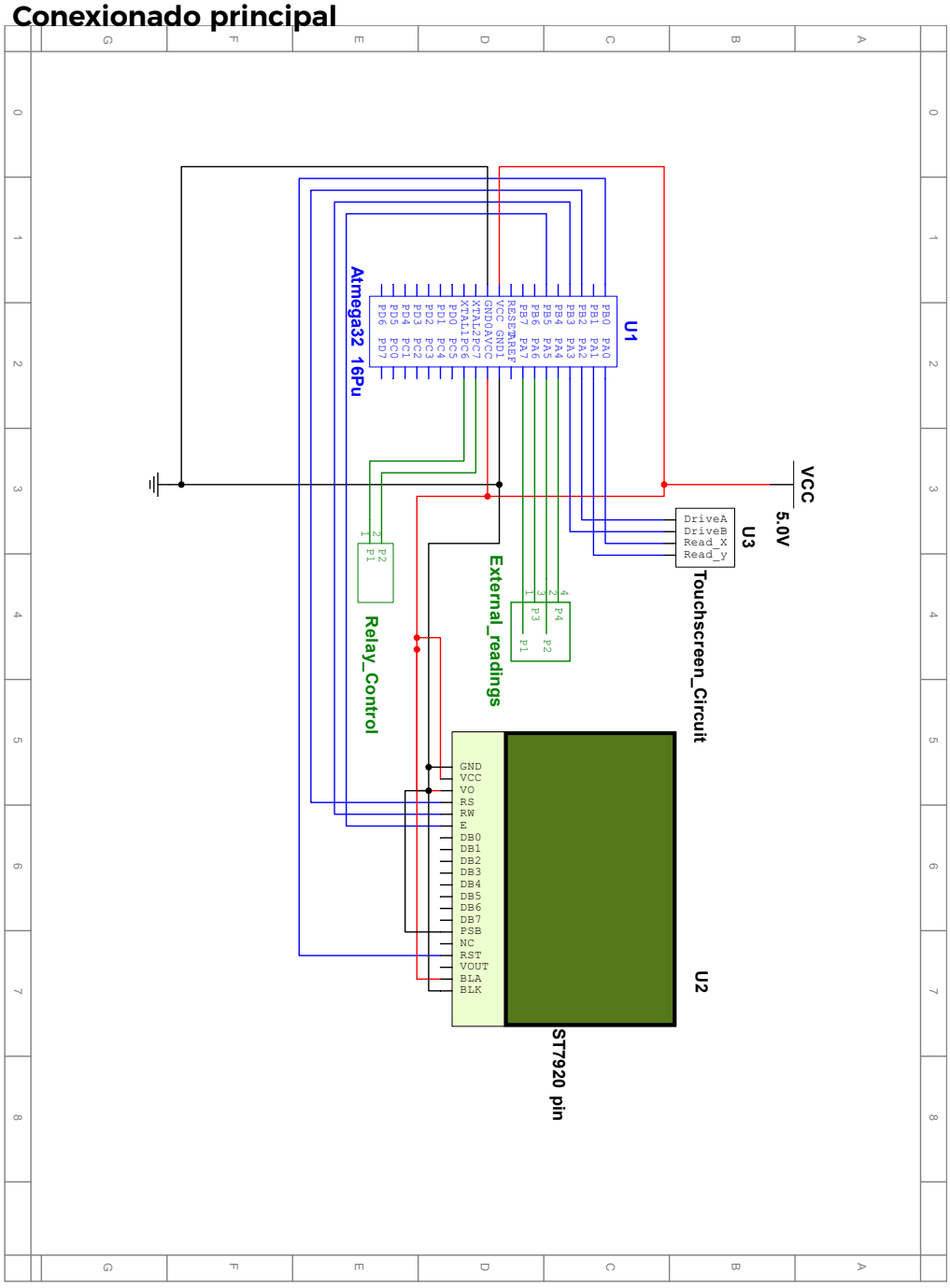
```
1032              manual_control = 1;
1033              alternatorcharge = 0;
1034            }
1035            if (manual_control == 1 && X_coord >= 51 && X_coord <= 87 && Y_coord <=
    58 && Y_coord >= 42)
1036            {
1037              u8g2_ClearBuffer(&u8g2);
1038              u8g2_DrawXBMP(&u8g2, 0, 0, VA_DEFAULT_WIDTH, VA_DEFAULT_HEIGHT,
    Pantalla_3_auto_off_carga_on_bits);
1039              u8g2_SendBuffer(&u8g2);
1040              alternatorcharge = 1;
1041            }
1042            if (manual_control == 1 && X_coord >= 90 && X_coord <= 124 && Y_coord <=
    58 && Y_coord >= 42)
1043            {
1044              u8g2_ClearBuffer(&u8g2);
1045              u8g2_DrawXBMP(&u8g2, 0, 0, VA_DEFAULT_WIDTH, VA_DEFAULT_HEIGHT,
    Pantalla_3_auto_off_carga_off_bits);
1046              u8g2_SendBuffer(&u8g2);
1047              alternatorcharge = 0;
1048            }
1049            if (X_coord >= 51 && X_coord <= 87 && Y_coord <= 35 && Y_coord >= 20)
1050            {
1051              u8g2_ClearBuffer(&u8g2);
1052              u8g2_DrawXBMP(&u8g2, 0, 0, VA_DEFAULT_WIDTH, VA_DEFAULT_HEIGHT,
    Pantalla_3_auto_on_bits);
1053              u8g2_SendBuffer(&u8g2);
1054              alternatorcharge = 0;
1055              manual_control = 0;
1056            }
1057          break;
1058
1059          case(4):
1060
1061          break;
1062
1063        }
1064      }
1065  }
1066
1067 }
```

Apéndice B

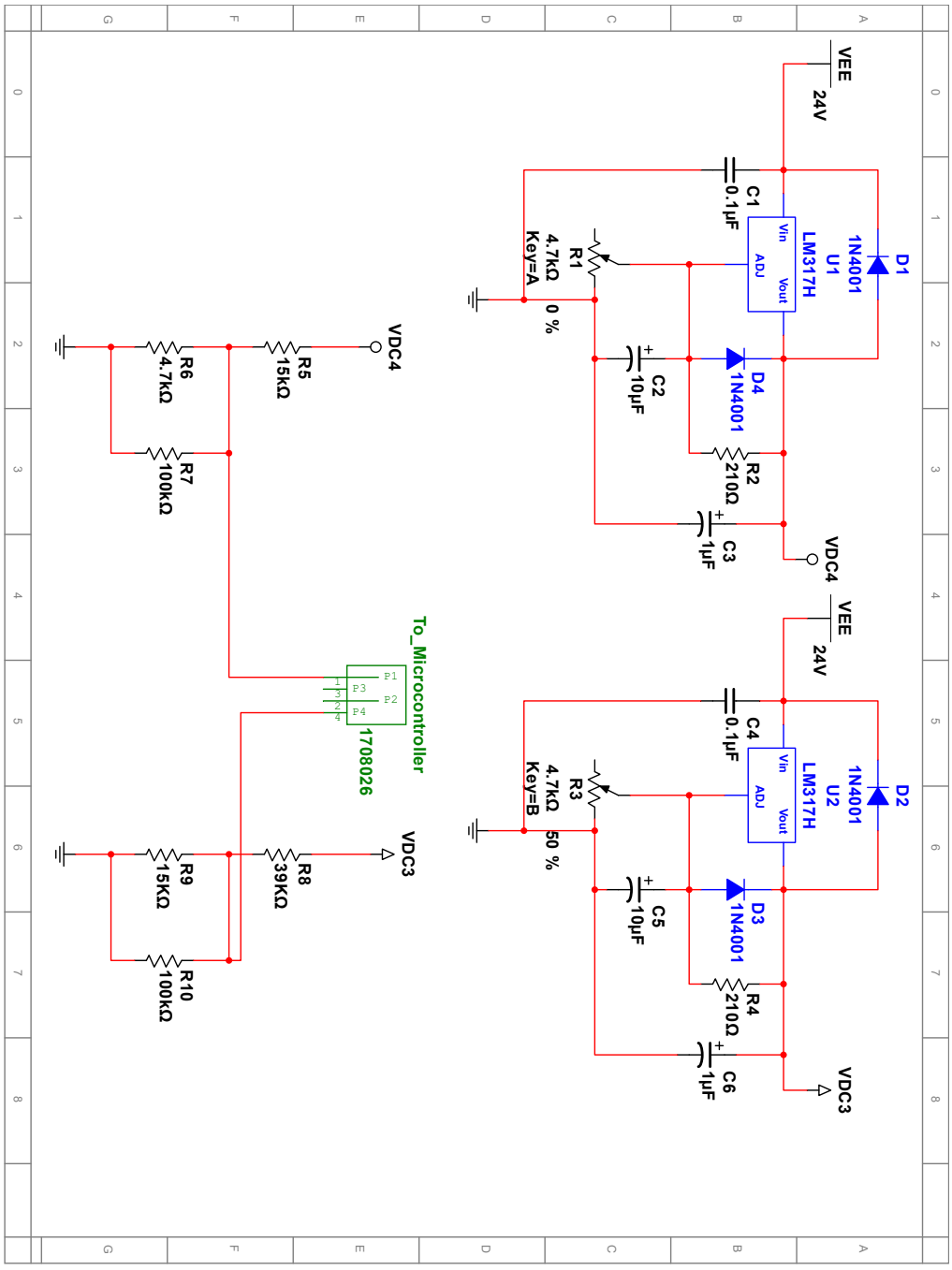
Circuitos, datasheets e informes

B.1. Esquemas de conexion

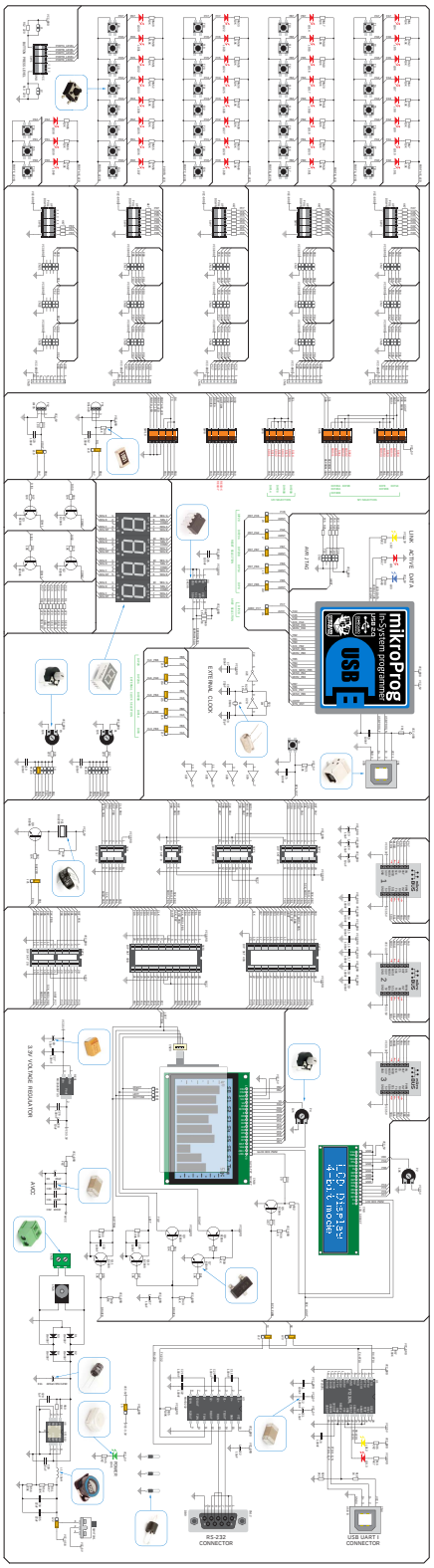
B.1.1. Conexionado principal



B.1.2. Esquema del los convertidores servidos para emular la instalación



B.1.3. Esquema general de la placa de desarrollo



B.2. Datasheets de periféricos

B.2.1. ATmega32

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 16 MHz
  - On-chip 2-cycle Multiplier
- High Endurance Non-volatile Memory segments
  - 32Kbytes of In-System Self-programmable Flash program memory
  - 1024Bytes EEPROM
  - 2Kbyte 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 1 MHz, 3V, 25°C
  - Active: 1.1mA
  - Idle Mode: 0.35mA
  - Power-down Mode: < 1µA

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

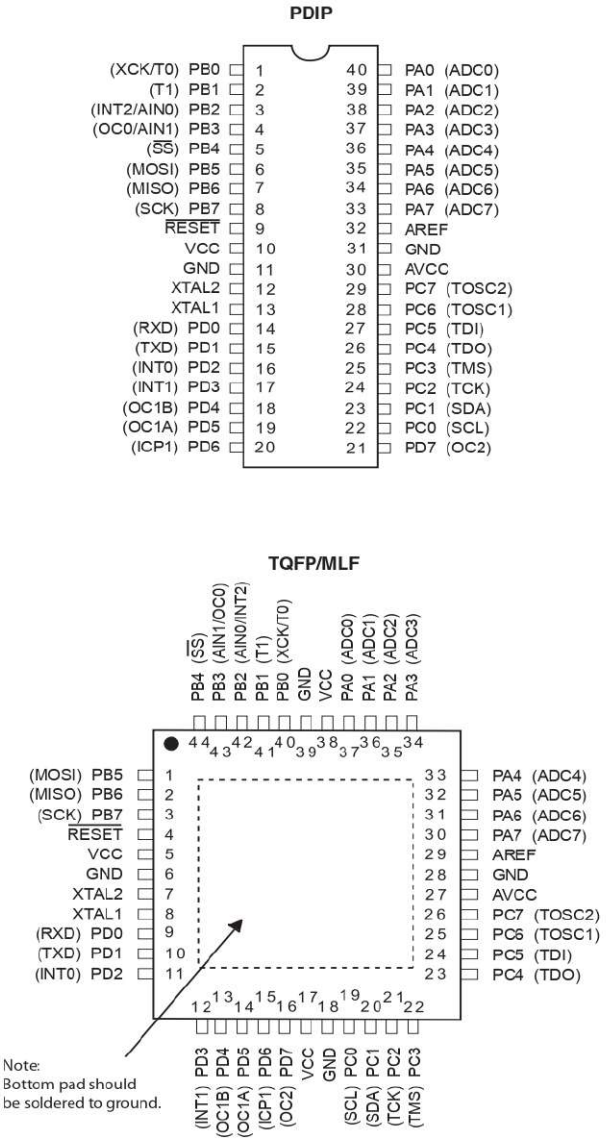
ATmega32
ATmega32L

Summary

2503QS–AVR–02/11

Pin Configurations

Figure 1. Pinout ATmega32

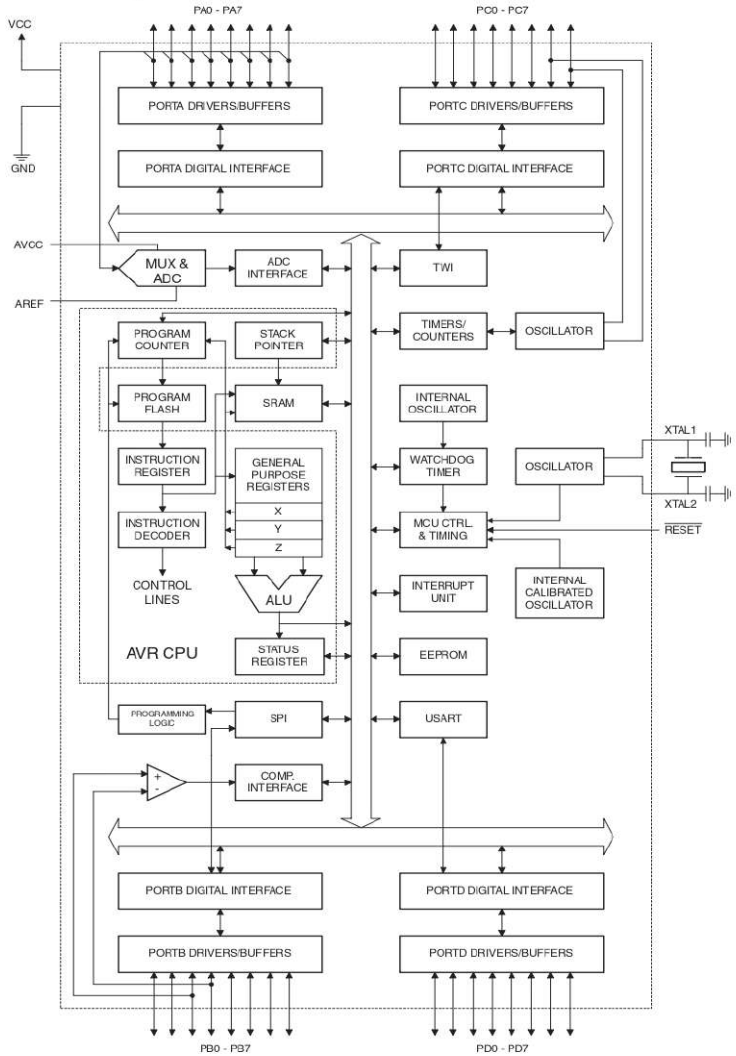


Overview

The Atmel® AVR® ATmega32 is a low-power CMOS 8-bit microcontroller based on the AVR enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the ATmega32 achieves throughputs approaching 1 MIPS per MHz allowing the system designer to optimize power consumption versus processing speed.

Block Diagram

Figure 2. Block Diagram



The Atmel® AVR® core combines a rich instruction set with 32 general purpose working registers. All the 32 registers are directly connected to the Arithmetic Logic Unit (ALU), allowing two independent registers to be accessed in 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.

Resources

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

Data Retention

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

About Code Examples

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

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

Ordering Information

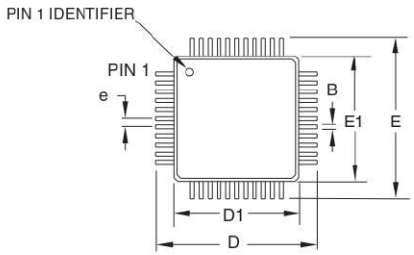
Speed (MHz)	Power Supply	Ordering Code[2]	Package[1]	Operational Range
8	2.7V - 5.5V	ATmega32L-8AU ATmega32L-8AUR[3] ATmega32L-8PU ATmega32L-8MU ATmega32L-8MUR[3]	44A 44A 40P6 44M1 44M1	Industrial (-40°C to 85°C)
16	4.5V - 5.5V	ATmega32-16AU ATmega32-16AUR[3] ATmega32-16PU ATmega32-16MU ATmega32-16MUR[3]	44A 44A 40P6 44M1 44M1	

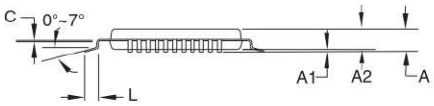
Notes:
1. This device can also be supplied in wafer form. Please contact your local Atmel sales office for detailed ordering information and minimum quantities.
2. Pb-free packaging complies to the European Directive for Restriction of Hazardous Substances (RoHS directive). Also Halide free and fully Green.
3. Tape & Reel

Package Type	
44A	44-lead, 10 × 10 × 1.0mm, Thin Profile Plastic Quad Flat Package (TQFP)
40P6	40-pin, 0.600" Wide, Plastic Dual Inline Package (PDIP)
44M1	44-pad, 7 × 7 × 1.0mm, Quad Flat No-Lead/Micro Lead Frame Package (QFN/MLF)

Packaging Information

44A





COMMON DIMENSIONS
(Unit of Measure = mm)

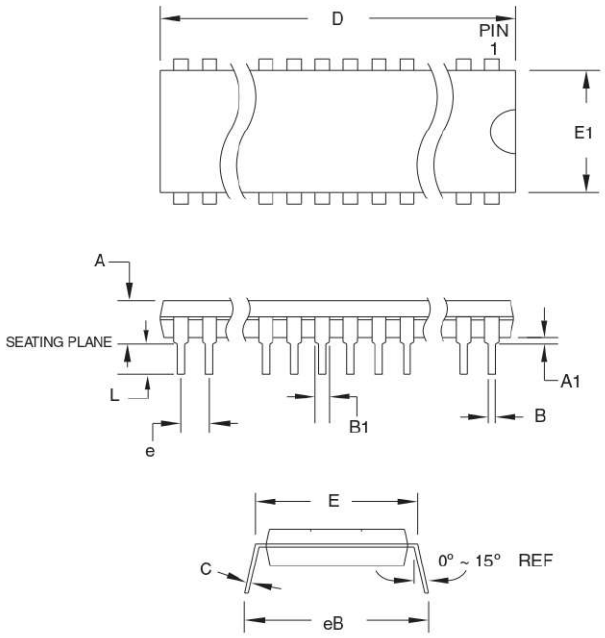
SYMBOL	MIN	NOM	MAX	NOTE
A	–	–	1.20	
A1	0.05	–	0.15	
A2	0.95	1.00	1.05	
D	11.75	12.00	12.25	
D1	9.90	10.00	10.10	Note 2
E	11.75	12.00	12.25	
E1	9.90	10.00	10.10	Note 2
B	0.30	–	0.45	
C	0.09	–	0.20	
L	0.45	–	0.75	
e	0.80 TYP			

Notes:
1. This package conforms to JEDEC reference MS-026, Variation ACB.
2. Dimensions D1 and E1 do not include mold protrusion. Allowable protrusion is 0.25 mm per side. Dimensions D1 and E1 are maximum plastic body size dimensions including mold mismatch.
3. Lead coplanarity is 0.10 mm maximum.

2010-10-20

	TITLE	DRAWING NO.	REV.
2325 Orchard Parkway San Jose, CA 95131	**44A,** 44-lead, 10 x 10 mm Body Size, 1.0 mm Body Thickness, 0.8 mm Lead Pitch, Thin Profile Plastic Quad Flat Package (TQFP)	44A	C

40P6



COMMON DIMENSIONS
(Unit of Measure = mm)

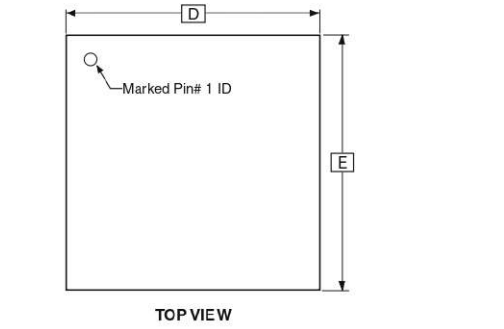
SYMBOL	MIN	NOM	MAX	NOTE
A	–	–	4.826	
A1	0.381	–	–	
D	52.070	–	52.578	Note 2
E	15.240	–	15.875	
E1	13.462	–	13.970	Note 2
B	0.356	–	0.559	
B1	1.041	–	1.651	
L	3.048	–	3.556	
C	0.203	–	0.381	
eB	15.494	–	17.526	
e	2.540 TYP			

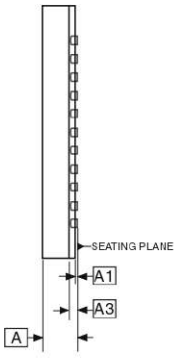
Notes: 1. This package conforms to JEDEC reference MS-011, Variation AC.
2. Dimensions D and E1 do not include mold Flash or Protrusion. Mold Flash or Protrusion shall not exceed 0.25 mm (0.010").

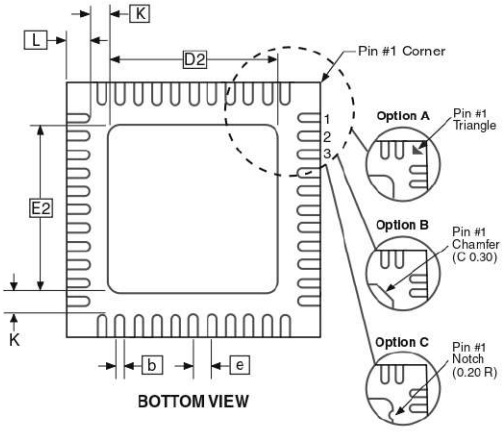
09/28/01

	TITLE	DRAWING NO.	REV.
2325 Orchard Parkway San Jose, CA 95131	**40P6**, 40-lead (0.600"/15.24 mm Wide) Plastic Dual Inline Package (PDIP)	40P6	B

44M1







Note: JEDEC Standard MO-220, Fig. 1 (SAW Singulation) VKKD-3.

COMMON DIMENSIONS
(Unit of Measure = mm)

SYMBOL	MIN	NOM	MAX	NOTE
A	0.80	0.90	1.00	
A1	–	0.02	0.05	
A3		0.20 REF		
b	0.18	0.23	0.30	
D	6.90	7.00	7.10	
D2	5.00	5.20	5.40	
E	6.90	7.00	7.10	
E2	5.00	5.20	5.40	
e		0.50 BSC		
L	0.59	0.64	0.69	
K	0.20	0.26	0.41	

9/26/08

Package Drawing Contact: packagedrawings@atmel.com	TITLE	GPC	DRAWING NO.	REV.
	44M1, 44-pad, 7 x 7 x 1.0 mm Body, Lead Pitch 0.50 mm, 5.20 mm Exposed Pad, Thermally Enhanced Plastic Very Thin Quad Flat No Lead Package (VQFN)	ZWS	44M1	H

Errata

ATmega32, rev. A to F

- **First Analog Comparator conversion may be delayed**
- **Interrupts may be lost when writing the timer registers in the asynchronous timer**
- **IDCODE masks data from TDI input**
- **Reading EEPROM by using ST or STS to set EERE bit triggers unexpected interrupt request.**

1. **First Analog Comparator conversion may be delayed**

   If the device is powered by a slow rising V_{CC}, the first Analog Comparator conversion will take longer than expected on some devices.

   Problem Fix/Workaround

   When the device has been powered or reset, disable then enable theAnalog Comparator before the first conversion.

2. **Interrupts may be lost when writing the timer registers in the asynchronous timer**

   The interrupt will be lost if a timer register that is synchronous timer clock is written when the asynchronous Timer/Counter register (TCNTx) is 0x00.

   Problem Fix/Workaround

   Always check that the asynchronous Timer/Counter register neither have the value 0xFF nor 0x00 before writing to the asynchronous Timer Control Register (TCCRx), asynchronous-Timer Counter Register (TCNTx), or asynchronous Output Compare Register (OCRx).

3. **IDCODE masks data from TDI input**

   The JTAG instruction IDCODE is not working correctly. Data to succeeding devices are replaced by all-ones during Update-DR.

   Problem Fix / Workaround

   - If ATmega32 is the only device in the scan chain, the problem is not visible.
   - Select the Device ID Register of the ATmega32 by issuing the IDCODE instruction or by entering the Test-Logic-Reset state of the TAP controller to read out the contents of its Device ID Register and possibly data from succeeding devices of the scan chain. Issue the BYPASS instruction to the ATmega32 while reading the Device ID Registers of preceding devices of the boundary scan chain.
   - If the Device IDs of all devices in the boundary scan chain must be captured simultaneously, the ATmega32 must be the fist device in the chain.

4. **Reading EEPROM by using ST or STS to set EERE bit triggers unexpected interrupt request.**

   Reading EEPROM by using the ST or STS command to set the EERE bit in the EECR register triggers an unexpected EEPROM interrupt request.

   Problem Fix / Workaround

   Always use OUT or SBI to set EERE in EECR.

B.2.2. GLCD ST7920

Sitronix

ST7920

Chinese Fonts built in LCD controller/driver

Main Features

- Voltage operating range:
  - 2.7 to 5.5V
- Support 8 bit, 4 bit, serial bus MPU interface
- 64 x 16-bits character display RAM (max. 16 chars x 4 lines, LCD display range 16 char. X 2 lines
- 64 x 256-bits graphic display RAM（GDRAM）
- 2M-bits Chinese fonts ROM (CGROM) supporting 8192 Chinese fonts (16x16 dot matrix)
- 16K-bits half height ROM (HCGROM) supporting 126 character set (16x8 dot matrix)
- 64 x 16-bits character generation RAM (CGRAM)
- 15 x 16-bits total 240 ICON RAM（IRAM）
- 33-common x 64-segment (2 lines display) LCD drivers
- Automatic power on reset
- External reset pin (XRESET)
- With extension segment drivers display area can up to 16x2 lines
- RC oscillator built in (with external R)
- Low power design
  Normal mode (450uA Typ V_{DD}=5V)
  Standby mode (30uA Max V_{DD}=5V)
- VLCD (V0~ V_{ss}): max 7V
- Graphic and character mix modes display
- Multiple instructions：
  - （Display clear）
  - （Return home）
  - （Display on/off）
  - （Cursor on/off）
  - （Display character blink）
  - （Cursor shift）
  - （Display shift）
  - （Vertical line scroll）
  - （By_line reverse display）
  - （Standby mode）
- Built in voltage booster (2 times)
- 1/33 Duty

Function Description

ST7920 LCD controller/driver IC can display alphabets, numbers, Chinese fonts and self-defined characters. It supports 3 kinds of bus interface, namely 8 bit/ 4bit and serial. All functions, including display RAM, character generator ROM, LCD display drivers and control circuits are all in a one-chip solution. With a minimum system configuration, a Chinese character display system can easily achieved.

ST7920 includes character ROM with 8192 16X16 dots Chinese fonts and 126 16X8 dots half height alphanumerical fonts. Also for graphic display it supports 64x256 dots graphic display area（GDRAM）and 240 dots ICON RAM. Mix mode display with both character and graphic data is possible. ST7920 has built in 4 sets CGRAM providing software programmable 16X16 font.

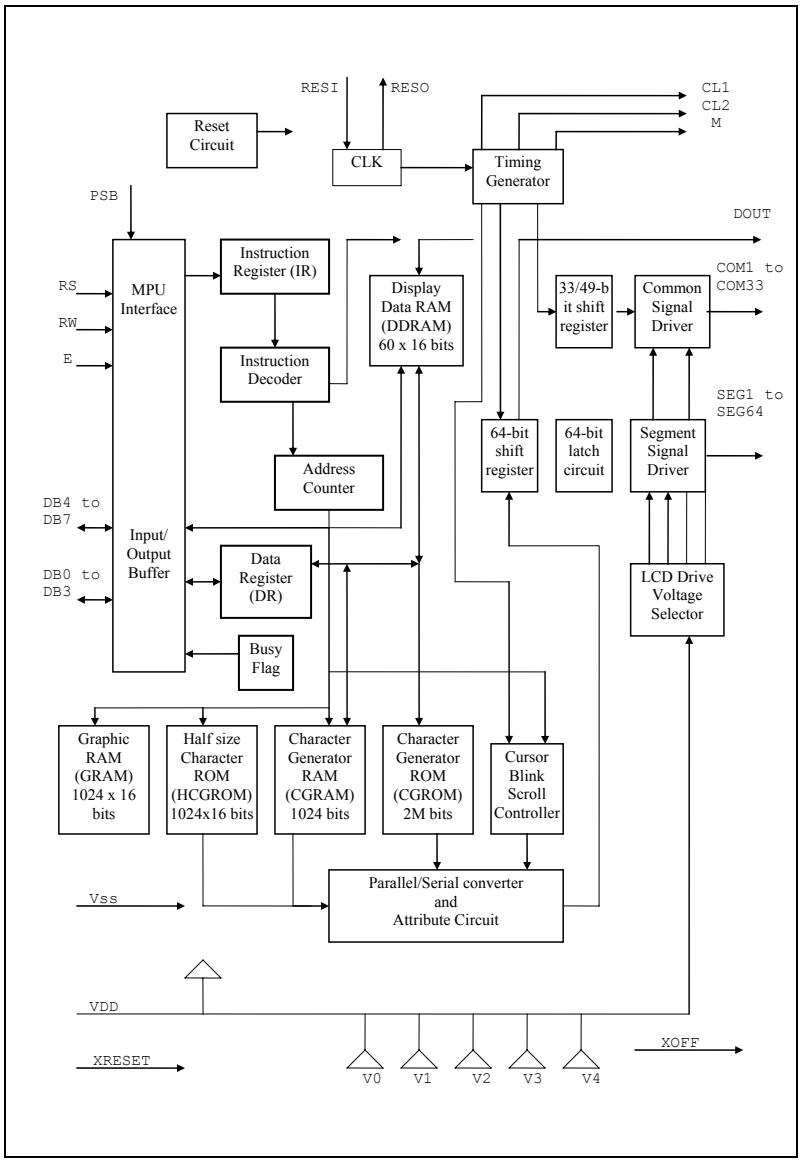
ST7920 has wide operating voltage (2.7V to 5.5V) and low power consumption suitable for battery power portable device.

ST7920 LCD driver consists of 33 common and 64 segments. Together with extension segment driver ST7921, ST7920 can support up to 33 common x 256 segments display.

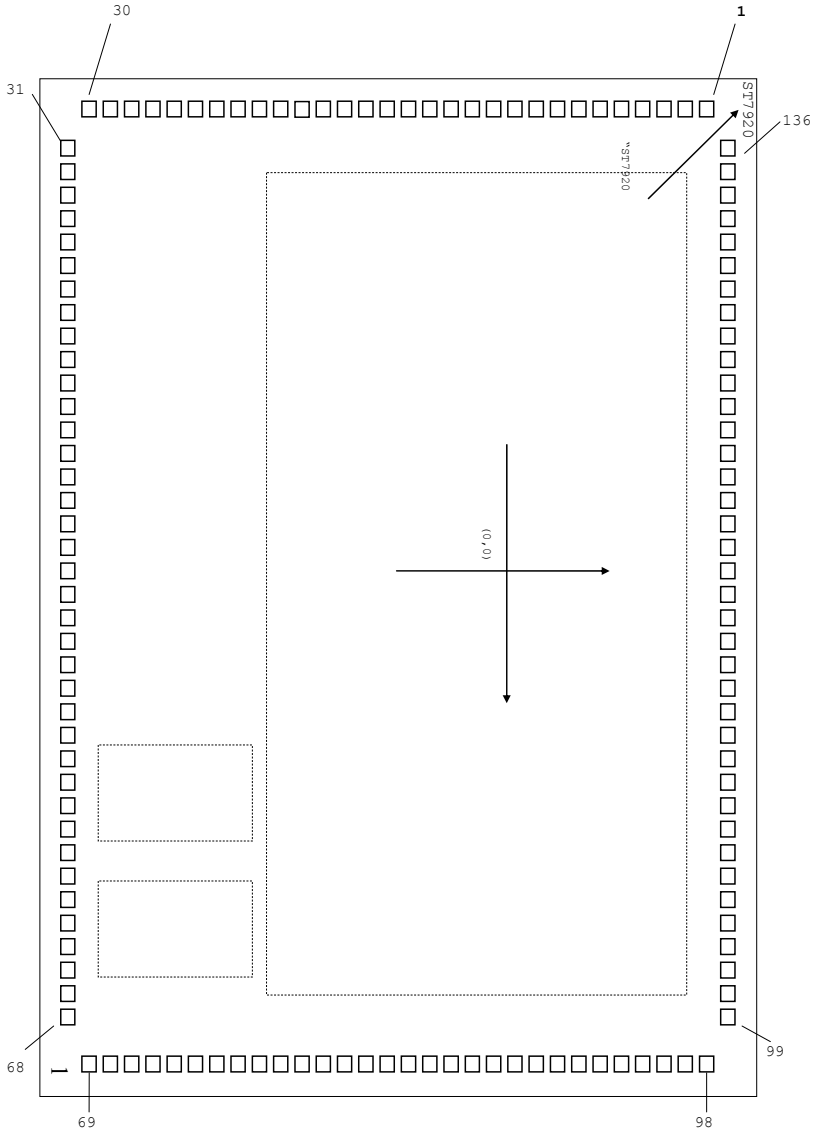
Product	Font type
ST7920-0A	BIG-5 code traditional character set
ST7920-0B	GB code simplified character set
ST7920-0C	GB code,BIG-5 code and Japanese code

ST7920 Specification Revision History		
Version	**Date**	**Description**
C1.7	2000/12/15	1. VCC changed to VDD. 2. VLCD changed from VCC-V5 to V0-VSS. 3. DC characteristics input High voltage (Vih) changed to 0.7VDD. 4. DC characteristics output High voltage (Voh) changed to 0.8VDD.
C1.8	2001/03/01	1. Chip Size changed. 2. ICON 256 dots changed to 240 dots. 3. XOFF normal high sleep Low changed to normal low sleep High. 4. Added XOFF application. 5. Modified application of ST7920 4,5,6 PIN floating. (4,5,6 爲 test pin) 6. Modified voltage doubler CAP1P, CAP1M, CAP2M capacitors polarity
C1.9	2001/05/28	1. Icon RAM TABLE changed. (TABLE-6) 2. Booster description modified. (PAGE-29) 3. AC Characteristics modified. 4. Added 2Line 16 Chinese Word (32Com X 256Seg) application circuit. 5. Added oscillation resistor's relation to power consumption and frequency.
C2.0	2001/07/03	1. Added Register initial values. 2. Voltage booster CAP1M CAP1P polarity changed. (PAGE-30)
V2.0	2001/08/17	1. Modified Table 7. (PAGE-14) 2. Change to English version.
V2.0c	2001/10/18	1. Modified page-38 Serial interface timing diagram
V2.0d	2002/05/09	1. Add the standard code (Japan、GB code、BIG-5 code)
V3.0	2002/10/11	1. Delete sleep mode function

System Block diagram



Pads diagram



origin: center of chip coordinates: from pad center

chip size: 5305 X 4074 Pad open: 90 X 90

Pad pitch: 125 unit: μm

*** chip substrate must connect to VSS**

Pin coordinates **unit: um**

No.	Name	X	Y
1	V0	-2548	1812
2	V1	-2548	1688
3	V2	-2548	1562
4	N.C.	-2548	1438
5	N.C.	-2548	1312
6	N.C.	-2548	1188
7	V3	-2548	1062
8	V4	-2548	938
9	VSS	-2548	812
10	VDD	-2548	688
11	XRESET	-2548	562
12	CL1	-2548	438
13	CL2	-2548	312
14	VDD	-2548	188
15	M	-2548	62
16	DOUT	-2548	-62
17	RS	-2548	-188
18	RW	-2548	-312
19	E	-2548	-438
20	VSS	-2548	-562
21	OSC1	-2548	-688
22	OSC2	-2548	-812
23	PSB	-2548	-938
24	D0	-2548	-1062
25	D1	-2548	-1188
26	D2	-2548	-1312
27	D3	-2548	-1438
28	D4	-2548	-1562
29	D5	-2548	-1688
30	D6	-2548	-1812
31	D7	-2306	-1933
32	XOFF	-2181	-1933
33	VOUT	-2056	-1933
34	CAP3M	-1931	-1933
35	CAP1P	-1806	-1933
36	CAP1M	-1681	-1933
37	CAP2P	-1556	-1933
38	CAP2M	-1431	-1933

No.	Name	X	Y
39	VD2	-1306	-1933
40	C[1]	-1181	-1933
41	C[2]	-1056	-1933
42	C[3]	-931	-1933
43	C[4]	-806	-1933
44	C[5]	-681	-1933
45	C[6]	-556	-1933
46	C[7]	-431	-1933
47	C[8]	-306	-1933
48	C[9]	-181	-1933
49	C[10]	-56	-1933
50	C[11]	69	-1933
51	C[12]	194	-1933
52	C[13]	319	-1933
53	C[14]	444	-1933
54	C[15]	569	-1933
55	C[16]	694	-1933
56	C[17]	819	-1933
57	C[18]	944	-1933
58	C[19]	1069	-1933
59	C[20]	1194	-1933
60	C[21]	1319	-1933
61	C[22]	1444	-1933
62	C[23]	1569	-1933
63	C[24]	1694	-1933
64	C[25]	1819	-1933
65	C[26]	1944	-1933
66	C[27]	2069	-1933
67	C[28]	2194	-1933
68	C[29]	2319	-1933
69	C[30]	2548	-1812
70	C[31]	2548	-1688
71	C[32]	2548	-1562
72	C[33]	2548	-1438
73	S[64]	2548	-1312
74	S[63]	2548	-1188
75	S[62]	2548	-1062
76	S[61]	2548	-938

No.	Name	X	Y
77	S[60]	2548	-812
78	S[59]	2548	-688
79	S[58]	2548	-562
80	S[57]	2548	-438
81	S[56]	2548	-312
82	S[55]	2548	-188
83	S[54]	2548	-62
84	S[53]	2548	62
85	S[52]	2548	188
86	S[51]	2548	312
87	S[50]	2548	438
88	S[49]	2548	562
89	S[48]	2548	688
90	S[47]	2548	812
91	S[46]	2548	938
92	S[45]	2548	1062
93	S[44]	2548	1188
94	S[43]	2548	1312
95	S[42]	2548	1438
96	S[41]	2548	1562
97	S[40]	2548	1688
98	S[39]	2548	1812
99	S[38]	2319	1933
100	S[37]	2194	1933
101	S[36]	2069	1933
102	S[35]	1944	1933
103	S[34]	1819	1933
104	S[33]	1694	1933
105	S[32]	1569	1933
106	S[31]	1444	1933
107	S[30]	1319	1933
108	S[29]	1194	1933
109	S[28]	1069	1933
110	S[27]	944	1933
111	S[26]	819	1933
112	S[25]	694	1933
113	S[24]	569	1933
114	S[23]	444	1933
115	S[22]	319	1933

No.	Name	X	Y
116	S[21]	194	1933
117	S[20]	69	1933
118	S[19]	-56	1933
119	S[18]	-181	1933
120	S[17]	-306	1933
121	S[16]	-431	1933
122	S[15]	-556	1933
123	S[14]	-681	1933
124	S[13]	-806	1933
125	S[12]	-931	1933
126	S[11]	-1056	1933
127	S[10]	-1181	1933
128	S[9]	-1306	1933
129	S[8]	-1431	1933
130	S[7]	-1556	1933
131	S[6]	-1681	1933
132	S[5]	-1806	1933
133	S[4]	-1931	1933
134	S[3]	-2056	1933
135	S[2]	-2181	1933
136	S[1]	-2306	1933

Pin Description

Name	No.	I/O	Connects to	Function
XRESET	11	I	—	System reset low active
PSB	23	I	—	Interface selection: 0: serial mode 1: 8/4-bits parallel bus mode
RS(CS*)	17	I	MPU	Register select 0: select instruction write, busy flag read, address counter read 1: select data write, read (Chip select) for serial mode 1: chip enable 0: chip disable
RW(SID*)	18	I	MPU	Read write control 0: write 1: read (serial data input)
E(SCLK*)	19	I	MPU	Enable trigger (serial clock)
D4 to D7	28～31	I/O	MPU	Higher nibble data bus for 8 bit interface and data bus for 4 bit interface
D0 to D3	24～27	I/O	MPU	Lower nibble data bus for 8 bit interface
CL1	12	O	Extension segment drv.	Latch signal for extension segment drivers
CL2	13	O	Extension segment drv.	Shift clock for extension segment drivers
M	15	O	Extension segment drv.	AC signal for extension segment drivers voltage inversion
DOUT	16	O	Extension segment drv.	Data output for extension segment drivers
COM1 to COM33	40～72	O	LCD	Common signals
SEG1 to SEG64	136～73	O	LCD	Segment signals
V0 to V4	1～3 7,8	—	—	LCD bias voltage V_0 - V4 ≦ 7 V
V_{DD}	10,14	I	Power	V_{DD}: 2.7V to 5.5V
Vss	9,20	I	Power	VSS: 0V
OSC1, OSC2	21,22	I, O	Resistors	For internal oscillation resistor 5.0V R=33K 2.7V R=18K use OSC1 for external clock input
VOUT	33	O	Resistors	LCD voltage doubler output

***note: The OSC pin must have the shortest wiring pattern of all other pins.To prevent noise from other signal lines , it should also be enclosed with the largest GND pattern possible. Poor noise characteristics on the OSC line will result in malfunction , or adversely affect the clock's duty ratio.**

Pin description

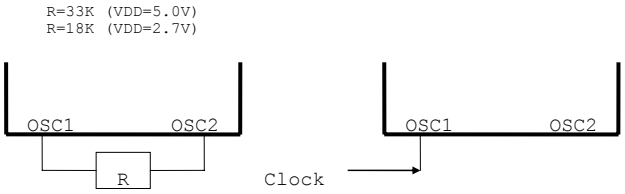
Name	No.	I/O	Connects to	Description
CAP3M CAP1P CAP1M CAP2M	34 35 36 38	I/O	Capacitors	Capacitor pins for voltage doubler
XOFF	32	O	—	Reserved (no connection)
CAP2P	37	—	—	Reserved (no connection)
VD2	39	I	Reference voltage	Voltage doubler reference voltage
N.C. N.C. N.C.	4 5 6	I	— — —	Test pins (no connection)

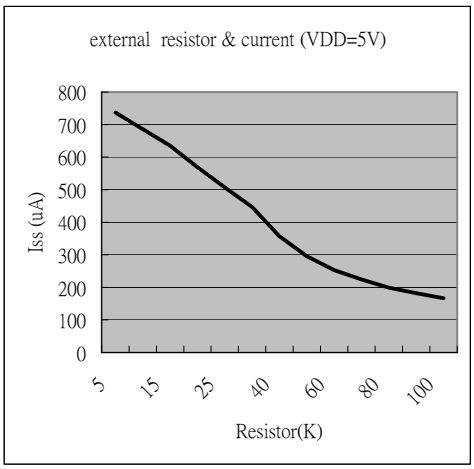
Note:

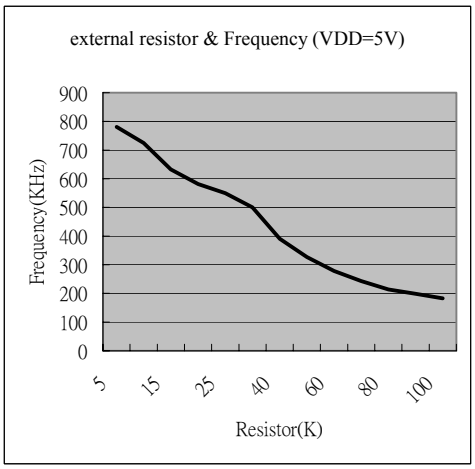
1. VDD>=V0>=V1>=V2>=V3>=V4 must be maintained

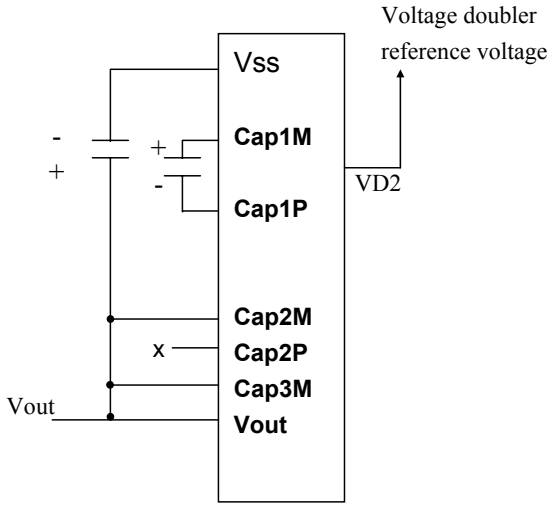
2. Two clock options:

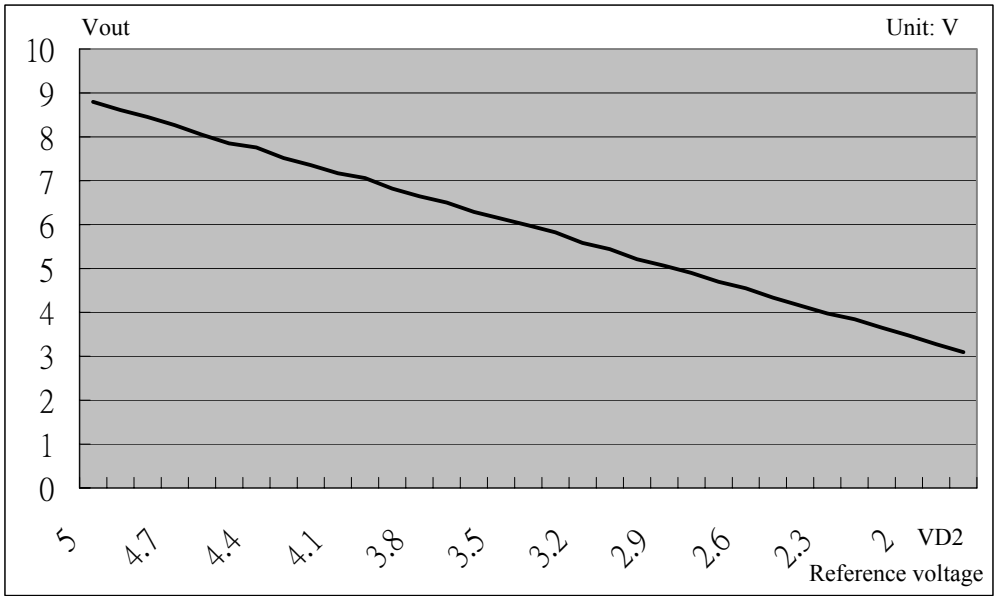
3.When using voltage doubler for VOUT it is recommended that the total sum of bleeder resistors R1~R5 should be larger than 20K Ohm











Doubler voltage mode VD2 & Vout output characteristic

Notes:
Follower loading resistor total 20k(ohm)
Boostaer Cap use 4.7uf
Panel size 80mm * 28mm (check display)

Function Description :

System interface
ST7920 supports 3 kinds of bus interface to MPU. 8 bits parallel, 4 bits parallel and clock synchronized serial interface. Parallel interface is selected by PSB="1" and serial interface by PSB="0". 8 bit / 4 bit interface is selected by function set instruction DL bit.
Two 8 bit registers (data register DR, instruction register IR) are used in ST7920's write and read operation. Data Register（DR）can access DDRAM/CGRAM/GDRAM and IRAM's data through the address pointer implemented by Address Counter (AC). Instruction Register (IR) stores the instruction by MPU to ST7920.

4 modes of read/write operation specified by RS and RW：

RS	RW	description
L	L	MPU write instruction to instruction register（IR）
L	H	MPU read busy flag（BF）and address counter（AC）
H	L	MPU write data to data register（DR）
H	H	MPU read data from data register（DR）

Busy Flag（BF）
Internal operation is in progress when BF="1", ST7920 is in busy state. No new instruction will be accepted until BF="0". MPU must check BF to determine whether the internal operation is finished and new instruction can be sent.

Address counter（AC）
Address counter（AC）is used for address pointer of DDRAM/CGRAM/IRAM/GDRAM. (AC) can be set by instruction and after data read or write to the memories (AC) will increase or decrease by 1 according to the setting in "entry mode set". When RS="0" and RW="1" and E="1" the value of（AC）will output to DB6～DB0.

16x16 character generation ROM (CGROM) and 8x16 half height ROM（HCGROM）

ST7920 provides character generation ROM supporting 8192 16 x 16 character fonts and 126 8 x 16 alphanumeric characters. It is easy to support multi languages application such as Chinese and English. Two consecutive bytes are used to specify one 16x16 character or two 8x16 half-height characters. Character codes are written into DDRAM and the corresponding fonts are mapped from CGROM or HCGROM to the display drivers.

Character generation RAM (CGRAM)

ST7920 provides RAM to support user-defined fonts. Four sets of 16x16 bit map area are available. These user-defined fonts are displayed the same ways as CGROM fonts through writing character cod data to DDRAM.

ICON RAM（IRAM）

ST7920 provides 240 ICON display. It consists of 15 sets of IRAM address. Each IRAM address has 16 bits data. IRAM address should be set first before writing to the IRAM. Two bytes for each address. First higher byte (D15~D8) and then lower byte (D7~D0).

Display data RAM（DDRAM）

There are 64x2 bytes for display data RAM area. Can store display data for 16 characters(16x16) by 4 lines or 32 characters(8x16) by 4 lines. However, only 2 lines can be displayed at a time. Character codes stored in DDRAM point to the fonts specified by CGROM，HCGROM and CGRAM. ST7920 display half height HCGROM fonts, user-defined CGRAM fonts and full 16x16 CGROM fonts. Data codes 0000H～0006H are for CGRAM user-defined fonts. Data codes 02H～7FH are for half height alpha numeric fonts. Data codes（A140～D75F）are for BIG5 code and (A1A0～F7FF) are for GB code.

1. display HCGROM fonts：Write 2 bytes data to DDRAM to display two 8x16 fonts. Each byte represents 1 character font. The data of each byte is 02H～7FH.
2. display CGRAM fonts：Write 2 bytes data to DDRAM to display one 16x16 font. Only 0000H，0002H，0004H， 0006H are allowed.
3. display CGROM fonts：Write 2 bytes data to DDRAM to display one 16x16 font. A140H～D75FH are for (BIG5) code, A1A0H～F7FFH are for (GB) code.

Higher byte（D15～D8）are written first and then lower byte（D7～D0）.

Refer to Table 5 for address map

CGRAM fonts and CGROM fonts can only be displayed in the start position of each address. (Refer toTable 4)

80		81		82		83		84		85		86		87		88		89		8A		8B		8C		8D		8E		8F	
H	L	H	L	H	L	H	L	H	L	H	L	H	L	H	L	H	L	H	L	H	L	H	L	H	L	H	L	H	L	H	L
S	**i**	**t**	**r**	**o**	**n**	**i**	**x**		**S**	**T**	**7**	**9**	**2**	**0**																	
矽		創		電		子		**.**	**.**	中		文		編		碼			**(**	正		確		**)**							
矽		創		電		子		**.**	**.**	**.**	中		文		編		碼														

Incorrect position

Table 4

Graphic RAM（GDRAM）

Graphic display RAM supports 64x256 bits bit-mapped memory space. GDRAM address is set by writing 2 consecutive bytes for vertical address and horizontal address. Two-bytes data write to GDRAM for one address. Address counter will automatically increase by one for the next two-byte data. The procedure is as followings.

1. Set vertical address（Y）for GDRAM
2. Set horizontal address（X）for GDRAM
3. Write D15～D8 to GDRAM 中(first byte)
4. Write D7～D0 to GDRAM 中(second byte)

Graphic display memory map please refer to Table-8

LCD driver

LCD driver have 33 common and 64 segments to drive the LCD panel. Segment data from CGRAM /CGROM /HCGROM are shifted into the 64 bits segment latches to display. Extended segment driver ST7921 can be used to extend the segment drivers to 256.

DDRAM data (char. code)					CGRAM Addr.						CGRAM data (higher byte)								CGRAM data (lower byte)							
B15~ B4	B3	B2	B1	B0	B5	B4	B3	B2	B1	B0	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
0	X	00	X	00	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	0	0	0	0	0
					0	0	0	1	1	1	1	1	1	1	1	1	0	0	0	1	0	0	0	0	0	0
					0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	1	0	0
					0	0	1	1	0	0	0	1	0	0	0	0	0	0	0	1	1	1	1	1	1	0
					0	1	0	0	0	0	1	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0
					0	1	0	1	0	0	1	1	1	1	0	0	0	0	1	0	0	0	0	1	0	0
					0	1	1	0	0	0	1	1	0	0	1	0	1	0	0	1	0	1	0	0	0	0
					0	1	1	1	1	0	1	0	1	0	0	1	1	0	0	1	0	0	1	0	0	0
					1	0	0	0	0	0	0	1	0	0	1	0	0	0	0	1	0	1	0	0	0	0
					1	0	0	1	0	0	0	1	0	0	1	0	0	0	0	0	0	0	1	0	0	0
					1	0	1	0	0	0	0	1	0	0	1	0	0	0	0	0	0	1	0	0	0	0
					1	0	1	1	0	0	0	1	1	1	1	0	0	0	0	0	0	1	0	0	0	0
					1	1	0	0	0	0	0	1	0	0	1	0	0	0	0	1	0	0	0	0	0	0
					1	1	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
					1	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
					1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	X	01	X	01	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	1	1	0
					0	0	0	1	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	1	0	0
					0	0	1	0	0	0	1	0	0	0	0	1	0	0	1	1	0	1	0	0	0	0
					0	0	1	1	0	1	0	1	1	1	0	1	1	0	1	0	0	1	0	0	0	0
					0	1	0	0	1	0	0	0	0	0	0	0	1	0	1	0	1	0	0	1	0	0
					0	1	0	1	0	1	1	1	1	1	1	1	0	0	1	0	0	1	0	0	1	0
					0	1	1	0	0	1	0	0	0	0	0	1	0	0	1	0	0	1	0	0	1	0
					0	1	1	1	0	1	1	1	1	1	1	1	0	0	1	0	0	1	0	0	1	0
					1	0	0	0	0	1	0	0	0	0	0	1	0	0	1	0	0	1	0	0	1	0
					1	0	0	1	0	1	1	1	1	1	1	1	0	0	1	0	0	1	0	0	1	0
					1	0	1	0	0	1	0	0	0	0	0	0	0	0	1	0	0	1	0	0	1	0
					1	0	1	1	0	1	1	1	1	1	1	1	1	0	1	0		0	0	1	0	0
					1	1	0	0	1	0	1	0	0	0	0	0	1	0	1	0	1	0	0	1	0	0
					1	1	0	1	1	0	1	0	1	1	1	1	1	1	1	0	0	1	1	1	0	0
					1	1	1	0	1	0	1	0	0	0	0	0	1	0	1	0	0	0	1	0	0	0
					1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 5： DDRAM data（character code）， CGRAM data / address map

Note：

1. DDRAM data (character code) bit1 and bit2 are the same as CGRAM address bit4 and bit5.
2. CGRAM address bit0 to bit3 specify total 16 rows. Row16 is for cursor display. The data in row 16 will be logical OR to the cursor.
3. CGRAM data for each address is 16 bits.
4. DDRAM data to select CGRAM bit4 to bit15 must be “0”. Bit0 and bit3 value are “don’t care”.

ICON RAM address Set SR "0", and then set IRAM address AC3….AC0				ICON RAM data															
				Higher byte								Lower byte							
AC3	AC2	AC1	AC0	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	SEG0	SEG1	SEG2	SEG3	SEG4	SEG5	SEG6	SEG7	SEG8	SEG9	SEG10	SEG11	SEG12	SEG13	SEG14	SEG15
0	0	0	1	SEG16	SEG17	SEG18	SEG19	SEG20	SEG21	SEG22	SEG23	SEG24	SEG25	SEG26	SEG27	SEG28	SEG29	SEG30	SEG31
0	0	1	0	SEG32	SEG33	SEG34	SEG35	SEG36	SEG37	SEG38	SEG39	SEG40	SEG41	SEG42	SEG43	SEG44	SEG45	SEG46	SEG47
0	0	1	1	SEG48	SEG49	SEG50	SEG51	SEG52	SEG53	SEG54	SEG55	SEG56	SEG57	SEG58	SEG59	SEG60	SEG61	SEG62	SEG63
0	1	0	0	SEG64	SEG65	SEG66	SEG67	SEG68	SEG69	SEG70	SEG71	SEG72	SEG73	SEG74	SEG75	SEG76	SEG77	SEG78	SEG79
0	1	0	1	SEG80	SEG81	SEG82	SEG83	SEG84	SEG85	SEG86	SEG87	SEG88	SEG89	SEG90	SEG91	SEG92	SEG93	SEG94	SEG95
0	1	1	0	SEG96	SEG97	SEG98	SEG99	SEG100	SEG101	SEG102	SEG103	SEG104	SEG105	SEG106	SEG107	SEG108	SEG109	SEG110	SEG111
0	1	1	1	SEG112	SEG113	SEG114	SEG115	SEG116	SEG117	SEG118	SEG119	SEG120	SEG121	SEG122	SEG123	SEG124	SEG125	SEG126	SEG127
1	0	0	0	SEG128	SEG129	SEG130	SEG131	SEG132	SEG133	SEG134	SEG135	SEG136	SEG137	SEG138	SEG139	SEG140	SEG141	SEG142	SEG143
1	0	0	1	SEG144	SEG145	SEG146	SEG147	SEG148	SEG149	SEG150	SEG151	SEG152	SEG153	SEG154	SEG155	SEG156	SEG157	SEG158	SEG159
1	0	1	0	SEG160	SEG161	SEG162	SEG163	SEG164	SEG165	SEG166	SEG167	SEG168	SEG169	SEG170	SEG171	SEG172	SEG173	SEG174	SEG175
1	0	1	1	SEG176	SEG177	SEG178	SEG179	SEG180	SEG181	SEG182	SEG183	SEG184	SEG185	SEG186	SEG187	SEG188	SEG189	SEG190	SEG191
1	1	0	0	SEG192	SEG193	SEG194	SEG195	SEG196	SEG197	SEG198	SEG199	SEG200	SEG201	SEG202	SEG203	SEG204	SEG205	SEG206	SEG207
1	1	0	1	SEG208	SEG209	SEG210	SEG211	SEG212	SEG213	SEG214	SEG215	SEG216	SEG217	SEG218	SEG219	SEG220	SEG221	SEG222	SEG223
1	1	1	0	SEG224	SEG225	SEG226	SEG227	SEG228	SEG229	SEG230	SEG231	SEG232	SEG233	SEG234	SEG235	SEG236	SEG237	SEG238	SEG239
1	1	1	1	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Table 6 ICON RAM address, data and Segment pins

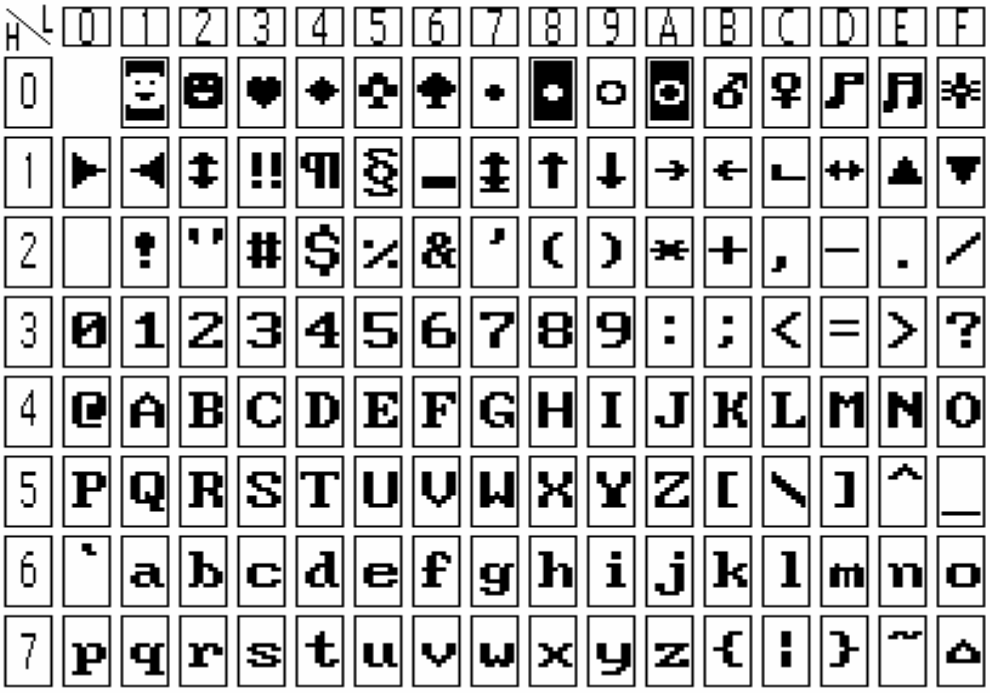


Table 7 16x8 half-height characters

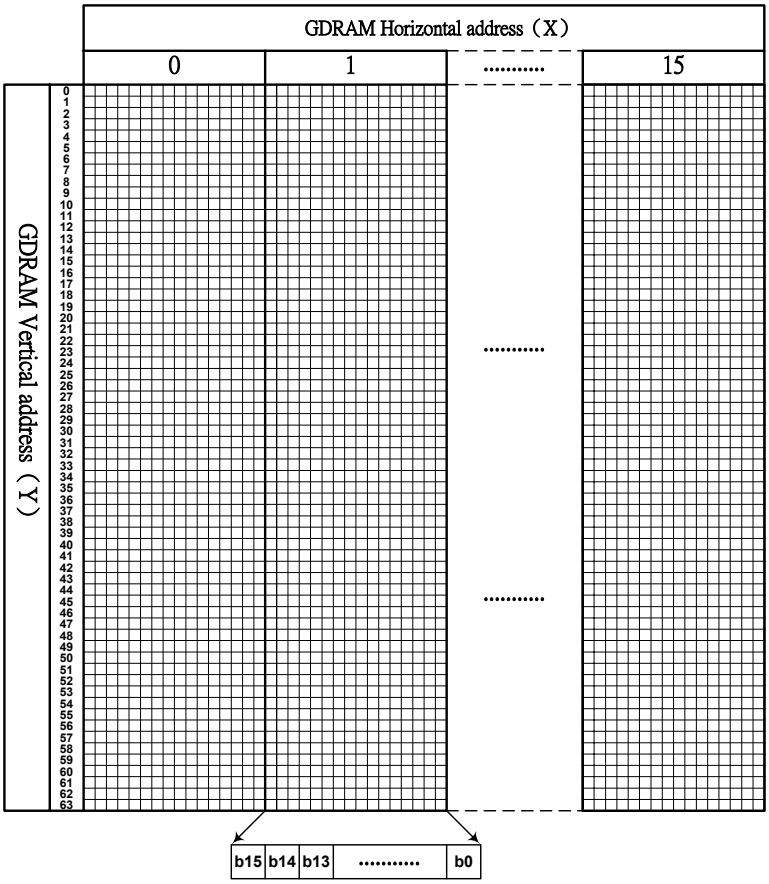


Table 8 GDRAM display coordinates and corresponding address

Instructions

ST7920 offers basic instruction set and extended instruction set：

Instruction set 1: (RE=0: basic instruction)

Ins	code										Description	Exec time (540KHZ)
	RS	RW	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
CLEAR	0	0	0	0	0	0	0	0	0	1	Fill DDRAM with "20H", and set DDRAM address counter （AC）to "00H"	1.6 ms
HOME	0	0	0	0	0	0	0	0	1	X	Set DDRAM address counter（AC）to "00H", and put cursor to origin ；the content of DDRAM are not changed	72us
ENTRY MODE	0	0	0	0	0	0	0	1	I/D	S	Set cursor position and display shift when doing write or read operation	72us
DISPLAY ON/OFF	0	0	0	0	0	0	1	D	C	B	D=1: display ON C=1: cursor ON B=1: blink ON	72 us
CURSOR DISPLAY CONTROL	0	0	0	0	0	1	S/C	R/L	X	X	Cursor position and display shift control ；the content of DDRAM are not changed	72 us
FUNCTION SET	0	0	0	0	1	DL	X	0 RE	X	X	DL=1 8-BIT interface DL=0 4-BIT interface **RE=1: extended instruction** **RE=0: basic instruction**	72 us
SET CGRAM ADDR.	0	0	0	1	AC5	AC4	AC3	AC2	AC1	AC0	Set CGRAM address to address counter（AC） **Make sure that in extended instruction SR=0 (scroll or RAM address select)**	72 us
SET DDRAM ADDR.	0	0	1	0 AC6	AC5	AC4	AC3	AC2	AC1	AC0	Set DDRAM address to address counter（AC） AC6 is fixed to 0	72 us
READ BUSY FLAG（BF）& ADDR.	0	1	BF	AC6	AC5	AC4	AC3	AC2	AC1	AC0	Read busy flag（BF）for completion of internal operation, also Read out the value of address counter（AC）	0 us
WRITE RAM	1	0	D7	D6	D5	D4	D3	D2	D1	D0	Write data to internal RAM (DDRAM/CGRAM/IRAM/GDRAM)	72 us
READ RAM	1	1	D7	D6	D5	D4	D3	D2	D1	D0	Read data from internal RAM (DDRAM/CGRAM/IRAM/GDRAM)	72 us

Instruction set 2: (RE=1: extended instruction)

Inst.	code										description	Exec. time (540KHZ)
	RS	RW	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
STAND BY	0	0	0	0	0	0	0	0	0	1	Enter stand by mode, any other instruction can terminate (Com1..32 halted, only Com33 ICON can display)	72 us
SCROLL or RAM ADDR. SELECT	0	0	0	0	0	0	0	0	1	SR	SR=1: enable vertical scroll position SR=0: enable IRAM address **(extended instruction)** SR=0: enable CGRAM address**(basic instruction)**	72 us
REVERSE	0	0	0	0	0	0	0	1	R1	R0	Select 1 out of 4 line (in DDRAM) and decide whether to reverse the display by toggling this instruction **R1,R0 initial value is 00**	72 us
EXTENDED FUNCTION SET	0	0	0	0	1	DL	X	1 RE	G	0	DL=1 8-BIT interface DL=0 4-BIT interface **RE=1: extended instruction set** **RE=0: basic instruction set** G=1 :graphic display ON G=0 :graphic display OFF	72 us
SET IRAM or SCROLL ADDR	0	0	0	1	AC5	AC4	AC3	AC2	AC1	AC0	SR=1: AC5~AC0 the address of vertical scroll SR=0: AC3~AC0 the address of ICON RAM	72 us
SET GRAPHIC RAM ADDR.	0	0	1	0 AC6	0 AC5	0 AC4	AC3 AC3	AC2 AC2	AC1 AC1	AC0 AC0	Set GDRAM address to address counter（AC） First set vertical address and the horizontal address by consecutive writing Vertical address range AC6...AC0 Horizontal address range AC3…AC0	72 us

Note：

1. Make sure that ST7920 is not in busy state by reading the busy flag before sending instruction or data. If use delay loop instead please make sure the delay time is enough. Please refer to the instruction execution time.
2. “RE” is the selection bit of basic and extended instruction set. Each time when altering the value of RE it will remain. There is no need to set RE every time when using the same group of instruction set.

Initial setting(Register flag) (RE=0: basic instruction)

Inst.	code										Description
	RS	RW	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	
ENTRY MODE SET	0	0	0	0	0	0	0	1	I/D	S	Cursor move to right ,DDRAM address counter（AC）plus 1
									1	**0**	
DISPLAY STATUS	0	0	0	0	0	0	1	D	C	B	Display, cursor and blink ALL OFF
								0	**0**	**0**	
CURSOR DISPLAY SHIFT	0	0	0	0	0	1	S/C	R/L	X	X	No cursor or display shift operation
							X	**X**			
FUNCTION SET	0	0	0	0	1	DL	X	0 RE	X	X	**8 BIT** MPU interface , basic instruction set
						1		**0**			

Initial setting(Register flag) (RE=1: extended instruction set)

Inst.	code										description
	RS	RW	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	
SCROLL OR RAM ADDR. SELECT	0	0	0	0	0	0	0	0	1	SR	Allow IRAMaddress or set CGRAM address
										0	
REVERSE	0	0	0	0	0	0	0	1	R1	R0	Begin with normal and toggle to reverse
									0	**0**	
EXTENDED FUNCTION SET	0	0	0	0	1	DL	X	1 RE	G	0	Graphic display OFF
									0		

Description of basic instruction set

- **CLEAR**

	RS	RW	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Code	0	0	0	0	0	0	0	0	0	1

Fill DDRAM with "20H"(space code). And set DDRAM address counter（AC）to"00H". Set entry mode I/D bit to be "1". Cursor moves right and AC adds 1 after write or read operation.

- **HOME**

	RS	RW	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Code	0	0	0	0	0	0	0	0	1	x

Set DDRAM address counter（AC）to "00H". Cursor moves to origin. Then content of DDRAM is not changed.

- **ENTRY MODE SET**

	RS	RW	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Code	0	0	0	0	0	0	0	1	I/D	S

Set the cursor movement and display shift direction when doing write or read operation.

I/D :address counter increase / decrease

When I/D = "1", cursor moves right, DRAM address counter（AC）add by 1.

When I/D = "0", cursor moves left, DRAM address counter（AC）subtract by 1.

S: Display shift

S	I/D	DESCRIPTION
H	H	Entire display shift left by 1
H	L	Entire display shift right by 1

- **DISPLAY STATUS**

	RS	RW	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Code	0	0	0	0	0	0	1	D	C	B

Controls display, cursor and blink ON/OFF.

D : Display ON/OFF control bit

When D = "1", display ON

When D = "0",display OFF , the content of DDRAM is not changed

C : Cursor ON/OFF control bit

When C = "1", cursor ON.

When C = "0", cursor OFF.

B : Blink ON/OFF control bit

When B = "1", cursor position blink ON. Then display data in cursor position will blink.

When B = "0", cursor position blink OFF

- **CURSOR AND DISPLAY SHIFT CONTROL**

	RS	RW	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Code	0	0	0	0	0	1	S/C	R/L	x	x

Instruction to move the cursor or shift the entire display. The content of DDRAM is not changed.

S/C	R/L	Description	AC Value
L	L	Cursor moves left by 1	AC=AC-1
L	H	Cursor moves right by 1	AC=AC+1
H	L	Display shift left by 1, cursor also follows to shift.	AC=AC
H	H	Display shift right by 1, cursor also follows to shift.	AC=AC

- **FUNCTION SET**

	RS	RW	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Code	0	0	0	0	1	DL	X	RE	x	x

DL : 4/8 BIT interface control bit

When DL = "1", **8 BIT** MPU bus interface

When DL = "0", 爲 **4 BIT** MPU bus interface

RE : extended instruction set control bit

When RE = "1", extended instruction set

When RE = "0", basic instruction set

In same instruction cannot alter DL and RE at once. Make sure that change DL first then RE.

- **SET CGRAM ADDRESS**

	RS	RW	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Code	0	0	0	1	AC5	AC4	AC3	AC2	AC1	AC0

Set CGRAM address to address counter（AC）

AC range is 00H..3FH

Make sure that in extended instruction SR=0 (scroll address or RAM address select)

- **SET DDRAM ADDRESS**

	RS	RW	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Code	0	0	1	AC6	AC5	AC4	AC3	AC2	AC1	AC0

Set DDRAM address to address counter（AC）.

First line AC range is 80H..8FH

Second line AC range is 90H..9FH

Third line AC range is A0H..AFH

Fourth line AC range is B0H..BFH

Please note that only 2 lines can be display at a time.

- **READ BUSY FLAG（BF）AND ADDRESS**

	RS	RW	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Code	0	1	BF	AC6	AC5	AC4	AC3	AC2	AC1	AC0

Read busy flag（BF）can check whether internal operation is finished. At the same time the value of address counter

（AC）is also read. When BF = “1” new instruction will not be accepted. Must wait for BF = “0” for new instruction.

- **WRITE DATA TO RAM**

	RS	RW	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Code	1	0	D7	D6	D5	D4	D3	D2	D1	D0

Write data to internal RAM and alter the (AC) by 1

Each RAM address (CGRAM,DDRAM,IRAM…..) must write 2 consecutive bytes for 16 bit data. After the second byte the address counter will add or subtract by 1 according to the entry mode set control bit.

- **READ RAM DATA**

	RS	RW	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Code	1	1	D7	D6	D5	D4	D3	D2	D1	D0

Read data from internal RAM and alter the (AC) by 1

After address set to read (CGRAM,DDRAM,IRAM…..)a DUMMY READ is required.

There is no need to DUMMY READ for the following bytes unless a new address set instruction is issued.

Description of extended instruction set

- **STAND BY**

	RS	RW	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Code	0	0	0	0	0	0	0	0	0	1

Instruction to enter stand by mode. Any other instruction follows this instruction can terminate stand by.

The content of DDRAM remain the same.

- **VERTICAL SCROLL OR RAM ADDRESS SELECT**

	RS	RW	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Code	0	0	0	0	0	0	0	0	1	SR

When SR = "1", the vertical scroll address set is enabled.

When SR = "0", the IRAM address set **(extended instruction)** and CGRAM address set**(basic instruction)** is enabled.

- **REVERSE**

	RS	RW	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Code	0	0	0	0	0	0	0	1	R1	R0

Select 1 out of 4 lines to reverse the display and to toggle the reverse condition by repeating this instruction.

R1,R0 initial vale is 00. When set the first time the display is reversed and set the second time the display become normal.

R1	R0	Description
L	L	First line normal or reverse
L	H	Second line normal or reverse
H	L	Third line normal or reverse
H	H	Fourth line normal or reverse

Please note that only 2 lines out of 4 line display data can be displayed.

- **EXTENED FUNCTION SET**

	RS	RW	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Code	0	0	0	0	1	DL	x	RE	G	x

DL : 4/8 BIT interface control bit

When DL = "1", **8 BIT** MPU interface

When DL = "0", **4 BIT** MPU interface

RE : extended instruction set control bit

When RE = "1", extended instruction set

When RE = "0", basic instruction set

G : Graphic display control bit

When G = "1", graphic display ON

When G = "0", Graphic display OFF

In same instruction cannot alter DL, RE and G at once. Make sure that change DL or G first and then RE.

- **SET IRAM OR SCROLL ADDRESS**

	RS	RW	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Code	0	0	0	1	AC5	AC4	AC3	AC2	AC1	AC0

SR=1: AC5~AC0 is vertical scroll displacement address

SR=0: AC3~AC0 is ICON RAM address

- **SET GRAPHIC RAM ADDRESS**

	RS	RW	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Code	0	0	1	AC6	AC5	AC4	AC3	AC2	AC1	AC0

Set GDRAM address to address counter（AC）.

First set vertical address and then horizontal address(write 2 consecutive bytes to complete vertical and horizontal address set)

Vertical address range is AC6...AC0

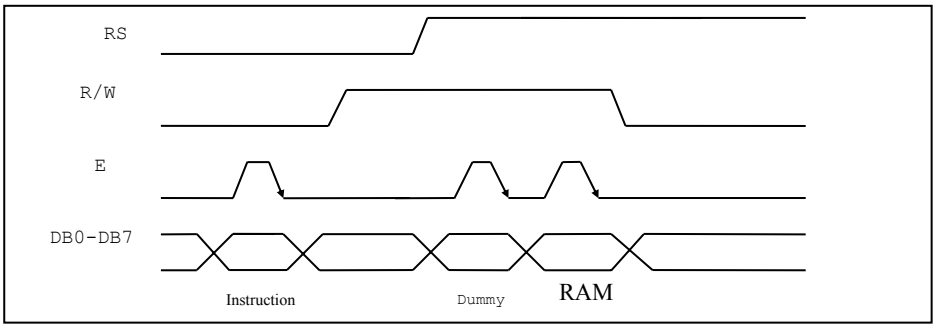
Horizontal address range is AC3…AC0

The address counter（AC）of graphic RAM(GRAM) only increment after write for horizontal address. After horizontal address =0FH it will automatically back to 00H. However, the vertical address will not increase as the result of the same action.

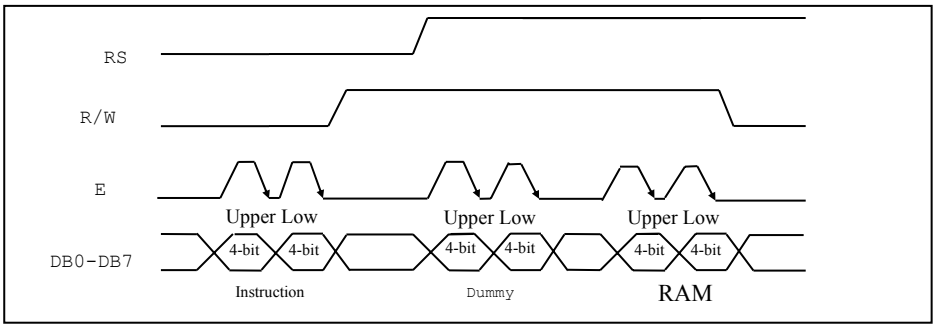
Parallel interface :

ST7920 is in parallel mode by pulling up PSB pin. And can select 8 bit or 4-bit bus interface by function set instruction DL control bit. MPU can control (RS , RW , E , and DB0..DB7) pins to complete the data transmission.

In 4-bit transfer mode, every 8 bits data or instruction is separated into 2 parts. Higher 4 bits（DB7~DB4）data will transfer First and placed into data pins（DB7~DB4）. Lower 4 bits（DB3~DB0）data will transfer second and placed into data pins （DB7~DB4）. (DB3~DB0) data pins are not used.



Timing Diagram of 8-bit Parallel Bus Mode Data Transfer



Timing Diagram of 4-bit Parallel Bus Mode Data Transfer

Serial interface :

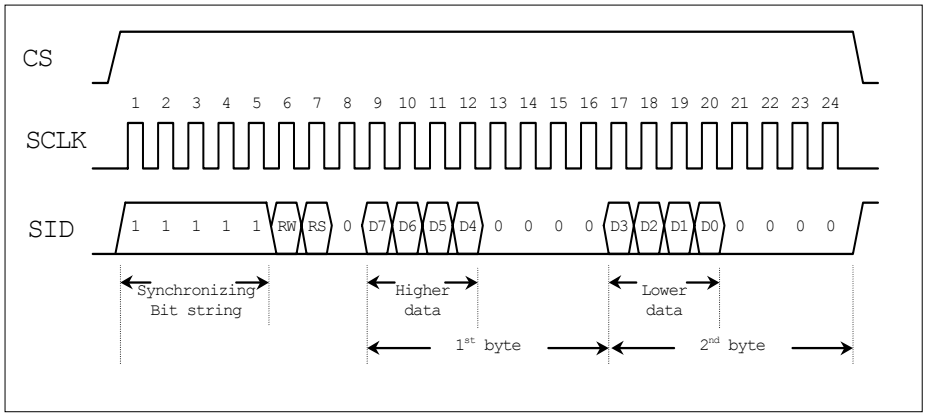
ST7920 is in serial interface mode when pull down PSB pin. Two pins (SCLK and SID) are used to complete the data transfer. Only write data is available.

When connecting several ST7920, chip select（CS）must be used. Only when（CS）is high the serial clock（SCLK）can be accepted. On the other hand, when chip select（CS）is low ST7920 serial counter and data will be reset. Transmission will be terminated and data will be cleared. Serial transfer counter is set to the first bit. For a minimal system with only one ST7920 and one MPU, only SCLK and SID pins are necessary. CS pin should pull to high.

ST7920's serial clock（SCLK）is asynchronous to the internal clock and is generated by MPU. When multiple instruction/data is transferred instruction execution time must be considered. Must wait for the previous instruction to finish before sending the next. ST7920 has no internal instruction buffer area.

When starting a transmission a start byte is required. It consists of 5 consecutive "1"（sync character）. Serial transfer counter will be reset and synchronized. Following 2 bits for read/write（RW）and register/data select（RS）. Last 4 bits is filled by "0"。

After receiving the sync character and RW and RS bits, every 8 bits instruction/data will be separated into 2 groups. Higher 4 bits（DB7~DB4）will be placed in first section followed by 4 "0". And lower 4 bits（DB3~DB0）will be placed in second section followed by 4 "0".



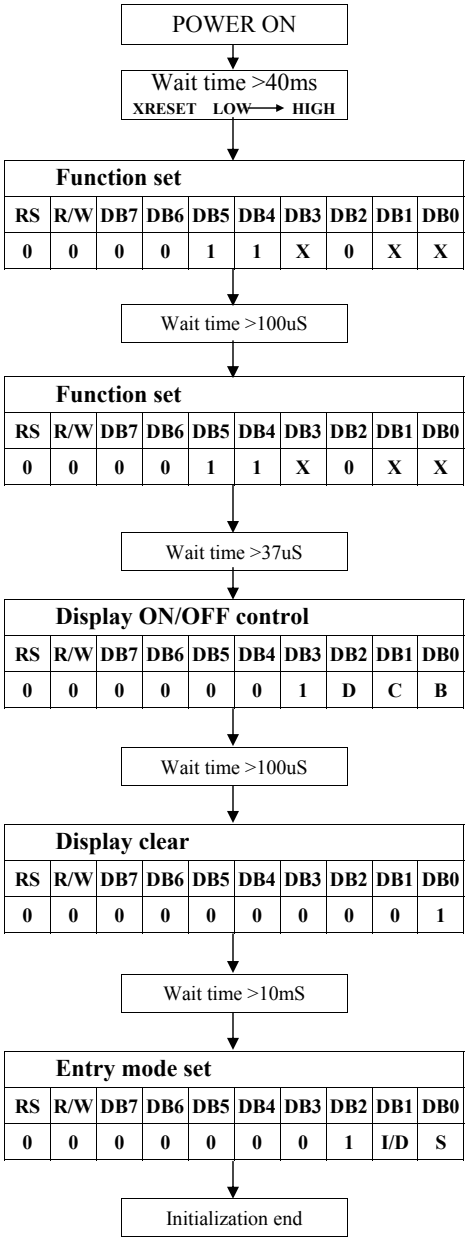
Timing Diagram of Serial Mode Data Transfer

8051 demo program for serial interface

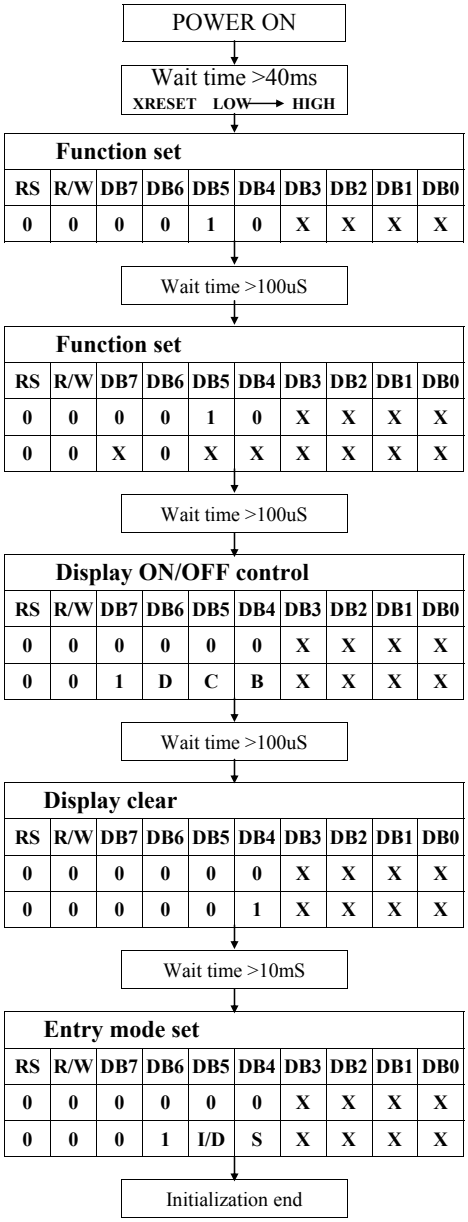
```
;-------------------------------------------------------
;    Write data from A into INSTRUCTION Register
;-------------------------------------------------------
WRINS:
        SETB    CS
        SETB    SID         ; SID = 1
        CLR     SCLK
        SETB    SCLK        ; READ DATA FROM SID
        CLR     SCLK
        SETB    SCLK        ; READ DATA FROM SID
        CLR     SCLK
        SETB    SCLK        ; READ DATA FROM SID
        CLR     SCLK
        SETB    SCLK        ; READ DATA FROM SID
        CLR     SCLK
        SETB    SCLK        ; READ DATA FROM SID
        CLR     SCLK
        SETB    SCLK        ; READ DATA FROM SID
        CLR     SCLK
        CLR     SID         ; SID = 0
        SETB    SCLK        ; READ DATA FROM SID
        CLR     SCLK
        SETB    SCLK        ; READ DATA FROM SID
        CLR     SCLK
        SETB    SCLK        ; READ DATA FROM SID
        CLR     SCLK
        MOVBIT  SID, A.7    ; SID = A.7
        SETB    SCLK        ; READ DATA FROM SID
        CLR     SCLK
        MOVBIT  SID, A.6    ; SID = A.6
        SETB    SCLK        ; READ DATA FROM SID
        CLR     SCLK
        MOVBIT  SID, A.5    ; SID = A.5
        SETB    SCLK        ; READ DATA FROM SID
        CLR     SCLK
        MOVBIT  SID, A.4    ; SID = A.4
        SETB    SCLK        ; READ DATA FROM SID
        CLR     SCLK
        CLR     SID         ; SID = 0
        SETB    SCLK        ; READ DATA FROM SID
        CLR     SCLK
        SETB    SCLK        ; READ DATA FROM SID
        CLR     SCLK
        SETB    SCLK        ; READ DATA FROM SID
        CLR     SCLK
        SETB    SCLK        ; READ DATA FROM SID
        CLR     SCLK
        MOVBIT  SID, A.3    ; SID = A.3
        SETB    SCLK        ; READ DATA FROM SID
        CLR     SCLK
        MOVBIT  SID, A.2    ; SID = A.2
        SETB    SCLK        ; READ DATA FROM SID
        CLR     SCLK
        MOVBIT  SID, A.1    ; SID = A.1
        SETB    SCLK        ; READ DATA FROM SID
        CLR     SCLK
        MOVBIT  SID, A.0    ; SID = A.0
        SETB    SCLK        ; READ DATA FROM SID
        CLR     SCLK
        CLR     SID         ; SID = 0
        SETB    SCLK        ; READ DATA FROM SID
        CLR     SCLK
        SETB    SCLK        ; READ DATA FROM SID
        CLR     SCLK
        SETB    SCLK        ; READ DATA FROM SID
        CLR     SCLK
        SETB    SCLK        ; READ DATA FROM SID
        CLR     SCLK
        CLR     CS
        CALL    DLY8
        RET
```

```
;-------------------------------------------------------
;    Write data from A into DATA Register
;-------------------------------------------------------
WRDATA:
        SETB    CS
        SETB    SID         ; SID = 1
        CLR     SCLK
        SETB    SCLK        ; READ DATA FROM SID
        CLR     SCLK
        SETB    SCLK        ; READ DATA FROM SID
        CLR     SCLK
        SETB    SCLK        ; READ DATA FROM SID
        CLR     SCLK
        SETB    SCLK        ; READ DATA FROM SID
        CLR     SCLK
        SETB    SCLK        ; READ DATA FROM SID
        CLR     SCLK
        SETB    SCLK        ; READ DATA FROM SID
        CLR     SCLK
        CLR     SID         ; SID = 0
        SETB    SCLK        ; READ DATA FROM SID
        CLR     SCLK
        SETB    SID         ; SID = 1
        SETB    SCLK        ; READ DATA FROM SID
        CLR     SCLK
        CLR     SID         ; SID = 0
        SETB    SCLK        ; READ DATA FROM SID
        CLR     SCLK
        MOVBIT  SID, A.7    ; SID = A.7
        SETB    SCLK        ; READ DATA FROM SID
        CLR     SCLK
        MOVBIT  SID, A.6    ; SID = A.6
        SETB    SCLK        ; READ DATA FROM SID
        CLR     SCLK
        MOVBIT  SID, A.5    ; SID = A.5
        SETB    SCLK        ; READ DATA FROM SID
        CLR     SCLK
        MOVBIT  SID, A.4    ; SID = A.4
        SETB    SCLK        ; READ DATA FROM SID
        CLR     SCLK
        CLR     SID         ; SID = 0
        SETB    SCLK        ; READ DATA FROM SID
        CLR     SCLK
        SETB    SCLK        ; READ DATA FROM SID
        CLR     SCLK
        SETB    SCLK        ; READ DATA FROM SID
        CLR     SCLK
        SETB    SCLK        ; READ DATA FROM SID
        CLR     SCLK
        MOVBIT  SID, A.3    ; SID = A.3
        SETB    SCLK        ; READ DATA FROM SID
        CLR     SCLK
        MOVBIT  SID, A.2    ; SID = A.2
        SETB    SCLK        ; READ DATA FROM SID
        CLR     SCLK
        MOVBIT  SID, A.1    ; SID = A.1
        SETB    SCLK        ; READ DATA FROM SID
        CLR     SCLK
        MOVBIT  SID, A.0    ; SID = A.0
        SETB    SCLK        ; READ DATA FROM SID
        CLR     SCLK
        CLR     SID         ; SID = 0
        SETB    SCLK        ; READ DATA FROM SID
        CLR     SCLK
        SETB    SCLK        ; READ DATA FROM SID
        CLR     SCLK
        SETB    SCLK        ; READ DATA FROM SID
        CLR     SCLK
        SETB    SCLK        ; READ DATA FROM SID
        CLR     SCLK
        CLR     CS
        CALL    DLY8
        RET
```

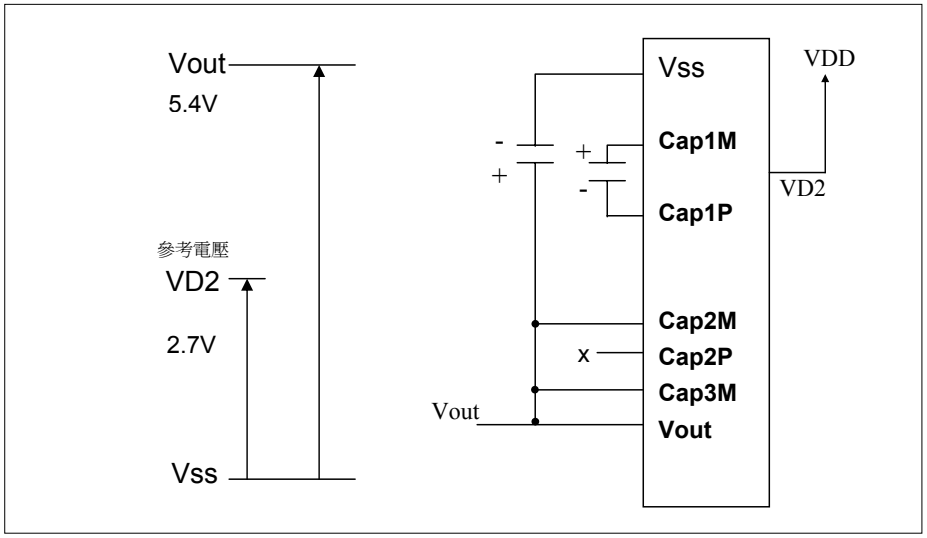
8 bit interface：



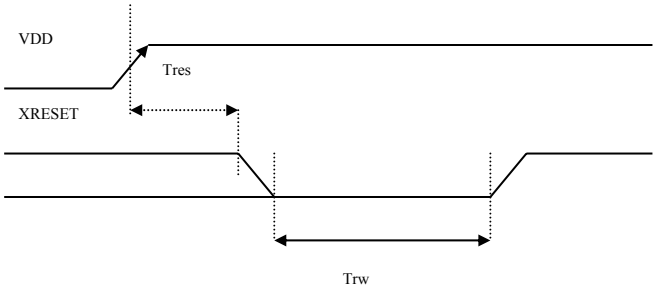
4 bit interface：



Built in voltage booster



External reset timing

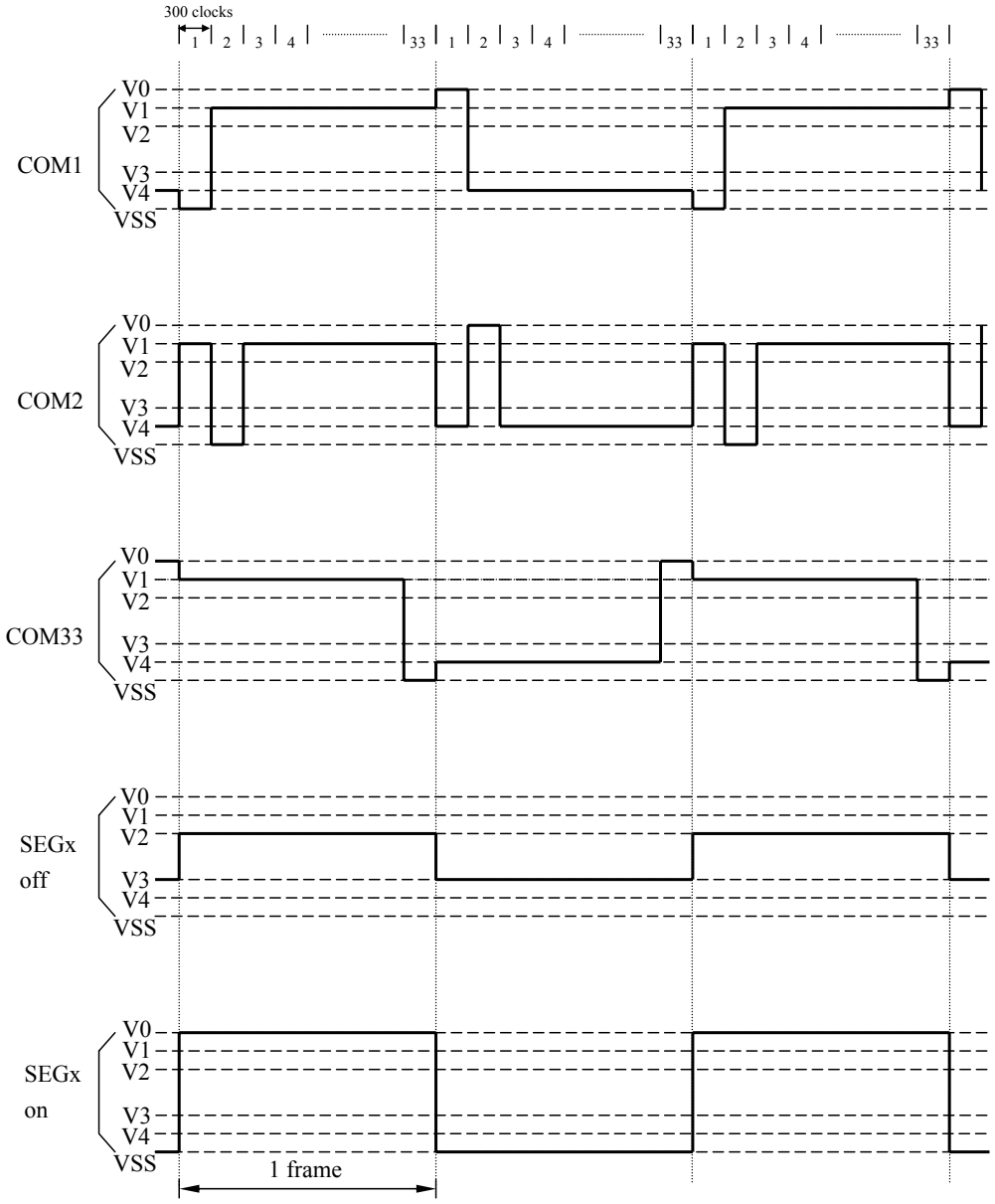


XRESET pulse width	Trw	10us
RESET start time	Tres	50ns

LCD driving wave form (1/33 duty , 1/5 bias)

When oscillation frequency is 540KHZ, 1 clock cycle time = 1.85us

1 frame = 1.85us x 300 x 33 = 18315us=18.3ms



Absolute Maximum Ratings

Characteristics	Symbol	Value
Power Supply Voltage	V_{DD}	-0.3V to +5.5V
LCD Driver Voltage	V_{LCD}	-0.3V to +7.0V
Input Voltage	V_{IN}	-0.3V to V_{DD}+0.3V
Operating Temperature	T_A	-20°C to + 85°C
Storage Temperature	T_{STO}	-55°C to + 125°C

DC Characteristics (T_A = 25°C , V_{DD} = 2.7 V – 4.5 V)

Symbol	Characteristics	Test Condition	Min.	Typ.	Max.	Unit
V_{DD}	Operating Voltage	-	2.7	-	5.5	V
V_{LCD}	LCD Voltage	V0-V_{SS}	3.0	-	7	V
I_{CC}	Power Supply Current	f_{OSC} = 530KHz, V_{DD}=3.0V Rf=18KΩ	-	0.20	0.45	mA
V_{IH1}	Input High Voltage (Except OSC1)	-	0.7V_{DD}	-	V_{DD}	V
V_{IL1}	Input Low Voltage (Except OSC1)	-	- 0.3	-	0.6	V
V_{IH2}	Input High Voltage (OSC1)	-	V_{DD} – 1	-	V_{DD}	V
V_{IL2}	Input Low Voltage (OSC1)	-	-	-	1.0	V
V_{OH1}	Output High Voltage (DB0 - DB7)	I_{OH} = -0.1mA	0.8V_{DD}	-	V_{DD}	V
V_{OL1}	Output Low Voltage (DB0 - DB7)	I_{OL} = 0.1mA	-	-	0.1	V
V_{OH2}	Output High Voltage (Except DB0 - DB7)	I_{OH} = -0.04mA	0.8V_{DD}	-	V_{DD}	V
V_{OL2}	Output Low Voltage (Except DB0 - DB7)	I_{OL} = 0.04mA	-	-	0.1V_{DD}	V
I_{LEAK}	Input Leakage Current	V_{IN} = 0V to V_{DD}	-1	-	1	µA
I_{PUP}	Pull Up MOS Current	V_{DD} = 3V	22	27	32	µA

DC Characteristics ($T_A = 25^oC$, $V_{DD} = 4.5$ V - 5.5 V)

Symbol	Characteristics	Test Condition	Min.	Typ.	Max.	Unit
V_{DD}	Operating Voltage	-	4.5	-	5.5	V
V_{LCD}	LCD Voltage	V0-V_{SS}	3.0	-	7	V
I_{CC}	Power Supply Current	f_{OSC} = 540KHz, V_{DD}=5V Rf=33KΩ	-	0.45	0.75	mA
V_{IH1}	Input High Voltage (Except OSC1)	-	0.7V_{DD}	-	V_{DD}	V
V_{IL1}	Input Low Voltage (Except OSC1)	-	-0.3	-	0.6	V
V_{IH2}	Input High Voltage (OSC1)	-	V_{DD}-1	-	V_{DD}	V
V_{IL2}	Input Low Voltage (OSC1)	-	-	-	1.0	V
V_{OH1}	Output High Voltage (DB0 - DB7)	I_{OH} = -0.1mA	0.8V_{DD}	-	V_{DD}	V
V_{OL1}	Output Low Voltage (DB0 - DB7)	I_{OL} = 0.1mA	-	-	0.4	V
V_{OH2}	Output High Voltage (Except DB0 - DB7)	I_{OH} = -0.04mA	0.8V_{DD}	-	V_{DD}	V
V_{OL2}	Output Low Voltage (Except DB0 - DB7)	I_{OL} = 0.04mA	-	-	0.1V_{DD}	V
I_{LEAK}	Input Leakage Current	V_{IN} = 0V to V_{DD}	-1	-	1	µA
I_{PUP}	Pull Up MOS Current	V_{DD} = 5V	75	80	85	µA

AC Characteristics (T_A = 25°C, V_{DD} = 4.5V) Parallel Mode Interface

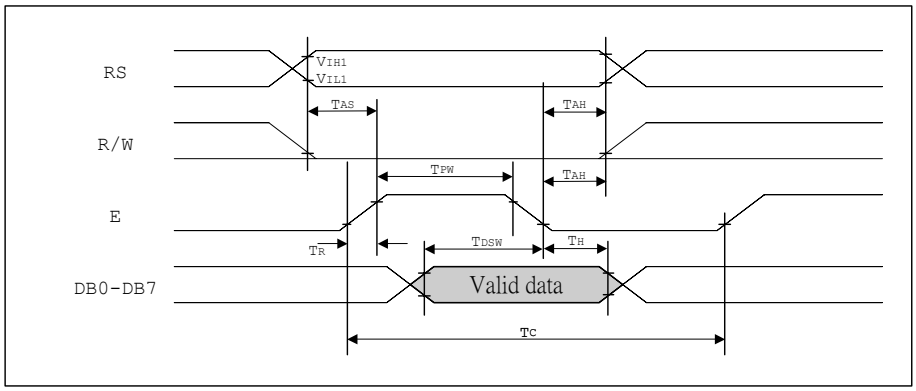
Symbol	Characteristics	Test Condition	Min.	Typ.	Max.	Unit
Internal Clock Operation						
f_{OSC}	OSC Frequency	R = 33KΩ	480	540	600	KHz
External Clock Operation						
f_{EX}	External Frequency	-	480	540	600	KHz
	Duty Cycle	-	45	50	55	%
T_R,T_F	Rise/Fall Time	-	-	-	0.2	μs
Write Mode (Writing data from MPU to ST7920)						
T_C	Enable Cycle Time	Pin E	1200	-	-	ns
T_{PW}	Enable Pulse Width	Pin E	140	-	-	ns
T_R,T_F	Enable Rise/Fall Time	Pin E	-	-	25	ns
T_{AS}	Address Setup Time	Pins: RS,RW,E	10	-	-	ns
T_{AH}	Address Hold Time	Pins: RS,RW,E	20	-	-	ns
T_{DSW}	Data Setup Time	Pins: DB0 - DB7	40	-	-	ns
T_H	Data Hold Time	Pins: DB0 - DB7	20	-	-	ns
Read Mode (Reading Data from ST7920 to MPU)						
T_C	Enable Cycle Time	Pin E	1200	-	-	ns
T_{PW}	Enable Pulse Width	Pin E	140	-	-	ns
T_R,T_F	Enable Rise/Fall Time	Pin E	-	-	25	ns
T_{AS}	Address Setup Time	Pins: RS,RW,E	10	-	-	ns
T_{AH}	Address Hold Time	Pins: RS,RW,E	20	-	-	ns
T_{DDR}	Data Delay Time	Pins: DB0 - DB7	-	-	100	ns
T_H	Data Hold Time	Pins: DB0 - DB7	20	-	-	ns
Interface Mode with LCD Driver(ST7921)						
T_{CWH}	Clock Pulse with High	Pins: CL1, CL2	800	-	-	ns
T_{CWL}	Clock Pulse with Low	Pins: CL1, CL2	800	-	-	ns
T_{CST}	Clock Setup Time	Pins: CL1, CL2	500	-	-	ns
T_{SU}	Data Setup Time	Pin: D	300	-	-	ns
T_{DH}	Data Hold Time	Pin: D	300	-	-	ns
T_{DM}	M Delay Time	Pin: M	-1000	-	1000	ns

AC Characteristics (T_A = 25°C, V_{DD} = 2.7V) Parallel Mode Interface

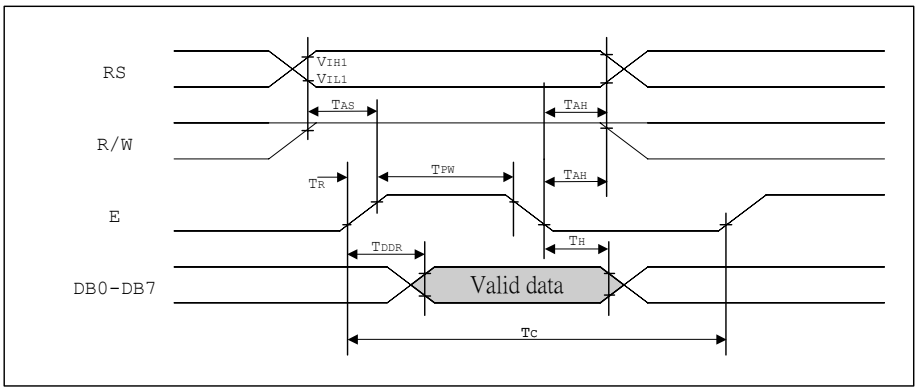
Symbol	Characteristics	Test Condition	Min.	Typ.	Max.	Unit
Internal Clock Operation						
f_{OSC}	OSC Frequency	R = 18KΩ	470	530	590	KHz
External Clock Operation						
f_{EX}	External Frequency	-	470	530	590	KHz
	Duty Cycle	-	45	50	55	%
T_R,T_F	Rise/Fall Time	-	-	-	0.2	μs
Write Mode (Writing data from MPU to ST7920)						
T_C	Enable Cycle Time	Pin E	1800	-	-	ns
T_{PW}	Enable Pulse Width	Pin E	160	-	-	ns
T_R,T_F	Enable Rise/Fall Time	Pin E	-	-	25	ns
T_{AS}	Address Setup Time	Pins: RS,RW,E	10	-	-	ns
T_{AH}	Address Hold Time	Pins: RS,RW,E	20	-	-	ns
T_{DSW}	Data Setup Time	Pins: DB0 - DB7	40	-	-	ns
T_H	Data Hold Time	Pins: DB0 - DB7	20	-	-	ns
Read Mode (Reading Data from ST7920 to MPU)						
T_C	Enable Cycle Time	Pin E	1800	-	-	ns
T_{PW}	Enable Pulse Width	Pin E	320	-	-	ns
T_R,T_F	Enable Rise/Fall Time	Pin E	-	-	25	ns
T_{AS}	Address Setup Time	Pins: RS,RW,E	10	-	-	ns
T_{AH}	Address Hold Time	Pins: RS,RW,E	20	-	-	ns
T_{DDR}	Data Delay Time	Pins: DB0 - DB7	-	-	260	ns
T_H	Data Hold Time	Pins: DB0 - DB7	20	-	-	ns
Interface Mode with LCD Driver(ST7921)						
T_{CWH}	Clock Pulse with High	Pins: CL1, CL2	800	-	-	ns
T_{CWL}	Clock Pulse with Low	Pins: CL1, CL2	800	-	-	ns
T_{CST}	Clock Setup Time	Pins: CL1, CL2	500	-	-	ns
T_{SU}	Data Setup Time	Pin: D	300	-	-	ns
T_{DH}	Data Hold Time	Pin: D	300	-	-	ns
T_{DM}	M Delay Time	Pin: M	-1000	-	1000	ns

8 bit interface timing diagram

- **MPU write data to ST7920**



- **MPU read data from ST7920**



AC Characteristics (T_A = 25°C, V_{DD} = 4.5V) Serial Mode Interface

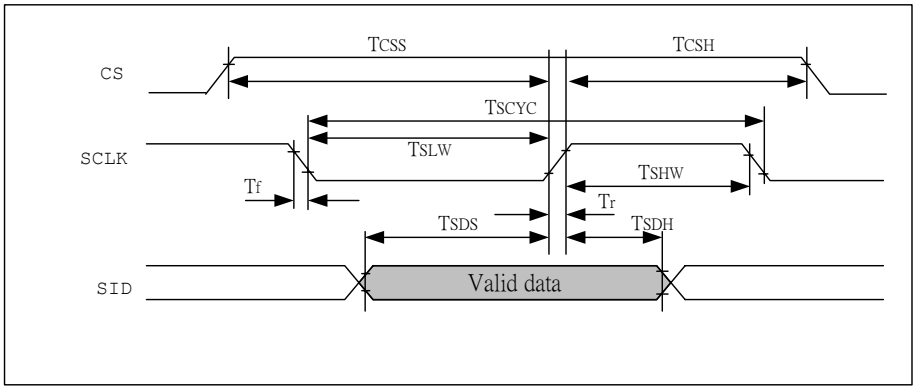
Symbol	Characteristics	Test Condition	Min.	Typ.	Max.	Unit
		Internal Clock Operation				
f_{OSC}	OSC Frequency	R = 33KΩ	470	530	590	KHz
		External Clock Operation				
f_{EX}	External Frequency	-	470	530	590	KHz
	Duty Cycle	-	45	50	55	%
T_R,T_F	Rise/Fall Time	-	-	-	0.2	μs
TSCYC	Serial clock cycle	Pin E	400	-	-	ns
TSHW	SCLK high pulse width	Pin E	200	-	-	ns
TSLW	SCLK low pulse width	Pin E	200	-	-	ns
TSDS	SID data setup time	Pins RW	40	-	-	ns
TSDH	SID data hold time	Pins RW	40	-	-	ns
TCSS	CS setup time	Pins RS	60	-	-	ns
TCSH	CS hold time	Pins RS	60	-	-	ns

AC Characteristics (T_A = 25°C, V_{DD} = 2.7V) Serial Mode Interface

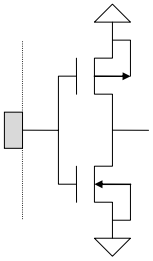
Symbol	Characteristics	Test Condition	Min.	Typ.	Max.	Unit
		Internal Clock Operation				
f_{OSC}	OSC Frequency	R = 18KΩ	470	530	590	KHz
		External Clock Operation				
f_{EX}	External Frequency	-	470	530	590	KHz
	Duty Cycle	-	45	50	55	%
T_R,T_F	Rise/Fall Time	-	-	-	0.2	μs
TSCYC	Serial clock cycle	Pin E	600	-	-	ns
TSHW	SCLK high pulse width	Pin E	300	-	-	ns
TSLW	SCLK low pulse width	Pin E	300	-	-	ns
TSDS	SID data setup time	Pins RW	40	-	-	ns
TSDH	SID data hold time	Pins RW	40	-	-	ns
TCSS	CS setup time	Pins RS	60	-	-	ns
TCSH	CS hold time	Pins RS	60	-	-	ns

Serial interface timing diagram

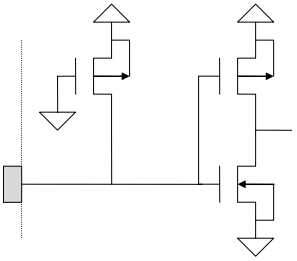
- **MPU write data to ST7920**



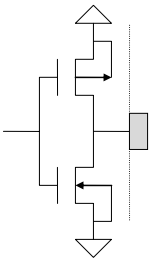
I/O pin diagram



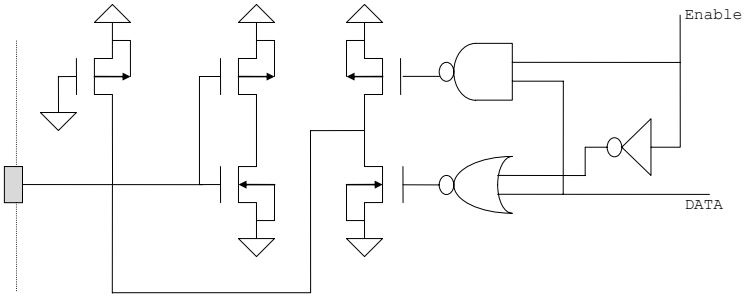
Input PAD: E (No Pull-up)



Input PAD: RS, RW(with Pull-up)



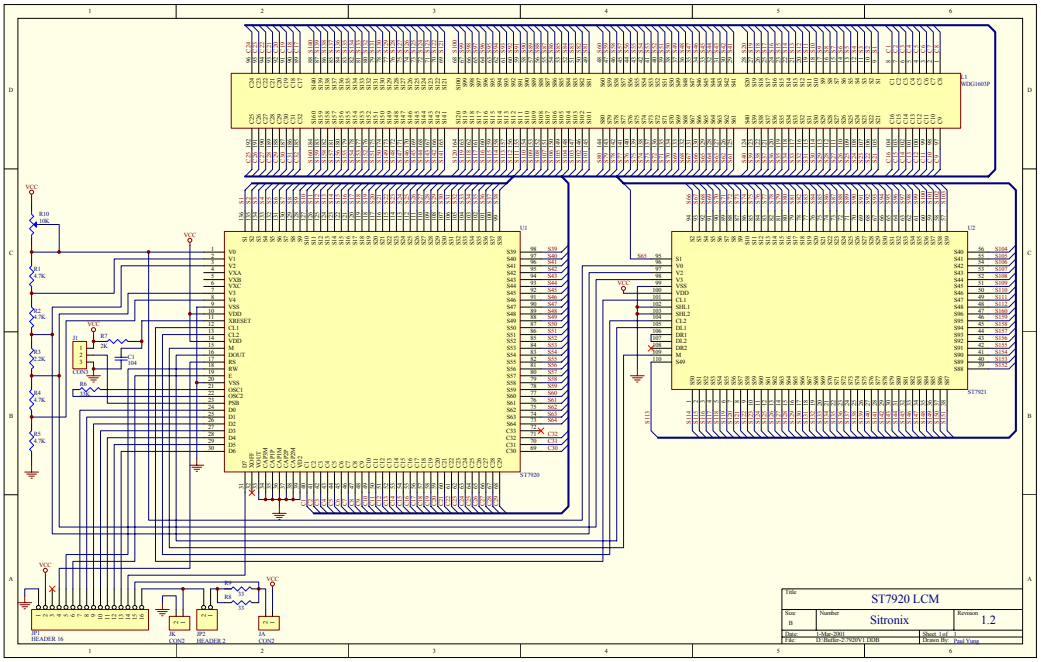
Output PAD: CL1, CL2, M, D



I/O PAD: DB0 – DB7

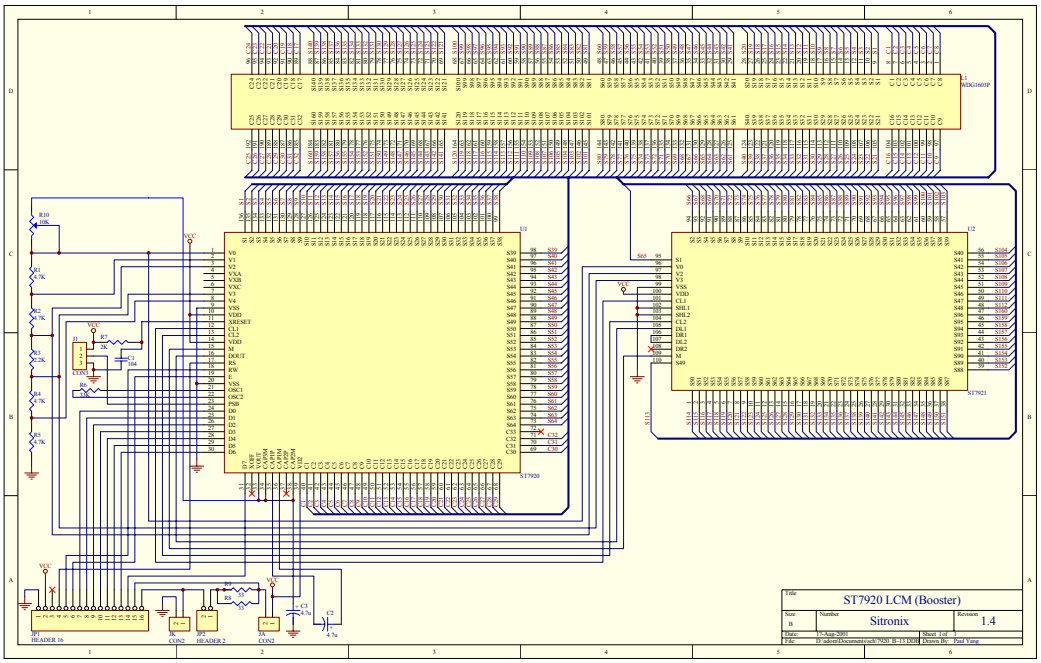
Application circuit 1：

LCD ： 32 COM x 160 SEG
LCD Voltage ： VCC



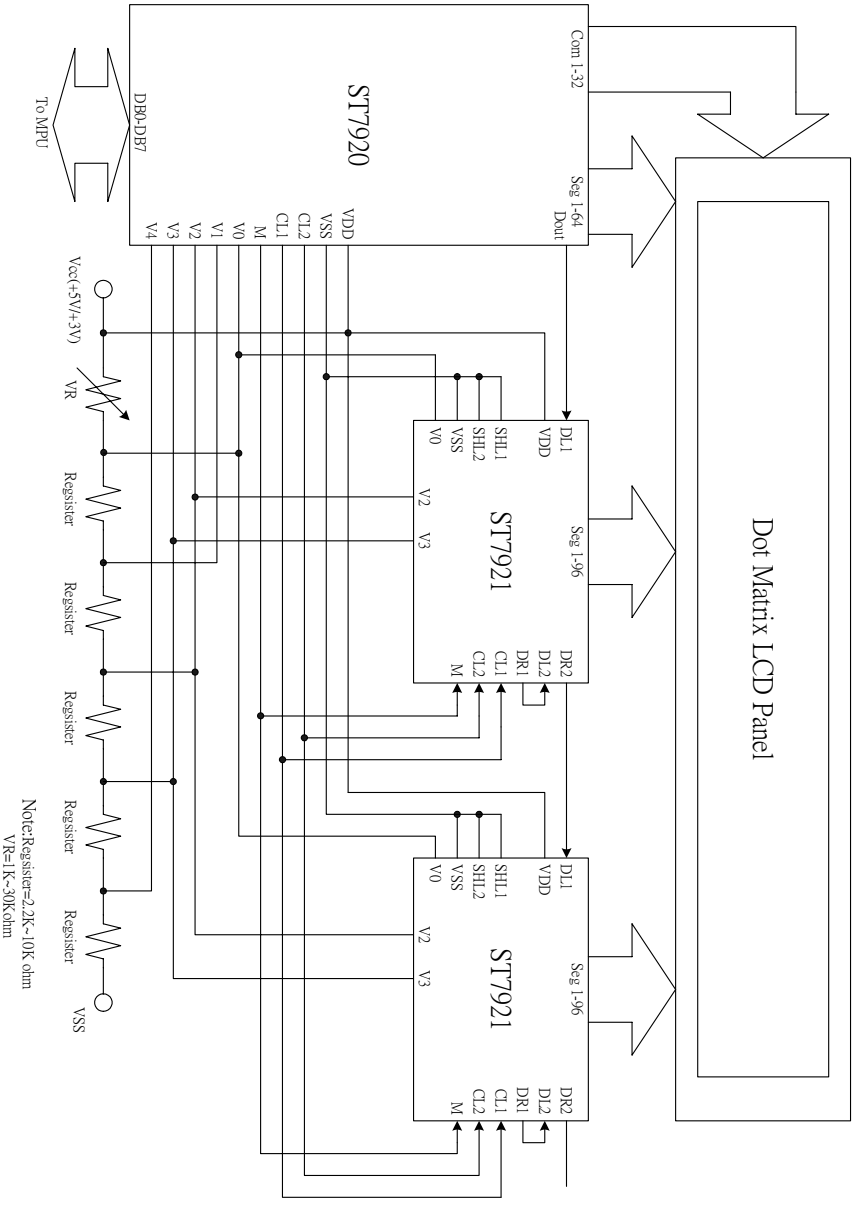
Application circuit 2：

LCD ： 32 COM x 160 SEG
LCD Voltage ： VCC x 2（Voltage doubler is used）*Vlcd should not over 7v.



Application circuit 3：

LCD : 2Line 16Chinese Word (32 COM x 256 SEG)



B.2.3. Sensor de efecto hall ACS712

ACS712

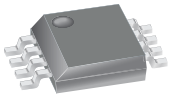
Fully Integrated, Hall Effect-Based Linear Current Sensor with 2.1 kVRMS Voltage Isolation and a Low-Resistance Current Conductor

Features and Benefits

- Low-noise analog signal path
- Device bandwidth is set via the new FILTER pin
- 5 µs output rise time in response to step input current
- 80 kHz bandwidth
- Total output error 1.5% at T_A = 25°C
- Small footprint, low-profile SOIC8 package
- 1.2 mΩ internal conductor resistance
- 2.1 kV_{RMS} minimum isolation voltage from pins 1-4 to pins 5-8
- 5.0 V, single supply operation
- 66 to 185 mV/A output sensitivity
- Output voltage proportional to AC or DC currents
- Factory-trimmed for accuracy
- Extremely stable output offset voltage
- Nearly zero magnetic hysteresis
- Ratiometric output from supply voltage

TÜV America
Certificate Number:
U8V 06 05 54214 010

Package: 8 Lead SOIC (suffix LC)



Approximate Scale 1:1

Description

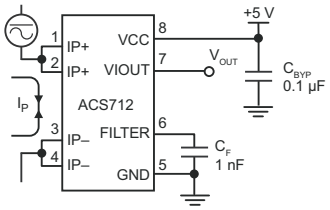
The Allegro® ACS712 provides economical and precise solutions for AC or DC current sensing in industrial, commercial, and communications systems. The device package allows for easy implementation by the customer. Typical applications include motor control, load detection and management, switched-mode power supplies, and overcurrent fault protection.

The device consists of a precise, low-offset, linear Hall sensor circuit with a copper conduction path located near the surface of the die. Applied current flowing through this copper conduction path generates a magnetic field which is sensed by the integrated Hall IC and converted into a proportional voltage. Device accuracy is optimized through the close proximity of the magnetic signal to the Hall transducer. A precise, proportional voltage is provided by the low-offset, chopper-stabilized BiCMOS Hall IC, which is programmed for accuracy after packaging.

The output of the device has a positive slope (>$V_{IOUT(Q)}$) when an increasing current flows through the primary copper conduction path (from pins 1 and 2, to pins 3 and 4), which is the path used for current sensing. The internal resistance of this conductive path is 1.2 mΩ typical, providing low power

Continued on the next page…

Typical Application



Application 1. The ACS712 outputs an analog signal, V_{OUT}, that varies linearly with the uni- or bi-directional AC or DC primary sensed current, I_P, within the range specified. C_F is recommended for noise management, with values that depend on the application.

Description (continued)

loss. The thickness of the copper conductor allows survival of the device at up to 5× overcurrent conditions. The terminals of the conductive path are electrically isolated from the sensor leads (pins 5 through 8). This allows the ACS712 current sensor to be used in applications requiring electrical isolation without the use of opto-isolators or other costly isolation techniques.

The ACS712 is provided in a small, surface mount SOIC8 package. The leadframe is plated with 100% matte tin, which is compatible with standard lead (Pb) free printed circuit board assembly processes. Internally, the device is Pb-free, except for flip-chip high-temperature Pb-based solder balls, currently exempt from RoHS. The device is fully calibrated prior to shipment from the factory.

Selection Guide

Part Number	Packing*	T_A (°C)	Optimized Range, I_P (A)	Sensitivity, Sens (Typ) (mV/A)
ACS712ELCTR-05B-T	Tape and reel, 3000 pieces/reel	–40 to 85	±5	185
ACS712ELCTR-20A-T	Tape and reel, 3000 pieces/reel	–40 to 85	±20	100
ACS712ELCTR-30A-T	Tape and reel, 3000 pieces/reel	–40 to 85	±30	66

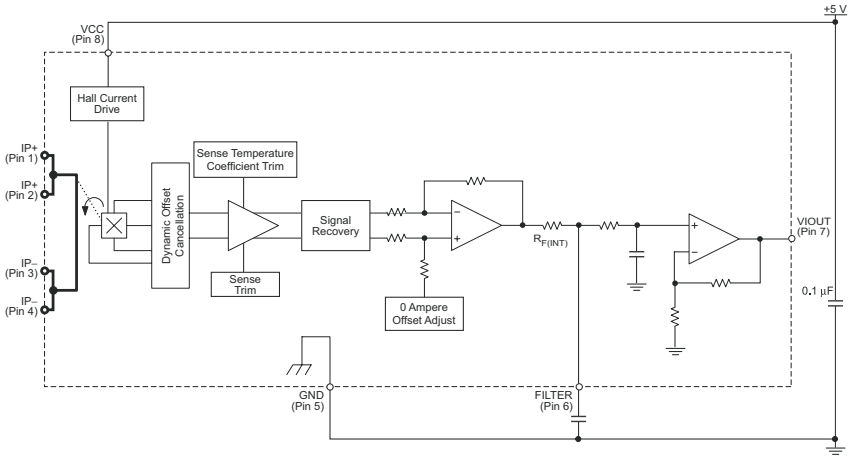
*Contact Allegro for additional packing options.

Absolute Maximum Ratings

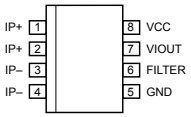
Characteristic	Symbol	Notes	Rating	Units
Supply Voltage	V_{CC}		8	V
Reverse Supply Voltage	V_{RCC}		–0.1	V
Output Voltage	V_{IOUT}		8	V
Reverse Output Voltage	V_{RIOUT}		–0.1	V
Reinforced Isolation Voltage	V_{ISO}	Pins 1-4 and 5-8; 60 Hz, 1 minute, T_A=25°C	2100	V
		Voltage applied to leadframe (Ip+ pins), based on IEC 60950	184	V_{peak}
Basic Isolation Voltage	$V_{ISO(bsc)}$	Pins 1-4 and 5-8; 60 Hz, 1 minute, T_A=25°C	1500	V
		Voltage applied to leadframe (Ip+ pins), based on IEC 60950	354	V_{peak}
Output Current Source	$I_{IOUT(Source)}$		3	mA
Output Current Sink	$I_{IOUT(Sink)}$		10	mA
Overcurrent Transient Tolerance	I_P	1 pulse, 100 ms	100	A
Nominal Operating Ambient Temperature	T_A	Range E	–40 to 85	ºC
Maximum Junction Temperature	T_J(max)		165	ºC
Storage Temperature	T_{stg}		–65 to 170	ºC

Parameter	Specification
Fire and Electric Shock	CAN/CSA-C22.2 No. 60950-1-03 UL 60950-1:2003 EN 60950-1:2001

Functional Block Diagram



Pin-out Diagram



Terminal List Table

Number	Name	Description
1 and 2	IP+	Terminals for current being sensed; fused internally
3 and 4	IP–	Terminals for current being sensed; fused internally
5	GND	Signal ground terminal
6	FILTER	Terminal for external capacitor that sets bandwidth
7	VIOUT	Analog output signal
8	VCC	Device power supply terminal

COMMON OPERATING CHARACTERISTICS[1] over full range of T_A, C_F = 1 nF, and V_{CC} = 5 V, unless otherwise specified

Characteristic	Symbol	Test Conditions	Min.	Typ.	Max.	Units
ELECTRICAL CHARACTERISTICS						
Supply Voltage	V_{CC}		4.5	5.0	5.5	V
Supply Current	I_{CC}	V_{CC} = 5.0 V, output open	–	10	13	mA
Output Capacitance Load	C_{LOAD}	VIOUT to GND	–	–	10	nF
Output Resistive Load	R_{LOAD}	VIOUT to GND	4.7	–	–	kΩ
Primary Conductor Resistance	$R_{PRIMARY}$	T_A = 25°C	–	1.2	–	mΩ
Rise Time	t_r	I_P = I_P(max), T_A = 25°C, C_{OUT} = open	–	5	–	µs
Frequency Bandwidth	f	–3 dB, T_A = 25°C; I_P is 10 A peak-to-peak	–	80	–	kHz
Nonlinearity	E_{LIN}	Over full range of I_P	–	1.5	–	%
Symmetry	E_{SYM}	Over full range of I_P	98	100	102	%
Zero Current Output Voltage	$V_{IOUT(Q)}$	Bidirectional; I_P = 0 A, T_A = 25°C	–	V_{CC} × 0.5	–	V
Power-On Time	t_{PO}	Output reaches 90% of steady-state level, T_J = 25°C, 20 A present on leadframe	–	35	–	µs
Magnetic Coupling[2]			–	12	–	G/A
Internal Filter Resistance[3]	$R_{F(INT)}$			1.7		kΩ

[1]Device may be operated at higher primary current levels, I_P, and ambient, T_A, and internal leadframe temperatures, T_A, provided that the Maximum Junction Temperature, T_J(max), is not exceeded.

[2]1G = 0.1 mT.

[3]$R_{F(INT)}$ forms an RC circuit via the FILTER pin.

COMMON THERMAL CHARACTERISTICS[1]

			Min.	Typ.	Max.	Units
Operating Internal Leadframe Temperature	T_A	E range	–40	–	85	°C
					Value	Units
Junction-to-Lead Thermal Resistance[2]	$R_{\theta JL}$	Mounted on the Allegro ASEK 712 evaluation board			5	°C/W
Junction-to-Ambient Thermal Resistance	$R_{\theta JA}$	Mounted on the Allegro 85-0322 evaluation board, includes the power consumed by the board			23	°C/W

[1]Additional thermal information is available on the Allegro website.

[2]The Allegro evaluation board has 1500 mm^2 of 2 oz. copper on each side, connected to pins 1 and 2, and to pins 3 and 4, with thermal vias connecting the layers. Performance values include the power consumed by the PCB. Further details on the board are available from the Frequently Asked Questions document on our website. Further information about board design and thermal performance also can be found in the Applications Information section of this datasheet.

x05B PERFORMANCE CHARACTERISTICS T_A = –40°C to 85°C[1], C_F = 1 nF, and V_{CC} = 5 V, unless otherwise specified

Characteristic	Symbol	Test Conditions	Min.	Typ.	Max.	Units
Optimized Accuracy Range	I_P		–5	–	5	A
Sensitivity	Sens	Over full range of I_P, T_A = 25°C	180	185	190	mV/A
Noise	$V_{NOISE(PP)}$	Peak-to-peak, T_A = 25°C, 185 mV/A programmed Sensitivity, C_F = 47 nF, C_{OUT} = open, 2 kHz bandwidth	–	21	–	mV
Zero Current Output Slope	$\Delta I_{OUT(Q)}$	T_A = –40°C to 25°C	–	–0.26	–	mV/°C
		T_A = 25°C to 150°C	–	–0.08	–	mV/°C
Sensitivity Slope	ΔSens	T_A = –40°C to 25°C	–	0.054	–	mV/A/°C
		T_A = 25°C to 150°C	–	–0.008	–	mV/A/°C
Total Output Error[2]	E_{TOT}	I_P =±5 A, T_A = 25°C	–	±1.5	–	%

[1]Device may be operated at higher primary current levels, I_P, and ambient temperatures, T_A, provided that the Maximum Junction Temperature, $T_{J(max)}$, is not exceeded.

[2]Percentage of I_P, with I_P = 5 A. Output filtered.

x20A PERFORMANCE CHARACTERISTICS T_A = –40°C to 85°C[1], C_F = 1 nF, and V_{CC} = 5 V, unless otherwise specified

Characteristic	Symbol	Test Conditions	Min.	Typ.	Max.	Units
Optimized Accuracy Range	I_P		–20	–	20	A
Sensitivity	Sens	Over full range of I_P, T_A = 25°C	96	100	104	mV/A
Noise	$V_{NOISE(PP)}$	Peak-to-peak, T_A = 25°C, 100 mV/A programmed Sensitivity, C_F = 47 nF, C_{OUT} = open, 2 kHz bandwidth	–	11	–	mV
Zero Current Output Slope	$\Delta I_{OUT(Q)}$	T_A = –40°C to 25°C	–	–0.34	–	mV/°C
		T_A = 25°C to 150°C	–	–0.07	–	mV/°C
Sensitivity Slope	ΔSens	T_A = –40°C to 25°C	–	0.017	–	mV/A/°C
		T_A = 25°C to 150°C	–	–0.004	–	mV/A/°C
Total Output Error[2]	E_{TOT}	I_P =±20 A, T_A = 25°C	–	±1.5	–	%

[1]Device may be operated at higher primary current levels, I_P, and ambient temperatures, T_A, provided that the Maximum Junction Temperature, T_J(max), is not exceeded.

[2]Percentage of I_P, with I_P = 20 A. Output filtered.

x30A PERFORMANCE CHARACTERISTICS T_A = –40°C to 85°C[1], C_F = 1 nF, and V_{CC} = 5 V, unless otherwise specified

Characteristic	Symbol	Test Conditions	Min.	Typ.	Max.	Units
Optimized Accuracy Range	I_P		–30	–	30	A
Sensitivity	Sens	Over full range of I_P, T_A = 25°C	64	66	68	mV/A
Noise	$V_{NOISE(PP)}$	Peak-to-peak, T_A = 25°C, 66 mV/A programmed Sensitivity, C_F = 47 nF, C_{OUT} = open, 2 kHz bandwidth	–	7	–	mV
Zero Current Output Slope	$\Delta I_{OUT(Q)}$	T_A = –40°C to 25°C	–	–0.35	–	mV/°C
		T_A = 25°C to 150°C	–	–0.08	–	mV/°C
Sensitivity Slope	ΔSens	T_A = –40°C to 25°C	–	0.007	–	mV/A/°C
		T_A = 25°C to 150°C	–	–0.002	–	mV/A/°C
Total Output Error[2]	E_{TOT}	I_P = ±30 A, T_A = 25°C	–	±1.5	–	%

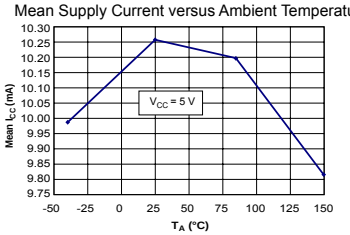
[1]Device may be operated at higher primary current levels, I_P, and ambient temperatures, T_A, provided that the Maximum Junction Temperature, T_J(max), is not exceeded.

[2]Percentage of I_P, with I_P = 30 A. Output filtered.

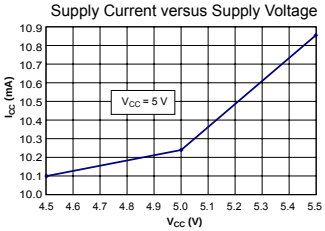
Characteristic Performance

I_P = 5 A, unless otherwise specified

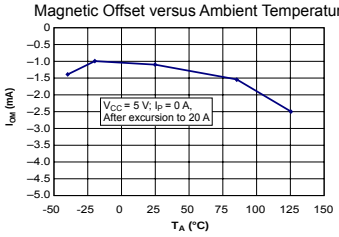
Mean Supply Current versus Ambient Temperature



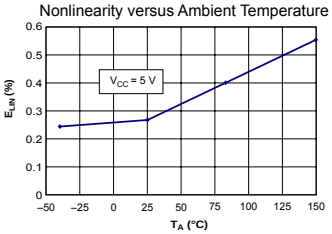
Supply Current versus Supply Voltage



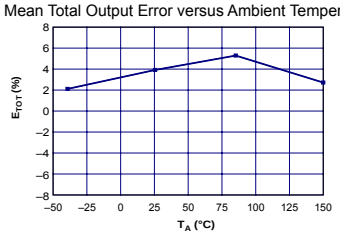
Magnetic Offset versus Ambient Temperature



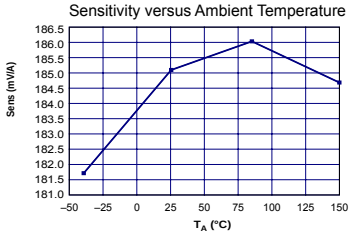
Nonlinearity versus Ambient Temperature



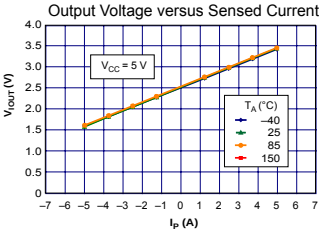
Mean Total Output Error versus Ambient Temperature



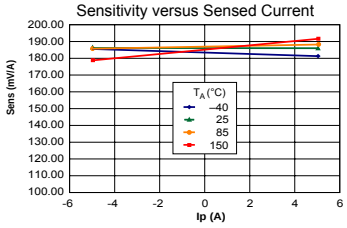
Sensitivity versus Ambient Temperature



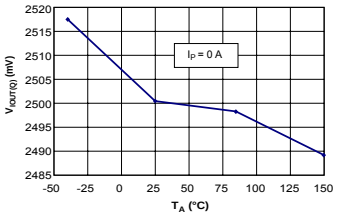
Output Voltage versus Sensed Current



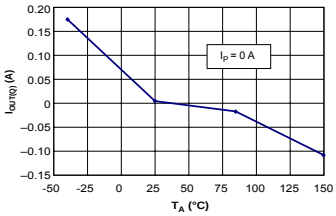
Sensitivity versus Sensed Current



0 A Output Voltage versus Ambient Temperature



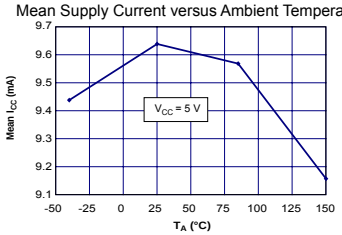
0 A Output Voltage Current versus Ambient Temperature



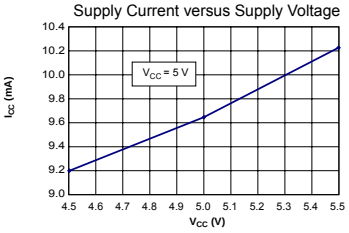
Characteristic Performance

I_P = 20 A, unless otherwise specified

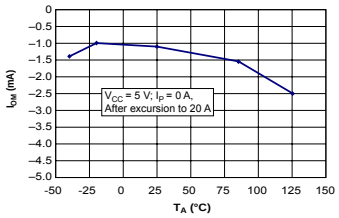
Mean Supply Current versus Ambient Temperature



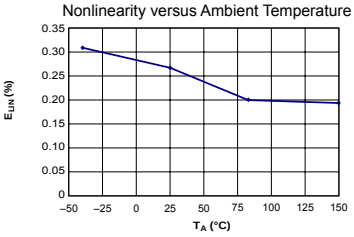
Supply Current versus Supply Voltage



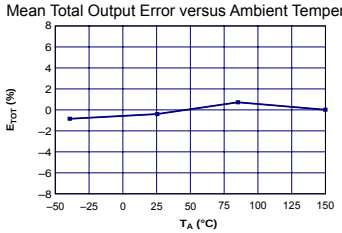
Magnetic Offset versus Ambient Temperature



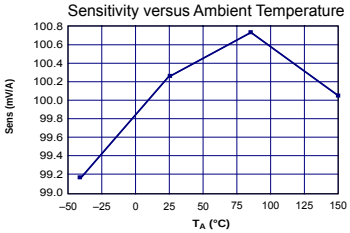
Nonlinearity versus Ambient Temperature



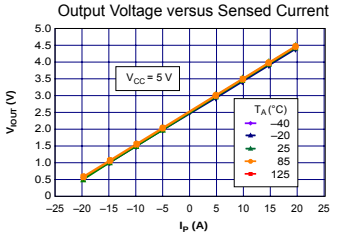
Mean Total Output Error versus Ambient Temperature



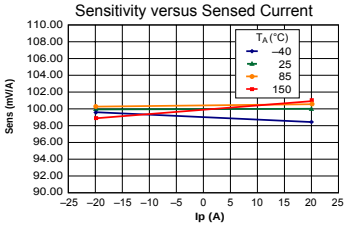
Sensitivity versus Ambient Temperature



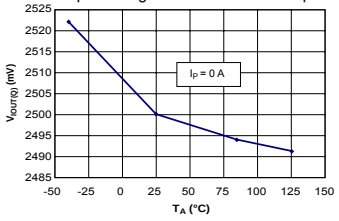
Output Voltage versus Sensed Current



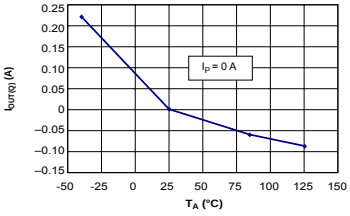
Sensitivity versus Sensed Current



0 A Output Voltage versus Ambient Temperature



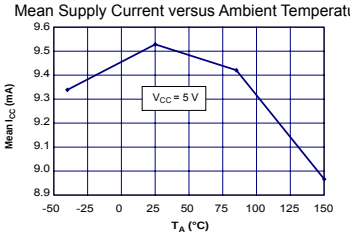
0 A Output Voltage Current versus Ambient Temperature



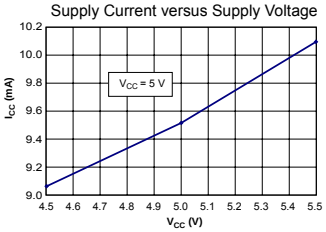
Characteristic Performance

I_P = 30 A, unless otherwise specified

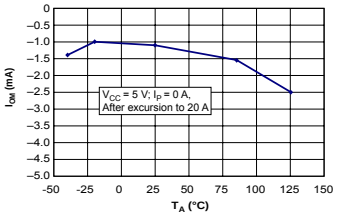
Mean Supply Current versus Ambient Temperature



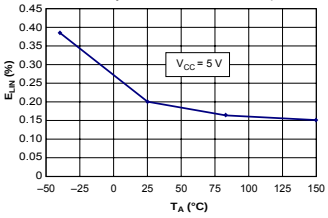
Supply Current versus Supply Voltage



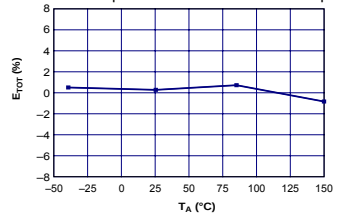
Magnetic Offset versus Ambient Temperature



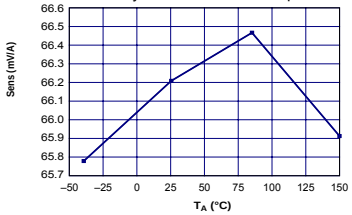
Nonlinearity versus Ambient Temperature



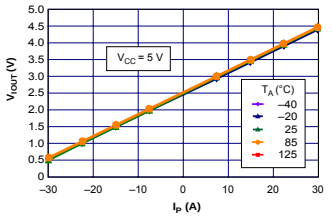
Mean Total Output Error versus Ambient Temperature



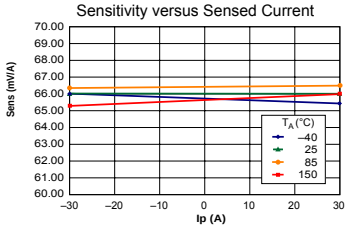
Sensitivity versus Ambient Temperature



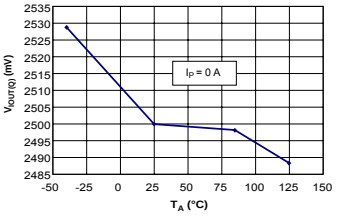
Output Voltage versus Sensed Current



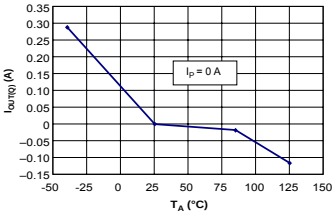
Sensitivity versus Sensed Current



0 A Output Voltage versus Ambient Temperature



0 A Output Voltage Current versus Ambient Temperature



Definitions of Accuracy Characteristics

Sensitivity (Sens). The change in sensor output in response to a 1 A change through the primary conductor. The sensitivity is the product of the magnetic circuit sensitivity (G/A) and the linear IC amplifier gain (mV/G). The linear IC amplifier gain is programmed at the factory to optimize the sensitivity (mV/A) for the full-scale current of the device.

Noise (V_{NOISE}). The product of the linear IC amplifier gain (mV/G) and the noise floor for the Allegro Hall effect linear IC (≈1 G). The noise floor is derived from the thermal and shot noise observed in Hall elements. Dividing the noise (mV) by the sensitivity (mV/A) provides the smallest current that the device is able to resolve.

Linearity (E_{LIN}). The degree to which the voltage output from the sensor varies in direct proportion to the primary current through its full-scale amplitude. Nonlinearity in the output can be attributed to the saturation of the flux concentrator approaching the full-scale current. The following equation is used to derive the linearity:

$$100\left\{1-\left[\frac{\Delta\text{ gain}\times\%\text{ sat}\,(V_{\text{IOUT\_full-scale amperes}}-V_{\text{IOUT(Q)}})}{2\,(V_{\text{IOUT\_half-scale amperes}}-V_{\text{IOUT(Q)}})}\right]\right\}$$

where $V_{\text{IOUT\_full-scale amperes}}$ = the output voltage (V) when the sensed current approximates full-scale $\pm I_P$.

Symmetry (E_{SYM}). The degree to which the absolute voltage output from the sensor varies in proportion to either a positive or negative full-scale primary current. The following formula is used to derive symmetry:

$$100\left(\frac{V_{\text{IOUT\_+ full-scale amperes}}-V_{\text{IOUT(Q)}}}{V_{\text{IOUT(Q)}}-V_{\text{IOUT\_–full-scale amperes}}}\right)$$

Quiescent output voltage ($V_{IOUT(Q)}$). The output of the sensor when the primary current is zero. For a unipolar supply voltage, it nominally remains at $V_{CC}/2$. Thus, V_{CC} = 5 V translates into $V_{IOUT(Q)}$ = 2.5 V. Variation in $V_{IOUT(Q)}$ can be attributed to the resolution of the Allegro linear IC quiescent voltage trim and thermal drift.

Electrical offset voltage (V_{OE}). The deviation of the device output from its ideal quiescent value of $V_{CC}/2$ due to nonmagnetic causes. To convert this voltage to amperes, divide by the device sensitivity, Sens.

Accuracy (E_{TOT}). The accuracy represents the maximum deviation of the actual output from its ideal value. This is also known as the total ouput error. The accuracy is illustrated graphically in the output voltage versus current chart at right.

Accuracy is divided into four areas:

- **0 A at 25°C.** Accuracy of sensing zero current flow at 25°C, without the effects of temperature.
- **0 A over Δ temperature.** Accuracy of sensing zero current flow including temperature effects.
- **Full-scale current at 25°C.** Accuracy of sensing the full-scale current at 25°C, without the effects of temperature.
- **Full-scale current over Δ temperature.** Accuracy of sensing full-scale current flow including temperature effects.

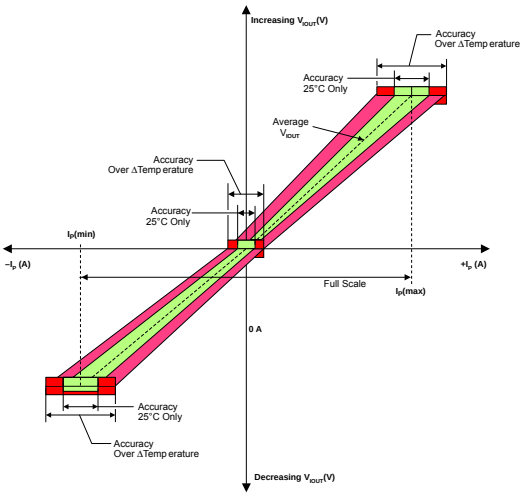
Ratiometry. The ratiometric feature means that its 0 A output, $V_{IOUT(Q)}$, (nominally equal to $V_{CC}/2$) and sensitivity, Sens, are proportional to its supply voltage, V_{CC}. The following formula is used to derive the ratiometric change in 0 A output voltage, $\Delta V_{IOUT(Q)RAT}$ (%).

$$100\left(\frac{V_{\text{IOUT(Q)VCC}}/V_{\text{IOUT(Q)5V}}}{V_{CC}/5\text{ V}}\right)$$

The ratiometric change in sensitivity, $\Delta Sens_{RAT}$ (%), is defined as:

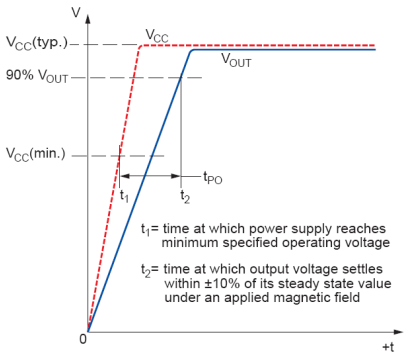
$$100\left(\frac{Sens_{VCC}/\,Sens_{5V}}{V_{CC}/5\text{ V}}\right)$$

Output Voltage versus Sensed Current
Accuracy at 0 A and at Full-Scale Current

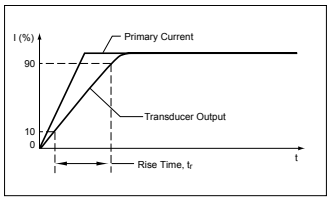


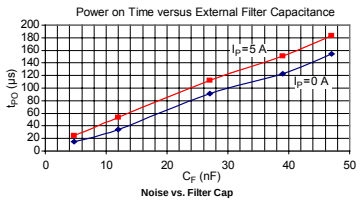
Definitions of Dynamic Response Characteristics

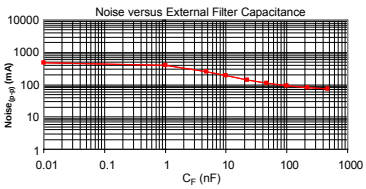
Power-On Time (t_{PO}). When the supply is ramped to its operating voltage, the device requires a finite time to power its internal components before responding to an input magnetic field. Power-On Time, t_{PO} , is defined as the time it takes for the output voltage to settle within ±10% of its steady state value under an applied magnetic field, after the power supply has reached its minimum specified operating voltage, V_{CC}(min), as shown in the chart at right.

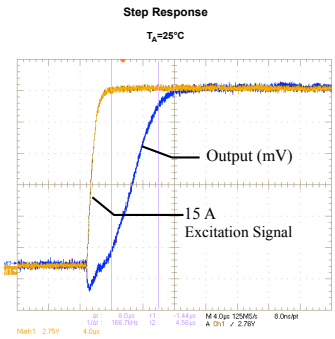


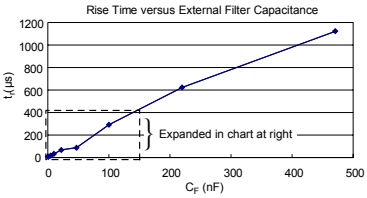
Rise time (t_r). The time interval between a) when the sensor reaches 10% of its full scale value, and b) when it reaches 90% of its full scale value. The rise time to a step response is used to derive the bandwidth of the current sensor, in which f(–3 dB) = 0.35 / t_r. Both t_r and $t_{RESPONSE}$ are detrimentally affected by eddy current losses observed in the conductive IC ground plane.



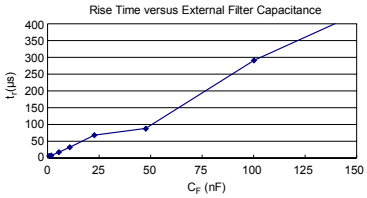








C_F (nF)	t_r (µs)
0	6.6
1	7.7
4.7	17.4
10	32.1
22	68.2
47	88.2
100	291.3
220	623.0
470	1120.0

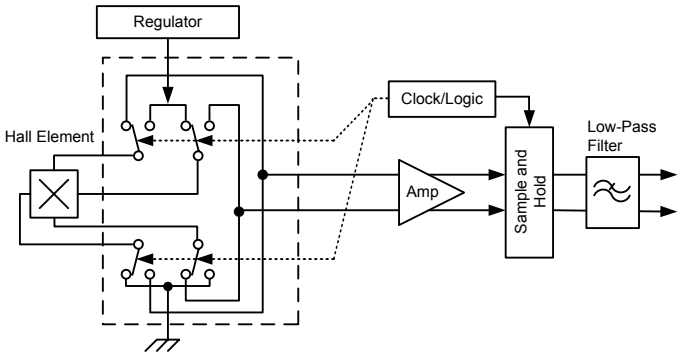


Chopper Stabilization Technique

Chopper Stabilization is an innovative circuit technique that is used to minimize the offset voltage of a Hall element and an associated on-chip amplifier. Allegro patented a Chopper Stabilization technique that nearly eliminates Hall IC output drift induced by temperature or package stress effects. This offset reduction technique is based on a signal modulation-demodulation process. Modulation is used to separate the undesired dc offset signal from the magnetically induced signal in the frequency domain. Then, using a low-pass filter, the modulated dc offset is suppressed while the magnetically induced signal passes through the filter.

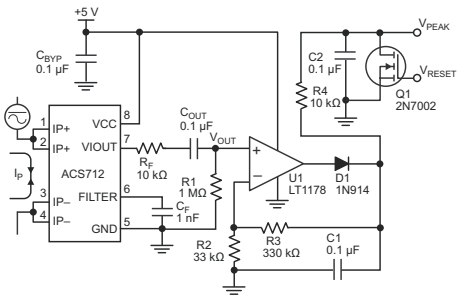
As a result of this chopper stabilization approach, the output voltage from the Hall IC is desensitized to the effects of temperature and mechanical stress. This technique produces devices that have an extremely stable Electrical Offset Voltage, are immune to thermal stress, and have precise recoverability after temperature cycling.

This technique is made possible through the use of a BiCMOS process that allows the use of low-offset and low-noise amplifiers in combination with high-density logic integration and sample and hold circuits.

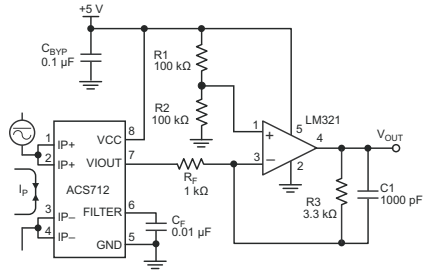


Concept of Chopper Stabilization Technique

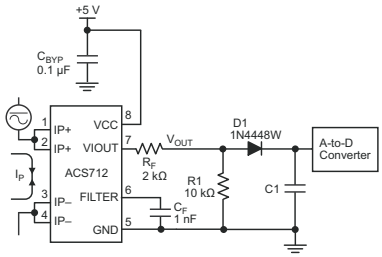
Typical Applications



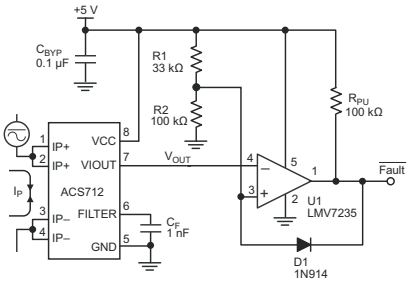
Application 2. Peak Detecting Circuit



Application 3. This configuration increases gain to 610 mV/A (tested using the ACS712ELC-05A).



Application 4. Rectified Output. 3.3 V scaling and rectification application for A-to-D converters. Replaces current transformer solutions with simpler ACS circuit. C1 is a function of the load resistance and filtering desired. R1 can be omitted if the full range is desired.



Application 5. 10 A Overcurrent Fault Latch. Fault threshold set by R1 and R2. This circuit latches an overcurrent fault and holds it until the 5 V rail is powered down.

Improving Sensing System Accuracy Using the FILTER Pin

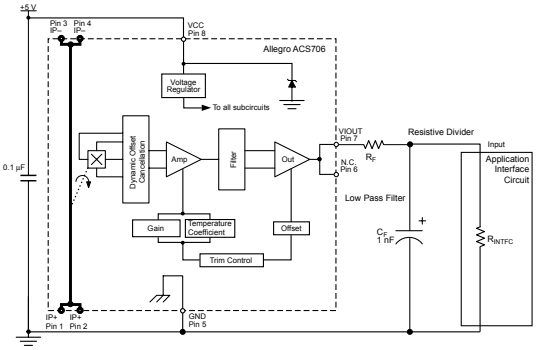
In low-frequency sensing applications, it is often advantageous to add a simple RC filter to the output of the sensor. Such a low-pass filter improves the signal-to-noise ratio, and therefore the resolution, of the sensor output signal. However, the addition of an RC filter to the output of a sensor IC can result in undesirable sensor output attenuation — even for dc signals.

Signal attenuation, ΔV_{ATT}, is a result of the resistive divider effect between the resistance of the external filter, R_F (see Application 6), and the input impedance and resistance of the customer interface circuit, R_{INTFC}. The transfer function of this resistive divider is given by:

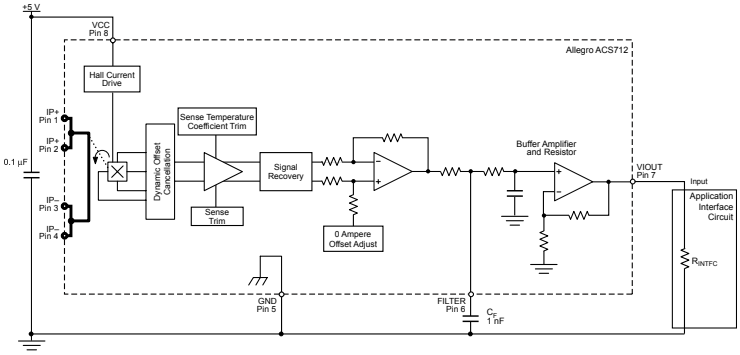
$$\Delta V_{ATT} = V_{IOUT}\left(\frac{R_{INTFC}}{R_F + R_{INTFC}}\right) .$$

Even if R_F and R_{INTFC} are designed to match, the two individual resistance values will most likely drift by different amounts over temperature. Therefore, signal attenuation will vary as a function of temperature. Note that, in many cases, the input impedance, R_{INTFC}, of a typical analog-to-digital converter (ADC) can be as low as 10 kΩ.

The ACS712 contains an internal resistor, a FILTER pin connection to the printed circuit board, and an internal buffer amplifier. With this circuit architecture, users can implement a simple RC filter via the addition of a capacitor, C_F (see Application 7) from the FILTER pin to ground. The buffer amplifier inside of the ACS712 (located after the internal resistor and FILTER pin connection) eliminates the attenuation caused by the resistive divider effect described in the equation for ΔV_{ATT}. Therefore, the ACS712 device is ideal for use in high-accuracy applications that cannot afford the signal attenuation associated with the use of an external RC low-pass filter.

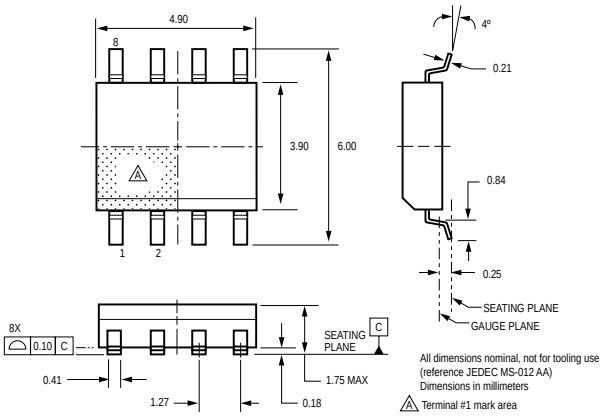


Application 6. When a low pass filter is constructed externally to a standard Hall effect device, a resistive divider may exist between the filter resistor, R_F, and the resistance of the customer interface circuit, R_{INTFC}. This resistive divider will cause excessive attenuation, as given by the transfer function for ΔV_{ATT}.



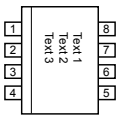
Application 7. Using the FILTER pin provided on the ACS712 eliminates the attenuation effects of the resistor divider between R_F and R_{INTFC}, shown in Application 6.

Package LC, 8-pin SOIC



Package Branding

Two alternative patterns are used



ACS712T *RLCPPP* *YYWWA*	ACS	Allegro Current Sensor
	712	Device family number
	T	Indicator of 100% matte tin leadframe plating
	R	Operating ambient temperature range code
	LC	Package type designator
	PPP	Primary sensed current
	YY	Date code: Calendar year (last two digits)
	WW	Date code: Calendar week
	A	Date code: Shift code

ACS712T *RLCPPP* *L...L* *YYWW*	ACS	Allegro Current Sensor
	712	Device family number
	T	Indicator of 100% matte tin leadframe plating
	R	Operating ambient temperature range code
	LC	Package type designator
	PPP	Primary sensed current
	L...L	Lot code
	YY	Date code: Calendar year (last two digits)
	WW	Date code: Calendar week

For the latest version of this document, visit our website:

www.allegromicro.com

B.2.4. Panel solar Aide Solar D185M5-Aa

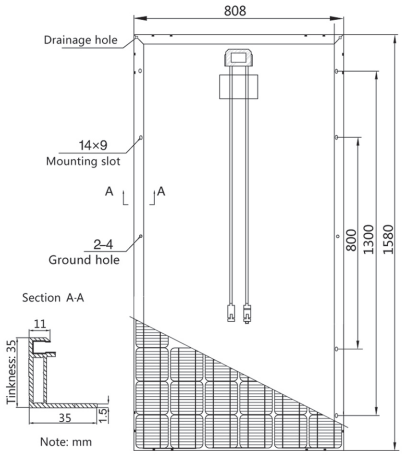
Aide Solar

AD180/185/190M5-Aa

MONOCRYSTALLINE SOLAR MODULE

Features

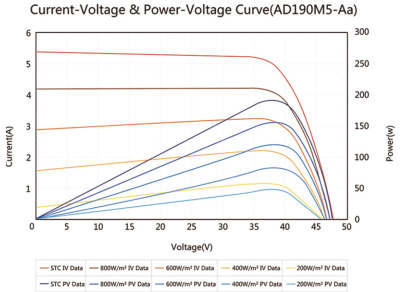
- Highly efficient energy conversion.
- High strength with wind and snow loads guaranteed up to 5400 Pascal.
- All manufactured modules are tested 100% by EL (Electroluminescence) before and after lamination.
- Drainage and other designs prevent deforming and fracturing due to freezing or other forces.
- Power catergorization one watt per pallet thus minimizing workload of classification at worksite.



Electrical Characteristics

STC	AD180M5-Aa	AD185M5-Aa	AD190M5-Aa
Maximum Power at STC (Pmax)	180W	185W	190W
Optimum Operating Voltage (Vmp)	35.66V	36.08V	36.50V
Optimum Operating Current (Imp)	5.05A	5.13A	5.21A
Open Circuit Voltage (Voc)	44.87V	45.04V	45.23V
Short Circuit Current (Isc)	5.35A	5.43A	5.51A
Module Efficiency	14.10%	14.49%	14.88%
Operating Temperature	-40～85℃	-40～85℃	-40～85℃
Maximum System Voltage	1000V DC	1000V DC	1000V DC
Maximum Series Fuse Rating	10A	10A	10A
Power Tolerance	0W~+5W	0W~+5W	0W~+5W

STC: Irradiance of 1000W/m², spectrum AM=1.5 , module temperature of 25℃



Mechanical Characteristics

Cell Type	Monocrystalline 125x125mm (5 inches)
Number of Cells	72 (6x12)
Dimensions	1580x808x35mm
Weight	15kg
Front Cover	Tempered glass
Frame Material	Anodized aluminium alloy
Standard Packaging (Modules per Pallet)	32pcs

Temperature Characteristics

Nominal Operating Cell Temperature	48±2℃
Temperature Coefficient of Pmax	-0.42%/℃
Temperature Coefficient of Voc	-0.30%/℃
Temperature Coefficient of Isc	0.06%/℃

*Specifications included in this datasheet are subject to change without further notification.

E-mail: Sales@aidesolar.com www.aidesolar.com

Version 2011 August (01)

B.2.5. Batería LivEN LEVG50-12

LiVEN BATTERY

12V 50Ah **LEVG50-12**

GEL Deep Cycle Battery

LIVEN LEVG Series

LEVG Pure Gel series are manufacturing with special PVC-SI02 separator and patented GEL electrolyte. The LEVG series Valve Regulated Lead Acid (VRLA) is Pure Gel with 12 years floating design life. This battery its ideal for light traction electric vehicles applications (wheelchairs, electric sweepers, golf trolleys…). Maintenance-Free Sealed Lead Acid Battery.

The number of deep discharge cycles its 450 cycles at 100% DOD and 1300 cycles at 50%.

Applications:

- Wheelchairs
- Golf trolleys
- Electric sweepers
- Floor machines
- Electric vehicles
- Lawn mowers
- Portable power
- Medical equipments

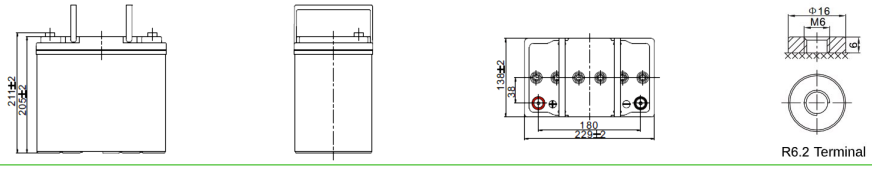
Dimensions:

Length	229±1.5mm (9.02in)
Width	138±1.5mm (5.43in)
Height	205±1.5mm (8.07in)
Total Height	211±1.5mm (8.31in)

Specifications:

Cells Per Unit	6
Voltage Per Unit	12V
Nominal Capacity	50.0Ah @20hour-rate to 1.80V per cell @25°C
Weight	Approx.17.6Kg ±2% (38.80lbs)
Internal Resistance	Approx. 9.0mΩ
Terminal	R6.2
Max. Discharge Current	500A (5sec)
Design Life	12 years floating Eurobat (20ºC): 10-12 years Long Life
Recommended Max. Charging Current	6.0A
Standby Use Voltage	13.5V~13.8V @ 25°C Temperature Compensation: -3mV/°C/Cell
Cycle Use Voltage	14.1V~14.4V @ 25°C Temperature Compensation: -4mV/°C/Cell
Operating Temperature Range	Discharge: -20°C~55°C Charge: 0°C~40°C Storage: -20°C~50°C
Normal Operating Temperature Range	25°C±5°C
Self Discharge	LIVEN Valve Regulated Lead Acid (VRLA) batteries can be stored for up to 6 months at 25°C and then recharging is recommended. Monthly Self-discharge ratio is less than 3% at 25°C.Please charge batteries before using.
Container Material	A.B.S. UL94-HB, UL94-V0 Optional.

Technical Drawings:



Constant Current Discharge (CC, Unit: A) at 25ºC (77ºF)

F.V. / Time	10min	15min	20min	30min	45min	1h	2h	3h	4h	5h	6h	8h	10h	20h
1.60V	102.00	78.70	66.60	49.50	36.60	29.70	17.80	13.10	10.40	8.74	7.50	5.87	4.91	2.63
1.67V	93.60	73.60	62.20	46.70	34.60	28.30	17.10	12.60	10.10	8.52	7.34	5.79	4.83	2.60
1.75V	78.70	64.30	55.60	42.40	31.90	26.20	15.90	11.90	9.59	8.09	7.00	5.58	4.68	2.54
1.80V	67.80	56.50	50.00	39.00	29.90	24.70	15.20	11.30	9.22	7.83	6.79	5.45	4.61	2.50
1.85V	56.50	48.90	43.90	35.00	26.60	22.20	14.00	10.50	8.61	7.37	6.40	5.16	4.38	2.44

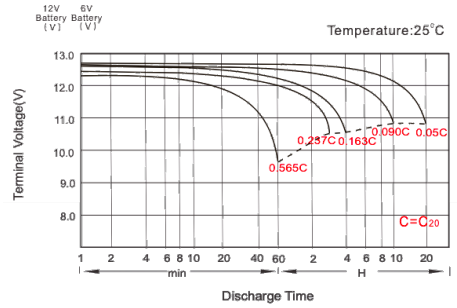
Constant Power Discharge (CP, Unit: W/Battery) at 25ºC (77ºF)

F.V. / Time	10min	15min	20min	30min	45min	1h	2h	3h	4h	5h	6h	8h	10h	20h
1.60V	1067.4	840.6	724.8	547.8	411.6	337.2	203.4	150.6	120.6	102.0	88.2	69.0	58.2	31.3
1.67V	997.2	796.8	684.6	522.6	393.0	324.0	197.4	146.4	117.6	99.6	86.4	68.4	57.4	31.1
1.75V	861.0	711.6	622.2	481.2	365.4	301.8	184.8	138.6	112.2	94.8	82.8	66.0	55.7	30.3
1.80V	753.6	632.4	565.2	445.2	345.0	286.8	176.4	132.6	108.0	92.4	80.4	64.8	54.9	29.9
1.85V	636.6	577.2	503.4	403.8	309.0	259.2	163.2	123.6	101.4	87.0	76.2	61.2	52.3	29.2

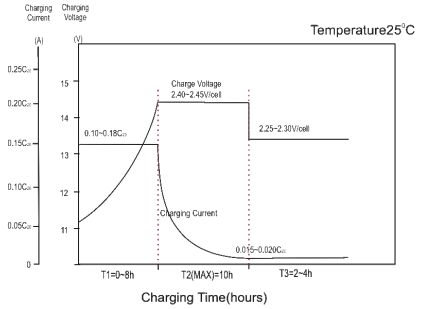
LIVEN BATTERY

12V 50Ah GEL VRLA battery LEVG50-12

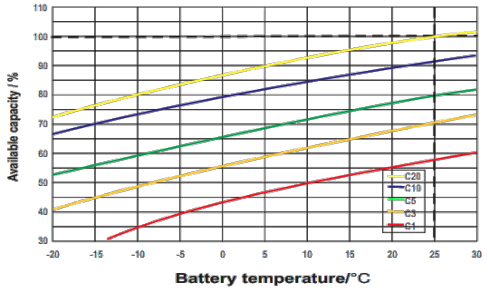
Discharge Characteristics



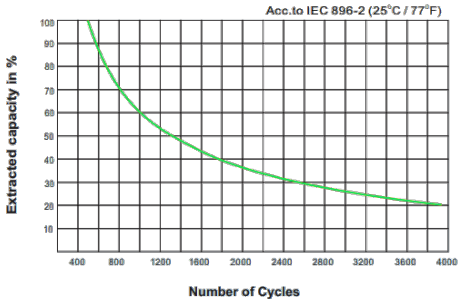
Charging Characteristics



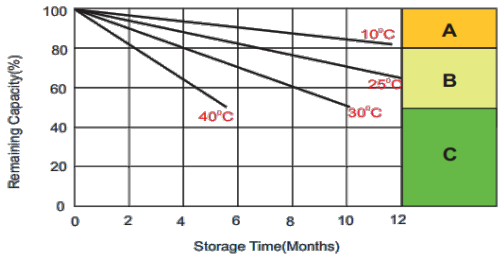
Temperature Effects in Relation to Capacity



Cycle Life in Relation to Depth of Discharge



Self Discharge Characteristics



A No supplementary charge required
(Carry out supplementary charge before use if 100% capacity is required).

B Supplementary charge required before use. Optional charging way as below:
1.Charged for above 3 days at limited current 0.25CA and constant voltage 2.25V/cell.
2.Charged for above 20 hours at limited current 0.25CA and constant voltage 2.45V/cell.
3.Charged for 8~10 hours at limited current 0.05CA.

C Supplementary charge may often fail to recover the capacity.
The battery should never be left standing till this is reached.

B.3. Informes

B.3.1. Instalación fotovoltaica en techo

European Commission

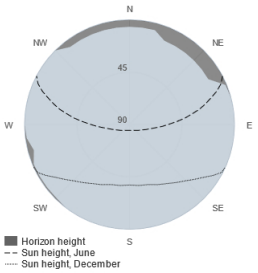
Performance of off-grid PV system

PVGIS-5 estimates of solar electricity generation

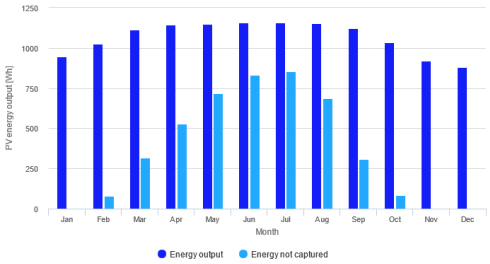
Provided inputs

Latitude/Longitude:	28.307,-16.503	Slope angle:	0 °
Horizon:	Calculated	Azimuth angle	0 °
Database used:	PVGIS-SARAH2	**Simulation outputs**	
PV installed:	360 Wp	Percentage days with full battery:	59.35 %
Battery capacity:	1200 Wh	Percentage days with empty battery:	32.16 %
Cutoff limit:	10 %	Average energy not captured:	622.25 Wh
Consumption per day:	1156 Wh	Average energy missing:	275.18 Wh

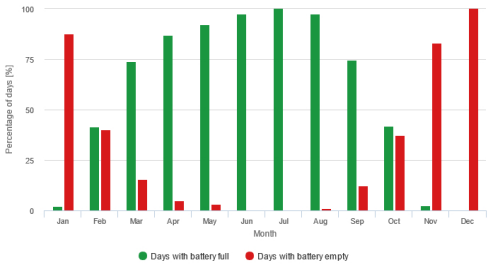
Outline of horizon at chosen location:



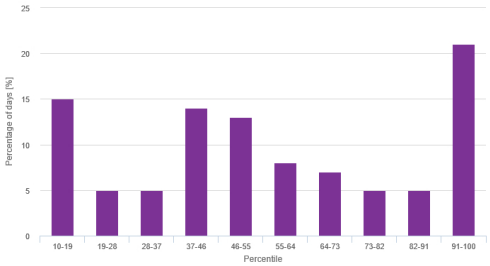
Power production estimate for off-grid PV:



Battery performance for off-grid PV system:



Probability of battery charge state at the end of the day:



Monthly average performance

Month	E_d	E_l	f_f	f_e
January	948.3	1.6	2.2	87.7
February	1027.2	79.9	41.5	40.2
March	1114.5	317.5	74.0	15.5
April	1144.3	528.3	86.9	5.0
May	1150.0	717.8	92.1	3.2
June	1155.7	832.4	97.7	0.2
July	1155.8	851.9	100.0	0.0
August	1152.8	684.5	97.6	1.2
September	1122.6	308.9	74.8	12.5
October	1034.5	83.1	41.7	37.5
November	919.2	1.4	2.3	83.1
December	880.9	0.0	0.0	100.0

E_d: Average energy production per day [Wh/day].
E_l: Average energy not captured per day [Wh/day].
f_f: Percentage of days when battery became full [%].
f_e: Percentage of days when battery became empty [%].

Cs	Cb
10-19	15.0
19-28	5.0
28-37	5.0
37-46	14.0
46-55	13.0
55-64	8.0
64-73	7.0
73-82	5.0
82-91	5.0
91-100	21.0

Cs: Charge state at the end of each day [%].
Cb: Percentage of days with this charge state [%].

Joint Research Centre

Report generated on 2022/09/07

B.3.2. Instalación fotovoltaica Instalable y orientable

European Commission

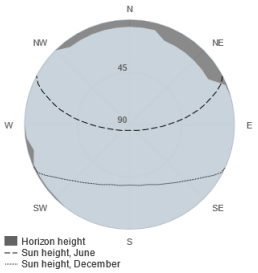
Performance of off-grid PV system

PVGIS-5 estimates of solar electricity generation

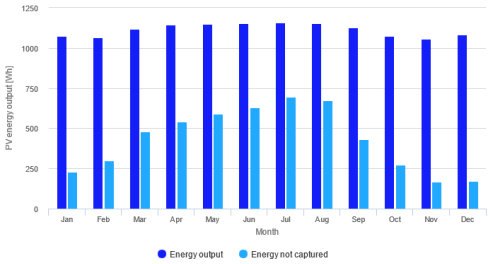
Provided inputs

Latitude/Longitude:	28.307,-16.503	Slope angle:	25 °
Horizon:	Calculated	Azimuth angle	-7 °
Database used:	PVGIS-SARAH2	**Simulation outputs**	
PV installed:	360 Wp	Percentage days with full battery:	78.36 %
Battery capacity:	1200 Wh	Percentage days with empty battery:	13.96 %
Cutoff limit:	10 %	Average energy not captured:	552.83 Wh
Consumption per day:	1156 Wh	Average energy missing:	307.73 Wh

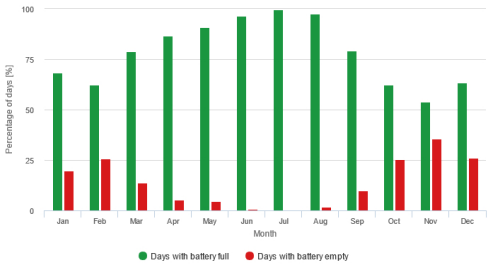
Outline of horizon at chosen location:



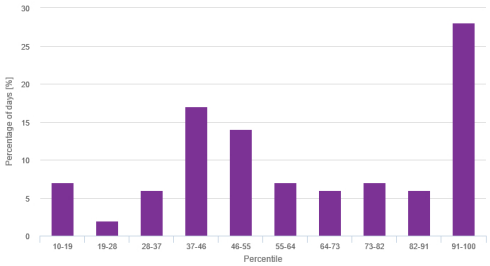
Power production estimate for off-grid PV:



Battery performance for off-grid PV system:



Probability of battery charge state at the end of the day:



Monthly average performance

Month	E_d	E_l	f_f	f_e
January	1072.8	230.8	68.2	19.8
February	1064.4	299.8	62.3	25.7
March	1119.1	481.2	79.0	13.7
April	1142.5	541.2	86.5	5.2
May	1147.4	588.6	90.7	4.4
June	1155.1	629.2	96.5	0.8
July	1155.9	697.4	99.6	0.0
August	1152.5	675.6	97.4	1.8
September	1128.0	429.2	79.2	10.0
October	1075.2	271.0	62.5	25.4
November	1055.2	168.2	53.8	35.6
December	1083.4	172.9	63.3	26.2

E_d: Average energy production per day [Wh/day].
E_l: Average energy not captured per day [Wh/day].
f_f: Percentage of days when battery became full [%].
f_e: Percentage of days when battery became empty [%].

Cs	Cb
10-19	7.0
19-28	2.0
28-37	6.0
37-46	17.0
46-55	14.0
55-64	7.0
64-73	6.0
73-82	7.0
82-91	6.0
91-100	28.0

Cs: Charge state at the end of each day [%].
Cb: Percentage of days with this charge state [%].

Joint Research Centre

Report generated on 2022/09/07