MICRO-PROCESSORS

AND

MICRO-CONTROLLERS

STEFAN HOLLENTHONER

INSTRUMENTATION UNIT

IAEA LABORATORIES SEIBERSDORF

2007-01-08

Parts of this manual were taken from manuals provided by courtesy of

MR. HEINZ RONGEN

FORSCHUNGSZENTRUM JÜLICH GMBH
ZENTRALINSTITUT FÜR ELEKTRONIK, ZEL
52425 JUELICH, GERMANY

EMAIL: H.RONGEN@FZ-JUELICH.DE

WEB: WWW.FZ-JUELICH.DE/ZEL/ZEL_HRONGEN

PHONE: [+49] (0)2461-61-4512

FAX: [+49] (0)2461-61-3990

List of contents

10. DATASHEETS

MICROPROCESSORS AND MICROCONTROLLERS

1. Introduction

Most of us know what a computer looks like. It usually has a keyboard, monitor, CPU (Central Processing Unit), printer, and a mouse. These types of computers, like the Mac or PC, are primarily designed to communicate (or "interface") with humans. Database management, financial analysis, or even word-processing are all accomplished inside the "big box" that contains the CPU, memory, hard drive, etc. The actual "computing", however, takes place within the CPU.

If you think about it, the whole purpose of a monitor, keyboard, mouse, & even the printer is to "connect" the CPU to the outside world.

But did you know that there are computers all around us, running programs & quietly doing calculations, not interacting with humans at all? These computers are in your car, on the Space Shuttle and in all modern instruments.

We call these devices "microcontrollers". Micro because they're small and controller because they "control" machines, gadgets, whatever. Microcontrollers by definition then are designed to connect to machines, rather than people. They're cool because, you can build a machine or instrument, write programs to control it and then let it work for you automatically. There are an infinite number of applications for microcontrollers. Your imagination is the only limiting factor! Hundreds (if not thousands) of different variations of microcontrollers are available. Some are programmed once & produced for specific applications, such as controlling your printer. Others are "re-programmable", which means they can be used over and over for different applications.

Microcontrollers are incredibly versatile - the same device may control a printer, a counter, or even your dosemeter.

So lets start and lets have a look what’s it all about.

2. What is a Microcomputer?

A Microcomputer is a complete computer system comprising at least three major components, the microprocessor (CPU), Memory and IO peripheral components; see Fig. 1.

A microcomputer could be a general purpose computer (like a PC) or a system designed to fulfil a special task (for example a controller system inside an instrument, microcontroller).

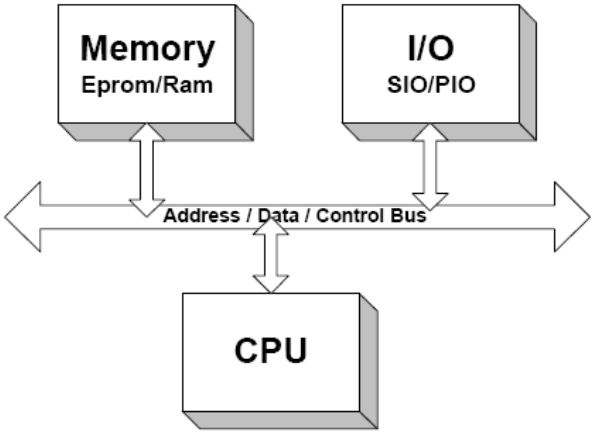


Fig. 1: MicroComputer

2.1. What is a Microprocessor?

A Microprocessor is a device containing functions equivalent to a small computer's Central Processing Unit (CPU). It is such capable of performing basic computer functions, and can be incorporated into system designs where such functions are required. A Microprocessor by definition means that this is only the central processing unit, with instruction decoder, registers and Arithmetic Logic processing Unit. A CPU does not include any memory or I/O components; see Fig. 2.

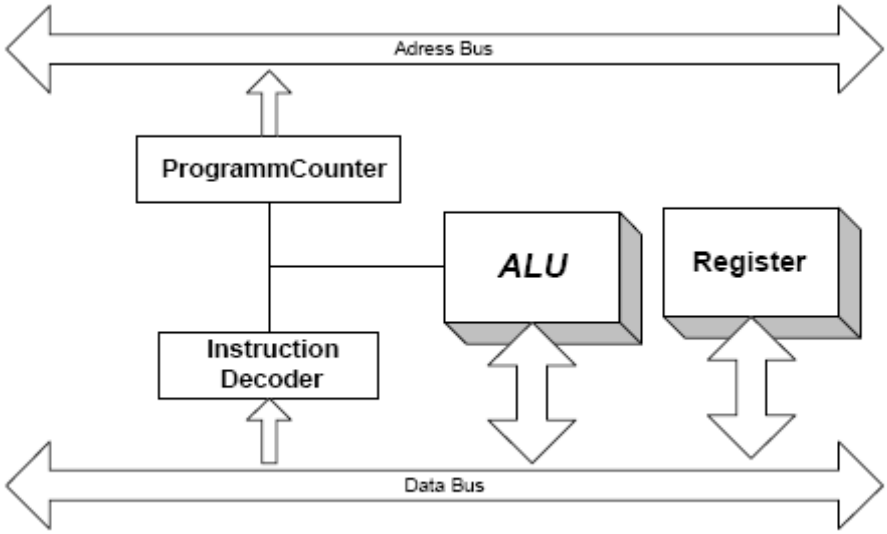


Fig. 2: MicroController

2.2 What is a Microcontroller?

A Microcontroller is a Microcomputer in a single Chip. That means that a microcontroller chip includes a microprocessor (CPU) as well as some often used peripherals. A controller is used to control (makes sense!) some process or aspect of the environment. A typical microcontroller application is the monitoring of my house. As the temperature rises, the controller causes the windows to open. If the temperature goes above a certain threshold, the air conditioner is activated.

As the process of miniaturization continued, all of the components needed for a controller were built right onto one chip (see Fig. 3). A one chip computer or microcontroller was born. A microcontroller is a highly integrated chip which includes, on one chip, all or most of the parts needed for a controller. The microcontroller could be called a "one-chip solution". It typically includes:

- CPU (central processing unit or the microprocessor)
- EPROM/PROM/ROM (Read Only Memory for the program code)
- RAM (Random Access Memory for the data)
- I/O (input/output) devices (serial, parallel, ADC, DAC etc.)
- Timers
- Interrupt controller

By only including the features specific to the task (control), cost is relatively low. A typical microcontroller has bit manipulation instructions, easy and direct access to I/O (input/output), and quick and efficient interrupt processing. Microcontrollers are a "one-chip solution" which drastically reduces parts count and design costs.

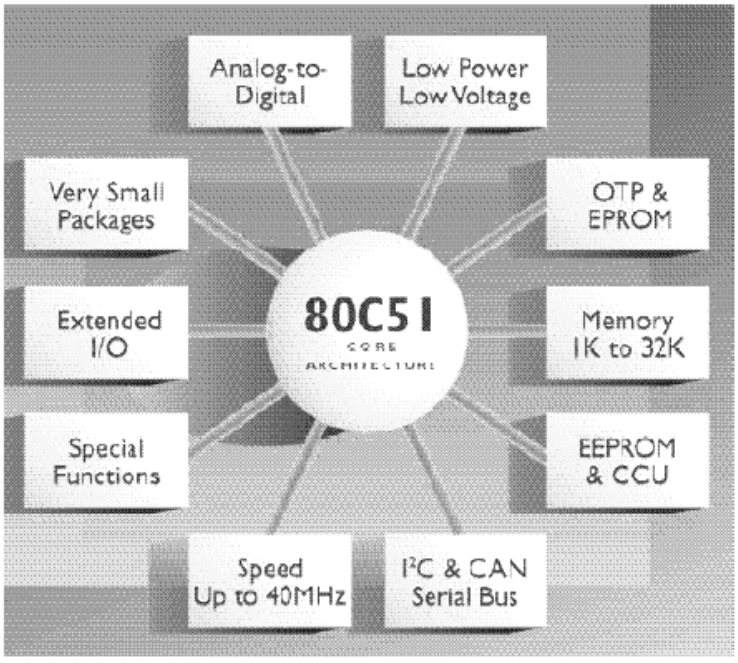


Fig. 3:

By only including the features specific to the task (control), cost is relatively low. A typical microcontroller has bit manipulation instructions, easy and direct access to I/O (input/output), and quick and efficient interrupt processing. Microcontrollers are a "one-chip solution" which drastically reduces parts count and design costs.

2.3 What is an Embedded Controller?

Simply (and naively stated) an embedded controller is a controller that is embedded in a greater system. You could say that an embedded controller is a controller (or computer) that is embedded into some device for some purpose other than to provide general purpose computing like a PC.

In addition to control applications such as the above home monitoring system, microcontrollers are frequently found in embedded applications (embedded controllers?).

Among the many uses you can find one or more microcontrollers

- in appliances like microwave oven, refrigerators, television and VCRs, stereos
- in automobiles engine control, diagnostics, climate control
- in environmental control like greenhouse, factories and home
- in instrumentation
- in aerospace
- and in thousands of other uses.

A special application that microcontrollers are well suited for is data logging. Stick one of these chips out in the middle of a corn field or up in a balloon, and monitor and record environmental parameters (temperature, humidity, rain, etc.). Small size, low power consumption, and flexibility make these devices ideal for unattended data monitoring and recording.

Microcontrollers come in many flavours and varieties. Depending on the power and features that are needed, you might choose a 4 bit, 8 bit, 16 bit, or 32 bit microcontroller. In addition, some specialized versions are available which include features specific for communications, keyboard handling, signal processing, video processing, and other tasks.

3. The major components of a microcomputer system:

The major components of every computer are the CPU, Memory, I/O devices and the system bus. These components can be found in any computer system, anyway if it is a microprocessor or microcontroller based system.

3.1. The CPU

CPU means Central Processing Unit. The CPU is the kernel of each computer. All data (instructions and user data) is read from the CPU into the CPU registers, see Fig. 4.

Instructions (the program code) are read into the instruction register, next they are decoded in the instruction decoder. Depending on the instruction as next following data fetches. The data are stored in the arithmetic registers (accumulator), from there they are processed in the ALU (Arithmetic & Logic Unit). The ALU performs all arithmetical and logical processes on the data. The result from the ALU is written back into a CPU register. From there the data can be written back to the memory or an I/O device.

The most important signals from a CPU are the address bus, data bus and control signals. These signals are connecting the CPU to the memory and the I/O devices.

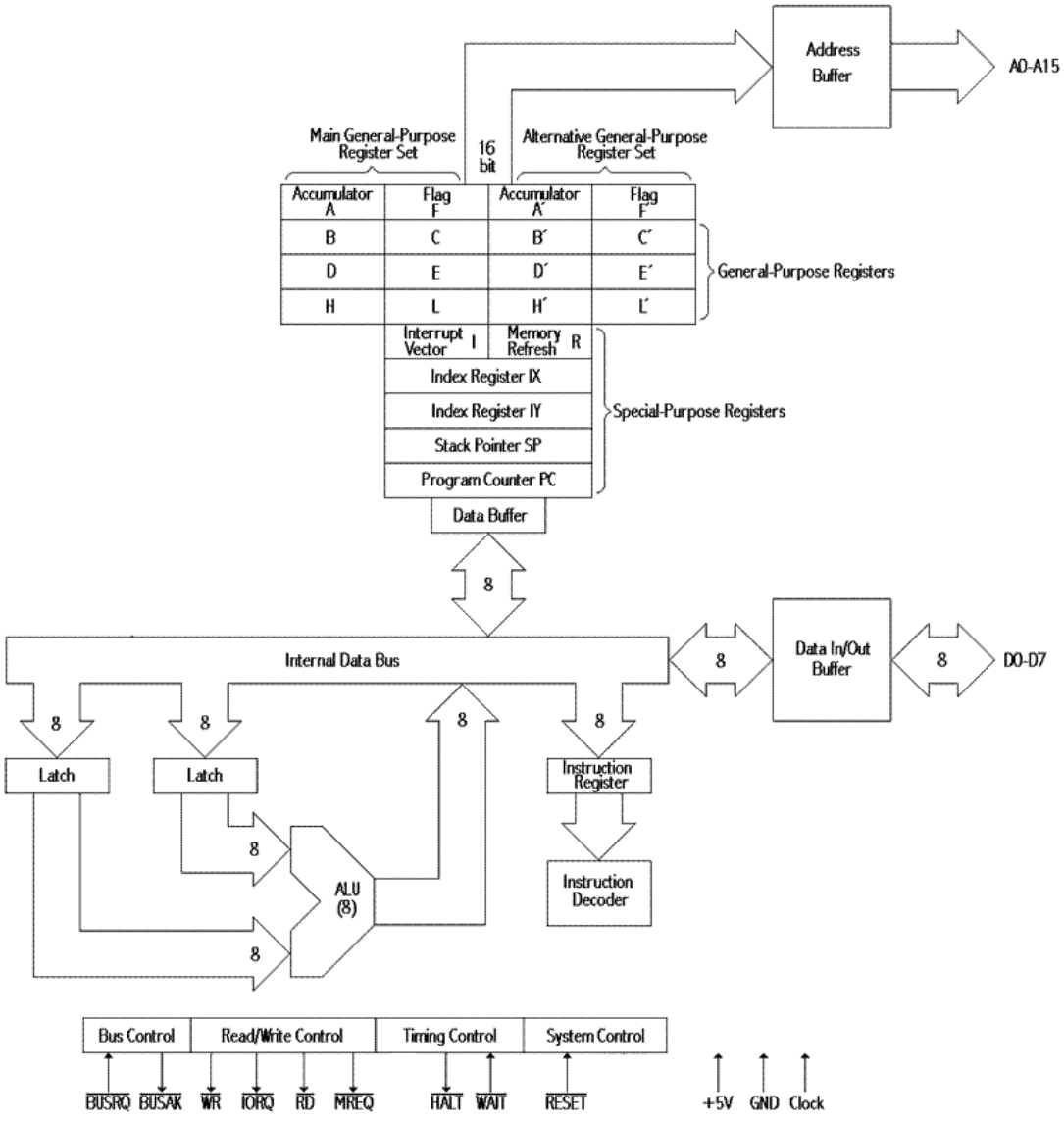


Fig. 4 Simplified block diagram of a CPU

3.2 The Memory

Every computer system needs a memory block, see Fig. 5. Memory can be external memory build by extra chips or the on chip memory of a microcontroller. In general there are two types of memory:

- Program memory
- Data memory

Program memory is used to hold the application program. This must be memory which doesn't loose its information when power is off. At power on the CPU begins to read the instructions from this memory. In most microcomputer applications the program memory is a READ ONLY MEMORY (ROM). There are different types of ROM available. The most used is the EPROM. This chip could be programmed with a special programmer and could be erased by applying ultra-violet light. During the normal use of this chip inside of the microcomputer system this memory can not be written, only read.

Data memory is used for the dynamic data which is generated by the application program and for the STACK. The stack is a portion of memory where the CPU saves his own internal register data for calling a subroutine. Data memory must be able to read and write from the CPU, it is so called Random Access Memory (RAM).

RAM chips loose the data when powered down. The amount of memory in a microcomputer system depends strongly on the application.

There are small controller applications which have only 512 Bytes ROM and 128 Bytes RAM. Bigger microcontroller application may have up to some Megabyte of EPROM and also RAM.

Because of the 16 bit address bus of the most popular microcontrollers the address space is limited to 64 Kbytes. So you will find in many applications 32 Kbytes EROM and 32 Kbytes RAM.

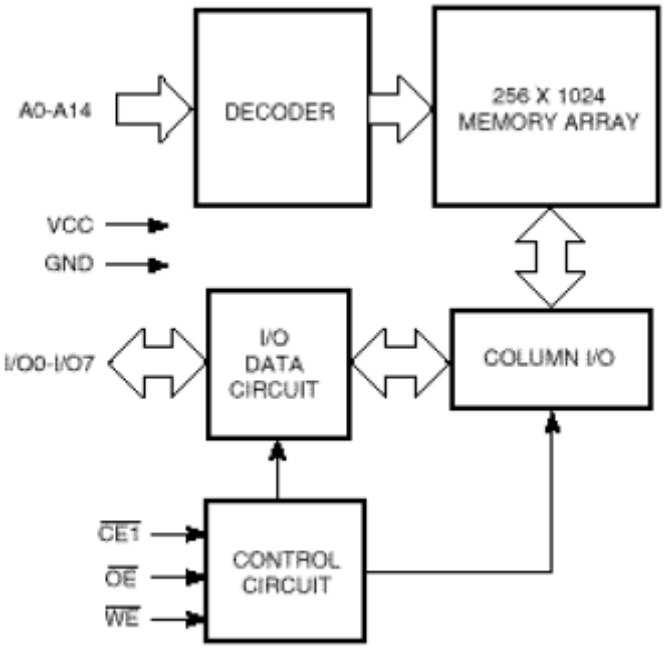


Fig. 5: Block diagram of a memory

3.2.1. Options for storing programs.

Another consideration in circuit design is how to store programs. Instead of using disk storage, most microcontroller circuits store their programs on-chip. For one-of kind projects or small-volume production, EPROM has long been the most popular method of program storage. Besides EPROM's, other options include EEPROM, ROM, non-volatile (NV), or battery-backed, RAM, and Flash EPROM. The program memory may be in the microcontroller chip, or a separate component. To save a program in **EPROM**, you must set the EPROM's data and address pins to the appropriate logic levels for each address and apply special programming voltages and control signals to store the data at the selected address. The programming process is sometimes called *burning* the EPROM. You erase the contents by exposing the chip's quartz window, and the circuits beneath it, to ultraviolet energy.

Some microcontrollers contain a one-time-programmable, or field-programmable, EPROM. This type has no window, so you can't erase its contents, but because it's cheaper than a windowed IC, it's a good choice when a program is finished and the device is ready for quantity production. Several techniques are available for programming EPROM's and other memory chips. With a manual programmer, you flip switches to toggle each bit and program the EPROM byte by byte. This is acceptable for short programs, but quickly becomes tedious with a program of any length. Computer control simplifies the job greatly. With an EPROM programmer that connects to a personal computer, you can write a program at your keyboard, save it to disk if you wish, and store the program in EPROM in a few easy steps. Data sheets for EPROM's rarely specify the number of erase and reprogramming cycles a device is guaranteed for, but a typical EPROM should endure 100 erase/program cycles, and usually many more.

EEPROM are much like EPROM's except that they are electrically erasable—no ultraviolet source is required. Limitations of EEPROM's include slow speed, high cost, and a limited number of times that they can be reprogrammed (typically 10,000 to 100,000).

ROM are most cost-effective when you need thousands of copies of a single program. ROMs must be factory-programmed and once programmed, can't be changed.

NVRAM typically includes a lithium cell, control circuits, and RAM encapsulated in a single IC package. When power is removed from the circuit, the lithium cell takes over and preserves the information in RAM, for 10 years or more. You can reprogram an NVRAM n infinite number of times, with the only limitation being battery life.

Flash EPROM is electrically erasable, like EEPROM, but most Flash devices erase all at once, or in a few large blocks, rather than byte-by-byte like EEPROM. Some Flash EPROMs require special programming voltages. As with EPROMs, the number of erase/program cycles is limited.

The 8052-BASIC uses two types of program memory. An 8-kilobyte, or 8K, on-chip ROM stores the BASIC-52 interpreter. For storing the BASIC-52 programs that you write, the BASIC-52 language has programming commands that enable you to save programs in external EPROM, EEPROM, or NVRAM.

Other memory Most systems also require a way to store data for temporary use. Usually, this is RAM, whose contents you can change as often as you wish. Unlike EPROM, ROM, EEPROM, and NVRAM, the contents of the RAM disappear when you remove power the chip (unless it has battery back-up).

3.3 The I/O Devices

In general a CPU with memory is ready to run. But of what need is a computer system without transaction to the real world? So every computer system needs some I/O devices. The word I/O-device is widespread. It could mean everything that is good for interfacing to the external world.

Typical I/O devices are:

- keyboard controller for inputs from the user
- display controller for information to the user
- parallel I/O to switch on/off some lights or relays
- serial I/O to communicate with other computers
- realtime clock so the computer knows the time and date
- analog inputs to measure a physical phenomena
- analog outputs to control a process

Finally, input/output (I/O) requires design decisions. Most systems require interfaces to things like sensors, keypads, switches, relays, and displays. Most microcontrollers have ports for interfacing to the world outside the chip. The 8052-BASIC uses many of its ports for accessing external memory and performing other special functions, but some port bits are available for user applications, and you can easily increase the available I/O by adding support chips.

Keyboards:

A typical keyboard in embedded controller systems (Fig. 6) is build with 16 keys designed as a 4 by 4 matrix. The keyboard controller scans the X-lines one by one at a time and looks for a response at the Y-lines. If a key is pressed then the keyboard controller knows from the X/Y information which key has been pressed and stores this information in an internal register. This information is read by the CPU.

A commonly used keyboard controller is the 74C922 or 74C923. In some applications there is no special keyboard controller. In this case the scan of the keyboard matrix is done by the CPU itself trough a parallel I/O interface (PIO).

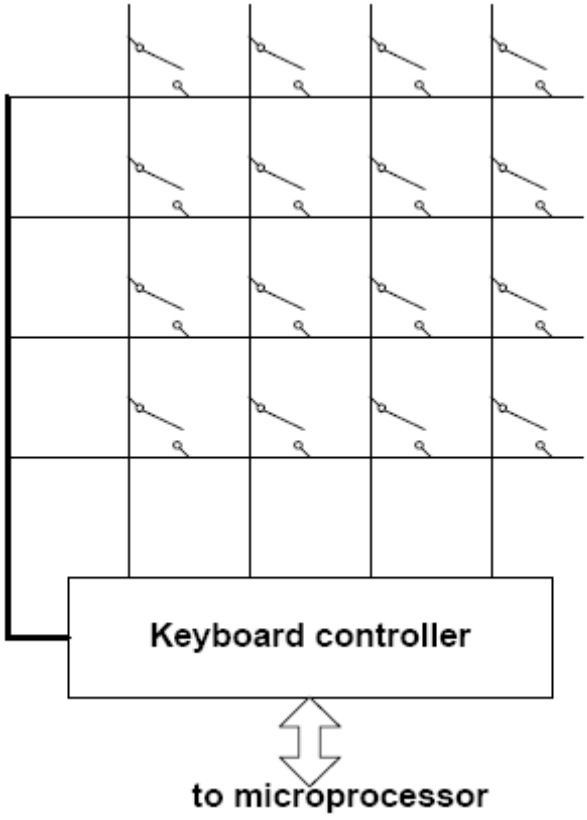
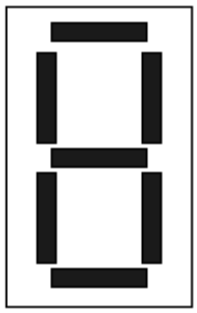


Fig. 6: Connecting a 16 key keypad

Displays:

The Hardware needed for the display depends strongly on the used display. One often used display is the so called 7-segment display.



These are LED based display modules for displaying only numbers 0 to 9 and also the hex characters A to F. For driving the LED's inside of this 7-segment display you need a component which delivers the needed current of about 10 mA per Segment. Also the decoding from the binary number (0 to 9 for example) to the segments which must be driven must be performed.

The component 74LS47 is a typical binary to 7-segment display decoder.

Other Displays are the LCD Displays. Alpha-numeric modules display characters, numerals, symbols and some limited graphics. These LCD modules have their own controller. Interface is achieved via a bi-directional, parallel ASCII data bus.

Necessary features such as Character Generation, Display RAM Addressing, Cursor Scrolling, Blanking, and Handshake are all included. User programmable fonts are supported. In summary, these modules are the simplest and most economic means to communicate meaningfully between any micro-system and the outside world.

Alpha-numeric modules range from 8 to 80 characters per line. One, two or four character lines may be chosen. Character height spans 0.130" (3.31 mm) to 0.500" (12.71 mm). Most formats are available in a variety of packages to meet various mounting requirements.

Parallel I/O:

Parallel I/O is the most used I/O device in small microcomputer applications. A parallel I/O means that there are a number of static input or output bits. These outputs can be used to drive some lights, to switch some relays, to drive a display and so on. The input bits can be used to read information from switches etc.

A simple parallel I/O can be done by a simple register component like a 74LS373. There are also some special parallel I/O components available, like the PIO 8255 or the Z80-PIO. With components the user can switch every bit as an input or output, so these components are very flexible for many applications.

Serial I/O:

Serial I/O is used to send data over longer distances to another computer system, display systems (terminal) or to a printer.

While speaking about serial interfaces there are two words which describe the type of the serial interface:

- **DTE** is acronym for **D**ata **T**erminal **E**quipment like computers, printers and terminals.
- **DCE** is acronym for **D**ata **C**ommunication **E**quipment like modems.

Wiring a cable for DTE to DCE communication is easy as all lines go straight from pin x to pin x.

But wiring a cable for DTE to DTE (nullmodem) or DCE to DCE requires that some lines are crossed. A signal should be wired from pin x to the opposite signal at the other end, which means that for example Transmit (TxD) goes to Receive (RxD) and vice versa.

Real-time clock:

A real-time clock is useful in application where the computer must known the absolute time and date, for example for switching on and off a peripheral device at a given day and time. Typical real-time clock component are the RTC62421 or RTC 72421 chip.

Analog inputs:

Many applications need an analog input for measuring a physical signal like a temperature or pressure. For measuring a physical phenomenon you first need a sensor, which converts the physical phenomena into an electrical signal. Most of the sensors delivers a very small signal so at next you need some operational amplifiers to bring the signal into the level which is needed for the **A**nalog to **D**igital **C**onverter (ADC) (in most cases +/- 5 or +/- 10 Volt). The ADC itself is a single chip. There are many ADCs on the market. All of them have different speed and resolution. For microprocessor applications there are many ADCs with converting time of about 10 to 100 µs. For simple measurements there are ADCs with 8 bit resolution, mostly used today are the 12 bit resolution, but there are also ADCs with 16, 18, 20 and 22 bit resolution used in microprocessor systems.

Analog outputs:

Analog outputs are used to control external devices by a linear voltage. This could be a high voltage supply where you can adjust the high voltage with an input voltage or another example could be a heater where you can adjust the heating power by a external voltage. **D**igital to **A**nalog **C**onverters (DAC) are used to generate a current or a voltage from the digital information written into a register. All the DAC's need an operational amplifier behind their output pins to adjust their output level to the voltage needed for the external device.

Also here are a lot of DAC's are on the market, with bit resolutions ranging from 8, 12 up to 16 bit.

Because of the big ADC and DAC component market it is very difficult to give here some component examples. Instead I will say that there are some component companies which are great on the analog IO market. If you see a component with a Label of: Analog Devices Burr Brown orMaxim then in most of the cases these are components for the analog I/O section.

3.4 The Address/Data/Control Bus (system bus)

The system bus (address/data/control lines) is the connection between the several components of a microprocessor system. The data bus width depends on the microprocessor (8, 16 or 32 bits), the address bus is at least 16 bit wide. The system bus is used for every data transfer between the Memory and the CPU for operation code fetches, for data transfer to and from the RAM for dynamic data fetch and also for data to and from the I/O devices. The transaction over the system bus can be grouped into two groups:

- a read cycle: data from the Memory or I/O device into the CPU
- a write cycle: data from the CPU to the Memory or I/O device

The control lines of the system bus controls what kind of data transfer is in progress.

The most important control lines are:

- the CHIPSELECT signal /CS
- the READ signal /RD
- the WRITE signal /WR

(/ - means that this signal is low active)

Every bus transfer starts with the output of an address onto the address lines. The lower address lines are direct connected to the Memory. Some address lines (the upper lines) are used for address decoding. Address decoding means that there is a logic circuit which decides based on the address what device (EPROM, RAM I/O device) should be activated. The address decoder then gives a unique CHIP SELECT (CS) signal to that device. For that, every device in a microprocessor system needs a unique address.

The CS signal enables the device to transmit or receive data during this bus cycle. As next one of the signals either /RD or /WR is activated from the CPU, the time during which the device transmits data to the CPU (Read cycle) or receives data from the CPU (Write cycle). For details please refer to Fig. 7.

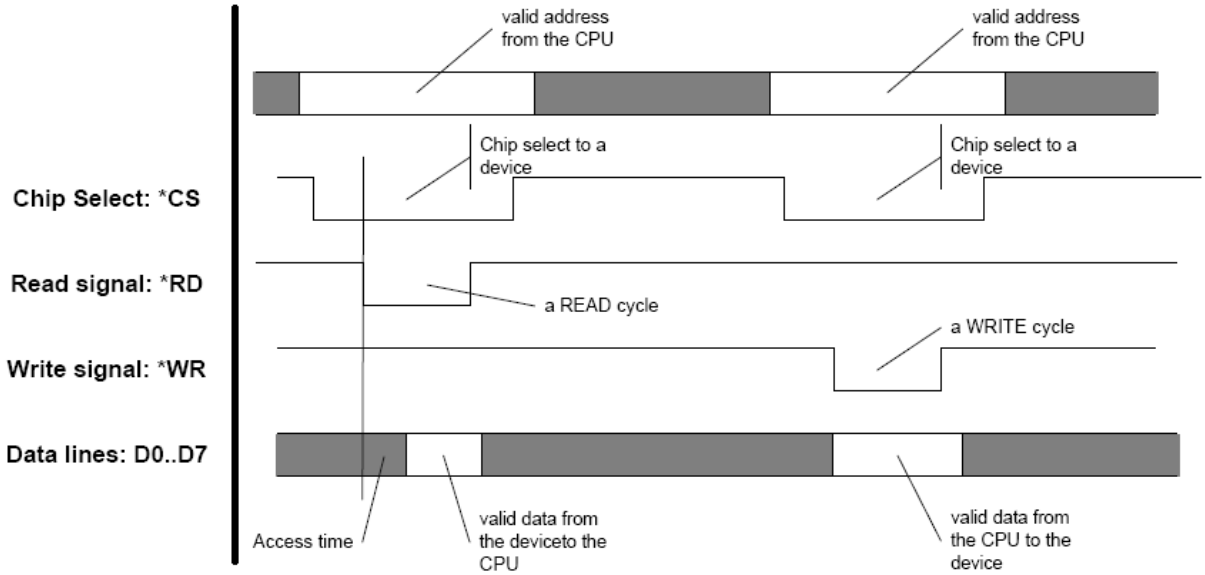


Fig. 7: General bus timing diagram

Every bus cycle (Read and Write) starts with a valid address from the CPU. As next the address decoder generates a unique CHIP SELECT for one device. Then the /RD or /WR signal is activated from the CPU to tell whether it is a read or write operation. By a read operation the device has to drive the data to the data bus, this could take a short time (data access time), but the data must be stable at the rising edge of the /RD signal. With the rising edge of the RD signal the CPU latches the data internal.

In a write cycle the CPU itself drive the data bus. Also in this case it is not sure that the data is stable at the beginning of the cycle. The data must be stable at the rising edge of the /WR cycle, so that the device could also latch the data with the rising edge of the /WR signal.

4. A typical MicroProcessor (Z80)

I get really nervous reading the trade press. Magazines will have you believe that the only viable processor for even the simplest applications is a 50 MHz 486 or a screaming RISC machine. How many designers are using these CPUs in their embedded designs?

Personally, I prefer 8 and 16 bit CPUs. They're cheap, they're cheap to interface to, and they are easy to work with. Remember that 16 and 32 bit machines will have 2 or 4 ROM and RAM chips per word, compared to 1 for an 8 bit processor, greatly increasing memory costs. In fact, an 8 bit processor exactly matches the bus width of the vast majority of memories and peripherals. An 8 bit bus uses memory efficiently.

Simpler circuitry, with fewer components, can be used. I feel four processor families can satisfy most embedded designs. I'll leave out 4 bit choices, as these applications are often so specialized that a custom or semi-custom chip is often the most cost effective solution. My choices are the 8051 family, the Z80 family, the 80186 (and V-Series) families, and the 68000 and its derivatives.

In this document I will do my explanations about microprocessor on the example of the Z80 and his derivates.

Current Z80 Technology

The Z80 is essentially unchanged from the original version introduced in the mid-70s. Now CMOS versions are common, and clock rates have skyrocketed. Zilog's newest offering runs at 20 MHz. This speed is no panacea- it comes at the price of expensive ROMs and RAM’s. Still, even at 6 to 10 MHz the Z80 has respectable performance. It would be a terrible Windows machine, but is more than adequate for an awful lot of embedded systems.

The Z80 itself is attractive due to its low cost and wide availability. It's pretty easy to find Z80s for less than a buck. Tools are everywhere. Though its index instructions are a little crude, it is a far better C machine than, say, the 8051.

The Z80 in isolation would be a dead-end line. What makes it interesting is the high integration derivatives CPUs spawned by the architecture. All four of the processor families I mentioned earlier exist in multiple proliferation versions. A processor by itself is useful; one integrated with a number of on-chip peripherals is compelling.

The 64180 is a Hitachi-supplied Z80 core with numerous on-chip "extras". Zilog's version is the Z180, which is essentially the same part. As of this writing, Zilog sells parts running at speeds to 15 MHz. The 64180/Z180 microprocessor integrates many of the functions traditionally assigned to peripheral circuitry onto a single chip.

The designers picked an architecture compatible with the Z80, giving Z80 users a completely software compatible upgrade path. Old Z80 designs can be converted to the 64180 with essentially no loss in software investment. New designs will benefit from the processor's low cost, powerful instruction set, minute power consumption, and high level of integration.

Both Toshiba and Zilog sell the 84013 and 84015, which are Z80 cores with conventional Z80 peripherals integrated on-board. These processors are natural migration paths for current users of Z80s wishing to reduce systems costs.

Both the 84013 and the 84015 include one SIO (Z80-specific Serial I/O), and one CTC (again, a Z80-specific timer part). The SIO and CTC are functionally identical to the discrete chip SIO/CTC used in so many older designs. In addition, the newer parts include a watchdog timer and clock generator. The 84015 comes in a 100 pin quad flat pack. To take advantage of the extra pins the vendors added in a Z80-like PIO (parallel I/O) port, with 16 parallel lines and 4 handshaking likes.

5. Microcontroller Basics

This chapter introduces you to the world of microcontrollers, including definitions, some history, and a summary of what's involved in designing and building a microcontroller project.

A microcontroller is a computer-on-a-chip, or, if you prefer, a single-chip computer. *Micro* suggests that the device is small, and *controller* tells you that the device might be used to control objects, processes, or events. Another term to describe a microcontroller is e*mbedded controller,* because the microcontroller and its support circuits are often built into, or embedded in, the devices they control. You can find microcontrollers in all kinds of things these days. Any device that measures, stores, controls, calculates, or displays information is a candidate for putting a microcontroller inside. The largest single use for microcontrollers is in automobiles; just about every car manufactured today includes at least one microcontroller for engine control, and often more to control additional systems in the car. In desktop computers, you can find microcontrollers inside keyboards, modems, printers, and other peripherals. In test equipment, microcontrollers make it easy to add features such as the ability to store measurements, to create and store user routines, and to display messages and waveforms. Consumer products that use microcontrollers include cameras, video recorders, compact-disk players, and ovens. And these are just a few examples.

A microcontroller is similar to the microprocessor inside a personal computer. Examples of microprocessors include Intel's 8086, Motorola's 68000, and Zilog's Z80. Both microprocessors and microcontrollers contain a central processing unit, or CPU. The CPU executes instructions that perform the basic logic, math, and data moving functions of a computer. To make a complete computer, a microprocessor requires memory for storing data and programs, and input/output (I/O) interfaces for connecting external devices like keyboards and displays. In contrast, a microcontroller is a single-chip computer because it contains memory and I/O interfaces in addition to the CPU. Because the amount of memory and interfaces that can fit on a single chip is limited, microcontrollers tend to be used in smaller systems that require little more than the microcontroller and a few support components. Examples of popular microcontrollers are Intel's 8052 (including the 8052-BASIC, which is the focus of this book), Motorola's 68HC11, and Zilog's Z8.

A Little History

To understand how microcontrollers fit into the always-expanding world of computers, we need to look back to the roots of micro computing. In its January 1975 issue, Popular Electronics magazine featured an article describing the Altair 8800 computer, which was the first microcomputer that hobbyists could build and program themselves. The basic Altair included no keyboard, video display, disk drives, or other elements we now think of as essential elements of a personal computer. Its 8080 microprocessor was programmed by flipping toggle switches on the front panel.

Standard RAM was 256 bytes and a kit version cost $397 ($498 assembled). A breakthrough in the Altair's usability occurred when a small company called Microsoft offered a version of the BASIC programming language for it.

Of course, the computer world has changed a lot since the introduction of the Altair. Microsoft has become an enormous software publisher, and a typical personal computer now includes a keyboard, video display, disk drives, and Megabytes of RAM. What's more, there's no longer any need to build a personal computer from scratch, since mass production has drastically lowered the price of assembled systems. At most, building a personal computer now involves only installing assembled boards and other major components in an enclosure.

A personal computer like Apple's Macintosh or IBM's PC is a general-purpose machine, since you can use it for many applications - word processing, spreadsheets, computer-aided design, and more - just by loading the appropriate software from disk into memory.

Interfaces to personal computers are for the most part standard ones like those to video displays, keyboards, and printers.

But along with cheap, powerful, and versatile personal computers has developed a new interest in small, customized computers for specific uses. Each of these small computers is dedicated to one task, or a set of closely related tasks. Adding computer power to a device can enable it to do more, or do it faster, better, or more cheaply. For example, automobile engine controllers have helped to reduce harmful exhaust emissions. And microcontrollers inside computer modems have made it easy to add features and abilities beyond the basic computer-to-phone-line interface.

In addition to their use in mass-produced products like these, it's also become feasible to design computer power into one-of-a-kind projects, such as an environmental controller for a scientific study or an intelligent test fixture that ensures that a product meets its specifications before it's shipped to a customer. At the core of many of these specialized computers is a microcontroller. The computer's program is typically stored permanently in semiconductor memory such as ROM or EPROM. The interfaces between the microcontroller and the outside world vary with the application, and may include a small display, a keypad or switches, sensors, relays, motors, and so on. These small, special-purpose computers are sometimes called single-board computers, or SBC's. The term can be misleading, however, since the computer doesn't have to be on a single circuit board, and many types of computer systems, such as laptop and notebook computers, are now manufactured on a single board.

New Tools

To design and build a computer-controlled device, you need skills in both circuit design and software programming. The good news is that a couple of recent advances have simplified the tasks involved.
One is the introduction of microcontrollers themselves, since they contain all of the elements of a computer on a single chip. Using a microcontroller can reduce the number of components and thus the amount of design work and wiring required for a project. The 8052-BASIC microcontroller even includes its own programming language, called BASIC-52. The other development is personal computers themselves. A desktop computer can help tremendously by serving as a *host system* for writing and testing programs. As you are developing a project, you can use a serial link to connect the host system to a *target system,* which contains the microcontroller circuits you are testing. You can then use the personal computer's keyboard, video display, disk drives, and other resources for writing and testing programs and transferring files between the two systems.

The 8052-BASIC chip described in this book is perfect for many simpler applications, especially control and monitoring tasks. Because the chip is easy to use, it's a good way to learn about microcontrollers and computers in general. Although you can't do the most complex projects with it, you can do a lot, at low cost and without a lot of hassle.

Choosing a chip

Does it matter which microcontroller chip you use? All microcontrollers contain a CPU, and chances are that you can use any of several devices for a specific project. Within each device family, you'll usually find a selection of family members, each with different combinations of options. For example, the 8052-BASIC is a member of the 8051 family of microcontrollers, which includes chips with program memory in ROM or EPROM, and with varying amounts of RAM and other features. You select the version that best suits your system's requirements.

Microcontrollers are also characterized by how many bits of data they process at once, with a higher number of bits generally indicating a faster or more powerful chip. Eight-bit chips are popular for simpler designs, but 4-bit, 16-bit, and 32-bit architectures are also available. The 8052-BASIC is an 8-bit chip. Power consumption is another consideration, especially for battery-powered systems. Chips manufactured with CMOS processes usually have lower

power consumption than those manufactured with NMOS processes. Many CMOS devices have special standby or “sleep” modes that limit current consumption to as low as a few microamperes when the circuits are inactive. Using these modes, a data logger can reduce its power consumption between samples, and power up only when it’s time to take data. The 8052-BASIC chip is available in both NMOS and CMOS versions. The original 8052-BASIC was an NMOS chip, offered directly from Intel. (Intel’s term for its NMOS process is HMOS.) Although Intel never offered a CMOS version directly, Micromint became a source by ordering a batch of CMOS 8052’s with the BASIC-52 programming language in ROM. The CMOS version, the 80C52-BASIC, has maximum power consumption of 30 mA, compared to 175 mA for the NMOS 8052-BASIC. All microcontrollers have a defined instruction set, which consists of the binary words that cause the CPU to carry out specific operations. For example, the instruction $0010\ 0110_B$ tells an 8052 to add the values in two locations. The binary instructions are also known as operation codes, or opcodes for short.

The opcodes perform basic functions like adding, subtracting, logic operations, moving and copying data, and controlling program branching. Control circuits often require reading or changing single bits of input or output, rather than reading and writing a byte at a time. For example, a microcontroller might use the eight bits of an output port to switch power to eight sockets. If each socket must operate independently of the others, a way is needed to change each bit without affecting the others. Many microcontrollers include bit-manipulation (also called Boolean) opcodes that easily allow programs to set, clear, compare, copy, or perform other logic operations on single bits of data, rather than a byte at a time.

5.1 A typical MircoController (8051)

The 8051 is an 8 bit microcontroller originally developed by Intel in 1980. It is the world's most popular microcontroller core, made by many independent manufacturers (truly multi-sourced). There were 126 million 8051s (and variants) shipped in 1993!!

A typical 8051 contains:

⇒ CPU with Boolean processor

⇒ 5 or 6 interrupts: 2 are external, 2 priority levels

⇒ 2 or 3 16-bit timer/counters

⇒ programmable full-duplex serial port (baud rate provided by one of the timers)

⇒ 32 I/O lines (four 8-bit ports)

⇒ RAM

⇒ ROM/EPROM in some models

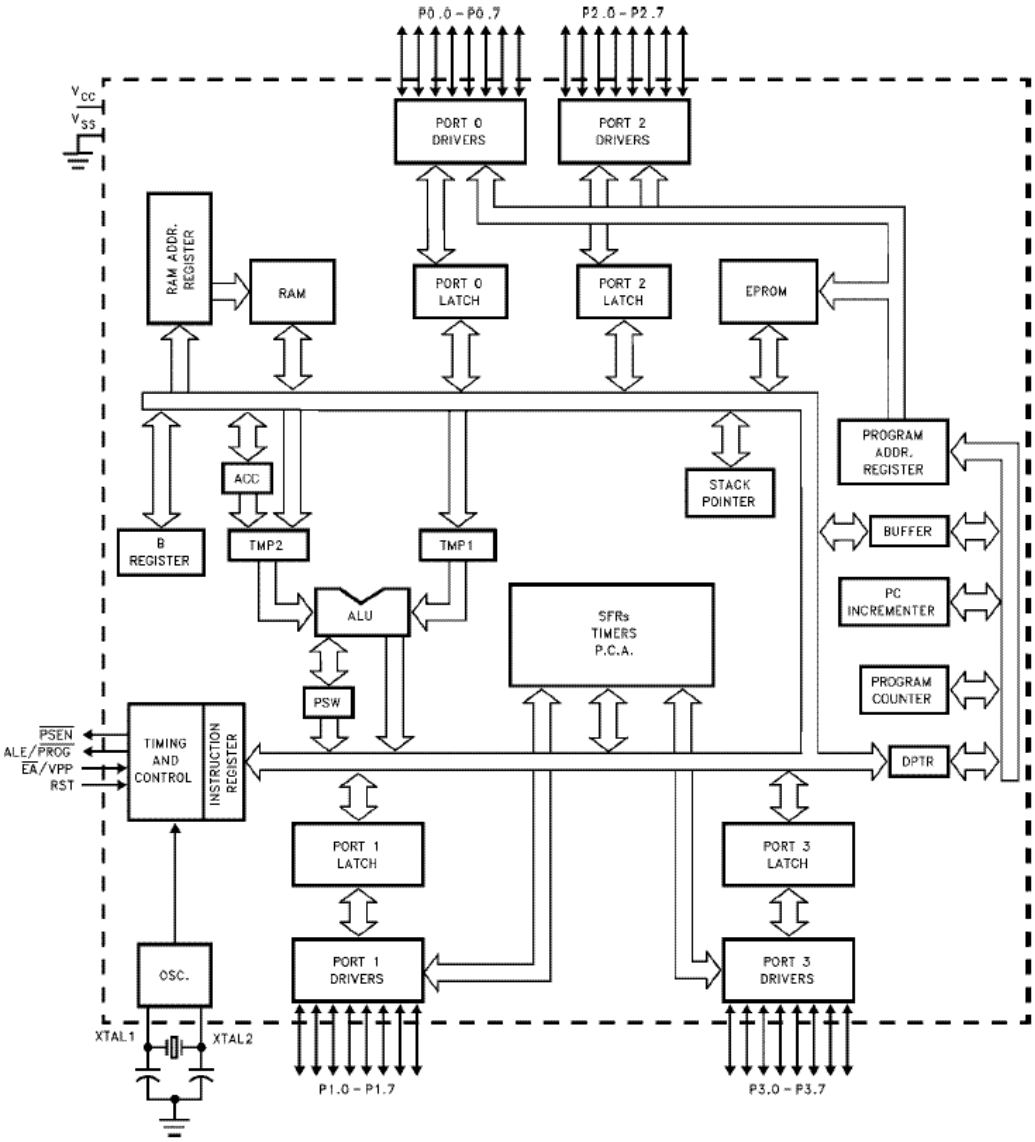


Fig. 8: Block diagram of a 8051 MCU

The 8051 architecture is a tad bizarre, but then so are the architectures of most microcontrollers due to their specialization (check out the PIC for creativity). One vexing problem with the 8051 is its very non-orthogonal instruction set – especially the restrictions on accessing the different address spaces. However, after some time programming the chip, you can get used to it - maybe even appreciate it.

One strong point of the 8051 is the way it handles interrupts. Vectoring to fixed 8-byte areas is convenient and efficient. Most interrupt routines are very short (or at least they should be), and generally can fit into the 8-byte area. Of course if your interrupt routine is longer, you can still jump to the appropriate routine from within the 8 byte interrupt region.

The 8051 instruction set is optimized for the one-bit operations so often desired in real-world, real-time control applications. The Boolean processor provides direct support for bit manipulation. This leads to more efficient programs that need to deal with binary input and output conditions inherent in digital-control problems. Bit addressing can be used for test pin monitoring or program control flags.

8051 Flavours

The 8051 has the widest range of variants of any embedded controller on the market. The smallest device is the Atmel 89c1051, a 20 Pin FLASH variant with 2 timers, UART, 20mA. The fastest parts are from Dallas, with performance close to 10 MIPS! The most powerful chip is the Siemens 80C517A, with 32 Bit ALU, 2 UARTS, 2K RAM, PLCC84 package, 8 x 16 Bit PWM's, and other features.

Among the major manufacturers are:

- AMD Enhanced 8051 parts (no longer producing 80x51 parts)
- Atmel FLASH and semi-custom parts
- Dallas Battery backed, program download, and fastest variants
- Intel 8051 through 80C51gb / 80C51sl
- Matra 80C154, low voltage static variants
- OKI 80c154, mask parts
- Philips 87C748 through 89Cc588 - more variants than anyone else
- **Siemens 80C501 through 80C517a, and SIECO cores**
- SMC COM20051 with ARCNET token bus network engine
- SSI 80x52, 2 x HDLC variant for MODEM use

8051 Types of Memory

Before we delve into the guts of the 8051, it is important to define the four types of memory that are available. The distinction between each type of memory is important because different assembly language instructions are used to access each type of memory, and each type of memory has unique qualities which must be considered when first designing the hardware and/or software.

The four types of memory are:

Internal RAM The microcontroller contains a small amount of on-chip RAM which is used primarily for internal program registers, the stack, and often user variables which need to be accessed quickly or on a frequent basis. In the 8051 and 8031 models, the total amount of Internal RAM is 128 bytes. In the 8052 and 8032 models, the total amount of Internal RAM is 256 bytes (an easy way to remember this is by looking at the last digit of the model of the chip: a "1" means 128 bytes, and a "2" means 256 bytes). It is important to remember that the internal "R" registers, the stack, and the "bit registers" (more on all these later) all share these precious 128 or 256 bytes of Internal RAM. Thus it is often important to use space within Internal RAM for variable storage on a very sparing basis to avoid problems, particularly stack overflows. The Internal RAM is volatile (it's contents are lost when the power fails) and is the fastest type of memory available in an 8051 architecture since it is "on-chip."

External RAM The microcontroller's ability to accept external RAM allows you to handle greater quantities of data than would otherwise be possible if you were restricted to the 8051's Internal RAM. As the name suggests, External RAM is additional Random-Access-Memory which is external to the chip itself. External RAM, like Internal RAM, is volatile so it's contents will be lost when the power fails. Like EPROM memory, External RAM is limited to 64k (Note: you may have 64k of RAM and 64k of EPROM at the same time). External RAM is not as fast as Internal RAM since the microcontroller has to move the bytes to/from the External RAM chip(s). Like External EPROM memory, the use of External RAM will cause you to lose 16 of the 32 input/output lines normally available. Note: If you use both External EPROM and External RAM, you will lose the same 16 input/output line. That is to say, you don't lose 16 bits for each type of External memory used, just for the fact that some type of external memory is used.

Internal EPROM Several models of the 8051 microcontroller (specifically, the 8751 and the 8752) offer an on-chip EPROM. This means rather than using an external EPROM chip, you may burn your program directly into the 8751 (or 8752). This has a number of important advantages:

- Since your program is stored on-chip, an additional EPROM chip (and circuit design to support it) is unnecessary.
- When your program is stored on-chip you do not lose the 16 input/output lines that you lose when using an External EPROM.
- The chip can be configured to prevent the contents of the EPROM from being read off the chip.

A disadvantage of external EPROM's is that anyone can read the contents and reverse engineer your code. Using Internal EPROM, you can burn your program onto the 8751/8752 and instruct it to deny read access by would-be hackers. The drawback to Internal EPROM's is that

the chips themselves are somewhat more expensive and that the amount of EPROM space is limited to 4k (8751) or 8k (8752) which may make it a non-option if your program is large.

External EPROM The microcontroller is generally pretty useless unless you can somehow run your "program" on it. One of the most common ways of running your program is to compile it and subsequently "burn" it into a separate EPROM chip. The 8051 can then be attached to the EPROM at the hardware level such that the microcontroller will run the program stored in the EPROM. Since your code is "burned" into the EPROM, the contents are not lost when the power fails. However, since it is a type of "ROM" (Read Only Memory) it is not possible for your program to modify the data stored therein. Thus, it is ideal for storing your program and other constants. The 8051 can access up to 64k (65536 bytes) of EPROM memory. Your programs, thus, are limited to 64k (there are actually tricks involving specialized circuit design and specialized support software that can break the 64k program limit, but without taking these fancy steps the limit is 64k). Additionally, if an External EPROM is used you will lose 16 of the 32 input/output lines normally available to you.

8051 Memory Access Procedure

When using external memory with the 8052 then port 0 and port 1 are used as a multiplexed address/data bus. By this you loose 16 of the 32 I/O bits, but for this you can extend your external address range to 64 Kbytes external locations. (EPROM, RAM or also memory mapped I/O devices)

A multiplexed address/data bus means that the lower lines of this 16 bit interface is used as a part of the address and later as datelines. The access to external memory is as followed, refer to Fig. 9. Whenever the 8051 requires access (either read or write) to External Data Memory (RAM), the following sequence of events takes place:

- The full address is loaded into pins Addr0 through Addr15.
- The ALE pin is strobed for a short period of time.
- The lower byte of the address (Addr0 through Addr7) is replaced by the bits of the data to be written.
- After a short delay, either the /WR, /RD or PSEN pin is strobed to execute the memory access operation. /WR and /RD deals with access to external data memory (RAM) whereas PSEN deals with access to external code memory (such as from an EPROM).

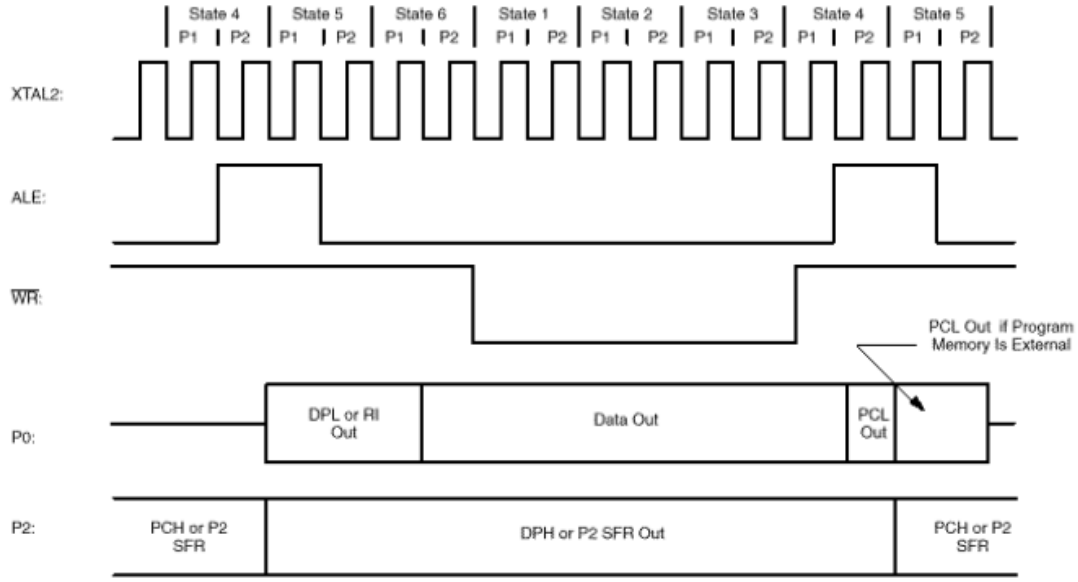


Fig. 9: Access to external memory

The logic behind this is that by doubling the function of pins port0-0 through port0-7 more effective use of pins is achieved. That is to say, if these pins did not serve double-functions the 8051 would either require an additional 8 pins (48 pins instead of 40) or would cause an additional 8-bit port to become unavailable when external memory was used. Since neither of these are desirable options, the designers of the 8051 chose to use pins port0-0 through port0-7both to transmit the low-byte of the address and to also transmit the data itself.

As you can see by reviewing the four step procedure above, the ALE pin is strobed when pins port0-0 through port0-7 contains the low-byte of the address. When ALE is strobed, therefore, we need to store and hold the value of the low-byte of memory so that those 8 bits are still available when the actual read or write operation takes place (when RD or WR is strobed). This is easily accomplished using a simple Octal Latch, such as the 74HCT373 (Fig. 10).

5.2 A more advanced Microcontroller board MB80C535

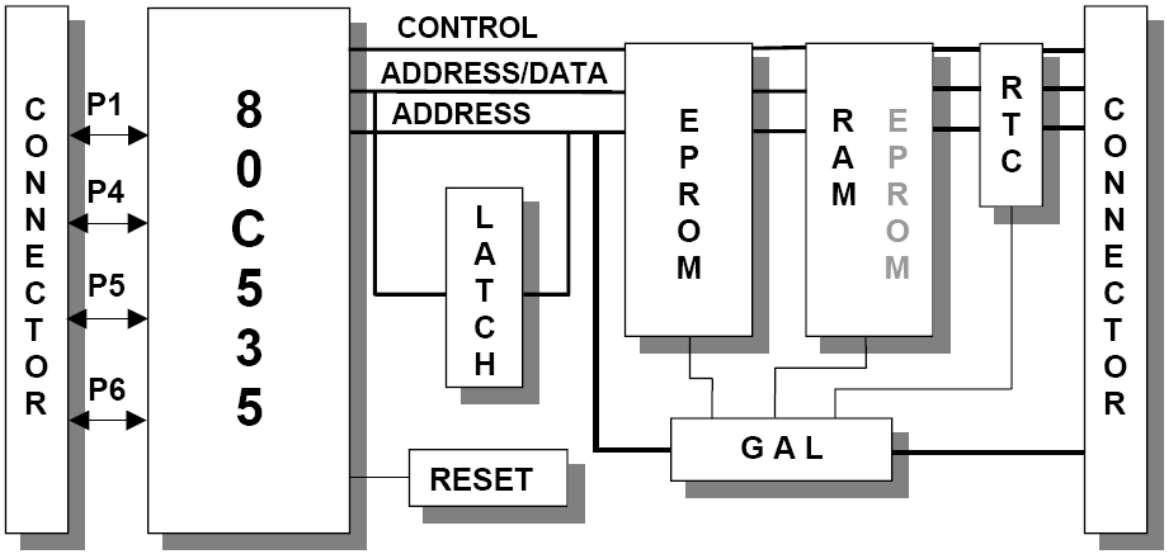


Fig. 11: Block diagram

As shown in Fig.11, the core of the system is the 8-bit micro-controller 80C535 (U1) from Siemens – Infenion. This controller is based on the well-known 8051 architecture but incorporates several enhancements that significantly increase the overall performance.

For the following descriptions refer to Fig.13.

When a **reset** (U2) is generated either by power-up or an external switch, the controller is automatically set-up when the EA input (U1 pin 51) is tied to low that three of its original six 8bit I/O ports are used to form the address-, data-, and control bus.

Port 0.0 to 0.7 - serves as the multiplexed address/data bus (AD0 - AD7)

Port 2.0 to 2.7 - serves as the high-order address bus (A8 - A15)

Port 3.0 to 3.7 - serves as control bus	P3.0 ... RxD	Receive data (RS232)
	P3.1 ... TxD	Transmit data (RS232)
	P3.2 ... /INT0	not used
	P3.3 ... /INT1	external Interrupt
	P3.4 ... T0	Counter 0 input
	P3.5 ... T1	Counter 1 input
	P3.6 ... /WR	Write data
	P3.7 ... /RD	Read data

Afterwards BASIC-535 clears the internal 80C535 memory, initializes the internal registers and pointers and tests, clears, and sizes the external memory. Afterwards it assigns the top of external RAM to the system control value - MTOP. BASIC then assigns the default crystal value, 12MHz, to the system control value - XTAL and uses this default value to calculate all time dependent functions, such as the interrupt driven internal TIMER.

The **latch** (U3) is used to de-multiplex the address/data bus. With the ALE signal the controller indicates that valid addresses are on the bus. Accordingly the signal is used to latch the lower address to maintain a valid sixteen bit wide address during the READ and WRITE operation (see Fig.12).

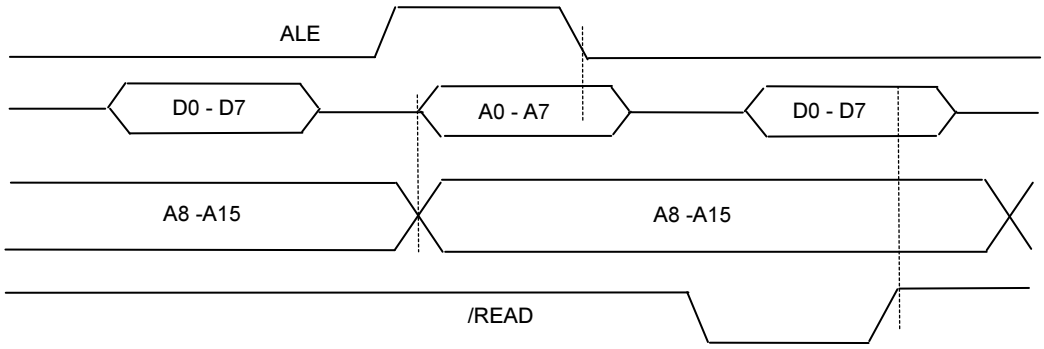


Fig. 12: Data-memory Read cycle

The **R**eal **T**ime **C**lock (U5) and the RAM are normally supplied by the +5V supply, but can be buffered when a battery ($\geq$ 3V) is connected to the BATTERY terminals.

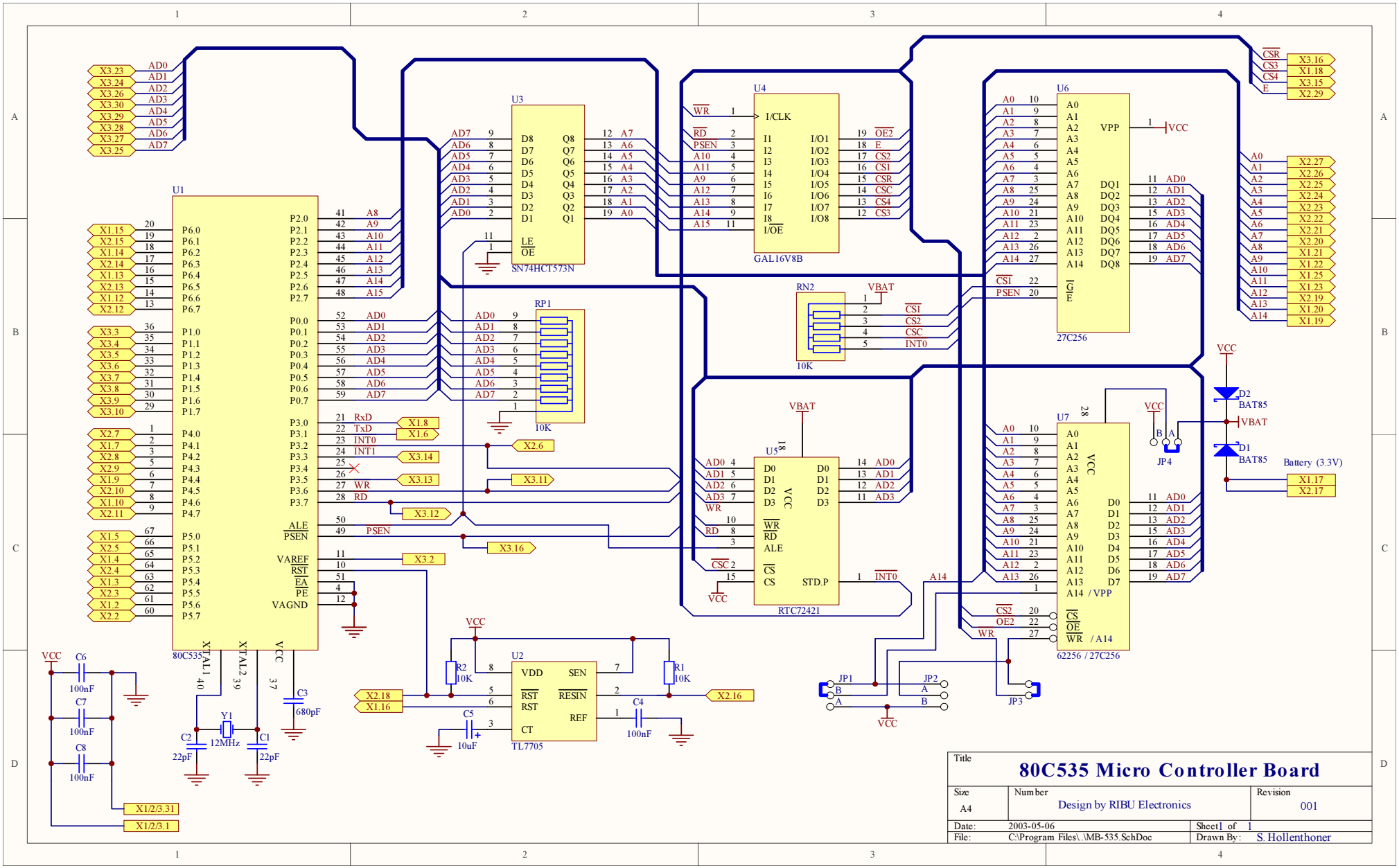


Fig. 13: Circuit diagram of a 80C535 MicroController

5.3 Introduction to Address Decoding

Although the memory space in a microcontroller system is said to be flat, it does not mean that the physical implementation of memory is homogenous.

- Different portions of memory are used for different purposes: RAM, ROM, I/O devices.
- Even if all memory was of one type, we still have to implement it using multiple ICs.
- This means that for a given valid address, one and only memory-mapped component must be accessed, simply said a unique address.

Address decoding is the process of generating chip selects (/CS) signals from the address bus for each device in the system.

To do this, the address bus lines are split into two sections (Fig. 14):

- the N most significant bits of the address bus are used to generate the /CS signals for the different devices.
- the M least significant signals are passed to the devices as addresses to address the different memory cells or internal registers.

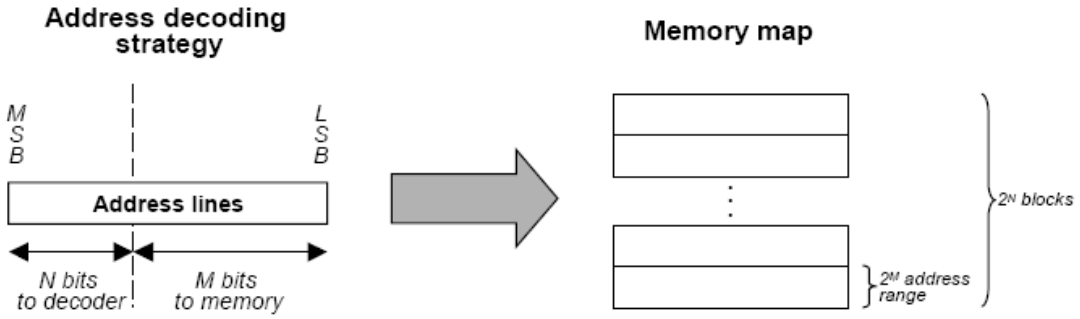


Fig. 14: Principle of address decoding

Let's assume a very simple microprocessor with 10 address lines (2^{10} = 1024Bytes = 1KByte) and in which we would like to implement all its memory space by using 128 x 8Byte memory chips. How does the solution look like (Fig. 15)?

- We will need 8 memory chips (8 x 128 = 1024)
- We will need 3 address lines to select each one of the chips (2^3 = 8)
- Each memory chip will need 7 address lines to address its internal cells (2^7 = 128)

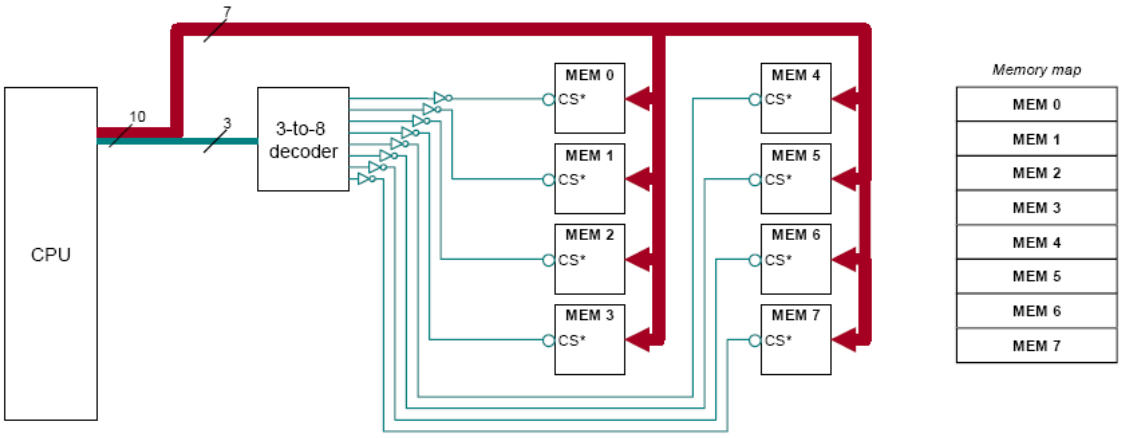


Fig. 15: Simple address decoder

The previous example specified that all addressable memory space was to be implemented but there are situations where this requirement is not necessarily affordable. If only a portion of the addressable space is going to be implemented there are two basic address decoding strategies:

Partial address decoding: since not all the address space is implemented, only a subset of the address lines are needed to point to the physical memory locations. Each physical memory location is identified by several possible addresses.

Let's assume the same microprocessor with 10 address lines (1KByte total memory space). However, this time we wish to implement only 512 Bytes of memory still using 128 x 8 Byte memory chips. Additional we would like to place the memory physically in the upper half of the memory map (Fig. 16).

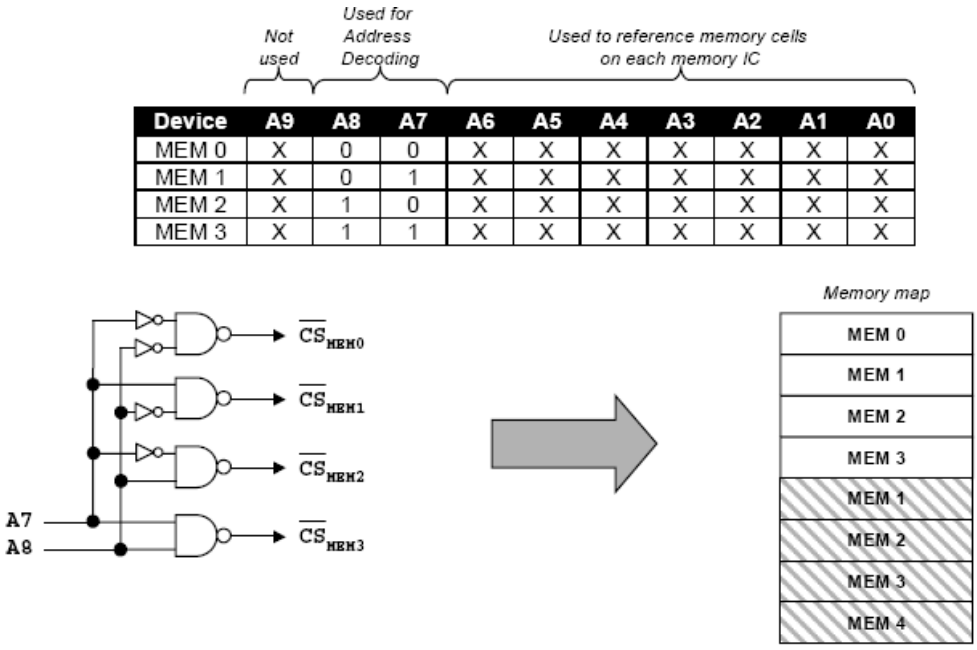


Fig. 16: Address decoder logic

Full address decoding: all the address lines are used to specify a memory location and each memory location is identified by a unique address.

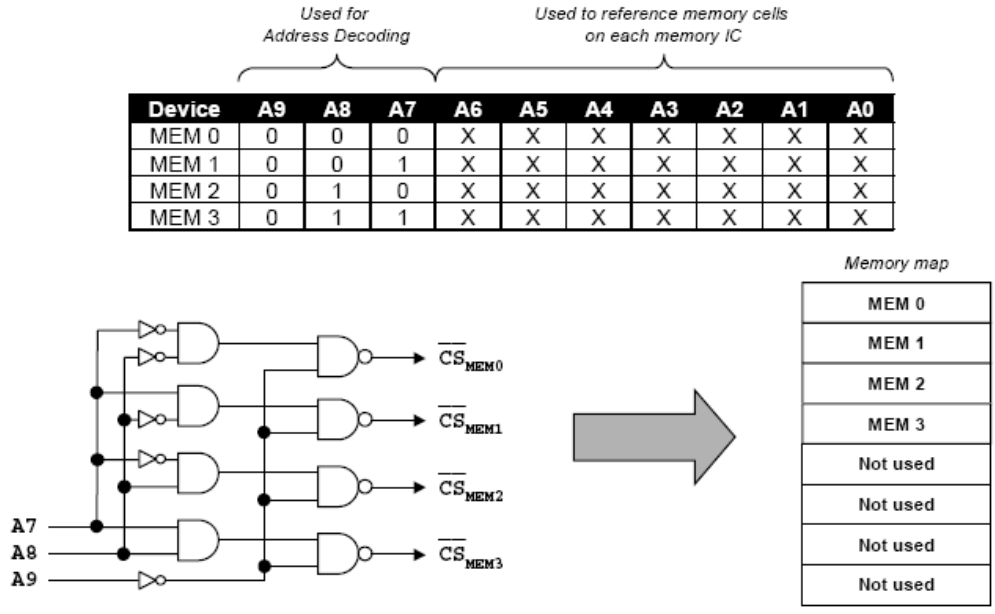


Fig. 17: Address decoder logic

5.3.1 Address map and decoding of the MB80C535

According the Boolean equations on the opposite page, the address assignment of the MB80C535 is as follows:

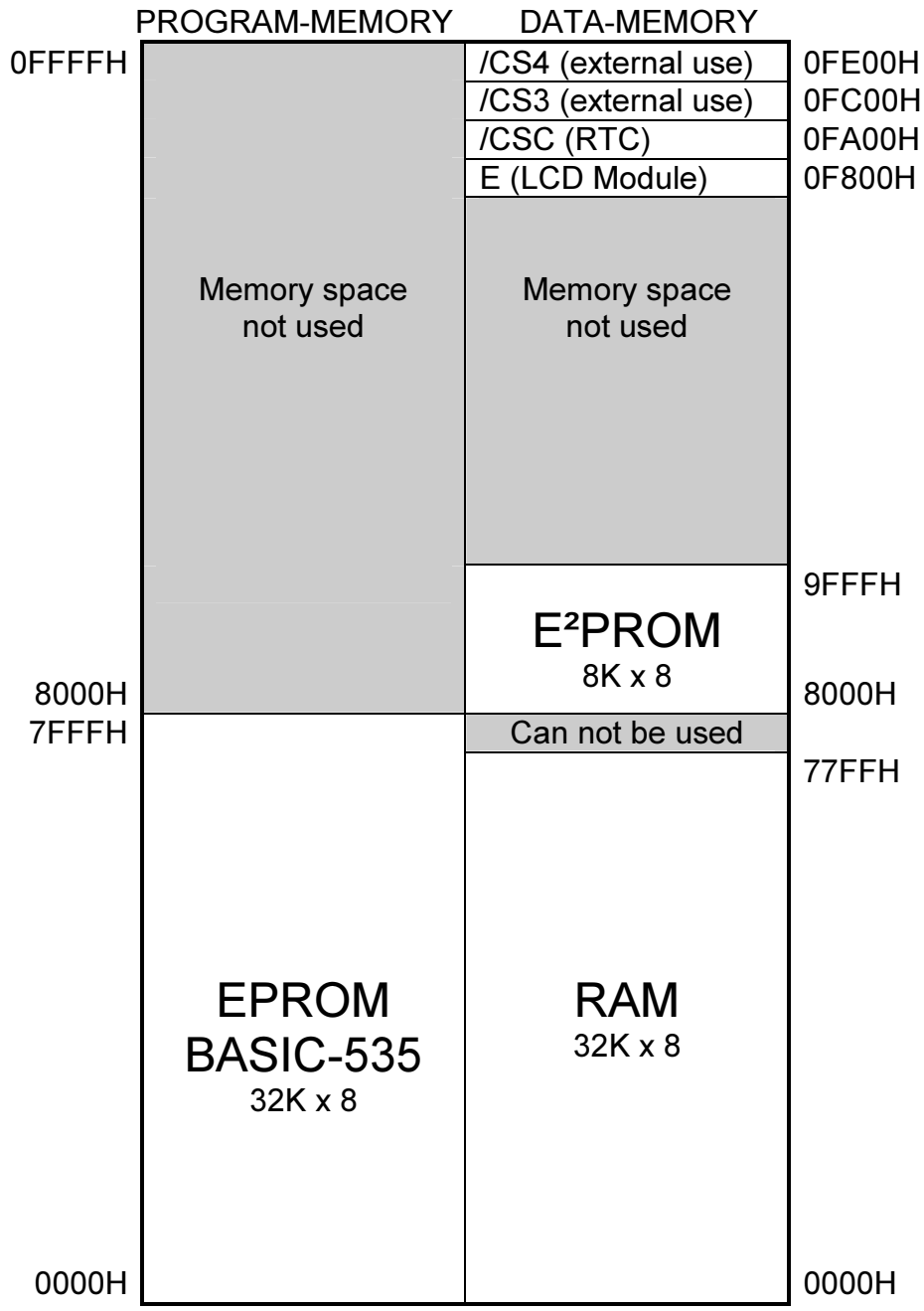


Fig. 18: Memory Assignment of the MB80C535

The apparent memory conflict between EPROM and RAM sharing the same address range (0000H - 77FFH) is avoided by using the /PSEN signal for address decoding.

/PSEN - the program store enable output is a control signal that enables the external program memory to the bus during external instruction fetch operations. It is activated every six oscillator periods except during external data memory accesses. It remains high during internal program execution.

For **address decoding** a GAL (U4) is used. Following you will find the GAL listing created with CUPL software:

```
NAME        BASIC-535 ADDRESS-DECODER;
COMPANY     RIBU;
DEVICE      GAL16V8;

/* INPUTS: Define inputs for the decoder */

            PIN1    = !WR;
            PIN2    = !RD;
            PIN3    = !PSEN;
            PIN4    = A10;
            PIN5    = A11;
            PIN6    = A9;
            PIN7    = A12;
            PIN8    = A13;
            PIN9    = A14;
            PIN11   = A15;

/* OUTPUTS: Define outputs for the decoder */

            PIN19   = !OE2;
            PIN18   = E;
            PIN17   = !CS2;
            PIN16   = !CS1;
            PIN15   = !CSR;
            PIN14   = !CSC;
            PIN13   = !CS4;
            PIN12   = !CS3;

/* LOGIC EQUATIONS */

            CS1     = !(!A15);
            CS2     = !(!A15 & !A14
                      # !A15 & A14 & !A13
                      # !A15 & A14 & A13 & !A12
                      # !A15 & A14 & A13 & A12 & !A11);
            OE2     = !(!RD);
            CSR     = !(A15 & !A14
                      # A15 & A14 & !A13
                      # A15 & A14 & A13 & !A12
                      # A15 & A14 & A13 & A12 & !A11);
            E       = !WR & A15 & A14 & A13 & A12 & A11 & !A10 & !A9
                      # !RD & A15 & A14 & A13 & A12 & A11 & !A10 & !A9;
            CSC     = !(A15 & A14 & A13 & A12 & A11 & !A10 & A9);
            CS3     = !(A15 & A14 & A13 & A12 & A11 & A10 & !A9);
            CS4     = !(A15 & A14 & A13 & A12 & A11 & A10 & A9);
```

6. The AVR MicroControllers

6.1 *Introduction*

This type of AVR Advanced Virtual RISC controller belongs to the cheapest but most versatile micro-controllers.

The AVR's are a family of ***RISC*** (**R**educed **I**nstruction **S**et **C**omputer) microcontrollers from Atmel. Their internal architecture was conceived by two students: Alf-Egil Bogen and Vergard Wollan, at the Norwegian Institute of Technology (NTH) and further developed at Atmel Norway, a subsidiary founded by the two architects.

The AVR is a ***Harvard*** architecture machine where programs and data are stored separately. Program instructions are stored in semi-permanent Flash memory which loads and manipulates data in the volatile SRAM. The AVR's have thirty-two single-byte registers.

The registers input/output ports and static RAM make up the data address space. The working registers are mapped in as the first thirty-two memory spaces followed by the I/O ports, which means that the actual usable RAM starts at memory location 100H. Atmel's AVR's have a two-stage pipeline design. The next machine instruction is fetched as the current one is executing. Most instructions take just one or two clock cycles, making AVR's relatively fast among the eight-bit microcontrollers. The AVR's were designed for the efficient execution of compiled code.

AVR's have a large following due to the free and inexpensive development tools available, including reasonably priced development boards and free development software. They are marketed under various names that share the same basic core but with different peripheral and memory combinations. Compatibility amongst chips is fairly good.

6.2 RISC versus CISC

In the early days of the computer industry, compiler technology did not exist. Programming was done in either machine code or assembly language. To make programming easier, computer architects created more and more complex instructions which were direct representations of high level functions of high level programming languages. The attitude at that time was that hardware design was easier than compiler design, so the complexity went into the hardware.

Another force that encouraged complex instructions was the lack of large memories. Since memories were small, it was advantageous for the density of information held in computer programs to be very high. When every byte of memory was precious, for example an entire system had only a few kilobytes of storage, it moved the industry to such features as highly encoded instructions. Instructions which could be variable sized, instructions which did multiple operations and instructions which did both data movement and data calculation. At that time, such instruction packing issues were of higher priority than the ease of decoding such instructions.

Memory was not only small, but rather slow since they were implemented using magnetic technology at the time. That was another reason to keep the density of information very high. By having dense information packing, one could decrease the frequency when one had to access this slow resource.

CPUs had few registers for two reasons:

- Bits in internal CPU registers are always more expensive than bits in external memory. The available level of silicon integration of the day meant large register sets would have been burdensome to the chip area or board areas available.
- Having a large number of registers would have required a large number of instruction bits (using precious RAM) to be used as register specifiers.

For the above reasons, CPU designers tried to make instructions that would do as much work as possible. This led to one instruction that would do all of the work in a single instruction: load up the two numbers to be added, add them, and then store the result back directly to memory. Another instruction would read the two numbers from memory, but store the result in a register. Another one would read a number from memory and another from a register and store it to memory again. And so on. This processor design philosophy eventually became known as **Complex Instruction Set Computer** (CISC).

A complex instruction set computer (CISC) is a microprocessor instruction set architecture (ISA) in which each instruction can execute several low-level operations, such as a load from memory, an arithmetic operation, and a memory store, all in a single instruction. The term was coined in contrast to reduced instruction set computer (RISC).

Before the first RISC processors were designed, many computer architects tried to bridge the "semantic gap" - to design instruction sets to support high-level programming languages by providing "high-level" instructions such as procedure call and return, loop instructions such as "decrement and branch if non-zero" and complex addressing modes to allow data structure and array accesses to be combined into single instructions. Additionally, the compact nature of a CISC ISA results in smaller program sizes and fewer calls to main memory, which at the time (the 1960s) resulted in a tremendous savings on the cost of a computer.

While they achieved their aim of allowing high-level language constructs to be expressed in fewer instructions, it was observed that they did not always result in improved performance. For example, on one processor it was discovered that it was possible to improve performance by *not* using the procedure call instruction but using a sequence of simpler instructions instead. Furthermore, the more complex the instruction set, the greater the overhead of decoding any given instruction, both in execution time and silicon area. This is particularly true for processors which used microcode to decode the (macro) instructions. In other words, adding a large and complex instruction set to the processor even slowed down the execution of simple instructions. Implementing all these complex instructions also required a great deal of work on the part of the chip designer, and many transistors; this left less room on the processor to optimize performance in other ways.

The term, like its antonym RISC, has become less meaningful with the continued evolution of both CISC and RISC designs and implementations. The first pipelined "CISC" CPUs, such as 486s from Intel, AMD, Cyrix, and IBM, certainly supported every instruction that their predecessors did, but achieved *high efficiency* only on a fairly simple x86 subset (resembling a non load/store "RISC" instruction set). Modern x86 processors also decode more complex instructions into series of smaller internal "micro-operations" which can thereby be executed in a pipelined (parallel) fashion; thus achieves high performance on a much larger subset of instructions.

Characteristics of the RISC architecture:

- As implied by the name, there exists a reduced subset of instructions
- Accumulator is replaced by a number of registers with equal rights
- Memories are organized according the Harvard model
- Standardized interface to the memories by using Load / Store instructions exclusively
- Executing all instructions within one machine cycle
- Optimized hardware and instruction set for high-level programming languages

First only used at high-performance computers, Industry soon discovered the RISC architecture for single chip controllers and implemented it like in the AVR family.

6.3 Harvard versus "von Neumann" architecture

The term **Harvard architecture** originally referred to computer architectures that used physically separate storage and signal pathways for their instructions and data (in contrast to the *von Neumann architecture*). The term originated from the Harvard Mark I relay-based computer, which stored instructions on punched tape (24-bits wide) and data in relay latches (23-digits wide). These early machines had very limited data storage, entirely contained within the data processing unit, and provided no access to the instruction storage as data (making loading, modifying, etc. of programs entirely an offline process).

In a computer with a "**von Neumann**" architecture, the CPU can be either reading an instruction or reading/writing data from/to the memory. Both cannot occur at the same time since the instructions and data use the same signal pathways and memory. In a computer with Harvard architecture, the CPU can read both an instruction and data from memory at the same time. A computer with Harvard architecture can be faster because it is able to fetch the next instruction at the same time it completes the current instruction. Speed is gained at the expense of more complex electrical circuitry.

In recent years the speed of the CPU has grown many times in comparison to the access speed of the main memory. Care needs to be taken to reduce the number of times main memory is accessed in order to maintain performance. If, for instance, every instruction run in the CPU requires an access to memory, the computer gains nothing for increased CPU speed - a problem referred to as being *memory bound*.

Memory can be made much faster, but only at high cost. The solution then is to provide a small amount of very fast memory known as a cache. As long as the information that the CPU needs is in the cache, the performance is much higher than when the cache has to turn around and get the data from the main memory. Tuning the cache is an important aspect of computer design.

Modern high performance CPU chip designs incorporate aspects of both Harvard and von Neumann architecture. On-chip cache memory is divided into an instruction cache and a data cache. Harvard architecture is used as the CPU accesses the cache. In the case of a cache miss, however, the data is retrieved from the main memory, which is not divided into separate instruction and data sections. Thus a "von Neumann" architecture is used for off chip memory access.

Harvard architectures are also frequently used in specialized DSPs (**D**igital **S**ignal **P**rocessors), commonly used in audio or video processing products.

Additionally, most general purpose small microcontrollers used in several electronics applications, such as the PIC microcontrollers made by Microchip Technology Inc and AVR Microcontrollers made by Atmel Corporation, are based on the Harvard architecture. These processors are characterized by having small amounts of program and data memory, and take advantage of the Harvard architecture and reduced instruction set (RISC) to ensure that most instructions can be executed within only one machine cycle. The separate storage means the program and data memories can be in different bit depths. For example, the PIC microcontrollers have an 8-bit data word but (depending on specific range of PICs) a 12-bit, 14-bit, or 16-bit program word. This allows a single instruction to contain a full-size data constant.

The AVR controllers have a separated data and memory using a one-stage pipeline, which means that during executing an instruction the next one is going to be loaded. This ensures that each instruction can be executed within one clock cycle (Fig. 1).

The first instruction is extended by an additional clock as no instruction is pre-loaded into the pipe-line.

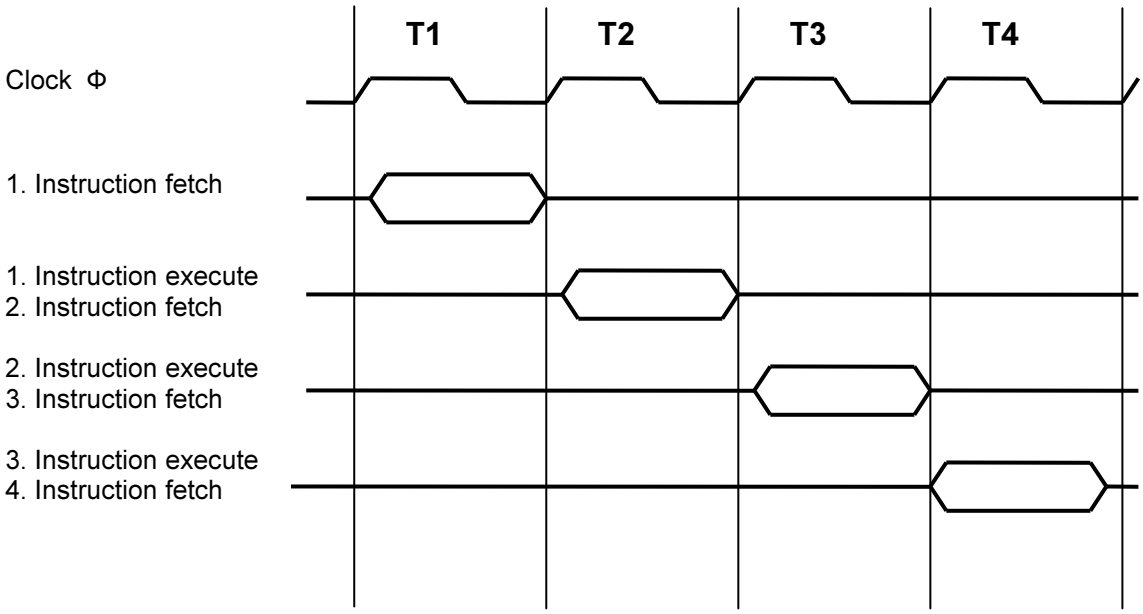


Fig. 19: Pipe-lining of AVR controllers

6.4 Basics about the AVR family

In principal all microcontrollers are all built on the same concept (Fig. 20). A sequencing unit consisting of

- a program counter and a decoding unit that takes care of reading and decoding the instructions out of the program memory.
- a computing unit executing the arithmetic and logic operations.
- an input/output interface enabling data exchange through the peripherals.
- and last but not least memory to store the program and data.

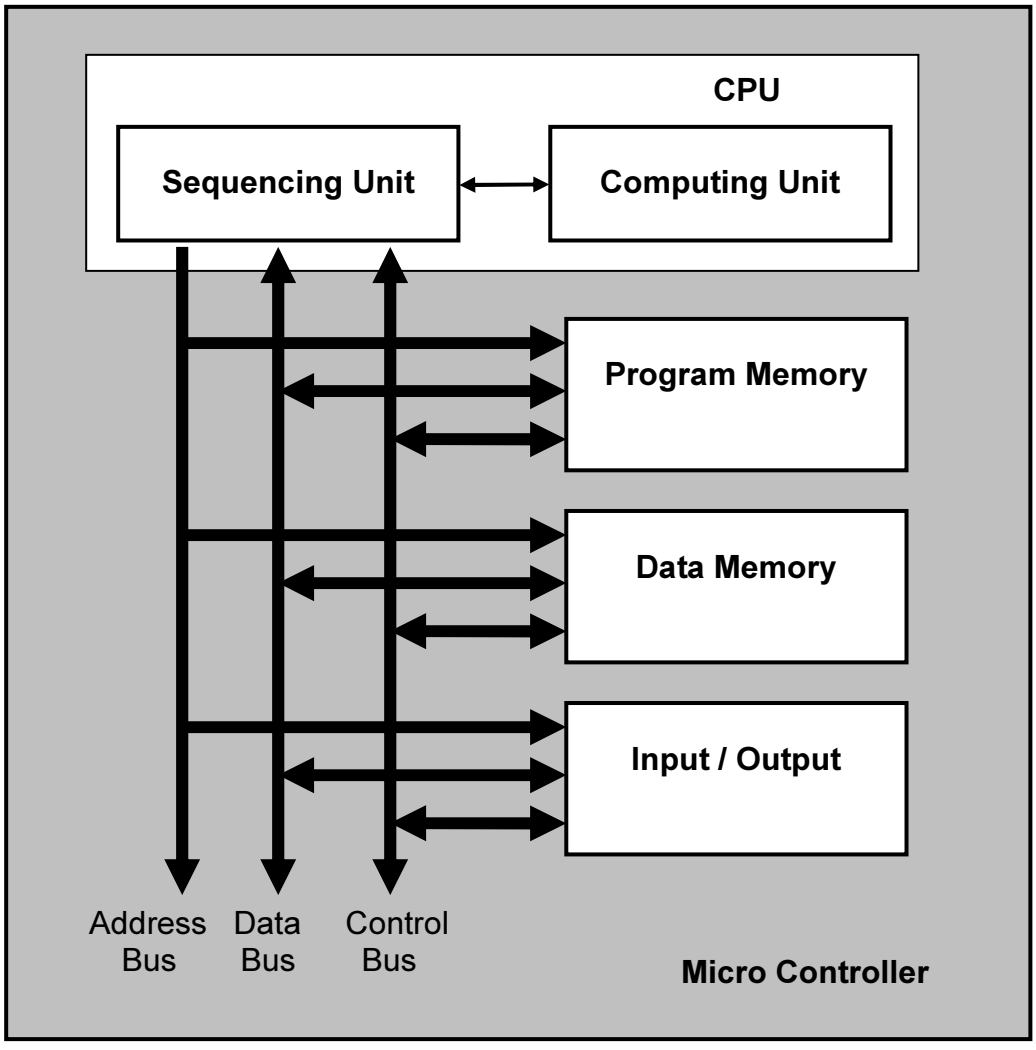


Fig. 20: Basic structure of a micro controller

The computing unit normally consists of the ALU (Arithmetic Logic Unit), the accumulator and a number of registers. In programs generally half of the instructions consist of – for the functioning of the program – useless moving instructions to transfer data from the memory and the registers to the accumulator and back.

In the AVR this “bottle-neck” of having only one accumulator has been solved by introducing 32 so called working registers, all of them directly connected with the ALU and through which the arithmetic and logical operations can be performed in a single clock cycle.

The ALU (Arithmetic Control Unit)

The ALU instructions can be divided into three main categories

- arithmetic
- logical
- Bit operations

For each of the above mentioned categories very powerful instructions exist and additional a 2 cycle multiplying unit is implemented in hardware.

The Instructions

Although a RISC processor it has a set of 130 instructions, most of them executed in one cycle. AVR controllers basically use 2 bytes (16 bits) for most of their instructions, including “JUMPs” and “CALLs”. Two exceptions exist, the LDS and STS instruction used for direct addressing the data memory occupying 4 bytes or a double word).

Instructions for jump, immediate operations and memory read / write need more than one system cycle.

Execution time

Fig. 21 shows a comparison of system cycles needed to execute two consecutive instructions by different controllers

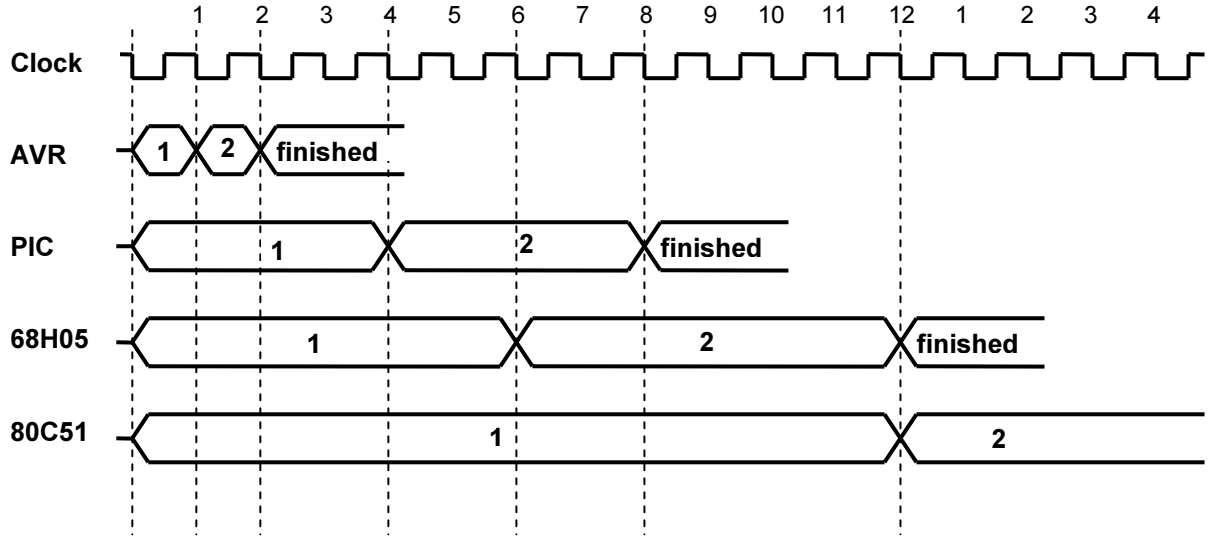


Fig. 21: Comparison of execution times of different processors

Memory architecture

In the classical “von Neumann” concept instruction and data are stored in the same memory. In contrast the “Harvard” architecture as used in AVR controllers. Here the program and data memory are separated, which means that instructions are handled different compared to data.

Especially in AVR controllers individual memory segments are physically built different.

a) The program memory is built in electric erasable and programmable Flash technology. In all AVR controllers it is 16 bit (2 byte) wide and always on-chip. Extension of the program memory by external EPROM or Flash is impossible.

b) The internal data memory is volatile and built as a static RAM (SRAM) and therefore does not need refresh cycle. This allows AVR controllers to run on very low clock speeds even down to 0 Hz. Additional there exists on different AVR controller the possibility to expand the data memory external, with the drawback of loosing valuable IO pins.

c) For data that has to be kept, a nonvolatile memory (EEPROM) is available. Data can be transferred during normal program routines and no special programming device is necessary.

IO ports

The IO area consists of 64 bytes and is as well located in the SRAM area, from address 20H to 5FH. In AVR controllers not only the real IO ports (Port A to Port D) are considered as IO functions, but also the status and control registers such as Timer, UART, Watchdog, EEPROM read / write, SPI interface, etc.

Interrupts and subroutines

Subroutine and interrupt techniques are implemented same as in other controllers. Parts of a program, that are used multiple or for structural reasons need to be separated, can be addressed by an *rcall* instruction.

Rcal is a kind of jump instruction, which automatically stores the next consecutive address of the program as return address in the stack memory. A subroutine needs to be terminated with a *ret* instruction, which will restore the address previously stored in the stack memory. The stack memory is located in the SRAM.

In Fig. 22 the stack is going to be assigned or initialized at Label Start. At the Label Call1 the first call of the subroutine UP1 is executed. As return address 78AH is going placed onto the stack. Subroutine UP1 is a dummy routine doing nothing, just returning to the main program. To allow returning from a subroutien, the previously stored return address is restored from the stack and the controller continues exactly with the instruction following the call instruction.

Subprograms might be nested, which means a call from inside a subroutine can be performed.

Interrupts are a special way to redirect the main program into a subroutine. Interrupts can be understood as a “hardware triggered call” generated by an external source like timer overflow or character received at a UART. Most of the AVR controllers provide 12 interrupt vector addresses.

Addr.	OP-Code	Label	Instruction		
0X780	0000	UP1:	nop		;No operation
0X781	9508		ret		;Return
0X782					
0X783					
0X784					
0X785	E50F	Start:	ldi	r16,0x5F	;Load immediate R16
0X786	BF0D		out	spl,r16	;Load Stackpointer L-Addr
0X787	E002		ldi	r16,0x02	
0X788	BF0E		out	sph,r16	;Load Stackpointer H-Addr
0X789	DFF6	Call1:	rcall	UP1	;Call subroutine UP1
0X78A	C001		rjmp	Call2	;Jump Call2
0X78B					
0X78C	DFF3	Call2:	rcall	UP1	;Call subroutine UP1
0X78D	C002		rjmp	Cont	;Jump Cont
0X78E					
0X78F	0000	Cont:	nop		;No operation
0X790					
0X791					

Addr.	Stack	
0X25F		← Stackpointer
0X25E		before 1. Call
0X25D		

Addr.	Stack	
0X25F	8A	
0X25E	07	
0X25D		← Stackpointer
		after 1. Call

Addr.	Stack	
0X25F	8D	
0X25E	07	
0X25D		← Stackpointer
		after 2.Call

Fig. 22: Stack handling during Call instructions

Peripheral Functions

A great number of peripheral functions are already implemented into the hardware of the AVR controllers like for example for the ATMega 8

- 23 programmable IO lines
- One programmable serial USART
- Byte oriented two-wire serial interface
- Master / slave SPI serial interface
- Two 8-bit and one 16-bit Timer/Counter with separate prescaler
- Three PWM channels
- 6 channel ADC (10-bit)
- Programmable watchdog timer with separate on-chip oscillator
- On-chip analog comparator
- In total 19 reset and interrupt vectors

6.5 Block diagram of the ATMega 8

The ATMega8 as shown in Fig. 23 is a low-power CMOS 8-bit microcontroller based on the AVR RISC architecture. By executing powerful instructions in a single clock cycle it achieves throughputs approaching 1 MIPS / MHz, allowing the system designer to optimize power consumption versus processing speed.

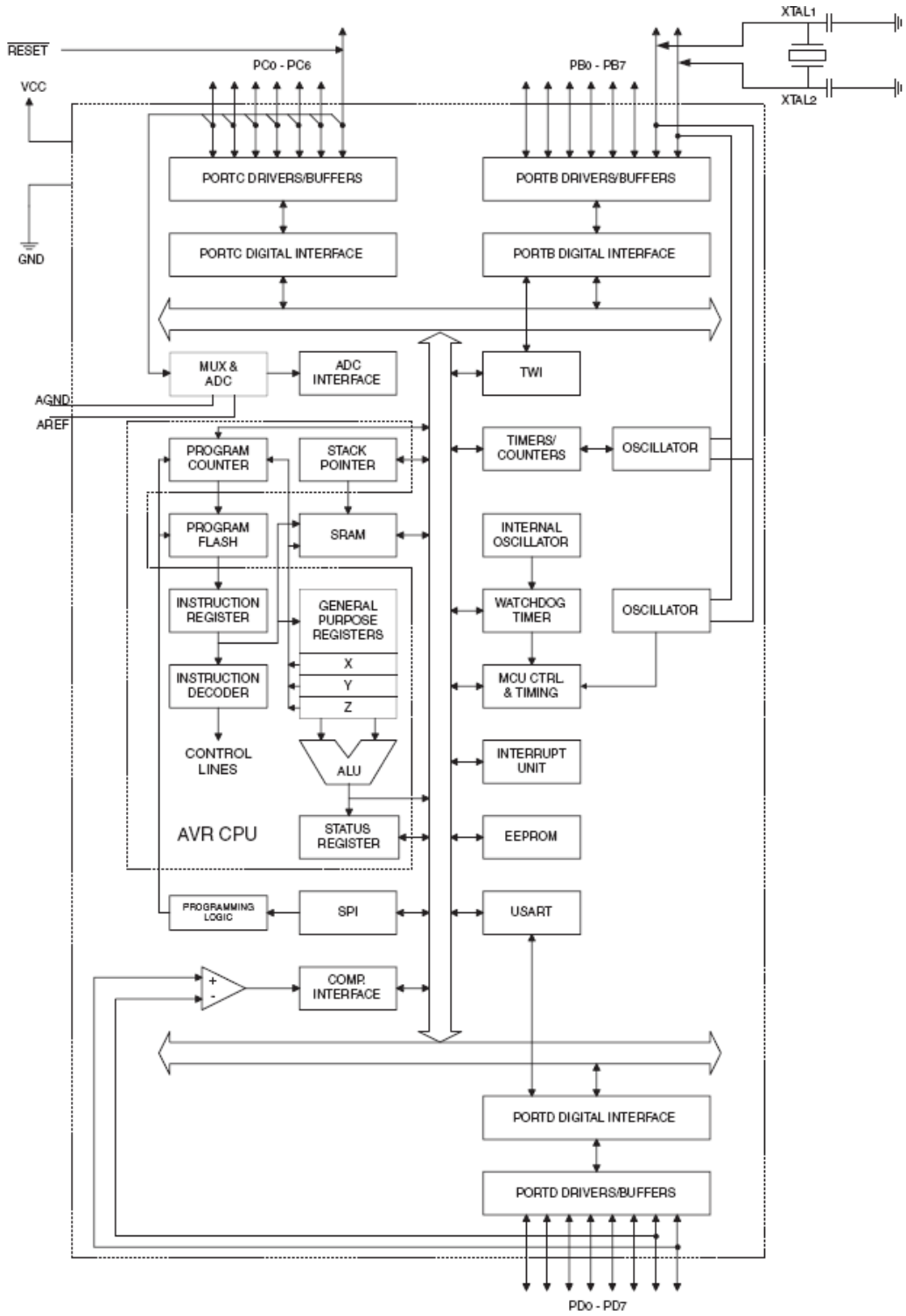


Fig. 23: Block diagram of the ATMega 8

CPU

The processor core, depicted here towards the centre of the image on the left side of the data bus, includes elements to read the program memory, decoding and executing the instructions within. The CPU can also fetch and store data to and from the EEPROM, SRAM and the 32 *registers*. The registers act as extremely efficient storage for 8 bit values (1 byte) and the ALU (**A**rithmetic **L**ogic **U**nit) can operate on each of the 32 registers directly.

The AVR processor features a real life stack and its instruction set is designed and optimized for use with high level languages - it is easy to program these chips using C or Basic. Note that certain implementations don't provide any SRAM, so the stack is actually hardware based (limiting the stack depth to three).

It is interesting to note that most instructions only take a single clock cycle to execute and there is no internal clock division. Also, the CPU will fetch and decode the next instruction as it is executing the current instruction (i.e. in parallel). This entire means that AVR's will reach performances of nearly 1 MIPS (Million Instructions per Second) per MHz, they are fast and efficient. Another advantage of reaching comparable performance at reduced clock rates (e.g. an AVR at 4 MHz will execute about as many instructions as a PIC microcontroller running at 16MHz) is reduced power consumption and lower electro-magnetic noise from high frequency signals.

Memory

The program memory is a contiguous block of flash memory. It may be programmed and reprogrammed numerous times easily. In fact, using the **I**n-**S**ystem **P**rogramming, you can design your system to allow chip firmware upgrades in-place. Program memory is 16 bits wide and can usually be erased/re-written at least 1000 times.

All AVR's have some EEPROM and most have SRAM available. Both are 8 bits wide. The EEPROM is programmable during execution (to retain data even without power) or directly during programming (which is very useful for things like production line calibration). Using EEPROM is slower than storing data in SRAM but, unlike the volatile RAM, once written the data will remain available indefinitely (even when the power source has been removed). The EEPROM withstand 100,000 erase/write cycles.

I/O and Peripherals

Peripherals are where the members of the AVR family distinguish themselves. They all have the same core and support a basically uniform instruction set, which means you can easily develop for all of them. However each has its own particular blend of peripherals and extras.

All AVRs, from the tiny 8 pin DIPs to the 44 pin Megas, have at least one data port. Data ports allow for input or output of logic level (binary, *HIGH* or *LOW*) data. The AVR ports are bi-directional, allowing you to set them for input or output on a pin-by-pin basis.

The ATMega8 depicted here has three 8-bit ports available. Often, the external pins which make up a port will serve dual (or triple!) purposes, and how the pins are used will depend on how you've configured the controller. For instance, you can see from the block diagram that all pins for port C (upper left) are also used as separate channels to the ADC (analog to digital converter) while certain pins of port D (lower right) are used for communication through the UART.

The Analog to Digital Converter is normally multiplexed; this means that you can monitor multiple analog values, through different pins, one at a time. The UARTs allow

asynchronous serial communication, directly with other chips or with computers systems using an RS-232 interface.

Internal peripherals include timers and counters (8 and 16 bit, with optional overflow interrupts), a watchdog timer (to automatically reset the chip after a preset delay of 16ms to 2s), RC oscillators and more.

Programming

The Flash Program memory can be reprogrammed In-System through an SPI serial interface, by a conventional non-volatile 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 ATMega8 is a powerful micro-controller that provides a highly-flexible and cost-effective solution to man embedded control applications.

Note: for more details please refer to the datasheet on Page 155

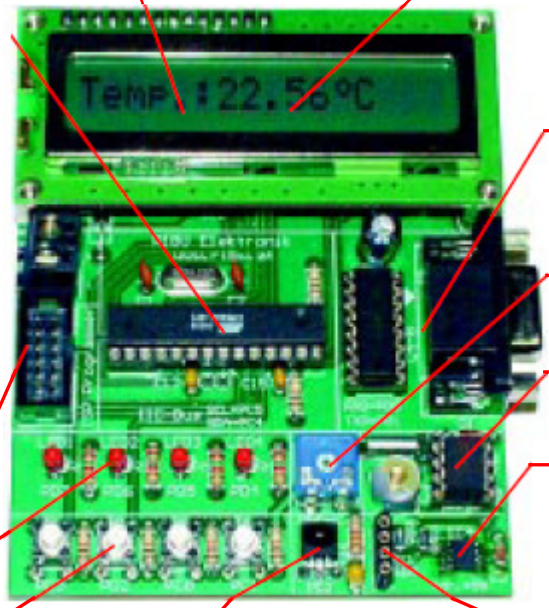
7. *Starter Kit ATMega8*

16 character LC dot matrix display

4 digit 7 segment LED display with I²C bus. The LED and LCD module are easily exchangeable

AVR controller ATMega8

- 8KByte Flash memory
- 1KByte RAM and 512Byte EEPROM
- 23 digital programmable I/Os
- 6 channel 10 bit ADC with internal reference
- Internal RC oscillator with external 16MHz crystal and pre-scaler for 1, 2, 4, and 8MHz
- Two 8 bit and one 16 bit Timer/counter
- Serial interface withprogrammable baudrate generator



RS232 interface for remote control or to exchange data with a PC

Potentiometer to adjust analog values

I²C bus Real Time Clock with calendar PCF 8583

12bit temperature sensor MP 9801 from Microchip with I²C bus for temperature measurement and control

ISP connenctor

4 Status LEDs

4 Control push buttons

IR receiver with built-in pre-amplifier and 36kHz demodulator to receive commands from an external IR remote control

Extension connector to connect external **I²C** devices

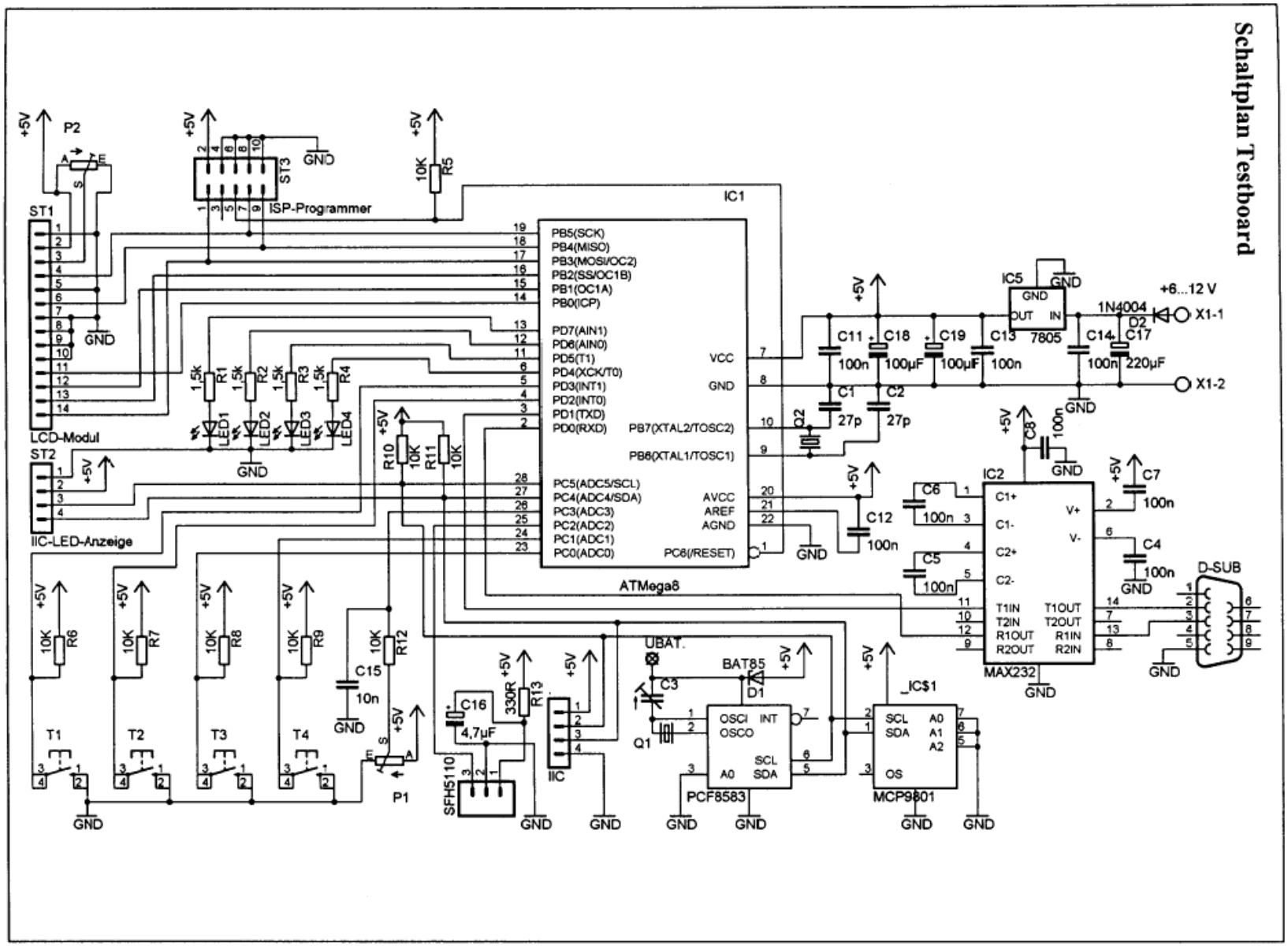


Fig. 24: Schematic of the ATMega 8 Starter Kit

7.1 Testboard

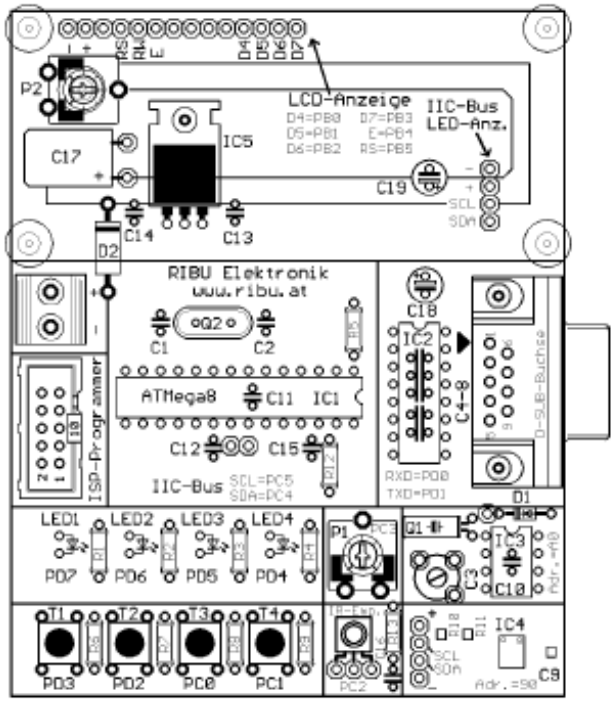
Components list

R1-R4	=	1,5 K
R5-R9, R12	=	10 K
R10-R11	=	10 K / SMD
R13	=	330 Ohm
P1,P2	=	10 K Poti
C1, C2	=	27 pF / RM2,5
C3	=	Variable capacitor
C4-C8, C10-C14	=	100 nF / RM2,5
C9	=	100 nF / SMD
C15	=	10 nF / RM2,5
C16	=	4,7 µF / RM2,5
C17	=	220 µF / RM5
C18-C19	=	100 µF / RM2,5
D1	=	BAT 85
D2	=	1N4007
LED1-LED4	=	3 mm red / 2 mA
IC1	=	ATMega 8
IC2	=	MAX 232
IC3	=	PCF 8583
IC4	=	MCP 9801
IC5	=	7805 T
Q1	=	32,768 KHz
Q2	=	16 MHz
IR Receiver	=	SFH 5110-36
T1 - T4	=	Push buttons

Miscellenaous

1 pcs.	9 pin Dsub female
1pcs.	10 pin square post
1pcs.	LCD module 1 line x 16 characters
1pcs.	14 pin single row connector male (LCD)
1pcs.	14 pin single row connector female (LCD)
1pcs.	4 pin single row connector female (LED)
1pcs.	4 pin single row connector female (IIC Bus)
1pcs.	8 pin IC socket
1pcs.	16 pin IC socket
1pcs.	28 pin IC socket
1pcs.	2 pole screw terminal
4pcs.	Distante holder 10mm (LED/LCD)
8pcs.	Screws M3x6 (LED/LCD)
4pcs.	Selfadhesive rubber pads
1pcs.	PCB 100 x 80mm

Component placement



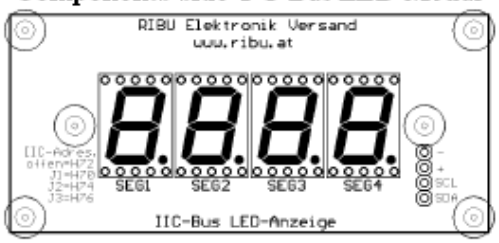
7.2 Display boards

Components list I²C-Bus LED-Modul

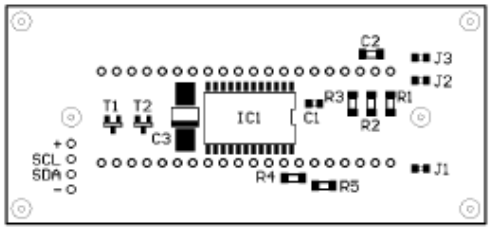
R1	=	10 K / SMD
R2	=	22 K / SMD
R3	=	18 K / SMD
R4-R5	=	330 R /SMD
C1	=	2,2 nF / SMD
C2	=	100 nF / SMD
C3	=	10 µF / SMD
T1, T2	=	BC 846
IC1	=	SAA 1064 / SMD

1 pcs. 4 pin header (insert from the solderside)
4 pcs. 7 segment LED display
1 pcs. PCB 80 x 36mm

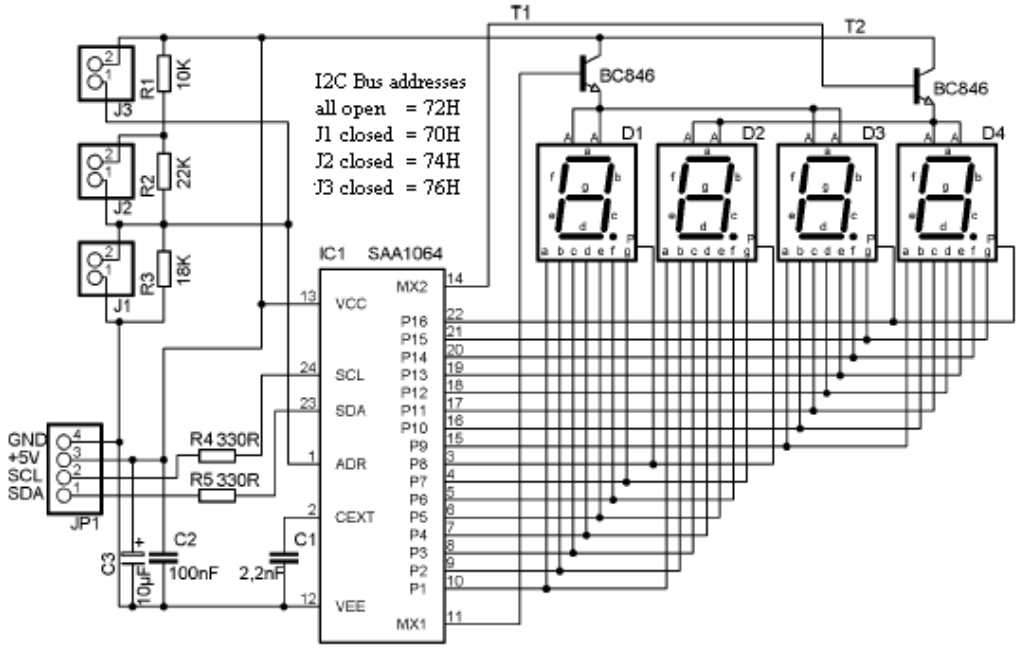
Components side I²C-Bus LED-Modul



Solder side I²C-Bus LED-Modul



Circuit diagram I²C-Bus LED-Modul



8. Troubleshooting in microprocessor based systems

Trouble shooting starts always with a good visual inspection of the whole system. This means inspections of all the printed circuit board, all cables and the power supply. There often the external devices in a microprocessor system causing the trouble, not the microprocessor board itself. External devices means: Power supply, cables to other PCBs and the sensors and displays.

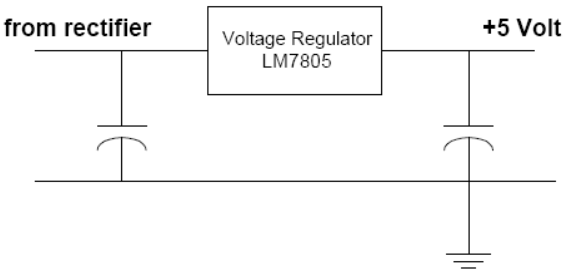
The most common problems in microprocessor based systems are:

✓ Power available?

The first topic to check is the power. In many cases the power supply causes the main problems inside of a microprocessor system. This can be the power supply itself, mostly build up from a normal transformer, a rectifier and some load capacitor.

In many cases I found the rectifier was defect. This can be checked if there is a voltage from the transformer is available (AC) and also a DC voltage behind the rectifier over the load capacitors.

Behind the capacitors the most power supplies contains a voltage regulator like the LM7805. (a 5 Volt, 1 Ampere regulator).

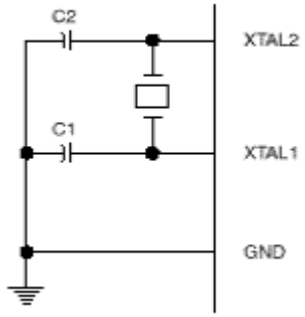


Check if there is a fuse which may be defect. Next check the voltage regulator itself. Check if there is voltage before and behind the regulator. The voltage before must be at least two volts higher as the desired output voltage.

✓ Is the clock available?

Most of the microprocessors contains a oscillation circuit. By connecting an oscillator to the external pins (XTAL1 and XTAL2) the required clock is generated.

By measuring at the two pins of the oscillator with a oscilloscope, one can decide whether the oscillator is running or not. At the pins of the oscillator should be a signal like a sinus, with the given frequency. Sometimes the oscillator itself is defect; sometimes one of the capacitors has a short to ground.

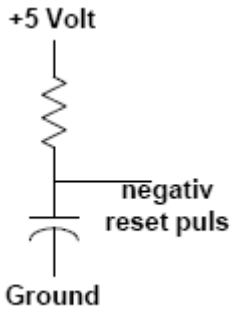


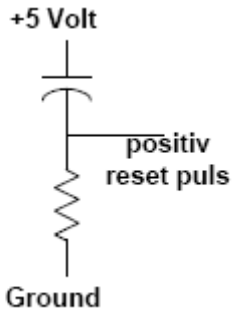
Many microprocessor also have an Clock_Out pin. By measuring at these pin with a oscilloscope, one can verify the internal clock generation. At the Clock_Out pin should be a TTL level clock signal.

✓ Is the Reset signal OK?

Every microprocessor needs a reset signal at power up time, while the clock is already running. In many applications the reset signal is generated by a simple Resistor/Capacitor circuit.

Sometimes the capacitor has a short.





✓ **Address/data line shorts (cuts)**

The address/data bus is the heart of every microprocessor system. The data bus with depends on the microprocessor (8, 16 or 32 bits), the address bus is at least 16 bit wide. This results in a great amount of tracks which must be routed in parallel over the whole printed circuit board. These lines are often very close together so that shorts between address or data lines are very often. On older PCBs without surface resist, shorts can also be generated by dust particles.

Check with a Ohmmeter if there is a short between the address/data lines. A second check could be to measure each line separate from the beginning at the microprocessor to the end (at the I/O devices).

✓ **Bus Buffer defect**

Sometimes RAM and IO is isolated from the processor by buffers. The buffers drive the bus lines from the processor into the rest of the PCB. These buffers must be bidirectional (for the data bus) and must be switch in the direction with the READ or WRITE signal. If something happens at the data bus sometimes the bus buffers are destroyed. Commonly used buffers are 74LS240, 74LS244, 74LS245, 74LS645 or derivates in ACT, HCT, ALS etc.

Measure with a oscilloscope at the input and output of the buffers, both sides must be have active signals.

✓ **Address Latch defect**

Many processors have a multiplexed data/address bus. This means that at the beginning of an fetch cycle first the address is given to that bus lines. This address must be latched into a register, because in the later period of the same fetch cycle these lines are used as data lines. Commonly used latches are 74LS273, 74LS373, 74LS374, 74LS573, 74LS574 or derivates in ACT, HCT, ALS etc.

The latch has a clock input, which is driven by the ALE signal (Address latch enable). Check this signal with an oscilloscope, it must be periodically.

Measure with an oscilloscope both sides of the latch. At the output the lines must be stable.

✓ **Pin not inserted**

Components placed in a socket like RAM's, EPROM's, PAL's etc. may not be properly seated. Sometimes a pin bends under the device. (Sometimes it happened that even the bended pin has some contact to the socket and then after a period of some years this contact is broken.) Visual inspection of all components inserted into a socket and replacing of these components are a good work here.

✓ **No chip select**

One or more signals may be needed to turn each device on. The chip select complexity can range from a simple undecoded address line to nightmarish spaghetti of PAL's and logic. Regardless, every RAM and I/O device must receive a proper chip select to function. Measure at each I/O device the chip select signal.

Check out what kind of address decoding is used (TTL chips or PAL's). If PAL's are used it could be that the PAL lost his internal function.

✓ **Multiple addressing**

This is a variant of the chip select problem. If more than one memory or IO device is used, several can sometimes be turned on at once. Each chip select signal must be unique at one time.

✓ **Refresh**

Dynamic RAM's require periodic accesses to keep the memory alive. This refresh signal is generated by external logic, or by the CPU itself. Sometimes the refresh circuitry fails. The entire memory array must be refreshed every few milliseconds to stay within the chips' specifications.

✓ **Bad device (EPROM)**

The semiconductor vendors do a wonderful job of delivering functioning chips. Rarely, though, a bad one may slip through (or, through miss handling, one may "toggle to the bad state"). This could be a problem with windowed EPROM's. EPROM's are erased by ultra-violet light. But the normal light has also a small part of these UV light. So an EPROM which window is not covered with a black tape could loose the information after a period of some years.

✓ **Interrupt Controllers**

Some of the hardest problems to diagnose are erratic interrupts. If an interrupt controller fails or is miss programmed it may assert an incorrect vector on the bus.

Again, pre-empt the problem by using a complete interrupt table with entries for all possible interrupts, not just the ones you are using. Unused vectors should point to an error handler.

Measure with an oscilloscope if there are any pulses at the external interrupt pins generating unexpected interrupts.

Note: a permanent active interrupt input may deadlock your system!

✓ **Watchdog timers**

Watchdog timers are another type of preventative medicine we often employ to detect failures. Too many, though, are designed for the convenience of the programmer or customer, with no thought to the poor electronics technician trying to repair things. A watchdog timer that resets and restarts the code invisibly is a nightmare - how can you tell if a system is operating marginally?

Note: if it is an external watchdog, you can check with a scope the Reset pin!

Test programs

A nice way of finding bus shorts, memory failures, and the like is to execute a looping program, letting the technician examine each address and data line with a scope to find the source of the trouble. Of course, if the memories don't work, or if the address bus is shorted, how can we run a program?

Using the Z80 RST 7 for bus testing On the Z80 and 8085 family the RST 7 instruction is a one byte CALL to location 38.

Was Intel clairvoyant, or was it just luck that caused them to use opcode FF for this instruction?) As a result, if you add pull up resistors to the bus, then simply removing all memory chips will make the processor execute CALLs to 38 all day long. The stack pointer will decrement through the processor's entire address space, so the technician can look at address lines and check that they cycle properly. The data bus will show return addresses after each RST 7 executes; since the stack pointer decrements, these addresses will change as well. This trivial test gives the repetitive signal needed to effectively use a scope to check out the hardest parts of the system.

Other CPUs usually have a similar instruction. On the 8088 family the INT 3 instruction is a similar one byte opcode. A one byte PUSH might even be better.

Since these instructions are not FF opcodes, pull up the bus and add a jumper field so the technician can set the proper opcode.

✓ RAM Test

A carefully engineered RAM test can be quite valuable. A Ram test program should be stored in the boot ROM, preferably near the reset address. The low order address lines must operate to run even a trivial program, and enough of the upper ones must work to select the boot ROM. The test itself certainly cannot make use of RAM!

Several of the failure modes manifest themselves by the inability to store data. For example, data bus problems, a bad device, chip select failures, or an incorrectly inserted pin will usually exhibit a simple read/write error. The traditional write and read of a 55, followed by an AA, will find these problems quickly.

Writing a pattern of 55H and 0AAH tests the ability of the devices to hold data, but it doesn't insure that the RAM's are being addressed correctly. Examples of failures that could pass this simple test are: a post-buffer address short, an open address line (say, from a pin not being inserted properly), or chip select failures causing multiple addressing. It's important to run a second routine that isolates these not uncommon problems.

An addressing test works by writing a unique data value to each location, and then reading the memory to see that that value is still stored. An easy-to-compute pattern is the low order address of the location; at 100 stores 00, at 101 store a 01, etc. This isn't really unique, since an 8 bit location can only store 256 different values. If we repeat the test, using the locations' high order address bytes as the data pattern, then (for memory sizes to 64k) after two passes the entire array will be tested uniquely. The first test insures that address lines A0 to A7 function correctly; the second checks lines A8 to A15 and also the chip select logic.

✓ ROM Test

If the ROM is not working, how can it test itself? As always, a completely dead kernel, one that just doesn't even boot, cannot run diagnostics. If the boot ROM does at least partially work, then some testing is valuable.

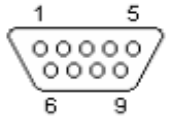
Unburned ROM space is set to 0xFFH. On the Z180/Z80 processors this is the RST7 instruction (a one byte call to location 0038). So most Z80's systems with an electronics failure that disables ROM will go into an infinite loop executing 0xFFH instructions, which is instantly visible on a scope. The characteristic double writes from the RST7's pushes tell an experienced scooper in a second what is going on. Teach your production folks this simple trick.

While the memories can fail in a number of ways, probably the most common is a miss inserted pin. If you've spent time troubleshooting electronics, you'll know that it can be awfully hard to tell if all pins are in the sockets. Other problems cover the usual range of broken circuit board tracks (i.e., address, data, control lines), miss programmed devices, and non-functioning chip select lines.

✓ **RS232 Pinning OK?**

Most problems caused by the RS232 serial line is the cable between the two devices, mostly the microprocessor SIO or microcontroller and the PC. There a two standard connectors for the RS232 serial line: a 25 pin DSUB connector and a 9 pin DSUB connector. Because of the PCs the today mostly used connector for the serial line is the 9 pin connector.

PIN	CONNECTION
1	Carrier Detect,DCD
2	Receive Data (RD)
3	Transmit Data (TD)
4	Data Terminal Ready
5	Signal Ground
6	Data Set Ready (DSR)
7	Request to Send (RTS)
8	Clear to Send (CTS)
9	Ring Indicator



9pin DSUB type connector

The Transmit Data line from the microprocessor board must be connected to the Receive Data line of the PC and the other way round. Very often a problem with the serial line is because TD and TD from the PC are connected together. To check the right connections of the serial line, measure with a normal voltmeter the signal levels at pin 2 and 3. In idle mode there must be a voltage below -5 Volt on both lines. If this is not the cause check at each end (with open cable) which pin is the TD pin (below -5 Volt) and adjust the cable.

With active data on the serial line, the level should switch between ± 8 Volt.

✓ **RS232 Line driver OK?**

If you can't measure a level below -5 Volt at pin 2 and pin 3 of the microprocessor board (9pin DSUB connector), then the line driver may be defect. The serial data stream is generated by the SIO (Serial I/O unit). All the SIO components have a TTL level output. To drive the serial line at the RS232 level there must be a RS232 line driver. Very commonly used is the MAX232 line driver. This component generates from a single +5 Volt supply all needed voltages. (± 12 Volt). For doing this the component needs external capacitors for the internal charge pump. Sometimes these capacitors are defect.

At pin 2 you should measure with a multimeter a voltage between +5 to +10 Volt. At pin 2 you should measure voltage between -5 to -10 Volt. If not so, the charge pump is not working. This may be caused by the chip itself or a defect capacitor.

Note: for details about the RS232 level converter, please refer to the datasheet (Chapter 10).

Troubleshooting the I/O devices

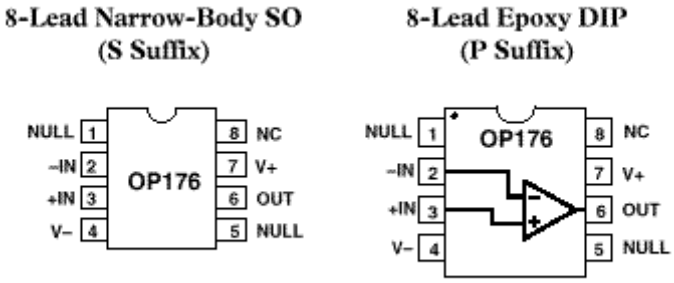
I/O devices are components like ADC, DAC, LCD Display, Keyboard controller, etc. The problems here are very wide because each device is very application specific. But still here there are some commonly known problems:

✓ Clock OK?

Also some I/O devices like a ADC needs a clock to work. In many cases the clock for the I/O components is generated from the processor clock. Check whether the I/O device has a clock. Measure the clock with a oscilloscope.

✓ Operational amplifier OK?

In front of an ADC and behind a DAC there are very often some operational amplifier for signal conditioning. If something happens at the front end then very often these OP's are destroyed.



When a operational amplifier is working, there is no voltage difference between the + and - input.

All the commonly used amplifiers with one amplifier in a package, have the same pinning.

Measure with a voltmeter between pin 2 and 3 if there is a voltage higher then 100 milliVolt, the operational amplifier chip is defect.

✓ Analog I/O power supply?

For the analog I/O section of a microcomputer system, mostly ± 12 Volt are needed. Check if there is the +/- voltage for the operational amplifiers. Measure with a voltmeter at pin 7 the positive supply voltage and at pin 4 the negative supply voltage.

✓ Adjustments OK?

Many analog I/O devices need an adjustment of working parameters with variable resistors (Trimmer). The feedback resistor of an operational amplifier could be a trimmer instead of a constant resistor, so that the gain of the OP and so the input voltage can be adjusted. A defect trimmer could result in a Gain=0, with causes that the ADC doesn't receive the input voltage to measure.

Also a LCD display has a trimmer for the contrast enhancement. The adjustment could be so bad, that you can't see any character on the LCD display even it is working.

So, check all adjustment points!

✓ Sensors

To measure physical phenomena you need at first a sensor which converts the physical phenomena into an electrical signal like a voltage. Very often these sensors causing troubles, a temperature sensor for example could be destroyed if it is overheated one times.

✓ Cables OK?

Cables between different printed circuit boards are causing many problems. Especially flat cables with pressed connectors can be defect after a longer period, if the pressing was not done absolute accurate. Check for loose cables, check for unstable contacts!

9. Test programs and Software Basics

Test Push Buttons (PB) and LED's

```
'************************************************** 30.07.2005 ************
'*                A V R – Example for ATMega8-Testboard                   *
'*            Copyright 1988-2005 RIBU Elektronik Versand                 *
'*          This program checks the status of the push-buttons            *
*            T1 to T4 and indicates it through the LED's 1 to 4          *
'*******************************************************************************
$regfile = "m8def.dat"                          'Select controller
$crystal = 16000000                             'Select crystal frequency
$baud = 9600                                    'Select Baudrate for RS232-interface
'Define pins of Port D and Port C
Config Portd = Output                           'PinD.4=LED4, PinD.5=LED3, ‘PinD.6=LED2, PinD.7=LED1
Config Pind.2 = Input                           'PinD2=PB 2
Config Pind.3 = Input                           'PinD3=PB 1
Config Portc = Input                            'PinC.0=PB 3, PinC1=PB 4,

Begin:                                          ‘Program start
    If Pind.3 = 0 Then Set Portd.7              ‘Read Port D.3 and if PB pressed then LED1 on
    If Pind.2 = 0 Then Set Portd.6
    If Pinc.0 = 0 Then Set Portd.5
    If Pinc.1 = 0 Then Set Portd.4
    Waitms 500                                  ‘Wait 500 ms before
    Reset Portd.7 : Reset Portd.6 : Reset Portd.5 : Reset Portd.4   ‘Reset of Port D
Goto Begin                                      ‘Loop back
End
```

Test ADC and LCD

```
'************************************ 30.03.2005 ***********
'*              A V R – Example for ATMega8-Testboard                *
'*            Copyright 1988-2005 RIBU Elektronik Versand            *
'*      This program reads the analog value from the potentiometer   *
'*               and displays it on the LC - Display                 *
'***************************************************************
$regfile = "m8def.dat"                                    'Select controller
$crystal = 16000000                                       'Select crystal frequency
$baud = 9600                                              'Select Baudrate for RS232-interface
'Define pins of Port D and Port C
Config Portd = Output                                     'PinD.4=LED4, PinD.5=LED3, ‘PinD.6=LED2, PinD.7=LED1
Config Pind.2 = Input                                     'PinD2=PB 2
Config Pind.3 = Input                                     'PinD3=PB 1
Config Portc = Input                                      'PinC.0=PB 3, PinC1=PB 4,
'Define pins (Port B) for LCD-Module
Config Lcdpin = Pin , Db4 = Portb.0 , Db5 = Portb.1 , Db6 = Portb.2 , Db7 = Portb.3 , E = Portb.4 , Rs = Portb.5
Config Lcd = 16 * 1a                                      'LCD-Module 1x16
Config Portb = Output
Deflcdchar 0 , 6 , 9 , 9 , 6 , 32 , 32 , 32 , 32         ' LCD – special characters
'Configure AD Converter
Config Adc = Single , Prescaler = Auto , Reference = Avcc 'Single measurement, Reference voltage = 5 Volt
‘Declare variable
Dim Res As Word , Ch As Byte
```

```
Start Adc                                   'Start ADC conversion, Stop ADC stops it
Begin:                                      'Programmstart
    Ch = 3                                  'Select AD-Input for Pot (Port C.3)
    Res = Getadc(Ch)                        'Read ADC and store it in variable Ch
    Cls                                     'Erase LCD
    Lcd "Value: "; Res                      'Display value in LCD-Display
    Waitms 500                              'Wait for 500 ms before
Goto Begin                                  'Loop back
End
```

Test I²C Bus and LED display module

```
'************************************ 03.04.2005 ***********
'*              A V R – Example for ATMega8-Testboard          *
'*            Copyright 1988-2005 RIBU Elektronik Versand      *
'*          This program counts automatically from 0 to 9999   *
'*       and displays the counter on the I2C bus LED display   *
'************************************************************
$regfile = "m8def.dat"                                  'Select controller
$crystal = 16000000                                     'Select crystal frequency
$baud = 9600                                            'Select Baudrate for RS232-interface

'Define pins of Port D and Port C
Config Portd = Output                                   'PinD.4=LED4, PinD.5=LED3, ‘PinD.6=LED2, PinD.7=LED1
Config Pind.2 = Input                                   'PinD2=PB 2
Config Pind.3 = Input                                   'PinD3=PB 1
Config Portc = Input                                    'PinC.0=PB 3, PinC1=PB 4

'Define pins used for I2C-Bus
Config Sda = Portc.4                                    ‘SDA = Serial Data
Config Scl = Portc.5                                    ‘SCL = Serial Clock

‘Declare sub-routines
Declare Sub Convert_LED

‘Declare variable
Dim A As Word, B As Byte, C As Byte, D As Byte, E As Byte, Res As Byte
```

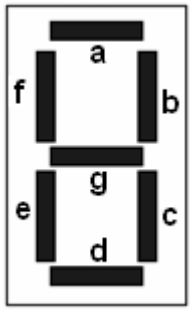
```
Begin:
    For E = 0 To 9                          ‘Create four decade counters from 0 to 9
      For D = 0 To 9                        ‘
        For C = 0 To 9
          For B = 0 To 9
            I2cstart                        'Generate an I2C start condition
            I2cwbyte &H70                   'Write I2C - Address (SAA 1064)
            I2cwbyte &H00                   'Write Register - Address
            I2cwbyte &H77                   'Write configuration

            A = E
            Call NumtoSeg                   'Convert number into segments
            I2cwbyte Res                    'Send 1st number (left first)
            A = D
            Call NumtoSeg
            Res = Res + 128                 'Turn on 2nd decimal point
            I2cwbyte Res                    'Send 2nd number
            A = C
            Call NumtoSeg
            I2cwbyte Res                    'Send 3rd number
            A = B
            Call NumtoSeg
            I2cwbyte Res                    'Send 4th number

            I2cstop                         'Generate an I2C stop condition
            Waitms 100                      'Wait for 100 ms
          Next B
        Next C
      Next D
    Next E
Goto Begin                                  'Loop back
```

```
Sub NumtoSeg                                    'Subroutine to convert number into segments
    If A = 0 Then Res   = &H3F
    If A = 1 Then Res   = &H06
    If A = 2 Then Res   = &H5B
    If A = 3 Then Res   = &H4F
    If A = 4 Then Res   = &H66
    If A = 5 Then Res   = &H6D
    If A = 6 Then Res   = &H7D
    If A = 7 Then Res   = &H07
    If A = 8 Then Res   = &H7F
    If A = 9 Then Res   = &H6F
    If A = 10 Then Res  = &H77
    If A = 11 Then Res  = &H7C
    If A = 12 Then Res  = &H39
    If A = 13 Then Res  = &H5E
    If A = 14 Then Res  = &H79
    If A = 15 Then Res  = &H71
End Sub

End
```



Example: to display number 3 turn on segments a, b, c, d, g

For details please refer to datasheet of SAA 1064

Test of Temperature Sensor MCP 9801

```
'************************************ 30.06.2005 ***********
'*              A V R – Example for ATMega8-Testboard                *
'*          Copyright 1988-2005 RIBU Elektronik Versand              *
'*        This program reads the value from sensor MCP 9801          *
'*      and displays the Temperature on the LC - Display             *
'************************************************************
$regfile = "m8def.dat"                                    'Select controller
$crystal = 16000000                                       'Select crystal frequency
$baud = 9600                                              'Select Baudrate for RS232-interface
'Define pins of Port D and Port C
Config Portd = Output                                     'PinD.4=LED4, PinD.5=LED3, ‘PinD.6=LED2, PinD.7=LED1
Config Pind.2 = Input                                     'PinD2=PB 2
Config Pind.3 = Input                                     'PinD3=PB 1
Config Portc = Input                                      'PinC.0=PB 3, PinC1=PB 4,
'Define pins (Port B) for LCD-Module
Config Lcdpin = Pin, Db4 = Portb.0, Db5 = Portb.1, Db6 = Portb.2, Db7 = Portb.3, E = Portb.4, Rs = Portb.5
Config Lcd = 16 * 1a                                      'LCD-Module 1 line x 16 characters
Config Portb = Output
Deflcdchar 0 , 6 , 9 , 9 , 6 , 32 , 32 , 32 , 32          ' LCD – special characters
‘Define Constant
Const Adr_chip_w = 144
Const Adr_chip_r = 145
‘Declare variable
Dim Temp_low As Byte, Temp_high As Byte, Adr_chip As Byte, Temp_dec As Single
'Define pins used for I2C-Bus
Config Sda = Portc.4                                      ‘SDA = Serial Data
Config Scl = Portc.5                                      ‘SCL = Serial Clock
```

```
Cursor Off Noblink                                  ’Cursor off
I2cstart                                            'Generate an I2C start condition
I2cwbyte Adr_chip_w                                 'Send address of MCP9801 and set the write-bit
I2cwbyte 1                                          'Send register-address of control-byte of MCP9801
I2cwbyte 96                                         'Send control-byte 0=9 Bit; 32=10 Bit; 64=11 Bit; 96=12 Bit
I2cstop                                             ‘Generate an I2C stop condition
I2cstart                                            'Generate an I2C start condition
I2cwbyte Adr_chip_w                                 'Send address of MCP9801 and set write-bit
I2cwbyte 0                                          ‘Send register-address of temperature reading of MCP9801
I2cstop                                             ‘Generate an I2C stop condition
Read:
    I2cstart                                        'Generate an I2C start condition
    I2cwbyte Adr_chip_r                             'Send address of MCP9801 and set read-bit
    I2crbyte Temp_high, Ack                         'Read high-byte, Temperature in Centigrade without decimals
    I2crbyte Temp_low, Nack                         'Read low-byte
    I2cstop                                         ‘Generate an I2C stop condition
    Rotate Temp_low, Right, 4                       'Rotate value 4 * right / 12bit, 5 * / 11bit, 6 * / 10bit, 7 * / 9bit
    Temp_dec = Temp_low * 0.0625                    ‘Adjust decimal places of temperature - * 0,0625 (12bit), * 0,125 (11bit)
    Temp_dec = Temp_high + Temp_dec                 ‘                                       * 025 (10bit), * 0,5 (9bit)
    Cls                                             'Clears LC - Display
    Lcd "Temp: "; Fusing(Temp_dec, “##.#”); Chr(0); " C"   'Output temperature to LC - Display
    Wait 1                                          'Wait for 1 second before
Goto Read                                           ’Loop back
End
```

Test of IR – Receiver SFH5111

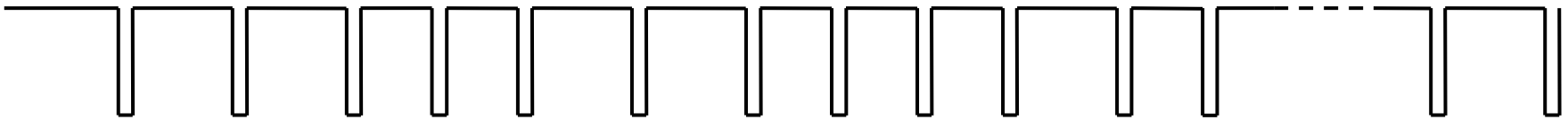
```
'************************************ 25.07.2006 ***********
'*              A V R – Example for ATMega8-Testboard          *
'*                    written by Mr. Hollenthoner              *
'*              This program reads and displays commands       *
'*                coming from the IR Sensor SFH5111            *
'*************************************************************
$regfile = "m8def.dat"                                  'Select controller
$crystal = 16000000                                     'Select crystal frequency
$baud = 9600                                            'Select Baudrate for RS232-interface
'Define pins of Port C
Config Portc.2 = Input                                  'Pulses from IR sensor
'Define Reference
Ir_Input Alias Pinc.2
'Declare variable
Dim Value As Word
'Declarations
Declare Function Ir() As Word                           'Measure time between pulses
'Configure internal timer
Config Timer0 = Timer, Prescale = 1024                  '
```

```
Enable Timer0
Print Chr(12)
Do
    Value = 0
    Print "Waiting for IR command:    ";
    Value = Ir()
    Value = Value And &H00FF
    Select Case Value
      Case &HC4 : Print "Key 1"
      Case &HC6 : Print "Key 2"
      Case &H84 : Print "Key 3"
      Case &H86 : Print "Key 4"
      Case &HA0 : Print "Key 5"
      Case &HA6 : Print "Key 6"
      Case &H88 : Print "Key 7"
      Case &HA2 : Print "Key 8"
      Case &HA4 : Print "Key 9"
      Case &HA8 : Print "Key 10"
      Case &H92 : Print "Key 11"
      Case &H98 : Print "Key 12"
      Case &H90 : Print "Key 13"
      Case &H94 : Print "Key 14"
      Case &H96 : Print "Key 15"
      Case &H82 : Print "Key 16"
    End Select
Loop

End
```

```
Function Ir() As Word
    Local I As Byte
    While Ir_input <> 0 : Wend                    'Waiting for high to low transition
    For I = 1 To 11
      Tcnt0 = 0                                   'Reset Timer 0
      Start Timer0
      While Ir_input = 0                          'Waiting until high again
      Wend
      While Ir_input <> 0                         'Waiting for next high to low transition
      Wend
      Stop Timer0
      If Tcnt0 > 110 Then Ir = Ir Or 1            'If > 6ms then "1" else "0"
      Rotate Ir , Left , 1
    Next I
End Function Ir()
```

KEY 1 pressed



1	**1**	**0**	**0**	**1**	**1**	**0**	**0**	**0**	**1**	**0**	**1**
Syn	Tog	A3	A2	A1	D5	D4	D3	D2	D1	D0	Syn

Time between 2 pulses:
0 ≈ 6ms
1 ≈ 8ms

16 CHANNEL IR TRANSMITTER K8049 TRANSMISSION CODES

Key	Syn	Tog	A3	A2	A1	D5	D4	D3	D2	D1	D0	Hex
1	1	X	0	0	1	1	0	0	0	1	0	**0462**
2	1	X	0	0	1	1	0	0	0	1	1	**0463**
3	1	X	0	0	1	0	0	0	0	1	0	**0442**
4	1	X	0	0	1	0	0	0	0	1	1	**0443**
5	1	X	0	0	1	0	1	0	0	0	0	**0450**
6	1	X	0	0	1	0	1	0	0	1	1	**0456**
7	1	X	0	0	1	0	0	0	1	0	0	**0444**
8	1	X	0	0	1	0	1	0	0	0	1	**0451**
S H I F T												
1	1	X	0	0	1	0	1	0	0	1	0	**0452**
2	1	X	0	0	1	0	1	0	1	0	0	**0454**
3	1	X	0	0	1	0	0	1	0	0	1	**0449**
4	1	X	0	0	1	0	0	1	1	0	0	**044C**
5	1	X	0	0	1	0	0	1	0	0	0	**0448**
6	1	X	0	0	1	0	0	1	0	1	0	**044A**
7	1	X	0	0	1	0	0	1	0	1	1	**044B**
8	1	X	0	0	1	0	0	0	0	0	1	**0441**

Test of Real Time Clock PCF8583

```
'************************************ 25.07.2006 ***********
'*              A V R – Example for ATMega8-Testboard                *
'*                    written by Mr. Hollenthoner                    *
'*              This program presets the RTC (PCF8583)               *
'*                    reads the time and displays it                 *
'*************************************************************
$regfile = "m8def.dat"                    ' Select controller
$crystal = 16000000                       'Select crystalfrequency
$baud = 9600                              'Select Baudrate for communication                '
'Declare variable
Dim Buffer As String * 8 , Buffer1 As String * 1 , Buffer2 As Byte , S As Byte , M As Byte , H As Byte
Dim S1 As Byte , M1 As Byte , H1 As Byte
'Define pins used for I2C-Bus
Config Scl = Portc.5                      'PORTC.5 is the SCL line
Config Sda = Portc.4                      'PORTC.4 is the SDA line

Begin:
Input "Time (HH:MM:SS): ", Buffer         'Wait for preset to be input, separate inputs by colon
Buffer1 = Mid(buffer , 7 , 1)             'Read 10 second
Buffer2 = Val(buffer1)                    'Convert string into number
Shift Buffer2 , Left , 4                  'Shift into correct position
S = Buffer2                               'Store it
Buffer1 = Right(buffer , 1)               'Read 1 second
Buffer2 = Val(buffer1) And &H0F           'Convert string into number and blank high nibble
S = S + Buffer2                           'Add the two numbers and assemble seconds in one Byte
Buffer1 = Mid(buffer , 4 , 1)             'Same as above for minutes
Buffer2 = Val(buffer1)
```

```
Shift Buffer2 , Left , 4
M = Buffer2
Buffer1 = Mid(buffer , 5 , 1)
Buffer2 = Val(buffer1) And &H0F
M = M + Buffer2
Buffer1 = Left(buffer , 1 )                ‘Same as above for hours
Buffer2 = Val(buffer1)
Shift Buffer2 , Left , 4
H = Buffer2
Buffer1 = Mid(buffer , 2 , 1)
Buffer2 = Val(buffer1) And &H0F
H = H + Buffer2
I2cstart                                   'Start I2C communication
I2cwbyte &HA0                              'Set I2C Address (PCF 8583 = &hAX)
I2cwbyte &H02                              'Set Register-Address to seconds (Addr. 02)
I2cwbyte S                                 'Write seconds
I2cwbyte M                                 'Write minutes
I2cwbyte H                                 'Write hours
I2cstop                                    'Stop communication
Waitms 10                                  'Wait 10ms to allow correct response
Lop:
    I2cstart                               'Start I2C communicationen new
    I2cwbyte &HA0                          'Write I2C Address (PCF 8583 = &hAX)
    I2cwbyte &H02                          'Write Register-Address (02 = seconds)
    I2cstart                               'Start I2C communicationen new
    I2cwbyte &HA1                          'Set I2C Address (PCF 8583 = &hAX)
    I2crbyte S , Ack                       'Read seconds
    I2crbyte M , Ack                       'Read minutes
    I2crbyte H , Nack                      'Read hours
```

```
    I2cstop                                          'Stop communication
    S1 = S                                           'Copy secondes into tenths of seconds
    Shift S1 , Right , 4                             'Shift 4x into correct position
    M1 = M                                           'Copy minutes into tenths of minutes
    Shift M1 , Right , 4                             'Shift 4x into correct position
    H1 = H                                           'Copy hours into tenths of hours
    Shift H1 , Right , 4                             'Shift 4x into correct position
    S = S And &H0F                                   'Blank high nibble
    M = M And &H0F
    H = H And &H0F
    Print chr(12)                                    ‘Clear screen
    Print "Time: " ; H1 ; H ; ":" ; M1 ; M ; ":" ; S1 ; S   'Show time
    Waitms 900                                       'Wait 900ms before
Goto Lop                                             'Looping

End
```

Datasheets

intersil

ICL232

August 1997

+5V Powered, Dual RS-232 Transmitter/Receiver

Features

- Meets All RS-232C and V.28 Specifications
- Requires Only Single +5V Power Supply
- Onboard Voltage Doubler/Inverter
- Low Power Consumption
- 2 Drivers
  - ±9V Output Swing for +5V Input
  - 300Ω Power-off Source Impedance
  - Output Current Limiting
  - TTL/CMOS Compatible
  - 30V/µs Maximum Slew Rate
- 2 Receivers
  - ±30V Input Voltage Range
  - 3kΩ to 7kΩ Input Impedance
  - 0.5V Hysteresis to Improve Noise Rejection
- All Critical Parameters are Guaranteed Over the Entire Commercial, Industrial and Military Temperature Ranges

Applications

- Any System Requiring RS-232 Communications Port
  - Computer - Portable and Mainframe
  - Peripheral - Printers and Terminals
  - Portable Instrumentation
  - Modems
- Dataloggers

Description

The ICL232 is a dual RS-232 transmitter/receiver interface circuit that meets all EIA RS-232C and V.28 specifications. It requires a single +5V power supply, and features two onboard charge pump voltage converters which generate +10V and -10V supplies from the 5V supply.

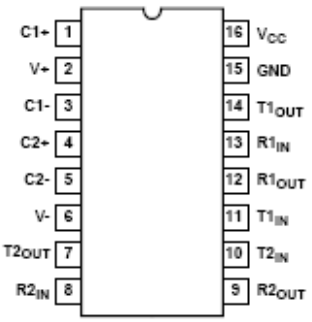
The drivers feature true TTL/CMOS input compatibility, slew-rate-limited output, and 300Ω power-off source impedance. The receivers can handle up to +30V, and have a 3kΩ to 7kΩ input impedance. The receivers also have hysteresis to improve noise rejection.

Ordering Information

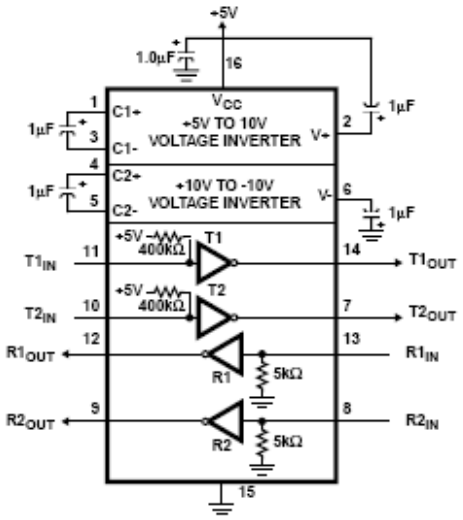
PART NUMBER	TEMP. RANGE (^{o}C)	PACKAGE	PKG. NO.
ICL232CPE	0 to 70	16 Ld PDIP	E16.3
ICL232CBE	0 to 70	16 Ld SOIC	M16.3
ICL232IPE	-40 to 85	16 Ld PDIP	E16.3
ICL232IJE	-40 to 85	16 Ld CERDIP	F16.3
ICL232IBE	-40 to 85	16 Ld SOIC	M16.3
ICL232MJE	-55 to 125	16 Ld CERDIP	F16.3

Pinout

ICL232 (PDIP, CERDIP, SOIC)
TOP VIEW



Functional Diagram



CAUTION: These devices are sensitive to electrostatic discharge; follow proper IC Handling Procedures.

File Number **3020.5**

ICL232

Absolute Maximum Ratings

V_{CC} to Ground . . . . . (GND -0.3V) < V_{CC} < 6V
V+ to Ground . . . . . (V_{CC} -0.3V) < V+ < 12V
V- to Ground . . . . . -12V < V- < (GND +0.3V)
Input Voltages
$T1_{IN}$, $T2_{IN}$ . . . . . (V- -0.3V) < V_{IN} < (V+ +0.3V)
$R1_{IN}$, $R2_{IN}$ . . . . . ±30V
Output Voltages
$T1_{OUT}$, $T2_{OUT}$ . . . . . (V- -0.3V) < V_{TXOUT} < (V+ +0.3V)
$R1_{OUT}$, $R2_{OUT}$ . . . . . (GND -0.3V) < V_{RXOUT} < (V_{CC} +0.3V)
Short Circuit Duration
$T1_{OUT}$, $T2_{OUT}$ . . . . . Continuous
$R1_{OUT}$, $R2_{OUT}$ . . . . . Continuous

Operating Conditions

Temperature Ranges
ICL232C . . . . . 0°C to 70°C
ICL232I . . . . . -40°C to 85°C
ICL232M . . . . . -55°C to 125°C

Thermal Information

Thermal Resistance (Typical, Note 1)	θ_{JA} (°C/W)	θ_{JC} (°C/W)
CERDIP Package	80	18
PDIP Package	100	N/A
SOIC Package	100	N/A

Maximum Junction Temperature
Plastic Packages . . . . . 150°C
Ceramic Package . . . . . 175°C
Maximum Storage Temperature Range . . . . . -65°C to 150°C
Maximum Lead Temperature (Soldering 10s) . . . . . 300°C

CAUTION: Stresses above those listed in "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress only rating and operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied.

NOTE:

1. θ_{JA} is measured with the component mounted on an evaluation PC board in free air.

Electrical Specifications

Test Conditions: V_{CC} = +5V ±10%, T_A = Operating Temperature Range. Test Circuit as in Figure 8 Unless Otherwise Specified

PARAMETER	TEST CONDITIONS	MIN	TYP	MAX	UNITS
Transmitter Output Voltage Swing, T_{OUT}	$T1_{OUT}$ and $T2_{OUT}$ Loaded with 3kΩ to Ground	±5	±9	±10	V
Power Supply Current, I_{CC}	Outputs Unloaded, T_A = 25°C	-	5	10	mA
T_{IN}, Input Logic Low, V_{IL}		-	-	0.8	V
T_{IN}, Input Logic High, V_{IH}		2.0	-	-	V
Logic Pullup Current, I_P	$T1_{IN}$, $T2_{IN}$ = 0V	-	15	200	µA
RS-232 Input Voltage Range, V_{IN}		-30	-	+30	V
Receiver Input Impedance, R_{IN}	V_{IN} = ±3V	3.0	5.0	7.0	kΩ
Receiver Input Low Threshold, V_{IN} (H-L)	V_{CC} = 5V, T_A = 25°C	0.8	1.2	-	V
Receiver Input High Threshold, V_{IN} (L-H)	V_{CC} = 5V, T_A = 25°C	-	1.7	2.4	V
Receiver Input Hysteresis, V_{HYST}		0.2	0.5	1.0	V
TTL/CMOS Receiver Output Voltage Low, V_{OL}	I_{OUT} = 3.2mA	-	0.1	0.4	V
TTL/CMOS Receiver Output Voltage High, V_{OH}	I_{OUT} = -1.0mA	3.5	4.6	-	V
Propagation Delay, t_{PD}	RS-232 to TTL	-	0.5	-	µs
Instantaneous Slew Rate, SR	C_L = 10pF, R_L = 3kΩ, T_A = 25°C (Notes 2, 3)	-	-	30	V/µs
Transition Region Slew Rate, SR_T	R_L = 3kΩ, C_L = 2500pF Measured from +3V to -3V or -3V to +3V	-	3	-	V/µs
Output Resistance, R_{OUT}	V_{CC} = V+ = V- = 0V, V_{OUT} = ±2V	300	-	-	Ω
RS-232 Output Short Circuit Current, I_{SC}	$T1_{OUT}$ or $T2_{OUT}$ Shorted to GND	-	±10	-	mA

NOTES:

2. Guaranteed by design.
3. See Figure 4 for definition.

Pin Descriptions (Continued)

PDIP, CERDIP	SOIC	PIN NAME	DESCRIPTION
12	12	$R1_{OUT}$	Receiver 1 TTL/CMOS output.
13	13	$R1_{IN}$	RS-232 Receiver 1 input, with internal 5K pulldown resistor to GND.
14	14	$T1_{OUT}$	RS-232 Transmitter 1 output ±10V (typical).
15	15	GND	Supply Ground.
16	16	V_{CC}	Positive Power Supply +5V ±10%

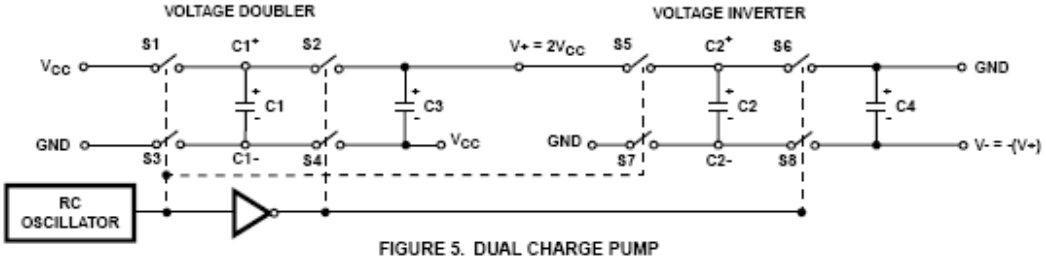


FIGURE 5. DUAL CHARGE PUMP

Detailed Description

The ICL232 is a dual RS-232 transmitter/receiver powered by a single +5V power supply which meets all EIA RS232C specifications and features low power consumption. The functional diagram illustrates the major elements of the ICL232. The circuit is divided into three sections: a voltage doubler/inverter, dual transmitters, and dual receivers Voltage Converter.

An equivalent circuit of the dual charge pump is illustrated in Figure 5.

The voltage quadrupler contains two charge pumps which use two phases of an internally generated clock to generate +10V and -10V. The nominal clock frequency is 16kHz. During phase one of the clock, capacitor C1 is charged to V_{CC}. During phase two, the voltage on C1 is added to V_{CC}, producing a signal across C2 equal to twice V_{CC}. At the same time, C3 is also charged to $2V_{CC}$, and then during phase one, it is inverted with respect to ground to produce a signal across C4 equal to $-2V_{CC}$. The voltage converter accepts input voltages up to 5.5V. The output impedance of the doubler (V+) is approximately 200Ω, and the output impedance of the inverter (V-) is approximately 450Ω. Typical graphs are presented which show the voltage converters output vs input voltage and output voltages vs load characteristics. The test circuit (Figure 3) uses 1µF capacitors for C1-C4, however, the value is not critical. Increasing the values of C1 and C2 will lower the output impedance of the voltage doubler and inverter, and increasing the values of the reservoir capacitors, C3 and C4, lowers the ripple on the V+ and V- supplies.

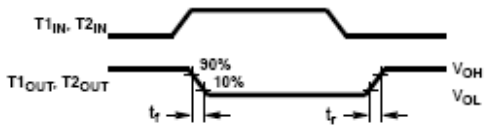


FIGURE 6. SLEW RATE DEFINITION

Transmitters

The transmitters are TTL/CMOS compatible inverters which translate the inputs to RS-232 outputs. The input logic threshold is about 26% of V_{CC}, or 1.3V for V_{CC} = 5V. A logic 1 at the input results in a voltage of between -5V and V- at the output, and a logic 0 results in a voltage between +5V and (V+ - 0.6V). Each transmitter input has an internal 400kΩ pullup resistor so any unused input can be left unconnected and its output remains in its low state. The output voltage swing meets the RS-232C specification of ±5V minimum with the worst case conditions of: both transmitters driving 3kΩ minimum load impedance, V_{CC} = 4.5V, and maximum allowable operating temperature. The transmitters have an internally limited output slew rate which is less than 30V/µs. The outputs are short circuit protected and can be shorted to ground indefinitely. The powered down output impedance is a minimum of 300Ω with ±2V applied to the outputs and V_{CC} = 0V.

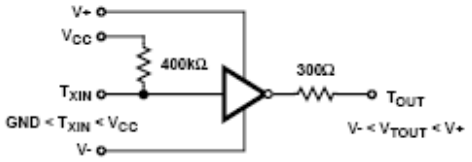


FIGURE 7. TRANSMITTER

Receivers

The receiver inputs accept up to ±30V while presenting the required 3kΩ to 7kΩ input impedance even it the power is off (V_{CC} = 0V). The receivers have a typical input threshold of 1.3V which is within the ±3V limits, known as the transition region, of the RS-232 specification. The receiver output is 0V to V_{CC}. The output will be low whenever the input is greater than 2.4V and high whenever the input is floating or driven between +0.8V and -30V. The receivers feature 0.5V hysteresis to improve noise rejection.

ICL232

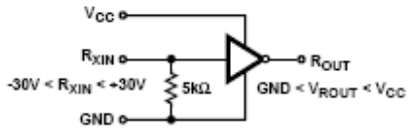
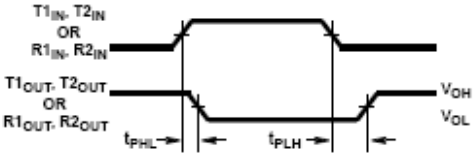


FIGURE 8. RECEIVER



Average Propagation Delay = $\frac{t_{PHL} + t_{PLH}}{2}$

FIGURE 9. PROPAGATION DELAY DEFINITION

Applications

The ICL232 may be used for all RS-232 data terminal and communication links. It is particularly useful in applications where ±12V power supplies are not available for conventional RS-232 interface circuits. The applications presented represent typical interface configurations.

A simple duplex RS-232 port with CTS/RTS handshaking is illustrated in Figure 10. Fixed output signals such as DTR (data terminal ready) and DSRS (data signaling rate select) is generated by driving them through a 5kΩ resistor connected to V+.

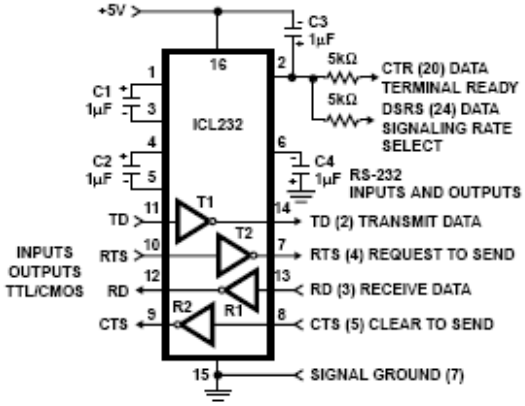


FIGURE 10. SIMPLE DUPLEX RS-232 PORT WITH CTS/RTS HANDSHAKING

In applications requiring four RS-232 inputs and outputs (Figure 11), note that each circuit requires two charge pump capacitors (C1 and C2) but can share common reservoir capacitors (C3 and C4). The benefit of sharing common reservoir capacitors is the elimination of two capacitors and the reduction of the charge pump source impedance which effectively increases the output swing of the transmitters.

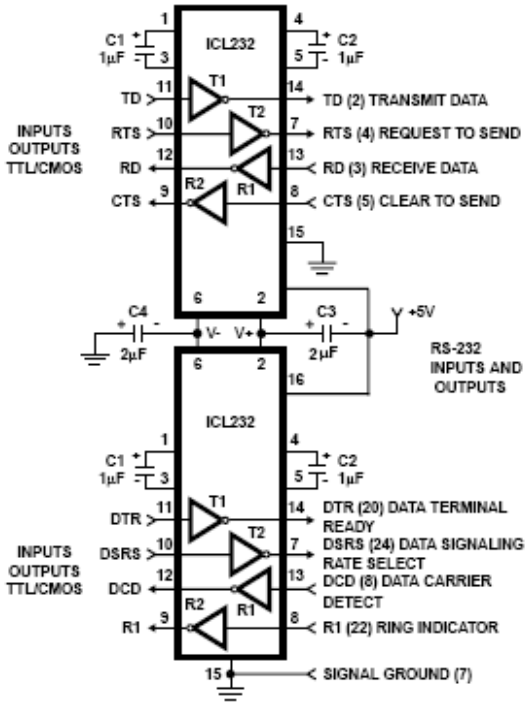
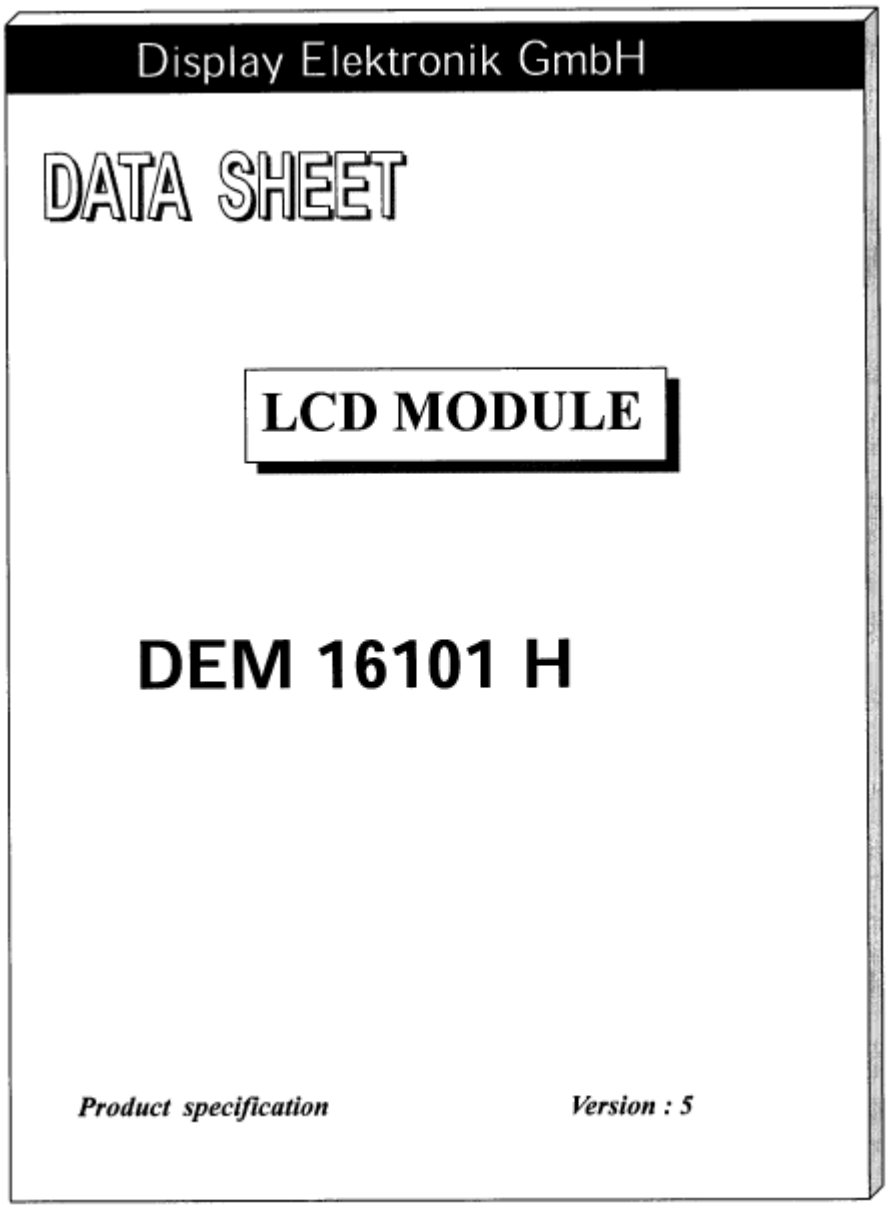


FIGURE 11. COMBINING TWO ICL232s FOR 4 PAIRS OF RS-232 INPUTS AND OUTPUTS


Display Elektronik GmbH
DATA SHEET
LCD MODULE
DEM 16101 H
Product specification
Version : 5

02/Apr/2003

CONTENTS

DEM 16101 H — Product Specification

1.FUNCTIONS & FEATURES

- Module-Type :

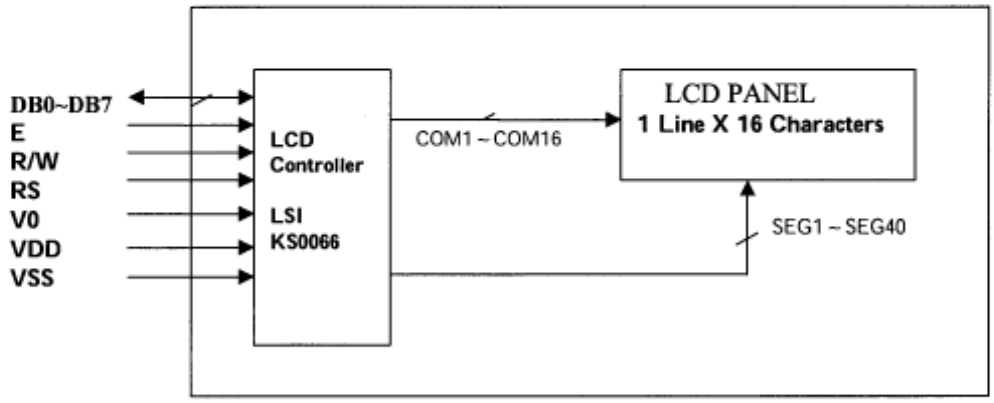
MODULE	LCD MODEL	LCD TYPE
DEM 16101 H	TN	Reflective Positive Mode

- Viewing Direction: : 6° clock
- Driving Scheme : 1/16 Duty Cycle, 1/5 Bias
- Power Supply Voltage : 5.0 V
- VLCD Adjustable For Best Contrast : 4.5 V (typ.)
- Display contents : 16 x 1Characters (5 x 8 dots, Format: 208 Kids)
- Internal Memory : CGROM (10,080 bits)
  : CGRAM (64 x 8 bits)
  : DDRAM (80 x 8 bits for Digits)
- Easy Interface with a 4-bit or 8-bit MPU

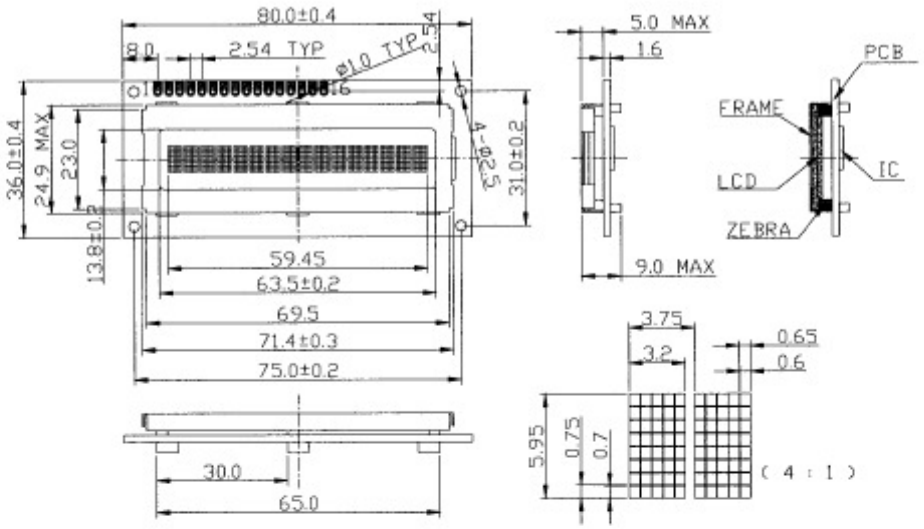
2. MECHANICAL SPECIFICATIONS

- Character Pitch : 3.75 (W) mm
- Character Size : 3.20 (W) x 5.95 (H) mm
- Character Font : 5 x 8 dots
- Dot Size : 0.60 (W) x 0.70 (H) mm
- Dot Pitch : 0.65 (W) x 0.75 (H) mm

3. BLOCK DIAGRAM



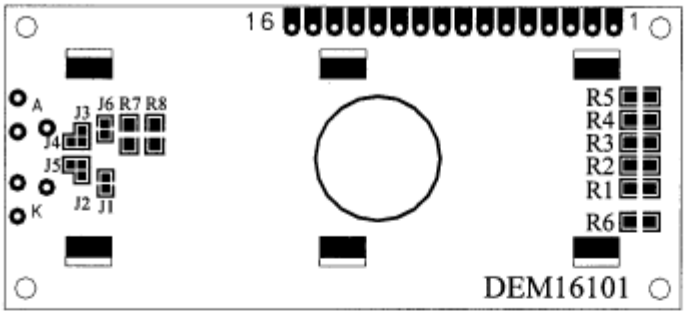
4. EXTERNAL DIMENSIONS



5. PIN ASSIGNMENT

Pin No.	Symbol	Function
1	Vss	Ground
2	VDD	Power supply (5V)
3	V0	Power Supply for LCD
4	RS	Select Display Data ("H") or Instructions ("L")
5	R/W	Read or Write Select Signal
6	E	Read/Write Enable Signal
7	DB0	Display Data Signal
8	DB1	
9	DB2	
10	DB3	
11	DB4	
12	DB5	
13	DB6	
14	DB7	
15	LED-(K)	Not Used (Prepared for version with backlight)
16	LED+(A)	Not Used (Prepared for version with backlight)

6.1 PCB DRAWING AND DESCRIPTION



DESCRIPTION: (Not applicable for this version)

6-1-1.The polarity of the pin 15 and the pin 16:

symbol	symbol state	J3,J5	J2, J4	LED Polarity	
				15 Pin	16 Pin
J2,J4	Each solder-bridge	Each open	------	Anode	Cathode
J3,J5	Each solder-bridge	------	Each open	Cathode	Anode

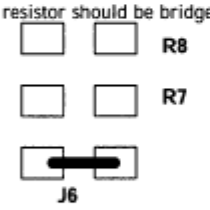
6-1-2. The metal-bezel should be on ground when the J1 is solder-Bridge.

6-1-3.The LED resistor should be bridged when the J6 is solder-Bridge.

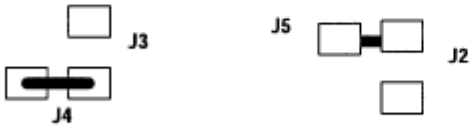
6-1-4.The R7 and the R8 are the LED resistor.

6.2 Example application (Not applicable for this version)

6-2-1. The LED resistor should be bridged as following.



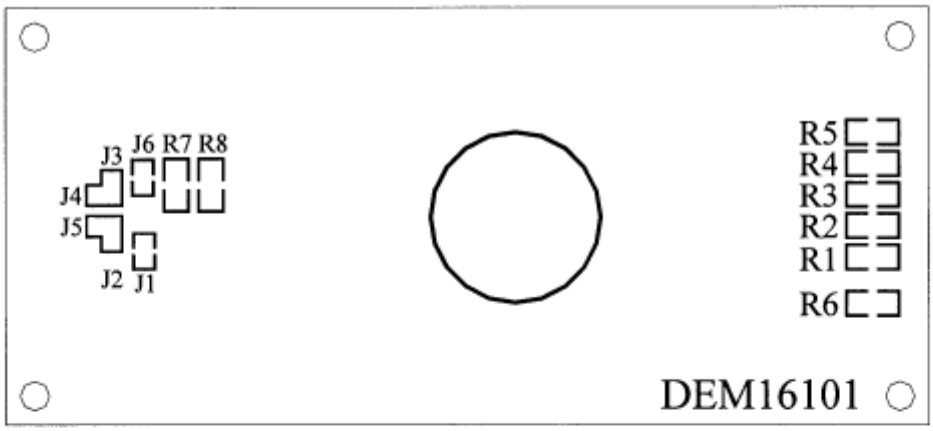
6-2-2. The 15 pin is the anode and the 16 pin is the cathode as following.



6-2-3.The 15 pin is the cathode and the 16 pin is the anode as following.

6-2-4. The metal-bezel is on ground as following.

6.3 The Module NO. Printed on the PCB.



6.4 DISPLAY DATA RAM(DDRAM)

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	← DISPLAY POSITION
00	01	02	03	04	05	06	07	40	41	42	43	44	45	46	47	← DDRAM ADDRESS

7. MAXIMUM ABSOLUTE RATINGS

Item	Symbol	Standard value	Unit
Power supply voltage (1)	V_{DD}	-0.3 ~ +7.0	V
Power supply voltage (2)	V_0	V_{DD}-13.5 ~ V_{DD}+0.3	V
Input voltage	V_{IN}	-0.3 ~ V_{DD}+0.3	V
Operating temperature	T_{opr}	-20 ~ +70	°C
Storage temperature	T_{stg}	-25 ~ +75	°C

*Voltage greater than above may damage to the Circuit.
VDD ≥ V1 ≥ V2 ≥ V3 ≥ V4 ≥ V5

8. ELECTRICAL CHARACTERISTICS

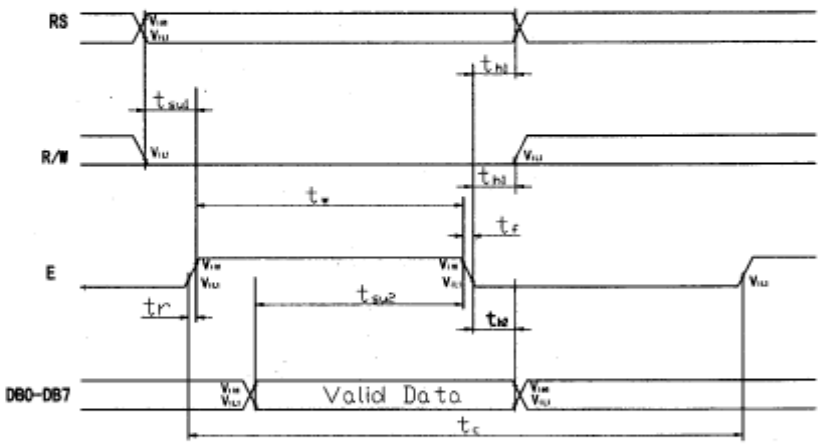
8-1 DC Characteristics (VDD=4.5V ~ 5.5V, Ta = -20 ~ +70°C)

Item	Symbol	Standard Value			Test Condition	Unit
		MIN	TYP	MAX		
Operating Voltage	V_{DD}	4.5	5	5.5	-------	V
Supply Current	I_{DD}	----	0.35	0.6	V_{DD}=5V, fosc=270kHz	mA
Input Voltage (1) (except OSC1)	V_{IL1}	-0.3	--	0.6	--------	V
	V_{IH1}	2.2	--	V_{DD}	---------	
Input Voltage (2) (OSC1)	V_{IL2}	-0.2	--	1.0	---------	V
	V_{IH2}	V_{DD}-1.0	--	V_{DD}	---------	
Output Voltage (1) (DB0 to DB7)	V_{OL1}	---	----	0.4	I_{OL}=1.2mA	V
	V_{OH1}	2.4	---	----	I_{OH}=-0.205mA	
Output Voltage (2) (except DB0 to DB7)	V_{OL2}	------	---	0.1V_{DD}	I_{OL}=40uA	V
	V_{OH2}	0.9V_{DD}	---	-----	I_{OH}=-40uA	
Voltage Drop	Vd_{COM}	-----	---	1	Io = ±0.1 mA	V
	Vd_{SEG}	------	---	1		
Input Leakage Current	I_{IKG}	-1	---	1	V_{IN}=0 V to V_{DD}	uA
Input Low Current	I_{IL}	-50	-125	-250	V_{IN}=0V, V_{DD}=5V(pull up)	uA
Internal Clock (external Rf)	f_{OSC1}	190	270	350	Rf = 91k±2% (V_{DD}=5V)	kHz
External Clock	f_{OSC}	125	270	410	----	kHz
	Duty	45	50	55	----	%
	t_R, t_F	---	---	0.2	----	us
LCD Driving Voltage	V_{LCD}	3.0	---	13.0	V_{DD}-V_5 (1/5, 1/4 Bias)	V

8-2 AC Characteristics (VDD = 4.5V ~ 5.5V , Ta = -20 ~ +70°C)

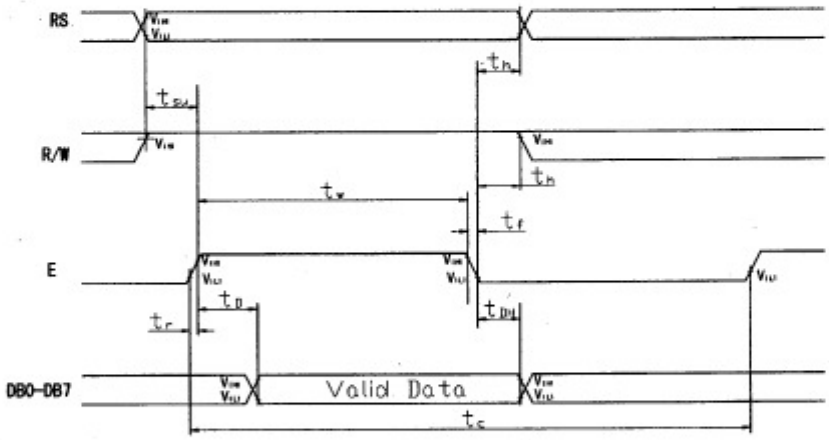
8-2-1 Write mode (writing data from MPU to DEM 16101-Series)

Characteristic	Symbol	Min	Type	Max	Unit	Test PIN
E Cycle Time	t_C	500	---	---	ns	E
E Rise Time	t_R	---	---	20	ns	E
E Fall Time	t_F	---	---	20	ns	E
E Pulse width (High,Low)	t_W	230	---	---	ns	E
R/W and RS Set-up Time	t_{SU1}	40	---	---	ns	R/W,RS
R/W and RS Hold Time	t_{H1}	10	---	---	ns	R/W,RS
Data Set-up Time	t_{SU2}	80	---	---	ns	DB0~DB7
Data Hold Time	t_{H2}	10	---	---	ns	DB0~DB7



8-2-2 Read mode (reading data from DEM 16101-Series to MPU)

Characteristic	Symbol	Min	Type	Max	Unit	Test PIN
E Cycle Time	t_C	500	---	---	ns	E
E Rise Time	t_R	---	---	20	ns	E
E Fall Time	t_F	---	---	20	ns	E
E Pulse width (High, Low)	t_W	230	---	---	ns	E
R/W and RS Set-up Time	t_{SU}	40	---	---	ns	R/W,RS
R/W and RS Hold Time	t_H	10	---	---	ns	R/W,RS
Data Output Delay Time	t_D	---	---	120	ns	DB0~DB7
Data Hold Time	t_{DH2}	5	---	---	ns	DB0~DB7



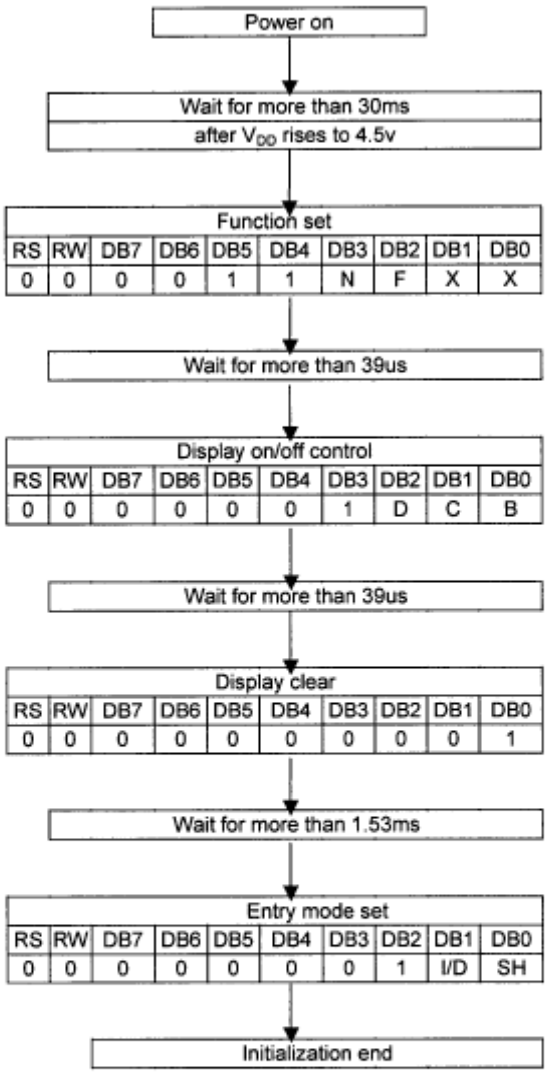
9. CONTROL AND DISPLAY COMMAND

Command	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Execution time (fosc=270KHz)	Remark
clear Display	L	L	L	L	L	L	L	L	L	H	1.53ms	Write "20H" to DDRAM and set DDRAM address to "00H" from AC
Return home	L	L	L	L	L	L	L	L	H	---	1.53ms	Cursor move to first digit
Entry mode set	L	L	L	L	L	L	L	H	I/D	SH	39us	I/D:set cursor move direction I/D: H Increase; I Decrease SH:Specifies shift of display SH: H Display is shifted; L Display is not shifted
Display on/off control	L	L	L	L	L	L	H	D	C	B	39us	Display D: H Display on; L Display off Cursor C: H Cursor on; L Cursor off Blinking B: H Blinking on; L Blinking off
Cursor or Display Shift	L	L	L	L	L	H	S/C	R/L	---	---	39us	SC: H Display shift; L Cursor move R/L: H Right shift; L Left shift
function Set	L	L	L	L	H	DL	N	F	---	---	39us	DL: H 8bits interface; L 4bits interface N: H 2 line display; L 1 line display F: H Display on; L Display off
Set CGRAM address	L	L	L	H	AC5	AC4	AC3	AC2	AC1	AC0	39us	CGRAM data is sent and received after this setting
Set DDRAM address	L	L	H	AC6	AC5	AC4	AC3	AC2	AC1	AC0	39us	DDRAM data is sent and received after this setting
Read busy flag& address	L	H	BF	AC6	AC5	AC4	AC3	AC2	AC1	AC0	0us	BF: H Busy; L Ready -Reads BF indication internal operating is being performed -Reads address counter contents
Write data to RAM	H	L	D7	D6	D5	D4	D3	D2	D1	D0	43us	Write data into DDRAM or CGRAM
Read data from RAM	H	H	D7	D6	D5	D4	D3	D2	D1	D0	43us	Read data from DDRAM or CGRAM

10. STANDARD CHARACTER PATTERN

11. LCM INITIALIZING BY INSTRUCTION

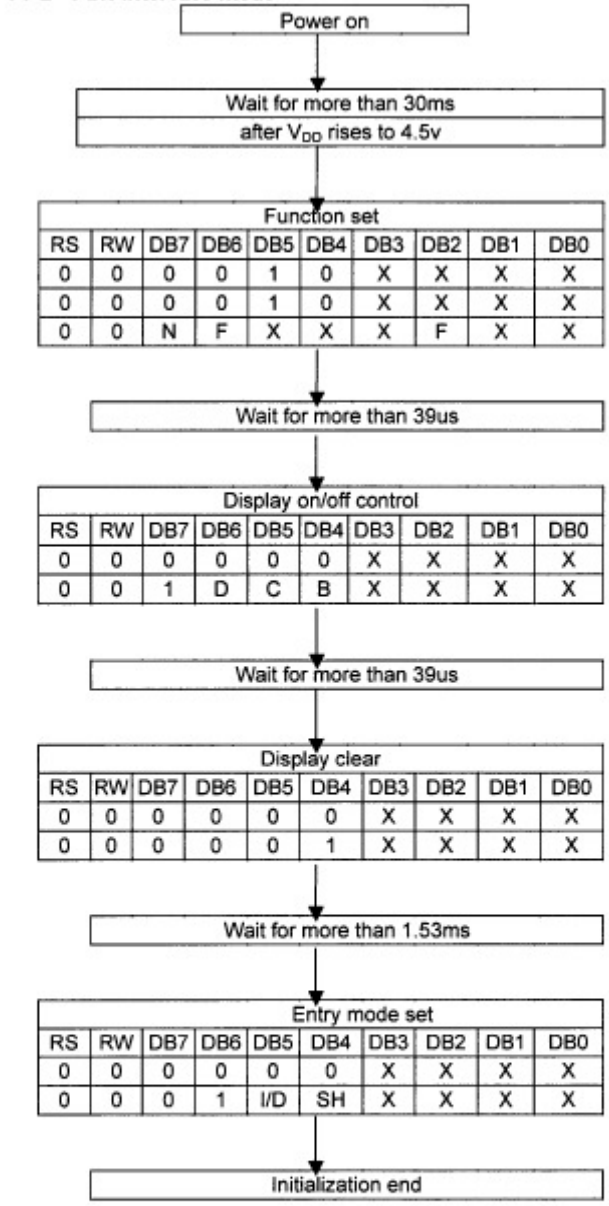
11-1 8-bit interface mode



Condition : fosc=270khz

N	0	1-line mode
	1	2-line mode
F	0	display off
	1	display on
D	0	display off
	1	display on
C	0	cursor off
	1	cursor on
B	0	blink off
	1	blink on
I/D	0	decrement mode
	1	increment mode
SH	0	entire shift off
	1	entire shift on

11-2 4-bit interface mode



Condition : fosc=270khz

N	0	1-line mode
	1	2-line mode
F	0	display off
	1	display on

D	0	display off
	1	Display on
C	0	cursor off
	1	cursor on
B	0	blink off
	1	blink on

I/D	0	decrement mode
	1	increment mode
SH	0	entire shift off
	1	entire shift on

12. LCD Modules Handling Precautions

- The display panel is made of glass. Do not subject it to a mechanical shock by dropping it from a high place, etc.

- If the display panel is damaged and the liquid crystal substance inside it leaks out, do not get any in your mouth. If the substance come into contact with your skin or clothes promptly wash it off using soap and water.

- Do not apply excessive force to the display surface or the adjoining areas since this may cause the color tone to vary.

- The polarizer covering the display surface of the LCD module is soft and easily scratched. Handle this polarize carefully.

- To prevent destruction of the elements by static electricity, be careful to maintain an optimum work environment.
  -Be sure to ground the body when handling the LCD module.
  -Tools required for assembly, such as soldering irons, must be properly grounded.
  -To reduce the amount of static electricity generated, do not conduct assembly and other work under dry conditions.
  -The LCD module is coated with a film to protect the display surface. Exercise care when peeling off this protective film since static electricity may be generated.

- Storage precautions
  When storing the LCD modules, avoid exposure to direct sunlight or to the light of fluorescent lamps. Keep the modules in bags designed to prevent static electricity charging under low temperature / normal humidity conditions (avoid high temperature / high humidity and low temperatures below 0°C).Whenever possible, the LCD modules should be stored in the same conditions in which they were shipped from our company.

13. Others

- Liquid crystals solidify at low temperature (below the storage temperature range) leading to defective orientation of liquid crystal or the generation of air bubbles (black or white). Air bubbles may also be generated if the module is subjected to a strong shock at a low temperature.

- If the LCD modules have been operating for a long time showing the same display patterns may remain on the screen as ghost images and a slight contrast irregularity may also appear. Abnormal operating status can be resumed to be normal condition by suspending use for some time. It should be noted that this phenomena does not adversely affect performance reliability.

- To minimize the performance degradation of the LCD modules resulting from caused by static electricity, etc. exercise care to avoid holding the following sections when handling the modules:
  - Exposed area of the printed circuit board
  - Terminal electrode sections

INTEGRATED CIRCUITS

DATA SHEET

SAA1064
4-digit LED-driver with I^2C-Bus interface

Product specification
File under Integrated Circuits, IC01

February 1991

Philips
Semiconductors

PHILIPS

4-digit LED-driver with I²C-Bus interface SAA1064

GENERAL DESCRIPTION

The LED-driver is a bipolar integrated circuit made in an I²L compatible 18 volts process. The circuit is especially designed to drive four 7-segment LED displays with decimal point by means of multiplexing between two pairs of digits. It features an I²C-Bus slave transceiver interface with the possibility to program four different SLAVE ADDRESSES, a POWER RESET flag, 16 current sink OUTPUTS, controllable by software up to 21 mA, two multiplex drive outputs for common anode segments, an on-chip multiplex oscillator, control bits to select static, dynamic and blank mode, and one bit for segment test.

QUICK REFERENCE DATA

PARAMETER	CONDITIONS	SYMBOL	MIN.	TYP.	MAX.	UNIT
Supply voltage	V_{EE} = 0 V	V_{CC}	4.5	5	15	V
Supply current all outputs OFF	V_{CC} = 5 V	I_{CC}[1]	7	9.5	14	mA
Total power dissipation						
24-lead DIL (SOT101B)		P_{tot}	–	–	1000	mW
24-lead DIL SO (SOT137A)		P_{tot}	–	–	500	mW
Operating ambient						
temperature range		T_{amb}	–40	–	+85	°C

Note

1. The positive current is defined as the conventional current flow into a device (sink current).

PACKAGE OUTLINE

SAA1064: 24-lead DIL; plastic with internal heat spreader (SOT101B); SOT101-1; 1996 August 30.

SAA1064T: 24-lead mini-pack; plastic (SO-24; SOT137A); SOT137-1; 1996 August 30.

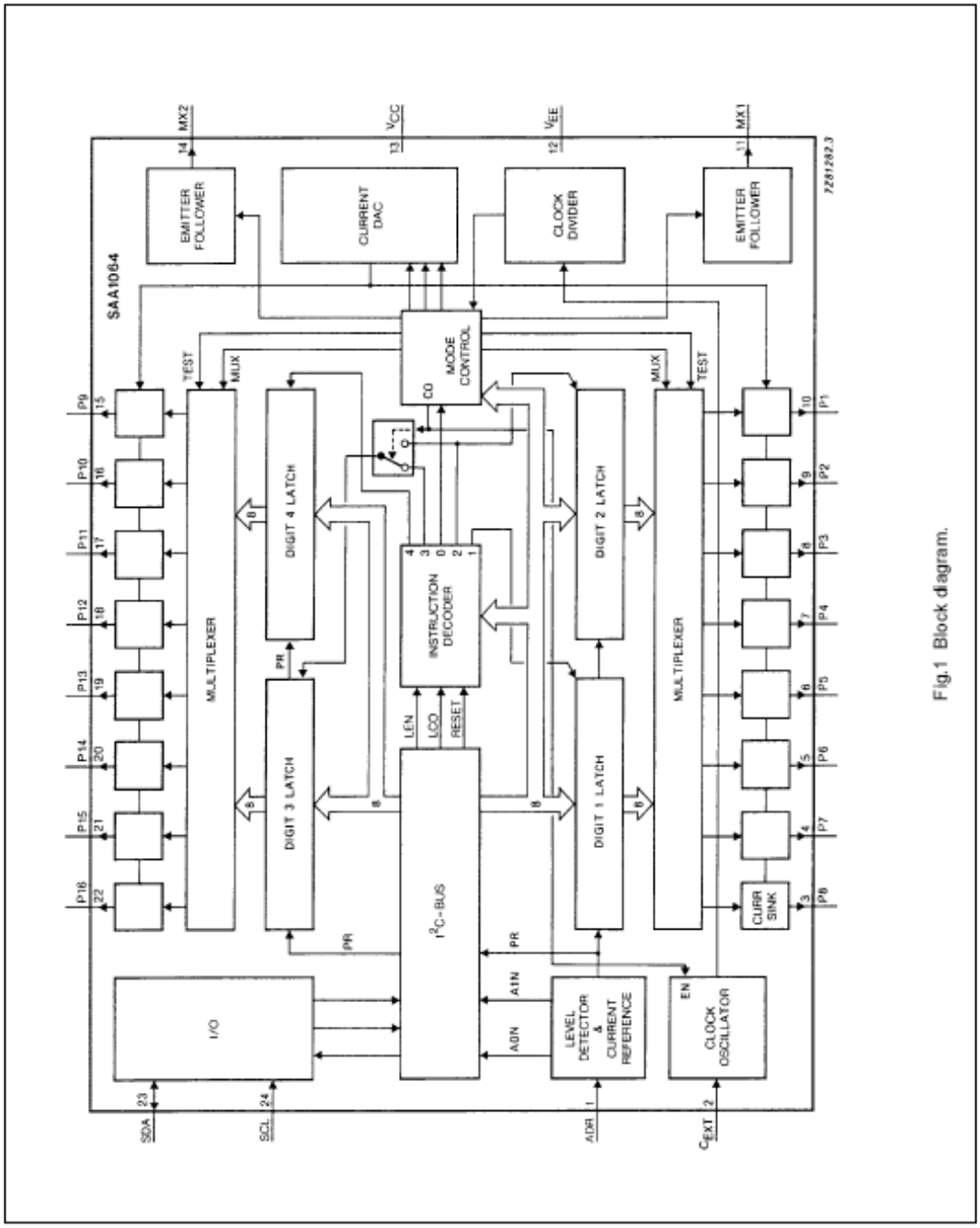


Fig.1 Block diagram.

PINNING

SYMBOL	PIN	DESCRIPTION
ADR	1	I²C-Bus slave address input
C_{EXT}	2	external control
P8 to P1	3-10	segment output
MX1	11	multiplex output
V_{EE}	12	ground
V_{CC}	13	positive supply
MX2	14	multiplex output
P9 to P16	15-22	segment output
SDA	23	I²C-Bus serial data line
SCL	24	I²C-Bus serial clock line

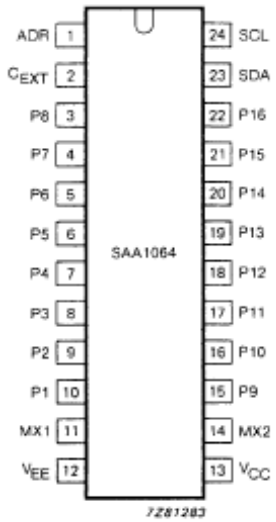
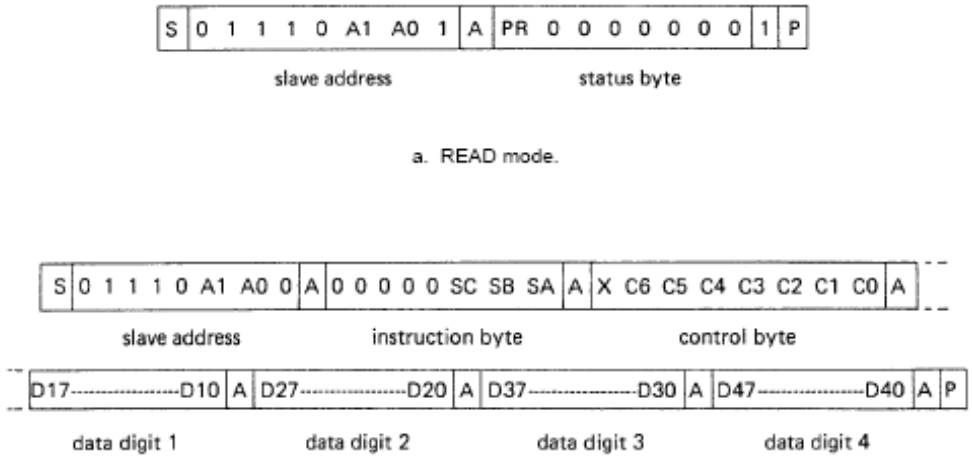


Fig.2 Pinning diagram.

FUNCTIONAL DESCRIPTION



S = start condition
P = stop condition
A = acknowledge
X = don't care

A1, A0 = programmable address bits
SC SB SA = subaddress bits
C6 to C0 = control bits
PR = POWER RESET flag

Fig.3 I²C-Bus format.

Address pin ADR

Four different slave addresses can be chosen by connecting ADR either to V_{EE}, 3/8 V_{CC}, 5/8 V_{CC} or V_{CC}. This results in the corresponding valid addresses HEX 70, 72, 74 and 76 for writing and 71, 73, 75 and 77 for reading. All other addresses cannot be acknowledged by the circuit.

Status byte

Only one bit is present in the status byte, the POWER RESET flag. A logic 1 indicates the occurence of a power failure since the last time it was read out. After completion of the READ action this flag will be set to logic 0.

Subaddressing

The bits SC, SB and SA form a pointer and determine to which register the data byte following the instruction byte will be written. All other bytes will then be stored in the registers with consecutive subaddresses. This feature is called Auto-Increment (AI) of the subaddress and enables a quick initialization by the master.
The subaddress pointer will wrap around from 7 to 0.
The subaddresses are given as follows:

SC	SB	SA	SUB-ADDRESS	FUNCTION
0	0	0	00	control register
0	0	1	01	digit 1
0	1	0	02	digit 2
0	1	1	03	digit 3
1	0	0	04	digit 4
1	0	1	05	reserved, not used
1	1	0	06	reserved, not used
1	1	1	07	reserved, not used

Control bits (see Fig.4)

The control bits C0 to C6 have the following meaning:

C0 = 0	static mode, i.e. continuous display of digits 1 and 2
C0 = 1	dynamic mode, i.e. alternating display of digit 1 + 3 and 2 + 4
C1 = 0/1	digits 1 + 3 are blanked/not blanked
C2 = 0/1	digits 2 + 4 are blanked/not blanked
C3 = 1	all segment outputs are switched-on for segment test(1)
C4 = 1	adds 3 mA to segment output current
C5 = 1	adds 6 mA to segment output current
C6 = 1	adds 12 mA to segment output current

Note

1. At a current determined by C4, C5 and C6.

Data

A segment is switched ON if the corresponding data bit is logic 1. Data bits D17 to D10 correspond with digit 1, D27 to D20 with digit 2, D37 to D30 with digit 3 and D47 to D40 with digit 4.

The MSBs correspond with the outputs P8 and P16, the LSBs with P1 and P9. Digit numbers 1 to 4 are equal to their subaddresses (hex) 1 to 4.

SDA, SCL

The SDA and SCL I/O meet the I²C-Bus specification. For protection against positive voltage pulses on these inputs voltage regulator diodes are connected to V_{EE}. This means that normal line voltage should not exceed 5,5 volt. Data will be latched on the positive-going edge of the acknowledge related clock pulse.

Power-on reset

The power-on reset signal is generated internally and sets all bits to zero, resulting in a completely blanked display. Only the POWER RESET flag is set.

External Control (C_{EXT})

With a capacitor connected to pin 2 the multiplex frequency can be set (see Fig.5). When static this pin can be connected to V_{EE} or V_{CC} or left floating since the oscillator will be switched off.

Segment outputs

The segment outputs P1 to P16 are controllable current-sink sources. They are switched on by the corresponding data bits and their current is adjusted by control bits C4, C5 and C6.

Multiplex outputs

The multiplex outputs MX1 and MX2 are switched alternately in dynamic mode with a frequency derived from the clock-oscillator. In static mode MX1 is switched on. The outputs consist of an emitter-follower, which can be used to drive the common anodes of two displays directly provided that the total power dissipation of the circuit is not exceeded. If this occurs external transistors should be connected to pins 11 and 14 as shown in Fig.5.

RATINGS

Limiting values in accordance with the Absolute Maximum System (IEC 134)

PARAMETER	CONDITIONS	SYMBOL	MIN.	MAX.	UNIT
Supply voltage (pin 13)	V_{EE} = 0 V	V_{CC}	–0.5	18	V
Supply current (pin 13)		I_{CC}	–50	200	mA
Total power dissipation					
24-lead DIL (SOT101B)		P_{tot}		1000	mW
24-lead SO (SO137A)		P_{tot}		500	mW
SDA, SCL voltages	V_{EE} = 0 V	$V_{23, 24}$	–0.5	5.9	V
Voltages ADR-MX1 and MX2-P16	V_{EE} = 0 V	V_{1-11}, V_{14-22}	–0.5	V_{CC} + 0.5	V
Input/output current all pins	outputs OFF	$\pm I_{I/O}$	–	10	mA
Operating ambient					
temperature range		T_{amb}	–40	+85	°C
Storage temperature range		T_{stg}	–55	+150	°C

THERMAL RESISTANCE

From crystal to ambient		
24-lead DIL	$R_{th\ j\text{-}a}$	35 K/W
24-lead SO (on ceramic substrate)	$R_{th\ j\text{-}a}$	75 K/W
24-lead SO (on printed circuit board)	$R_{th\ j\text{-}a}$	105 K/W

CHARACTERISTICS

V_{CC} = 5 V; T_{amb} = 25 °C; voltages are referenced to ground (V_{EE} = 0 V); unless otherwise specified

PARAMETER	CONDITIONS	SYMBOL	MIN.	TYP.	MAX.	UNIT
Supply						
Supply voltage (pin 13)		V_{CC}	4,5	5,0	15	V
Supply current	all outputs OFF					
	V_{CC} = 5 V	I_{CC}	7,0	9,5	14,0	mA
Power dissipation	all outputs OFF	P_d	–	50	–	mW
SDA; SCL (pins 23 and 24)						
Input voltages		$V_{23,24}$	0	–	5,5	V
Logic input voltage LOW		$V_{IL(L)}$	–	–	1,5	V
Logic input voltage HIGH		$V_{IH(L)}$	3,0	–	–	V
Input current LOW	$V_{23,24} = V_{EE}$	$-I_{IL}$	–	–	10	µA
Input current HIGH	$V_{23,24} = V_{CC}$	I_{IH}	–	–	10	µA
SDA						
Logic output voltage LOW	I_O = 3 mA	$V_{OL(L)}$	–	–	0,4	V
Output sink current		I_{SDA}	3	–	–	mA
Address input (pin 1)						
Input voltage						
programmable address bits:						
A0 = 0; A1 = 0		V_1	V_{EE}	–	$3/16V_{CC}$	V
A0 = 1; A1 = 0		V_1	$5/16V_{CC}$	$3/8V_{CC}$	$7/16V_{CC}$	V
A0 = 0; A1 = 1		V_1	$9/16V_{CC}$	$5/8V_{CC}$	$11/16V_{CC}$	V
A0 = 1; A1 = 1		V_1	$13/16V_{CC}$	–	V_{CC}	V
Input current LOW	$V_1 = V_{EE}$	$-I_1$	–	–	10	µA
Input current HIGH	$V_1 = V_{CC}$	I_1	–	–	10	µA
External control (C_{EXT}) pin 2						
Switching level input						
Input voltage LOW		V_{IL}	–	–	$V_{CC}-3,3$	V
Input voltage HIGH		V_{IH}	$V_{CC}-1,5$	–	–	V
Input current	V_2 = 2 V	I_2	–140	–160	–180	µA
	V_2 = 4 V	I_2	140	160	180	µA

PARAMETER	CONDITIONS	SYMBOL	MIN.	TYP.	MAX.	UNIT
Segment outputs						
(P8 to P1; pins 3 to 10)						
P9 to P16; pins 15 to 22)						
Output voltages	I_O = 15 mA	V_O	–	–	0.5	V
Output leakage current HIGH	$V_O = V_{CC}$ = 15 V	I_{LO}	–	–	± 10	µA
Output current LOW						
All control bits (C4, C5						
and C6) are HIGH	V_{OL} = 5 V	I_{OL}	17.85	21	25.2	mA
Contribution of:						
control bit C4		I_O	2.55	3.0	3.6	mA
control bit C5		I_O	5.1	6.0	7.2	mA
control bit C6		I_O	10.2	12.0	14.4	mA
Relative segment output current accuracy						
with respect to highest value		ΔI_O	–	–	7.5	%
Multiplex 1 and 2 (pins 11 and 14)						
Maximum output voltage						
(when ON)	$-I_{MPX}$ = 50 mA	V_{MPX}	V_{CC}–1.5	–	–	V
Maximum output current HIGH						
(when ON)	V_{MPX} = 2 V	$-I_{MPX}$	50	–	110	mA
Maximum output current LOW						
(when OFF)	V_O = 2 V	$+I_{MPX}$	50	70	110	µA
Multiplex output period	C_{EXT} = 2.7 nF	T_{MPX}	5	–	10	ms
Multiplexed duty factor			–	48.4	–	%

* Value to be fixed.

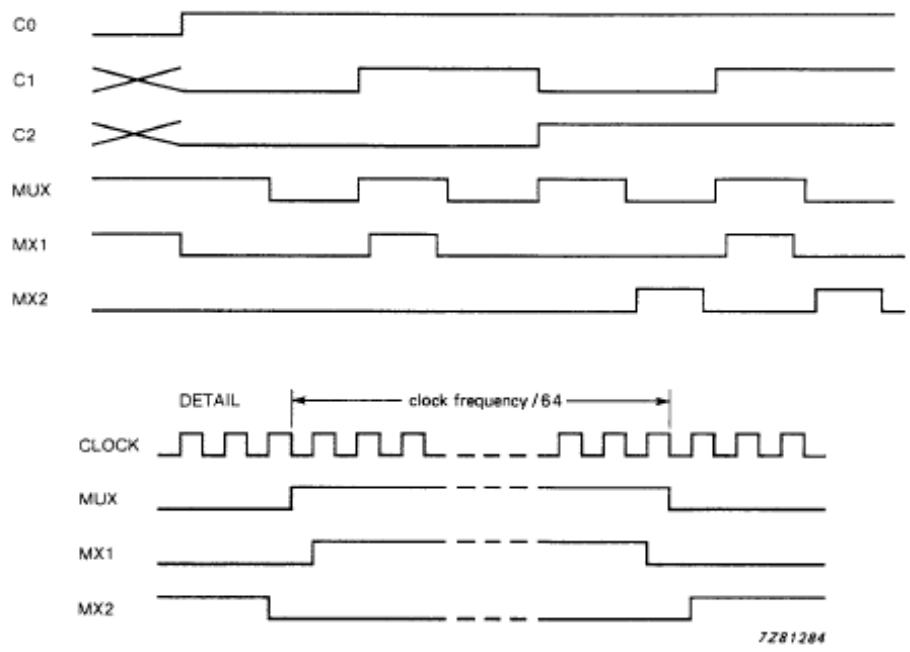


Fig.4 Timing diagram.

APPLICATION INFORMATION

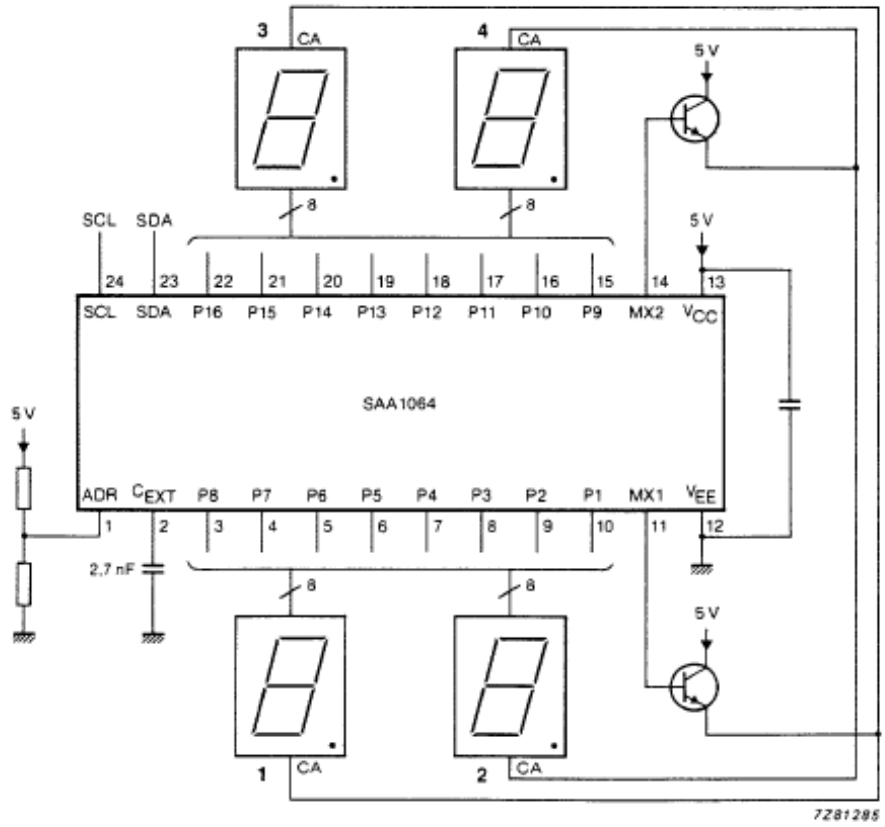


Fig.5 Dynamic mode application diagram.

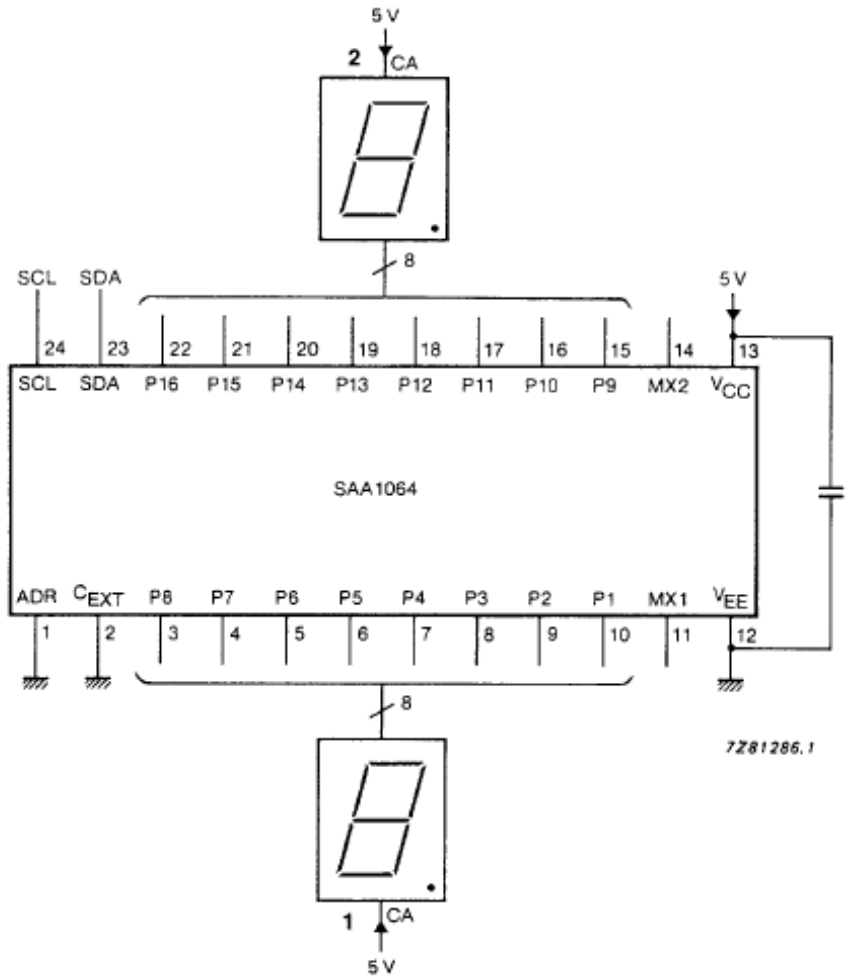


Fig.6 Static mode application diagram.

POWER DISSIPATION

The total maximum power dissipation of the SAA1064 is made up by the following parts:

1. Maximum dissipation when none of the outputs are programmed (continuous line in Fig.7).
2. Maximum dissipation of each programmed output. The dashed line in Fig.7 visualises the dissipation when all the segments are programmed (max. 16 in the static, and max. 32 in the dynamic mode). When less segments are programmed one should take a proportional part of the maximum value.
3. Maximum dissipation of the programmed segment drivers which can be expressed as:
   $P_{add} = V_O \times I_O \times N$.

Where:
- P_{add} = The additional power dissipation of the segment drivers
- V_O = The low state segment driver output voltage
- I_O = The programmed segment output current
- N = The number of programmed segments in the static mode, or half the number of programmed segment drivers in the dynamic mode.

Under no conditions the total maximum dissipation (500 mW for the SO and 1000 mW for the DIL package) should be exceeded.

Example:
- V_{CC} = 5 V
- V_O = 0.25 V
- I_O = 12 mA

24 programmed segments in dynamic mode

$P_{tot} = P_1 + P_2 + P_3$

$= 75\ mW + (50 * 24/32)\ mW + (0.25 * 12.10^{-3} * 12)\ mW$

$= 148.5\ mW$

MCP9800/1/2/3

2-Wire High-Accuracy Temperature Sensor

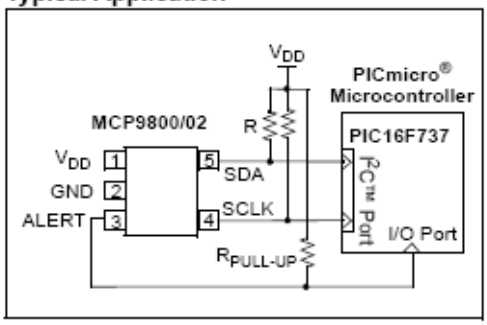
Features

- Temperature-to-Digital Converter
- Accuracy with 12-bit Resolution:
  - ±0.5°C (typ.) at +25°C
  - ±1°C (max.) from -10°C to +85°C
  - ±2°C (max.) from -10°C to +125°C
  - ±3°C (max.) from -55°C to +125°C
- User-selectable Resolution: 9 – 12 bit
- Operating Voltage Range: 2.7V to 5.5V
- 2-wire Interface: I^2C™/SMBus Compatible
- Operating Current: 200 µA (typ.)
- Shutdown Current: 1 µA (max.)
- Power-saving One-shot Temperature Measurement
- Available Packages: SOT-23-5, MSOP-8, SOIC-8

Typical Applications

- Personal Computers and Servers
- Hard Disk Drives and Other PC Peripherals
- Entertainment Systems
- Office Equipment
- Data Communication Equipment
- Mobile Phones
- General-purpose Temperature Monitoring

Typical Application



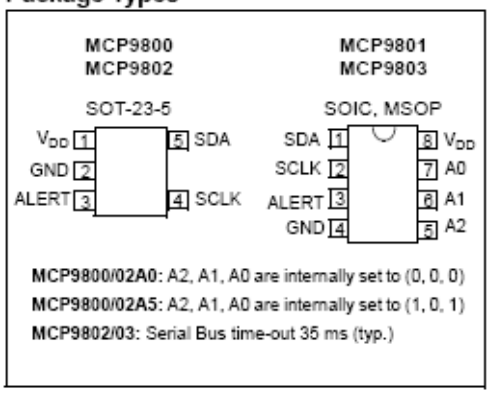
Description

Microchip Technology Inc.'s MCP9800/1/2/3 family of digital temperature sensors converts temperatures between -55°C and +125°C to a digital word. They provide an accuracy of ±1°C (max.) from -10°C to +85°C.

The MCP9800/1/2/3 family comes with user-programmable registers that provide flexibility for temperature sensing applications. The register settings allow user-selectable 9-bit to 12-bit temperature measurement resolution, configuration of the power-saving Shutdown and One-shot (single conversion on command while in the Shutdown) modes and the specification of both temperature alert output and hysteresis limits. When the temperature changes beyond the specified limits, the MCP9800/1/2/3 outputs an alert signal. The user has the option of setting the alert output signal polarity as an active-low or active-high comparator output for thermostat operation, or as temperature event interrupt output for microprocessor-based systems.

This sensor has an industry standard 2-wire, I^2C™/SMBus compatible serial interface, allowing up to eight devices to be controlled in a single serial bus. These features make the MCP9800/1/2/3 ideal for sophisticated multi-zone temperature-monitoring applications.

Package Types



MCP9800/02A0: A2, A1, A0 are internally set to (0, 0, 0)
MCP9800/02A5: A2, A1, A0 are internally set to (1, 0, 1)
MCP9802/03: Serial Bus time-out 35 ms (typ.)

MCP9800/1/2/3

1.0 ELECTRICAL CHARACTERISTICS

Absolute Maximum Ratings †

V_{DD} 6.0V

Voltage at all Input/Output pins .... GND – 0.3V to 5.5V

Storage temperature .....-65°C to +150°C

Ambient temp. with power applied .....-55°C to +125°C

Junction Temperature (T_J)..... 150°C

ESD protection on all pins (HBM:MM)..... (4 kV:400V)

Latch-Up Current at each pin ..... ±200 mA

†Notice: Stresses above those listed under "Maximum ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at those or any other conditions above those indicated in the operational listings of this specification is not implied. Exposure to maximum rating conditions for extended periods may affect device reliability.

PIN FUNCTION TABLE

NAME	FUNCTION
SDA	Bidirectional Serial Data (open-drain output)
SCLK	Serial Clock Input
ALERT	Temperature Alert Output (open-drain)
A2	Address Select Pin (bit 2)
A1	Address Select Pin (bit 1)
A0	Address Select Pin (bit 0)
V_{DD}	Power Supply Input
GND	Ground

DC CHARACTERISTICS

Electrical Specifications: Unless otherwise indicated, V_{DD} = 2.7V to 5.5V, GND = Ground, and T_A = -55°C to +125°C.

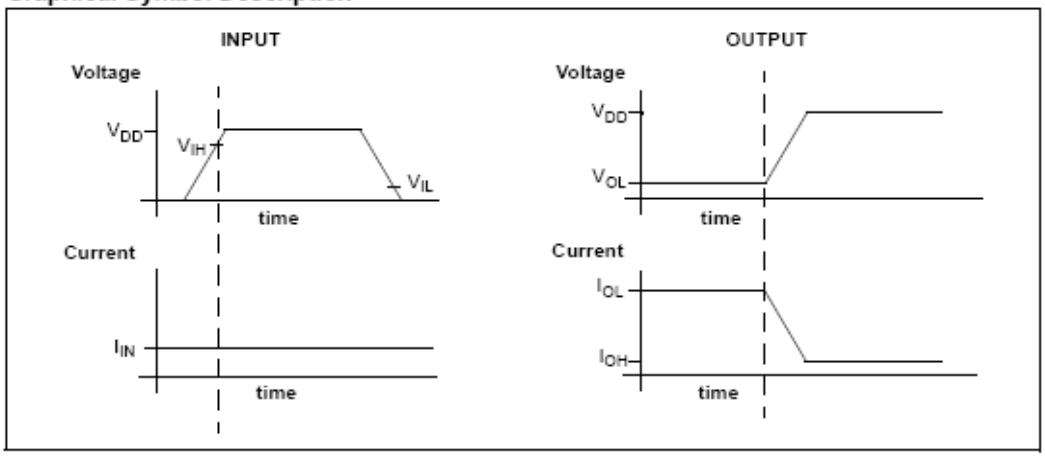
Parameters	Sym	Min	Typ	Max	Unit	Conditions
Power Supply						
Operating Voltage Range	V_{DD}	2.7	—	5.5	V	
Operating Current	I_{DD}	—	200	400	µA	Continuous Operation
Shutdown Current	I_{SHDN}	—	0.1	1	µA	Shutdown Mode
Power On Reset Threshold (POR)	V_{POR}	—	1.7	—	V	V_{DD} falling edge
Temperature Sensor Accuracy						
Accuracy with 12-bit Resolution:						
T_A = +25°C	T_{ACY}	—	±0.5	—	°C	V_{DD} = 3.3V
-10°C < T_A ≤ +85°C	T_{ACY}	-1.0	—	+1.0	°C	V_{DD} = 3.3V
-10°C < T_A ≤ +125°C	T_{ACY}	-2.0	—	+2.0	°C	V_{DD} = 3.3V
-55°C < T_A ≤ +125°C	T_{ACY}	-3.0	—	+3.0	°C	V_{DD} = 3.3V
Internal ΣΔ ADC						
Conversion Time:						
9-bit Resolution	t_{CONV}	—	30	75	ms	33 samples/sec (typ.)
10-bit Resolution	t_{CONV}	—	60	150	ms	17 samples/sec (typ.)
11-bit Resolution	t_{CONV}	—	120	300	ms	8 samples/sec (typ.)
12-bit Resolution	t_{CONV}	—	240	600	ms	4 samples/sec (typ.)
Alert Output (Open-drain)						
High-level Current	I_{OH}	—	—	1	µA	V_{OH} = 5V
Low-level Voltage	V_{OL}	—	—	0.4	V	I_{OL}= 3 mA
Thermal Response						
Response Time	t_{RES}	—	1.4	—	s	Time to 63% (88°C) 27°C (Air) to 125°C (oil bath)

DIGITAL INPUT/OUTPUT PIN CHARACTERISTICS

Electrical Specifications: Unless otherwise indicated, V_{DD} = 2.7V to 5.5V, GND = Ground and T_A = -55°C to +125°C.

Parameters	Sym	Min	Typ	Max	Units	Conditions
Serial Input/Output (SCLK, SDA, A0, A1, A2)						
Input						
High-level Voltage	V_{IH}	0.7 V_{DD}	—	—	V	
Low-level Voltage	V_{IL}	—	—	0.3 V_{DD}	V	
Input Current	I_{IN}	-1	—	+1	µA	
Output (SDA)						
Low-level Voltage	V_{OL}	—	—	0.4	V	I_{OL} = 3 mA
High-level Current	I_{OH}	—	—	1	µA	V_{OH} = 5V
Low-level Current	I_{OL}	6	—	—	mA	V_{OL} = 0.6V
Capacitance	C_{IN}	—	10	—	pF	
SDA and SCLK Inputs						
Hysteresis	V_{HYST}	0.05 V_{DD}	—	—	V	

Graphical Symbol Description



TEMPERATURE CHARACTERISTICS

Electrical Specifications: Unless otherwise indicated, V_{DD} = +2.7V to +5.5V, GND = Ground.

Parameters	Sym	Min	Typ	Max	Units	Conditions
Temperature Ranges						
Specified Temperature Range	T_A	-55	—	+125	°C	**(Note 1)**
Operating Temperature Range	T_A	-55	—	+125	°C	
Storage Temperature Range	T_A	-65	—	+150	°C	
Thermal Package Resistances						
Thermal Resistance, 5L-SOT23	θ_{JA}	—	256	—	°C/W	
Thermal Resistance, 8L-SOIC	θ_{JA}	—	163	—	°C/W	
Thermal Resistance, 8L-MSOP	θ_{JA}	—	206	—	°C/W	

Note 1: Operation in this range must not cause T_J to exceed Maximum Junction Temperature (+150°C).

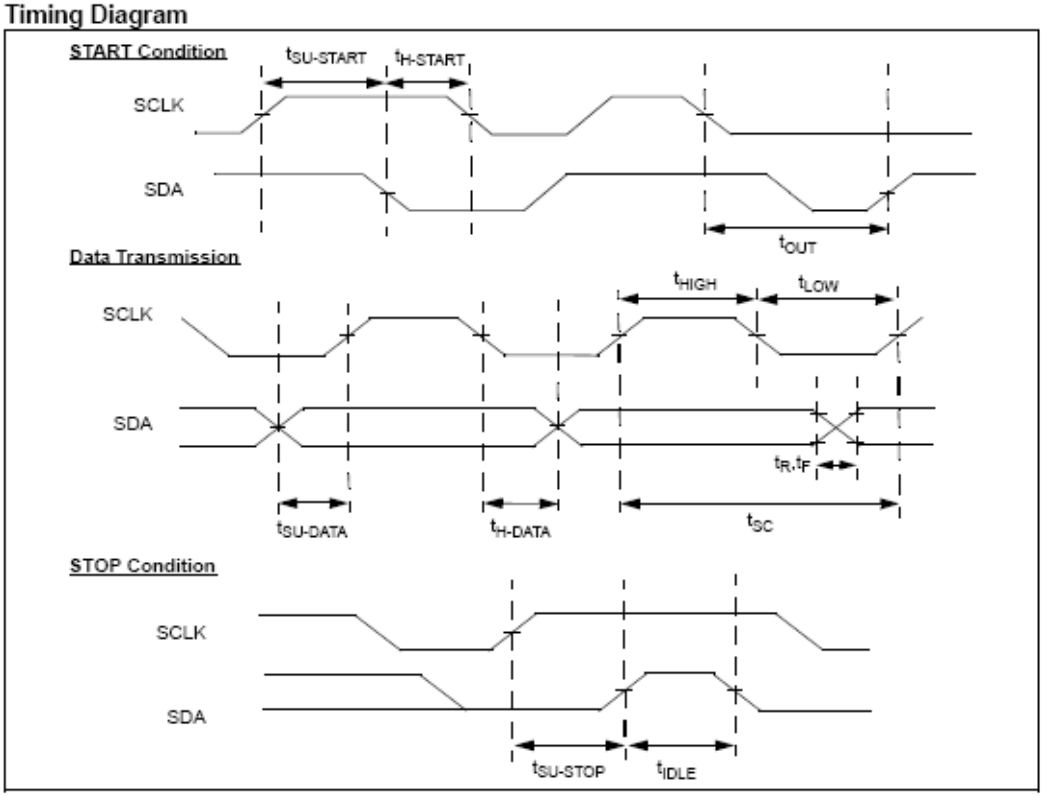
MCP9800/1/2/3

SERIAL INTERFACE TIMING SPECIFICATIONS

Electrical Specifications: Unless otherwise indicated, V_{DD} = 2.7V to 5.5V, GND = Ground, -55°C < T_A < +125°C, C_L = 80 pF, and all limits measured to 50% point.

Parameters	Sym	Min	Typ	Max	Units	Conditions
2-Wire I²C/SMBus Compatible Interface						
Serial Port Frequency	f_{SC}	0	—	400	kHz	I²C MCP9800/01
	f_{SC}	10	—	400	kHz	SMBus MCP9802/03
Clock Period	t_{SC}	2.5	—	—	µs	
Low Clock	t_{LOW}	1.3	—	—	µs	
High Clock	t_{HIGH}	0.6	—	—	µs	
Rise Time	t_R	20	—	300	ns	10% to 90% of V_{DD} (SCLK, SDA)
Fall Time	t_F	20	—	300	ns	90% to 10% of V_{DD} (SCLK, SDA)
Data Setup Before SCLK High	$t_{SU-DATA}$	0.1	—	—	µs	
Data Hold After SCLK Low	t_{H-DATA}	0	—	0.9	µs	
Start Condition Setup Time	$t_{SU-START}$	0.6	—	—	µs	
Start Condition Hold Time	$t_{H-START}$	0.6	—	—	µs	
Stop Condition Setup Time	$t_{SU-STOP}$	0.6	—	—	µs	
Bus Idle	t_{IDLE}	1.3	—	—	µs	
Time Out	t_{OUT}	25	35	50	ms	MCP9802/03 only

Timing Diagram



2.0 TYPICAL PERFORMANCE CURVES

Note: Unless otherwise noted: V_{DD} = 2.7V to 5.5V.

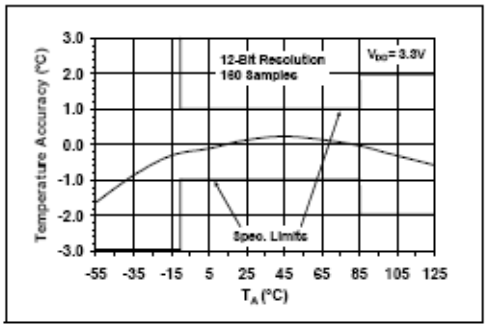


FIGURE 2-1: Average Temperature Accuracy vs. Ambient Temperature, V_{DD} = 3.3V.

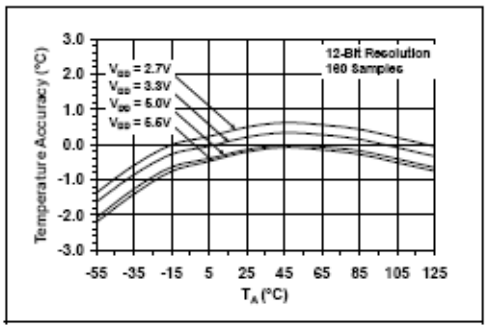


FIGURE 2-2: Average Temperature Accuracy vs. Ambient Temperature.

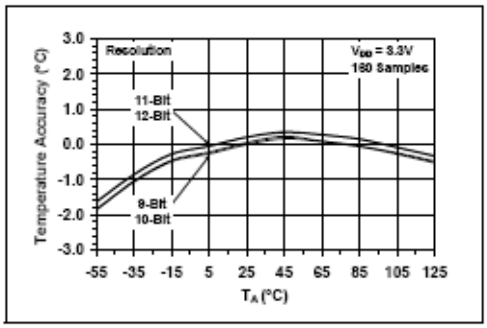


FIGURE 2-3: Average Temperature Accuracy vs. Ambient Temperature, V_{DD} = 3.3V.

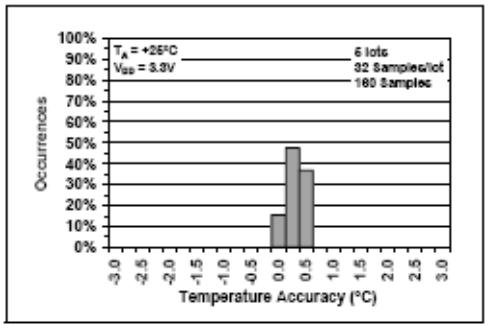


FIGURE 2-4: Temperature Accuracy Histogram, T_A = +25°C.

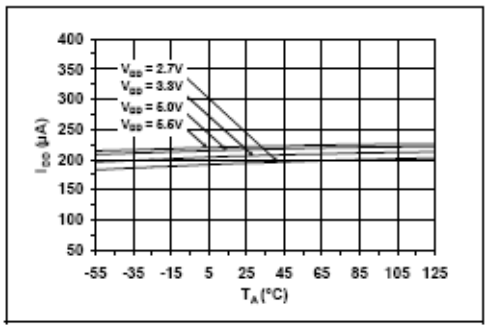


FIGURE 2-5: Supply Current vs. Ambient Temperature.

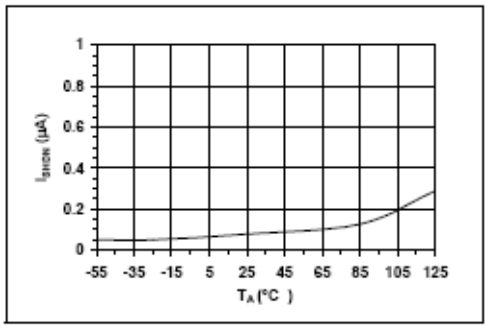


FIGURE 2-6: Shutdown Current vs. Ambient Temperature.

MCP9800/1/2/3

Note: Unless otherwise noted: V_{DD} = 2.7V to 5.5V.

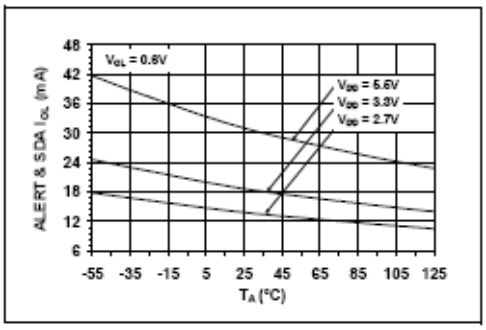


FIGURE 2-7: ALERT and SDA I_{OL} vs. Ambient Temperature.

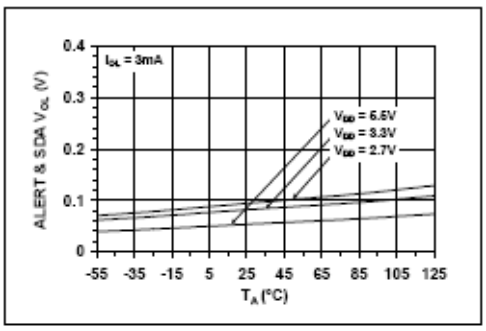


FIGURE 2-8: ALERT and SDA Output V_{OL} vs. Ambient Temperature.

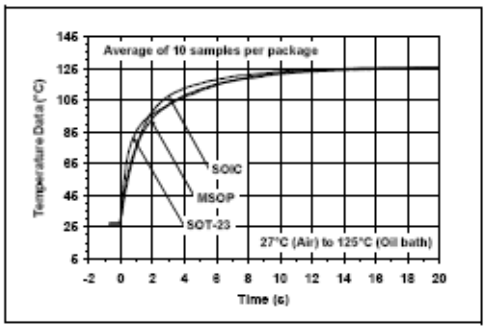


FIGURE 2-9: MCP980X Thermal Response vs Time.

3.0 PIN DESCRIPTION

The descriptions of the pins are listed in Table 3-1.

TABLE 3-1: PIN FUNCTION TABLE

MCP9800 MCP9802 SOT-23-5	MCP9801 MCP9803 MSOP, SOIC	Symbol	Function
5	1	SDA	Bidirectional Serial Data
4	2	SCLK	Serial Clock Input
3	3	ALERT	Temperature Alert Output
2	4	GND	Ground
—	5	A2	Address Select Pin (bit 2)
—	6	A1	Address Select Pin (bit 1)
—	7	A0	Address Select Pin (bit 0)
1	8	V_{DD}	Power Supply Input

3.1 Serial Data Pin (SDA)

The SDA is a bidirectional input/output pin, used to serially transmit data to and from the host controller. This pin requires a pull-up resistor to output data.

3.2 Serial Clock Pin (SCLK)

The SCLK is a clock input pin. All communication and timing is relative to the signal on this pin. The clock is generated by the host controller on the bus.

3.3 Power Supply Input (V_{DD})

The V_{DD} pin is the power pin. The operating voltage, as specified in the DC electrical specification table, is applied on this pin.

3.4 Ground (GND)

The GND pin is the system ground pin.

3.5 ALERT Output

The MCP9800/1/2/3's ALERT pin is an open-drain output pin. The device outputs an alert signal when the ambient temperature goes beyond the user-programmed temperature limit.

3.6 Address Pins (A2, A1, A0)

These pins are device or slave address input pins and are available only with the MCP9801/03. The device addresses for the MCP9800/02 are factory-set.

The address pins are the Least Significant bits (LSb) of the device address bits. The Most Significant bits (MSb) (A6, A5, A4, A3) are factory-set to <1001>. This is illustrated in Table 3-2.

TABLE 3-2: SLAVE ADDRESS

Device	A6	A5	A4	A3	A2	A1	A0
MCP9800/02A0	1	0	0	1	0	0	0
MCP9800/02A5	1	0	0	1	1	0	1
MCP9801/03	1	0	0	1	X	X	X

Note: User-selectable address is shown by X.

MCP9800/1/2/3

4.0 FUNCTIONAL DESCRIPTION

The MCP9800/1/2/3 family of temperature sensors consists of a band-gap type temperature sensor, a ΣΔ Analog-to-Digital Converter (ADC), user-programmable registers and a 2-wire I^2C/SMBus protocol compatible serial interface.

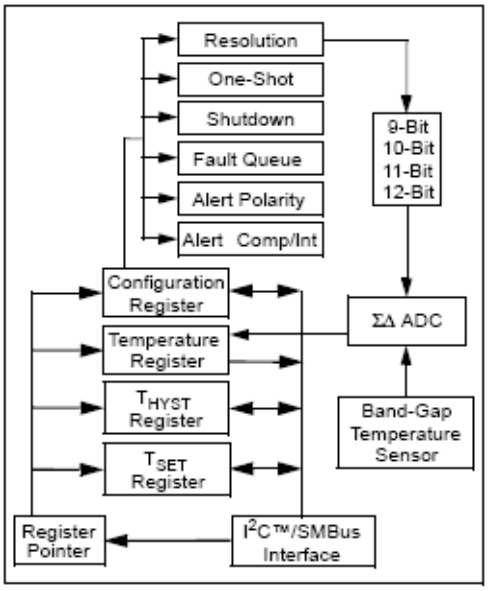


FIGURE 4-1: Functional Block Diagram.

4.1 Temperature Sensor

The MCP9800/1/2/3 uses the difference in the base-emitter voltage of a transistor while its collector current is changed from IC_1 to IC_2. With this method, the ΔV_{BE} depends only on the ratio of the two currents and the ambient temperature, as shown in Equation 4-1.

EQUATION 4-1:

$$\Delta V_{BE} = \left(\frac{kT}{q}\right) \times ln(IC_1/IC_2)$$

Where:

T = temperature in kelvin
ΔV_{BE} = change in diode base-emitter voltage
k = Boltzmann's constant
q = electron charge
IC_1 and IC_2 = currents with n:1 ratio

4.2 ΣΔ Analog-to-Digital Converter

A sigma-delta analog-to-digital converter is used to convert ΔV_{BE} to a digital word that corresponds to the transistor temperature. The converter has an adjustable resolution from 9-bits (at 30 ms conversion time) to 12-bits (at 240 ms conversion time). Thus, it allows the user to make trade-offs between resolution and conversion time. Refer to **Section 4.3.4 "Sensor Configuration Register (CONFIG)"** and **Section 4.3.4.7 "ΣΔ ADC Resolution"** for details.

MCP9800/1/2/3

4.3.1 AMBIENT TEMPERATURE REGISTER (T_A)

The MCP9800/1/2/3 has a 16-bit read-only ambient temperature register (T_A) that contains 9-bit to 12-bit temperature data. This data is formatted in two's complement. The bit assignments, as well as the corresponding resolution, is shown in the register assignment below.

The refresh rate of this register depends on the selected ADC resolution. It takes 30 ms (typ.) for 9-bit data and 240 ms (typ.) for 12-bit data. Since this register is double-buffered, the user can read the register while the MCP9800/1/2/3 performs analog-to-digital conversion in the background. The decimal code to ambient temperature conversion is shown in Equation 4-2:

EQUATION 4-2:

$$T_A = Code \times 2^n$$

Where:

n = -1, -2, -3 and -4 for 9-bit, 10-bit, 11-bit and 12-bit resolution, respectively

T_A = Ambient Temperature (°C)

Code = MCP980X output in decimal (Table 4-1)

REGISTER 4-2: AMBIENT TEMPERATURE REGISTER (T_A)

Upper Half:							
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
Sign	2^6 °C/bit	2^5 °C/bit	2^4 °C/bit	2^3 °C/bit	2^2 °C/bit	2^1 °C/bit	2^0 °C/bit
bit 15							bit 8

Lower Half:							
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
2^{-1} °C/bit	2^{-2} °C/bit	2^{-3} °C/bit	2^{-4} °C/bit	0	0	0	0
bit 7							bit 0

Note: When the 9-bit, 10-bit or 11-bit resolutions are selected, bit 6, bit 7 or bit 8 will remain clear <0>, respectively.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

TABLE 4-1: AMBIENT TEMPERATURE TO CODE CONVERSION

Ambient Temperature				Code			T_A (°C)
9-Bit	10-Bit	11-Bit	12-Bit	Binary	Hexadecimal	Decimal	
+125°C				0111 1101 0uuu uuuu(1)	0FA	250	+125
+25.4375°C				0001 1001 0uuu uuuu	032	50	+25
+0.5°C				0000 0000 1uuu uuuu	001	1	+0.5
	+125°C			0111 1101 00uu uuuu	1F4	500	+125
	+25.4375°C			0001 1001 01uu uuuu	065	101	+25.25
	+0.25°C			0000 0000 01uu uuuu	001	1	+0.25
		+125°C		0111 1101 000u uuuu	3E8	1000	+125
		+25.4375°C		0001 1001 011u uuuu	0CB	203	+25.375
		+0.125°C		0000 0000 001u uuuu	001	1	+0.125
			+125°C	0111 1101 0000 uuuu	7D0	2000	+125
			+25.4375°C	0001 1001 0111 uuuu	197	407	+25.4375
			+0.0625°C	0000 0000 0001 uuuu	001	1	+0.0625
0°C				0000 0000 0000 uuuu	000	0	0
			–0.0625°C	1111 1111 1111 uuuu(2)	001(3)	-1	-0.0625
			–25.4375°C	1110 0110 1001 uuuu	197	-407	-25.4375
			–55°C	1100 1001 0000 uuuu	370	-880	-55

Note 1: 'u' represents unused bits. The MCP9800/1/2/3 clears <0> the unused bits.

2: This data is in two's complement format, which indicates ambient temperature below 0°C.

3: Negative temperature magnitude in Hexadecimal. This conversion is done by complimenting each binary bit and adding 1.

4.3.2 TEMPERATURE LIMIT-SET REGISTER (T_{SET})

The MCP9800/1/2/3 has a 16-bit read/write Temperature Limit-Set register (T_{SET}) which contains a 9-bit data in two's compliment format. This data represents a maximum temperature limit. If the ambient temperature exceeds this specified limit, the MCP9800/1/2/3 asserts an alert output. (Refer to **Section 4.3.4.3 "ALERT Output Configuration"**).

This register uses the nine Most Significant bits (MSb) and all other bits are don't cares.

The power-up default value of T_{SET} register is 80°C <0 1010 0000> in binary.

REGISTER 4-3: TEMPERATURE LIMIT-SET REGISTER (T_{SET})

Upper Half:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
Sign	2^6 °C/bit	2^5 °C/bit	2^4 °C/bit	2^3 °C/bit	2^2 °C/bit	2^1 °C/bit	2^0 °C/bit
bit 15							bit 8

Lower Half:							
R/W-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
2^{-1} °C/bit	0	0	0	0	0	0	0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

MCP9800/1/2/3

4.3.3 TEMPERATURE HYSTERESIS REGISTER (T_{HYST})

The MCP9800/1/2/3 has a 16-bit read/write temperature hysteresis register (T_{HYST}) that contains a 9-bit data in two's compliment format. This register is used to set a hysteresis for the T_{SET} limit. Therefore, the data represents a minimum temperature limit. If the ambient temperature drifts below the specified limit, the MCP9800/1/2/3 asserts an alert output (refer to **Section 4.3.4.3 "ALERT Output Configuration"**).

This register uses the nine Most Significant bits (MSb) and all other bits are don't cares.

The power-up default value of T_{HYST} register is 75°C <0 1001 0110> in binary.

REGISTER 4-4: TEMPERATURE HYSTERESIS REGISTER (T_{HYST})

Upper Half:

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
Sign	2^6 °C/bit	2^5 °C/bit	2^4 °C/bit	2^3 °C/bit	2^2 °C/bit	2^1 °C/bit	2^0 °C/bit
bit 15							bit 8

Lower Half:

R/W-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
2^{-1} °C/bit	0	0	0	0	0	0	0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

4.3.4 SENSOR CONFIGURATION REGISTER (CONFIG)

The MCP9800/1/2/3 has an 8-bit read/write configuration register (CONFIG) that allows the user to select the different features. These features include shutdown, ALERT output select as comparator or interrupt output, ALERT output polarity, fault queue cycle, temperature measurement resolution and One-shot mode (single conversion while in shutdown). These functions are described in detail in the following sections.

REGISTER 4-5: CONFIGURATION REGISTER (CONFIG)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
One-Shot	Resolution		Fault Queue		ALERT Polarity	COMP/ INT	Shut-down
bit 7							bit 0

bit 7 **ONE-SHOT bit**
1 = Enabled
0 = Disabled (Power-up default)

bit 5-6 **ΣΔ ADC RESOLUTION bit**
00 = 9 bit (Power-up default)
01 = 10 bit
10 = 11 bit
11 = 12 bit

bit 3-4 **FAULT QUEUE bit**
00 = 1 (Power-up default)
01 = 2
10 = 4
11 = 6

bit 2 **ALERT POLARITY bit**
1 = Active-High
0 = Active-Low (Power-up default)

bit 1 **COMP/INT bit**
1 = Interrupt Mode
0 = Comparator Mode (Power-up default)

bit 0 **SHUTDOWN bit**
1 = Enable
0 = Disable (Power-up default)

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

4.3.4.1 Shutdown Mode

The Shutdown mode disables all power-consuming activities (including temperature sampling operations) while leaving the serial interface active. The device consumes 1 µA (max.) in this mode. It remains in this mode until the configuration register is updated to enable continuous conversion or until power is recycled.

In Shutdown mode, the CONFIG, T_A, T_{SET} and T_{HYST} registers can be read or written. However, the serial bus activity will increase the shutdown current.

4.3.4.2 One-Shot Mode

The MCP9800/1/2/3 can also be used in a One-shot mode that can be selected using bit 7 of the CONFIG register. The One-shot mode performs a single temperature measurement and returns to Shutdown mode. This mode is especially useful for low-power applications where temperature is measured upon command from a controller. For example, a 9-bit T_A in One-shot mode consumes 200 µA (typ.) for 30 ms and 0.1 µA (typ.) during shutdown.

To access this feature, the device needs to initially be in Shutdown mode. This is done by sending a byte to the CONFIG register with bit 0 set <1> and bit 7 cleared <0>. Once the device is in Shutdown mode, CONFIG needs to be written again with bit 0 and bit 7 set <1>. This begins the single conversion cycle of 30 ms for 9-bit data. Once the conversion is completed, T_A is updated and bit 7 of CONFIG becomes cleared <0> by the MCP9800/1/2/3.

TABLE 4-6: SHUTDOWN AND ONE-SHOT MODE DESCRIPTION

Operational Mode	One-Shot (Bit 7)	Shutdown (Bit 0)
Continuous Conversion	0	0
Shutdown	0	1
Continuous Conversion (One-shot is ignored)	1	0
One-Shot	1	1

Note: The shutdown command <01> needs to be programmed before sending a one-shot command <11>.

4.3.4.3 ALERT Output Configuration

The ALERT output can be configured as either a comparator output or as Interrupt Output mode using bit 1 of CONFIG. The polarity can also be specified as an active-high or active-low, using bit 2 of CONFIG. The following sections describe each Output mode and Figure 4-2 shows graphical description.

4.3.4.4 Comparator Mode

In the Comparator mode, the ALERT output is asserted when T_A is greater than T_{SET}. The pin remains active until T_A is lower than T_{HYST}. The Comparator mode is useful for thermostat-type applications such as turning on a cooling fan or triggering a system shutdown when the temperature exceeds a safe operating range.

In the Comparator mode, if the device enters the Shutdown mode with asserted ALERT output, the output remains active during shutdown. The device must be operating in continuous conversion, with T_A below T_{HYST}, for the ALERT output to be deasserted.

4.3.4.5 Interrupt Mode

In the Interrupt mode, the ALERT output is asserted when T_A is greater than T_{SET}. However, the output is deasserted when the user performs a read from any register. This mode is designed for interrupt driven microcontroller based systems. The microcontroller receiving the interrupt will have to acknowledge the interrupt by reading any register from the MCP9800/1/2/3. This will clear the interrupt and the ALERT pin will become deasserted. When T_A drifts below T_{HYST}, the MCP9800/1/2/3 outputs another interrupt and the controller needs to read a register to deassert the ALERT output. Shutting down the device will also reset or deassert the ALERT output.

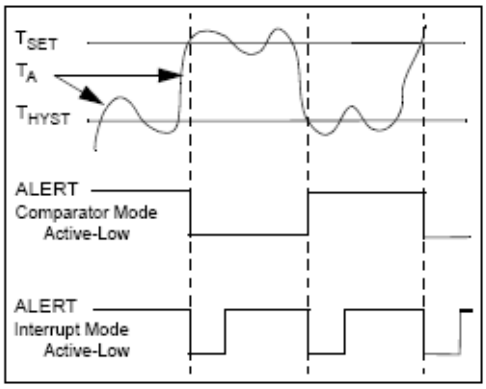


FIGURE 4-2: Alert Output.

4.3.4.6 Fault Queue

The fault queue feature can be used as a filter to lessen the probability of spurious activation of the ALERT pin. T_A must remain above T_{SET} for the consecutive number of conversion cycles selected using the Fault Queue bits. Bit 3 and bit 4 of CONFIG can be used to select up to six fault queue cycles. For example, if six fault queues are selected, T_A must be greater than T_{SET} for six consecutive conversions before ALERT is asserted as a comparator or an interrupt output.

This queue setting also applies for T_{HYST}. T_A must remain below T_{HYST} for six consecutive conversions before ALERT is deasserted (comparator mode) or before another interrupt is asserted (interrupt mode).

4.3.4.7 ΣΔ ADC Resolution

The MCP9800/1/2/3 provides access to select the ADC resolution from 9-bit to 12-bit using bit 6 and bit 5 of the CONFIG register. The user can gain better insight into the trends and characteristics of the ambient temperature by using a finer resolution. Increasing the resolution also reduces the quantization error. Figure 2-4 shows accuracy versus resolution.

Table 4-1 shows the T_A register conversion time for the corresponding resolution.

TABLE 4-1: RESOLUTION AND CONVERSION TIME

Bits	Resolution °C/Bit (typ.)	Conversion time t_{CONV} ms (typ.)
9	0.5	30
10	0.25	60
11	0.125	120
12	0.0625	240

4.4 Summary of Power-up Default

The MCP9800/1/2/3 has an internal Power-on Reset (POR) circuit. If the power supply voltage V_{DD} glitches down to the 1.7V (typ.) threshold, the device resets the registers to the power-up default settings.

Table 4-2 shows the power-up default summary.

TABLE 4-2: POWER-UP DEFAULTS

Register	Data (Hex)	Power-up Defaults
T_A	0000	0°C
T_{SET}	A000	80°C
T_{HYST}	9600	75°C
Pointer	00	Temperature register
CONFIG	00	Continuous Conversion Comparator mode Active-Low Output Fault Queue 1 9-bit Resolution

MCP9800/1/2/3

5.0 SERIAL COMMUNICATION

5.1 2-Wire I²C/SMBus Compatible Interface

The MCP9800/1/2/3 serial clock input (SCLK) and the bidirectional serial data line (SDA) form a 2-Wire bidirectional serial port for communication.

The following bus protocol has been defined:

TABLE 5-1: MCP980X SERIAL BUS CONVENTIONS

Term	Description
Transmitter	Device sending data to the bus
Receiver	Device receiving data from the bus
Master	The device that controls the serial bus, typically a microcontroller
Slave	The device addressed by the master, such as the MCP9800/1/2/3
START	A unique signal from master to initiate serial interface with a slave
STOP	A unique signal from the master to terminate serial interface from a slave
Read/Write	A read or write to the MCP9800/1/2/3 registers
ACK	A receiver Acknowledges (ACK) the reception of each byte by polling the bus
NAK	A receiver Not-Acknowledges (NAK) or releases the bus to show End-of-Data (EOD)
Busy	Communication is not possible because the bus is in use
Not Busy	The bus is in the idle state, both SDA and SCLK remain high
Data Valid	SDA must remain stable before SCLK becomes high in order for a data bit to be considered valid. During normal data transfers, SDA only changes state while SCLK is low

5.1.1 DATA TRANSFER

Data transfers are initiated by a start condition (START), followed by a 7-bit device address and a 1-bit read/write. Acknowledge (ACK) from slave confirms the reception of each byte. Each access must be terminated by a stop condition (STOP).

Data transfer may be initiated when the bus is in IDLE.

5.1.2 MASTER/SLAVE

The bus is controlled by a master device (typically a microcontroller) that controls the bus access and generates the start and stop conditions. The MCP9800/1/2/3 is a slave device and does not control other devices in the bus. Both master and slave devices can operate as either transmitter or receiver. However, the master device determines which mode is activated.

5.1.3 START/STOP CONDITION

A high-to-low transition of the SDA line (while SCLK is high) is the start condition. All data transfers must be preceded by a start condition from the master. If a start condition is generated during data transfer, the MCP9800/1/2/3 resets and accepts the new start condition.

A low-to-high transition of the SDA line (while SCLK is high) is the stop condition. All data transfers must be ended by a stop condition from the master. If a stop condition is introduced during data transmission, the MCP9800/1/2/3 releases the bus.

5.1.4 ADDRESS BYTE

Following the start condition, the host must transmit the address byte to the MCP9800/1/2/3. The 7-bit address for the MCP9800/02A0 and MCP9800/02A5 is <1001000> and <1001101> in binary, respectively. The address for the MCP9802/03 is <1001,A2,A1,A0> in binary, where the A0, A1 and A2 bits are set externally by connecting the corresponding pins to V_{DD} <1> or GND <0>. The 7-bit address transmitted in the serial bit stream must match the selected address for the MCP9800/1/2/3 to respond with an ACK.

Bit 8 in the address byte is a read/write bit. Setting this bit to '1' commands a read operation, while '0' commands a write operation.

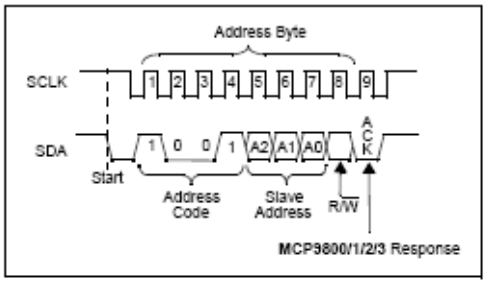


FIGURE 5-1: Device Addressing.

5.1.5 DATA VALID

After the start condition, each bit of data in transmission needs to be settled for time specified by $t_{SU-DATA}$ before SCLK toggles from low-to-high (refer to the Serial Interface Timing Specification).

5.1.6 ACKNOWLEDGE (ACK)

Each receiving device, when addressed, is obliged to generate an acknowledge bit after the reception of each byte. The master device must generate an extra clock pulse for ACK to be recognized.

The acknowledging device has to pull down the SDA line for $t_{SU-DATA}$ before the low-to-high transition of SCLK from the Master and remains pulled down for t_{H-DATA} after high-to-low transition of SCLK.

During read, the master must signal an End-of-Data (EOD) to the slave by not generating an ACK bit once the last bit has been clocked out of the slave. In this case, the slave will leave the data line released to enable the master to generate the stop condition.

5.1.7 TIME OUT (MCP9802/03)

If the SCLK stays low for time specified by t_{OUT}, the MCP9802/03 resets the serial interface. This dictates the minimum clock speed as specified in the SMBus specification. The I^2C bus specification does not limit clock speed and, therefore, the master can hold the clock indefinitely to process data (MCP9800/01 only).

MCP9800/1/2/3

5.2 Graphical Representation of the MCP9800/1/2/3 Serial Protocols

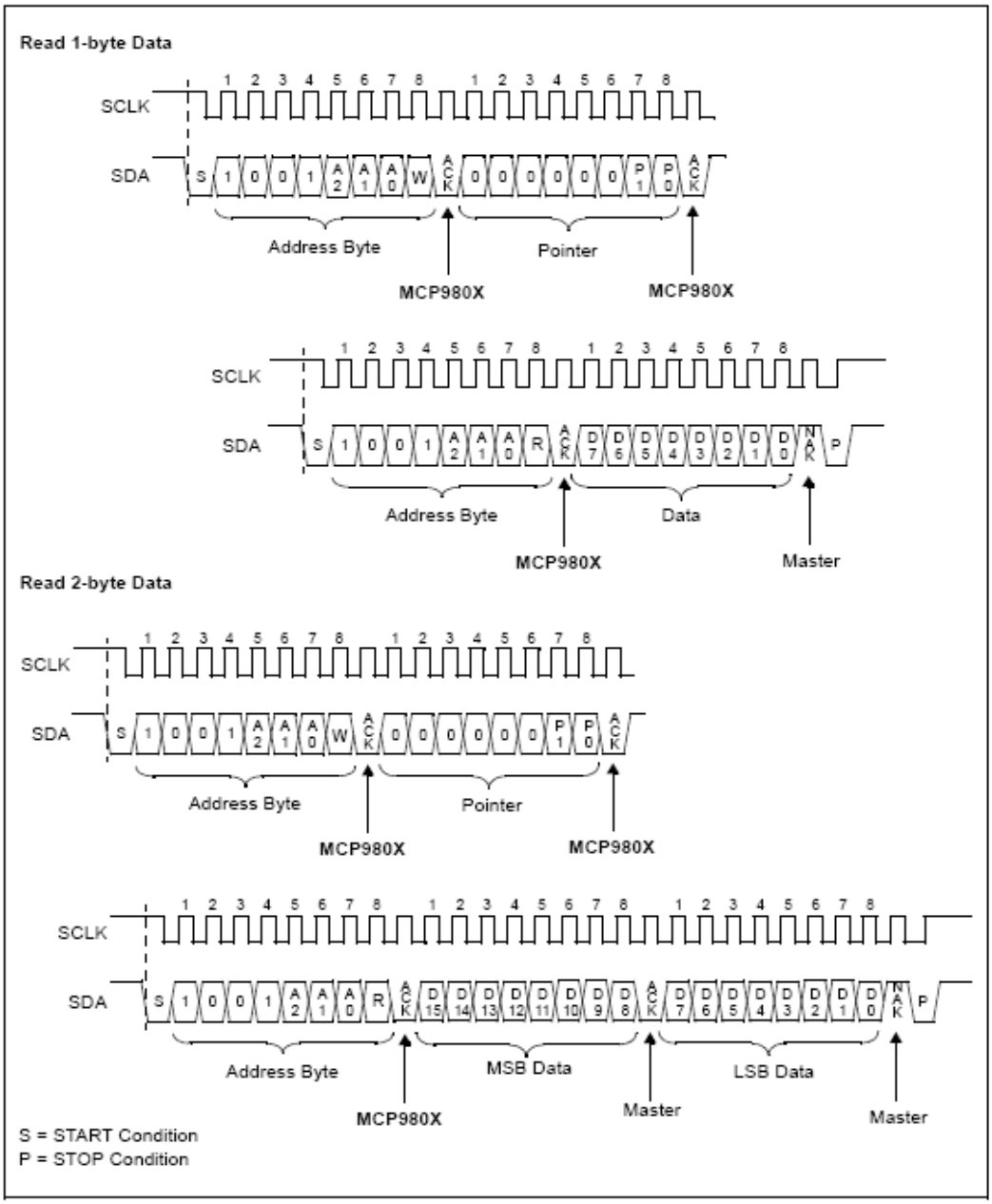


FIGURE 5-2: Read 1-byte and 2-byte data from a Register.

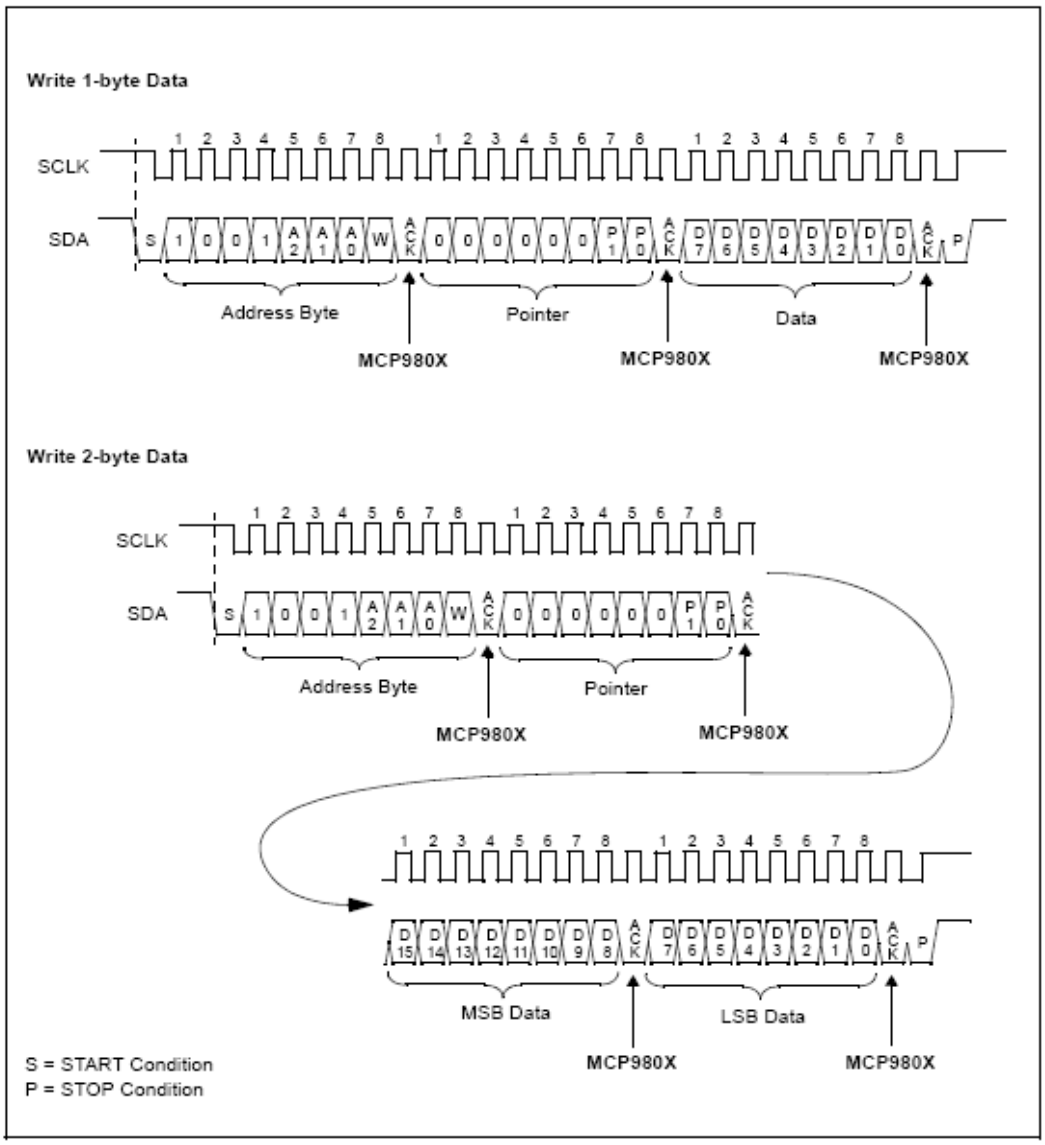


FIGURE 5-3: Write 1-byte and 2-byte data from a Register.

MCP9800/1/2/3

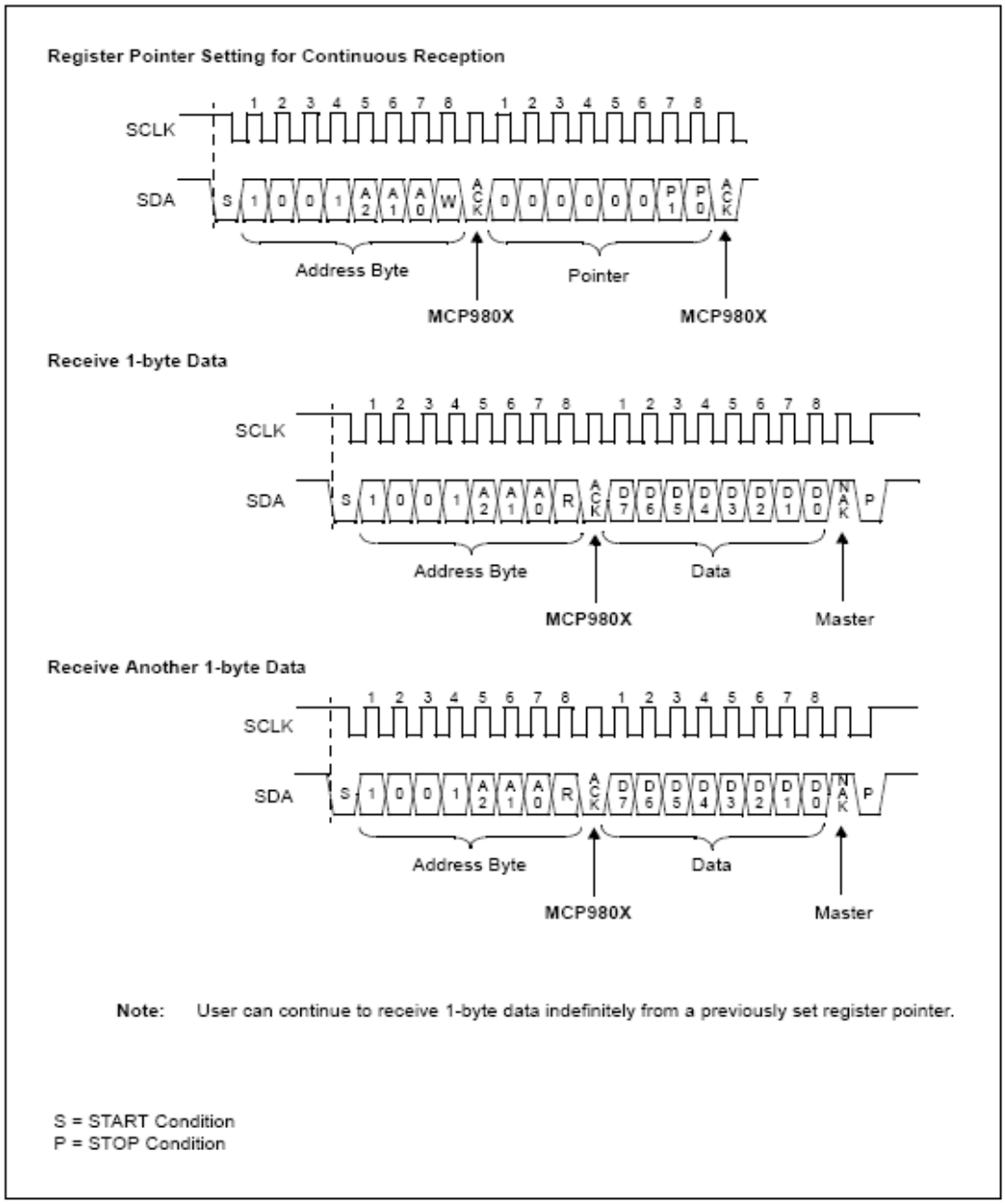


FIGURE 5-4: Receive 1-byte data from previously set pointer.

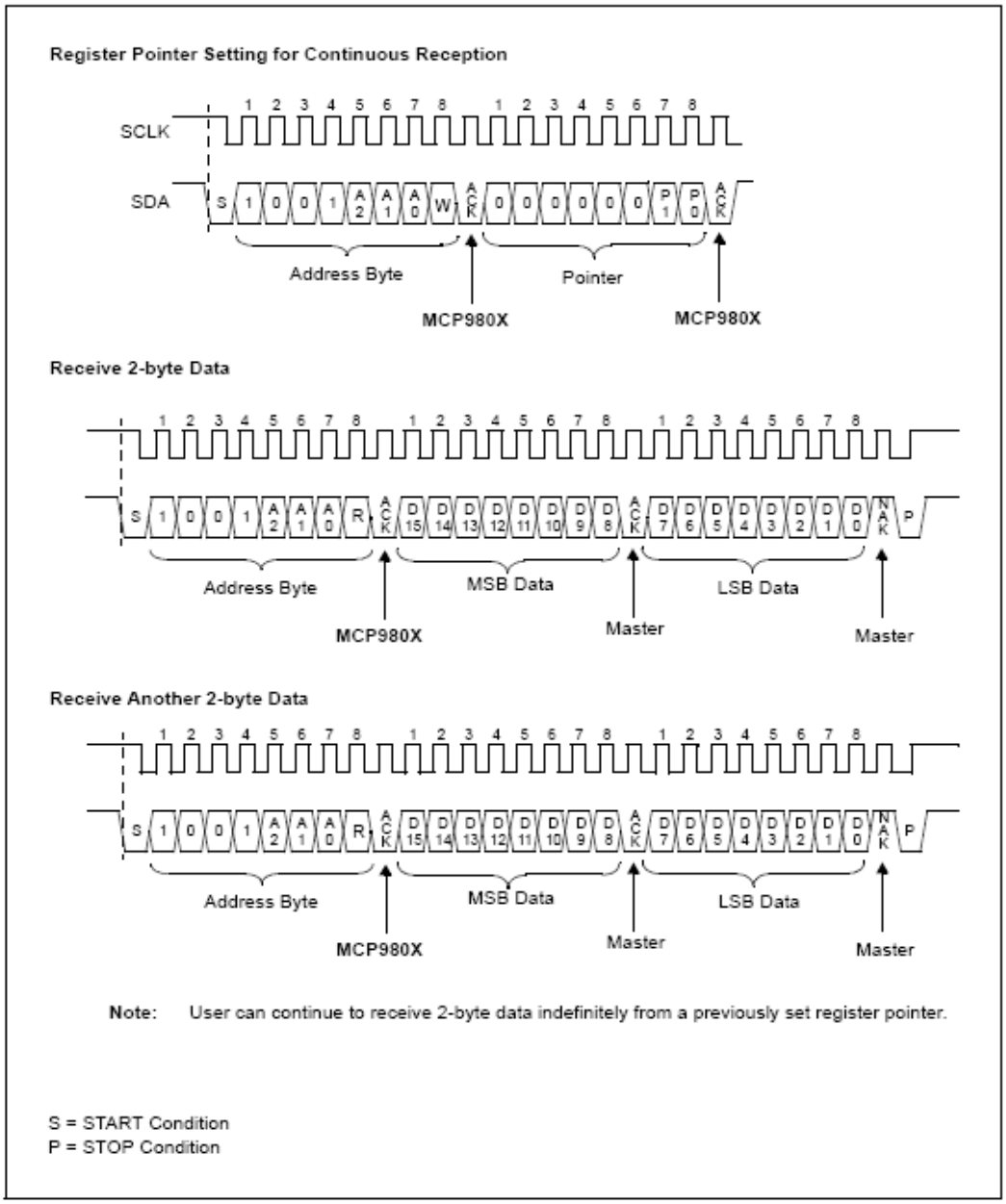


FIGURE 5-5: Receive 2-byte data from previously set pointer.

MCP9800/1/2/3

6.0 APPLICATIONS INFORMATION

6.1 Connecting to the Serial Bus

The SDA and SCLK serial interface are open-drain pins that require pull-up resistors. This configuration is shown in Figure 6-1.

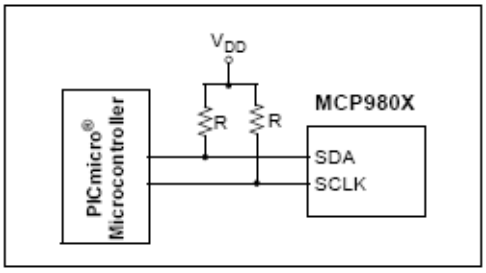


FIGURE 6-1: Pull-up Resistors On Serial Interface.

For the SMBus protocol, the number of devices connected to the bus are limited only by the maximum rise and fall times of the SDA and SCLK lines. Unlike the I^2C specifications, SMBus does not specify a maximum bus capacitance value. Rather, it specifies 350 µA (max.) current through the pull-up resistor. Therefore, the value of the pull-up resistors will vary depending on the system's supply voltage (V_{DD}). The pull-up resistor values for a 5V system ranges 14.3 kΩ to 50 kΩ. Minimizing bus capacitance is still very important, as it directly affects the rise and fall times of the SDA and SCLK lines.

Although SMBus specifications only require the SDA and SCLK lines to pull down 350 µA (max.) with 0.4V (max.) voltage drop, the MCP9800/1/2/3 is designed to meet 0.4V (max.) voltage drop at 3 mA of current. This allows the MCP9800/1/2/3 to drive lower values of pull-up resistors and higher bus capacitance. In this application, all devices on the bus must meet the same pull-down current requirements.

6.2 Typical Application

Microchip provides several microcontroller product lines with Master Synchronous Serial Port Modules (MSSP) that include I^2C interface mode. This module implements all master and slave functions and simplifies the firmware development overhead. Figure 6-2 shows a typical application using the PIC16F737 as a master to control other Microchip slave products, such as EEPROM, fan speed controllers and the MCP980X temperature sensor connected to the bus.

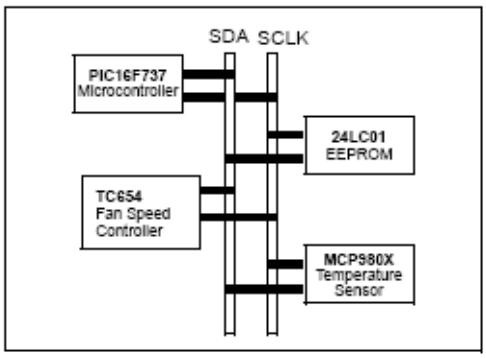


FIGURE 6-2: Multiple Devices on SMBus.

The ALERT output can be wire-ORed with a number of other open-drain devices. In such applications, the output needs to be programmed as an active-low output. Most systems will require pull-up resistors for this configuration.

6.3 Layout Considerations

The MCP9800/1/2/3 does not require any additional components besides the Master controller in order to measure temperature. However, it is recommended that a decoupling capacitor of 0.1 µF to 1 µF be used between the V_{DD} and GND pins. A high-frequency ceramic capacitor is recommended. It is necessary for the capacitor to be located as close as possible to the power pins in order to provide effective noise protection.

6.4 Thermal Considerations

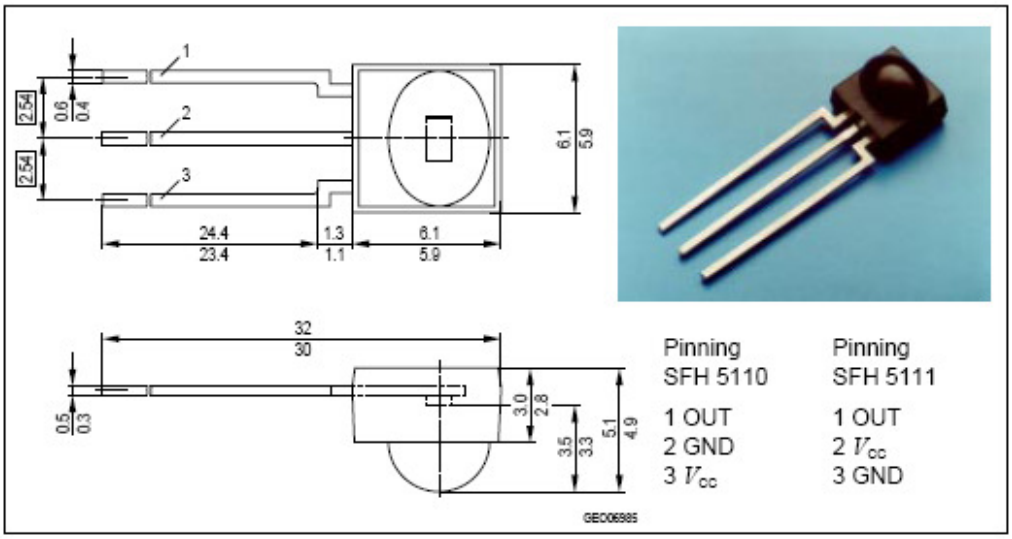
The MCP9800/1/2/3 measures temperature by monitoring the voltage of a diode located in the die. A low impedance thermal path between the die and the Printed Circuit Board (PCB) is provided by the pins. Therefore, the MCP9800/1/2/3 effectively monitors the temperature of the PCB. However, the thermal path for the ambient air is not as efficient because the plastic device package functions as a thermal insulator.

A potential for self-heating errors can exist if the MCP9800/1/2/3 SDA and SCLK communication lines are heavily loaded with pull-ups. Typically, the self-heating error is negligible because of the relatively small current consumption of the MCP9800/1/2/3. However, in order to maximize the temperature accuracy, the SDA and SCLK pins need to be lightly loaded.

SIEMENS

IR-Empfänger für Fernbedienungen **SFH 5110**
IR-Receiver for Remote Control Systems **SFH 5111**

Vorläufige Daten / Preliminary Data



Maße in mm, wenn nicht anders angegeben/Dimensions in mm, unless otherwise specified.

Beschreibung

SFH 5110 und SFH 5111 sind Infrarot-Empfänger für die Erkennung von Signalen aus Infrarot-Fernbedienungssystemen und bestehen aus Fotodiode, Vorverstärker, automatischer Verstärkungsregelung, Bandpaß-Filter und Demodulator. Das schwarz eingefärbte Gehäuse dient zur Unterdrückung des Tageslichteinflusses.

Description

SFH 5110 and SFH 5111 are IR receivers to detect light from infrared remote control systems. The IC includes photodiode, preamplifier, automatic gain control, bandpass and demodulator. The black-colored package is designed as daylight-cutoff filter.

Typ Type	Trägerfrequ. Carrier Frequency kHz	Bestellnr. Ordering Code	Typ Type	Trägerfrequ. Carrier Frequency kHz	Bestellnr. Ordering Code
SFH 5110-30	30	Q62702-P5088	SFH 5110-38	38	Q62702-P5091
SFH 5110-33	33	Q62702-P5089	SFH 5110-40	40	Q62702-P5092
SFH 5110-36	36	Q62702-P5090	SFH 5111	on request	

SIEMENS

SFH 5110
SFH 5111

Wesentliche Merkmale

- IC mit monolithisch integrierter Fotodiode (Ein-Chip Lösung)
- Speziell geeignet für Anwendungen von 940 … 950 nm (IR Filter)
- Hohe Empfindlichkeit
- Variable Bandpaß-Filterfrequenz
- TTL und CMOS kompatibel
- Ausgang: aktiv „Low"
- Keine externe Beschaltung nötig

Features

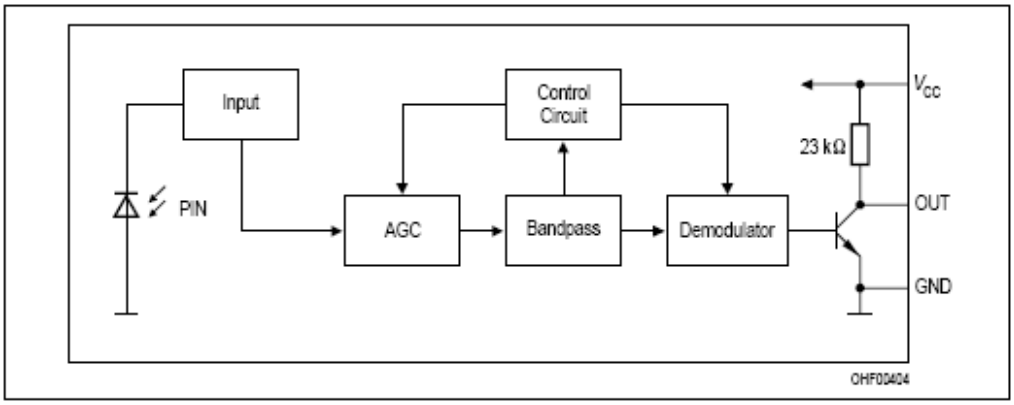
- IC with monolithic integrated photodiode (single chip solution)
- Especially suitable for applications of 940 … 950 nm (IR filter)
- High sensitivity
- Various bandpass filter frequency
- TTL and CMOS compatibility
- Output: active Low
- No external components necessary

Anwendungen

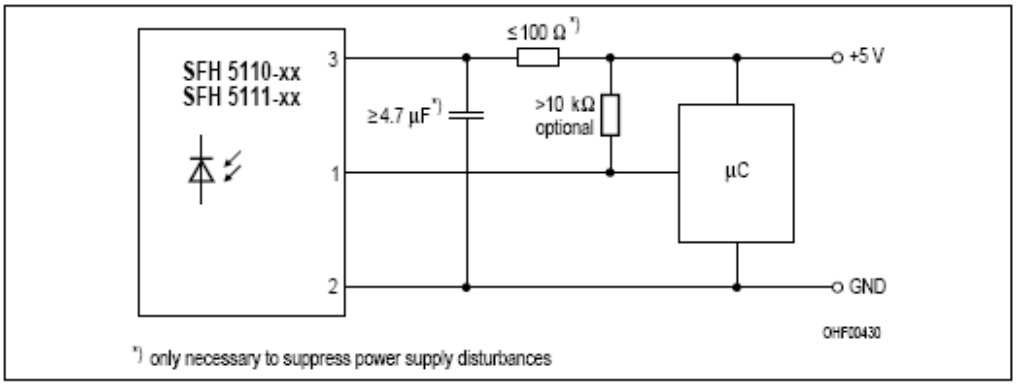
- Empfänger in Fernbedienungen für TV, Videorekorder, HiFi, Satellitenempfänger und CD-Spieler
- Optischer Schalter

Applications

- Remote control module for TV sets, VCRs, hi-fi audio receivers, SAT receivers and compact disk players
- Optical Switch



Blockschaltbild
Block Diagram



Externe Beschaltung
External Circuit

SIEMENS

SFH 5110
SFH 5111

Grenzwerte (T_A = 25 °C)
Maximum Ratings

Bezeichnung Description	Symbol Symbol	Wert Value	Einheit Unit
Betriebs- und Lagertemperatur Operation and storage temperature range	T_{op} T_{stg}	– 10 ... + 75 – 30 ... + 100	°C
Betriebsspannung Supply voltage	V_S	6.3	V
Betriebsstrom Supply current	I_{CC}	5	mA
Ausgangsspannung Output voltage	V_{OUT}	6.3	V
Ausgangsstrom Output current	I_{OUT}	3	mA
Verlustleistung Total power dissipation $T_A \leq 85$ °C	P_{tot}	50	mW

Kennwerte (T_A = 25 °C)
Characteristics

Bezeichnung Description	Symbol Symbol	Wert Value			Einheit Unit
		min.	typ.	max.	
Betriebsspannung Supply voltage	V_S	4.5	5.0	5.5	V
Stromaufnahme, V_{CC} = 5 V, E = 0 Current consumption	I_{CC}	–	1.3	–	mA
Wellenlänge der max. Fotoempfindlichkeit Wavelength of max. sensitivity	$\lambda_{S\,max}$	–	940	–	nm
Spektraler Bereich der Fotoempfindlichkeit Spectral range of sensitivity	λ	830	–	1100	nm
Ausgangsspannung Output voltage Output "High" - (I_Q = 10 µA) Output "Low" - (I_Q = 500 µA)	 $V_{OUT\,high}$ $V_{OUT\,low}$	 V_S – 0.5 –	 – –	 – 0.5	V
Trägerfrequenz Carrier frequency	f_0	– – – – –	30 33 36 38 40	– – – – –	kHz

Kennwerte (T_A = 25 °C)
Characteristics (cont'd)

Bezeichnung Description	Symbol Symbol	Wert Value			Einheit Unit
		min.	typ.	max.	
Bestrahlungsstärke (Testsignal, s. Figure 1) Threshold irradiance (test signal, see Figure 1) $f = f_o$, $t_{p,I}$ = 600 µs	$E_{e\,min}$ $E_{e\,max}$	– 20	0.35 –	– –	mW/m² W/m²
Reichweite Transmission distance SFH4510/SFH4515, I_F = 500 mA	d	–	30	–	m
Eingangspulsbreite „ON" (Testsignal, s. Figure 1) Input pulse width "ON" (test signal, see Figure 1)	$t_{p,I}$	$6/f_o$	–	$35/f_o$	µs
Ausgangspulsbreite „ON" (Testsignal, s. Figure 1) Output pulse width "ON" (test signal, see Figure 1, E_e = 1 mW/m²)	$t_{p,O}$	$t_{p,I}$ $- 6/f_o$	–	$t_{p,I}$ $+ 6/f_o$	µs
50%-Filterbandbreite, $f = f_o$, E = 0, V_S = 5 V 50%-Filter bandwidth	$\Delta f_{50\%}$	3	–	6	kHz

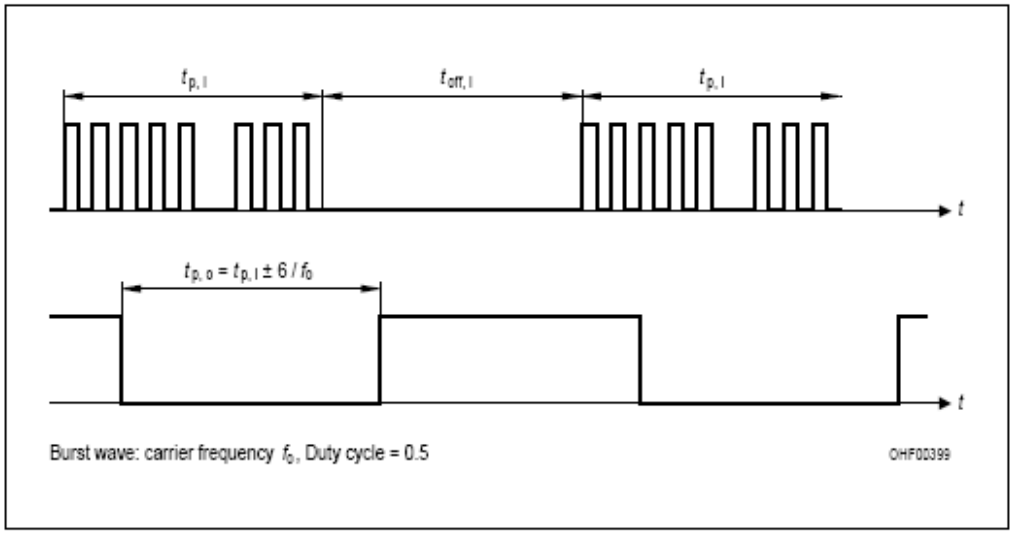
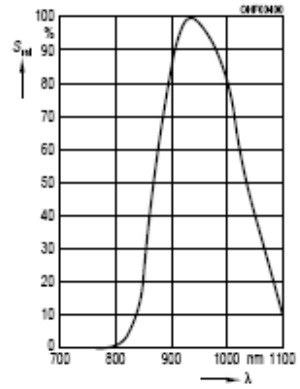
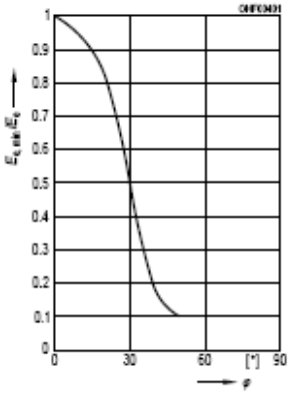


Figure 1 Optisches Testsignal
Optical Test Signal

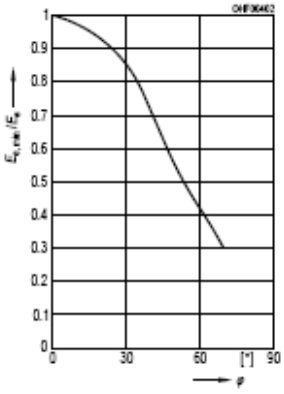
Relative luminous sensitivity
$S_{rel} = f(\lambda)$



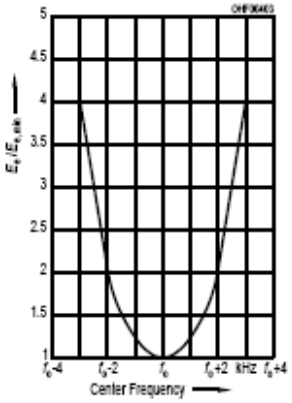
Vertical directivity φ_y



Horizontal directivity φ_x



Relative sensitivity $E_e/E_{e,\,min} = f(f_0)$



INTEGRATED CIRCUITS

DATA SHEET

PCF8583
Clock/calendar with 240 × 8-bit RAM

Product specification
Supersedes data of 1997 Mar 28
File under Integrated Circuits, IC12

1997 Jul 15

Philips Semiconductors

PHILIPS

CONTENTS

Clock/calendar with 240 × 8-bit RAM PCF8583

1 FEATURES

- I²C-bus interface operating supply voltage: 2.5 V to 6 V
- Clock operating supply voltage (0 to +70 °C): 1.0 V to 6.0 V
- 240 × 8-bit low-voltage RAM
- Data retention voltage: 1.0 V to 6 V
- Operating current (at f_{SCL} = 0 Hz): max. 50 µA
- Clock function with four year calendar
- Universal timer with alarm and overflow indication
- 24 or 12 hour format
- 32.768 kHz or 50 Hz time base
- Serial input/output bus (I²C)
- Automatic word address incrementing
- Programmable alarm, timer and interrupt function
- Slave address:
  - READ: A1 or A3
  - WRITE: A0 or A2.

2 GENERAL DESCRIPTION

The PCF8583 is a clock/calendar circuit based on a 2048-bit static CMOS RAM organized as 256 words by 8 bits. Addresses and data are transferred serially via the two-line bidirectional I²C-bus. The built-in word address register is incremented automatically after each written or read data byte. Address pin A0 is used for programming the hardware address, allowing the connection of two devices to the bus without additional hardware.

The built-in 32.768 kHz oscillator circuit and the first 8 bytes of the RAM are used for the clock/calendar and counter functions. The next 8 bytes may be programmed as alarm registers or used as free RAM space. The remaining 240 bytes are free RAM locations.

3 QUICK REFERENCE DATA

SYMBOL	PARAMETER	CONDITION	MIN.	TYP.	MAX.	UNIT
V_{DD}	supply voltage operating mode	I²C-bus active	2.5	–	6.0	V
		I²C-bus inactive	1.0	–	6.0	V
I_{DD}	supply current operating mode	f_{SCL} = 100 kHz	–	–	200	µA
I_{DDO}	supply current clock mode	f_{SCL} = 0 Hz; V_{DD} = 5 V	–	10	50	µA
		f_{SCL} = 0 Hz; V_{DD} = 1 V	–	2	10	µA
T_{amb}	operating ambient temperature range		–40	–	+85	°C
T_{stg}	storage temperature range		–65	–	+150	°C

4 ORDERING INFORMATION

TYPE NUMBER	PACKAGE		
	NAME	DESCRIPTION	VERSION
PCF8583P	DIP8	plastic dual in-line package; 8 leads (300 mil)	SOT97-1
PCF8583T	SO8	plastic small outline package; 8 leads; body width 7.5 mm	SOT176-1

5 BLOCK DIAGRAM

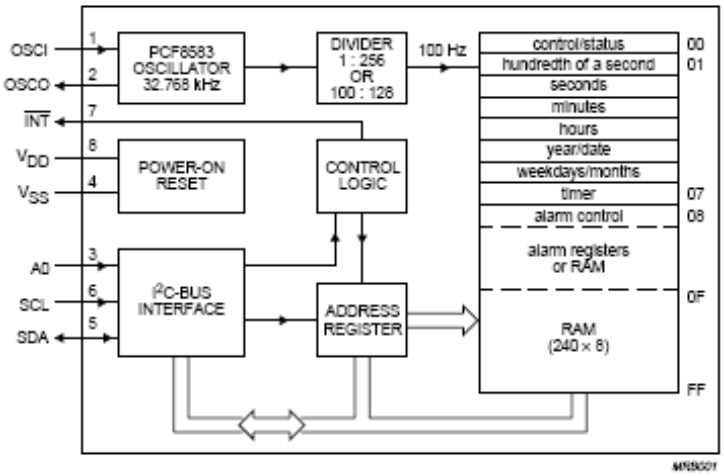


Fig.1 Block diagram.

6 PINNING

SYMBOL	PIN	DESCRIPTION
OSCI	1	oscillator input, 50 Hz or event-pulse input
OSCO	2	oscillator output
A0	3	address input
V_{SS}	4	negative supply
SDA	5	serial data line
SCL	6	serial clock line
$\overline{INT}$	7	open drain interrupt output (active LOW)
V_{DD}	8	positive supply

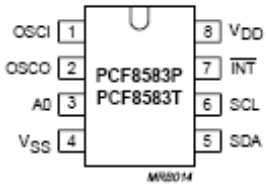


Fig.2 Pinning diagram.

7 FUNCTIONAL DESCRIPTION

The PCF8583 contains a 256 by 8-bit RAM with an 8-bit auto-increment address register, an on-chip 32.768 kHz oscillator circuit, a frequency divider, a serial two-line bidirectional I^2C-bus interface and a power-on reset circuit.

The first 16 bytes of the RAM (memory addresses 00 to 0F) are designed as addressable 8-bit parallel special function registers. The first register (memory address 00) is used as a control/status register. The memory addresses 01 to 07 are used as counters for the clock function. The memory addresses 08 to 0F may be programmed as alarm registers or used as free RAM locations, when the alarm is disabled.

7.1 Counter function modes

When the control/status register is programmed, a 32.768 kHz clock mode, a 50 Hz clock mode or an event-counter mode can be selected.

In the clock modes the hundredths of a second, seconds, minutes, hours, date, month (four year calendar) and weekday are stored in a BCD format. The timer register stores up to 99 days. The event counter mode is used to count pulses applied to the oscillator input (OSCO left open-circuit). The event counter stores up to 6 digits of data.

When one of the counters is read (memory locations 01 to 07), the contents of all counters are strobed into capture latches at the beginning of a read cycle. Therefore, faulty reading of the count during a carry condition is prevented.

When a counter is written, other counters are not affected.

7.2 Alarm function modes

By setting the alarm enable bit of the control/status register the alarm control register (address 08) is activated.

By setting the alarm control register a dated alarm, a daily alarm, a weekday alarm or a timer alarm may be programmed. In the clock modes, the timer register (address 07) may be programmed to count hundredths of a second, seconds, minutes, hours or days. Days are counted when an alarm is not programmed.

Whenever an alarm event occurs the alarm flag of the control/status register is set. A timer alarm event will set the alarm flag and an overflow condition of the timer will set the timer flag. The open drain interrupt output is switched on (active LOW) when the alarm or timer flag is set (enabled). The flags remain set until directly reset by a write operation.

When the alarm is disabled (Bit 2 of control/status register = 0) the alarm registers at addresses 08 to 0F may be used as free RAM.

7.3 Control/status register

The control/status register is defined as the memory location 00 with free access for reading and writing via the I^2C-bus. All functions and options are controlled by the contents of the control/status register (see Fig.3).

7.4 Counter registers

In the clock modes 24 h or 12 h format can be selected by setting the most significant bit of the hours counter register. The format of the hours counter is shown in Fig.5.

The year and date are packed into memory location 05 (see Fig.6). The weekdays and months are packed into memory location 06 (see Fig.7). When reading these memory locations the year and weekdays are masked out when the mask flag of the control/status register is set. This allows the user to read the date and month count directly.

In the event-counter mode events are stored in BCD format. D5 is the most significant and D0 the least significant digit. The divider is by-passed.

In the different modes the counter registers are programmed and arranged as shown in Fig.4. Counter cycles are listed in Table 1.

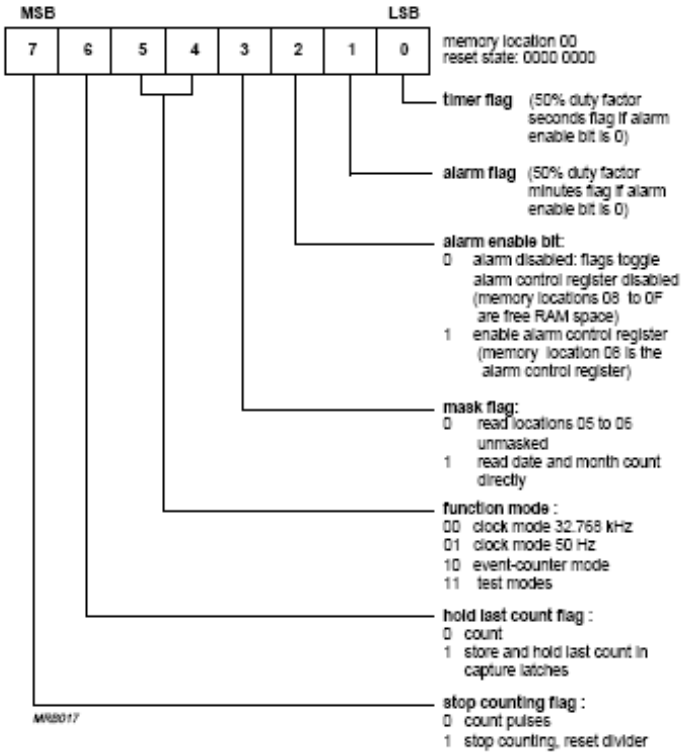


Fig.3 Control/status register.

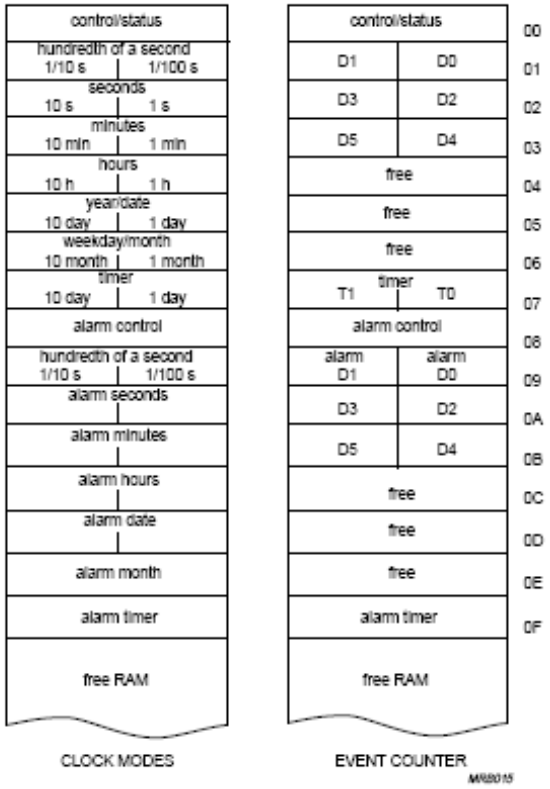


Fig.4 Register arrangement.

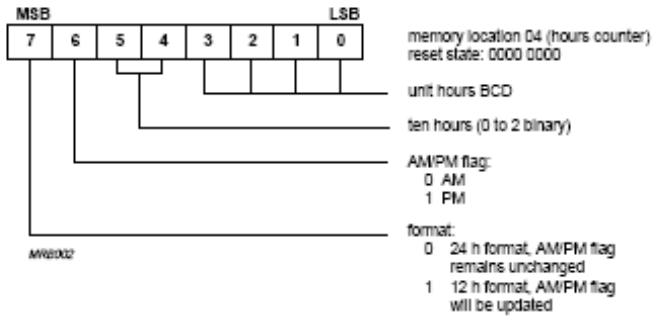


Fig.5 Format of the hours counter.

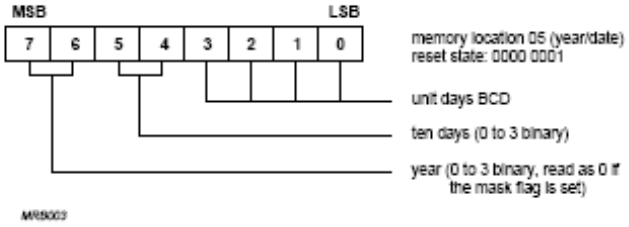


Fig.6 Format of the year/date counter.

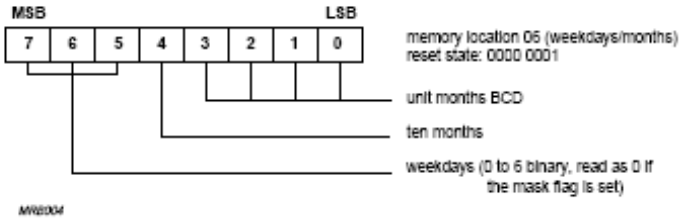


Fig.7 Format of the weekdays/month counter.

Table 1 Cycle length of the time counters, clock modes

UNIT	COUNTING CYCLE	CARRY TO NEXT UNIT	CONTENTS OF THE MONTH COUNTER
Hundredths of a second	00 to 99	99 to 00	–
Seconds	00 to 59	59 to 00	–
Minutes	00 to 59	59 to 00	–
Hours (24 h)	00 to 23	23 to 00	–
Hours (12 h)	12 AM	–	–
	01 AM to 11 AM	–	–
	12 PM	–	–
	01 PM to 11 PM	11 PM to 12 AM	–
Date	01 to 31	31 to 01	1, 3, 5, 7, 8, 10 and 12
	01 to 30	30 to 01	4, 6, 9 and 11
	01 to 29	29 to 01	2, year = 0
	01 to 28	28 to 01	2, year = 1, 2 and 3
Months	01 to 12	12 to 01	–
Year	0 to 3	–	–
Weekdays	0 to 6	6 to 0	–
Timer	00 to 99	no carry	–

7.5 Alarm control register

When the alarm enable bit of the control/status register is set (address 00, bit 2) the alarm control register (address 08) is activated. All alarm, timer, and interrupt output functions are controlled by the contents of the alarm control register (see Fig.8).

7.6 Alarm registers

All alarm registers are allocated with a constant address offset of hexadecimal 08 to the corresponding counter registers (see Fig.4, Register arrangement).

An alarm signal is generated when the contents of the alarm registers matches bit-by-bit the contents of the involved counter registers. The year and weekday bits are ignored in a dated alarm. A daily alarm ignores the month and date bits. When a weekday alarm is selected, the contents of the alarm weekday/month register will select the weekdays on which an alarm is activated (see Fig.9).

Remark: In the 12 h mode, bits 6 and 7 of the alarm hours register must be the same as the hours counter.

Fig.8 Alarm control register; clock mode.

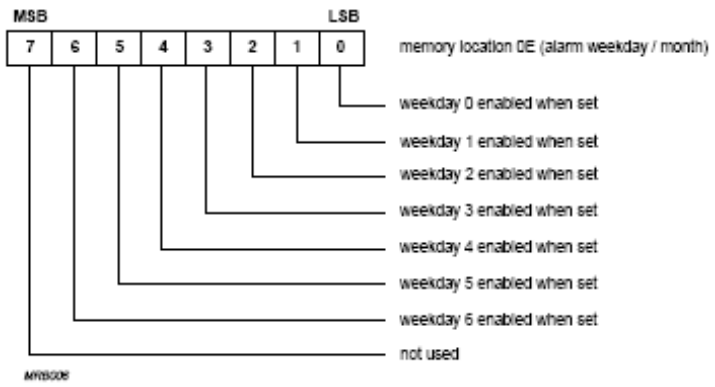


Fig.9 Selection of alarm weekdays.

7.7 Timer

The timer (location 07) is enabled by setting the control/status register = XX0X X1XX. The timer counts up from 0 (or a programmed value) to 99. On overflow, the timer resets to 0. The timer flag (LSB of control/status register) is set on overflow of the timer. This flag must be reset by software. The inverted value of this flag can be transferred to the external interrupt by setting bit 3 of the alarm control register.

Additionally, a timer alarm can be programmed by setting the timer alarm enable (bit 6 of the alarm control register). When the value of the timer equals a pre-programmed value in the alarm timer register (location 0F), the alarm flag is set (bit 1 of the control/status register). The inverted value of the alarm flag can be transferred to the external interrupt by enabling the alarm interrupt (bit 6 of the alarm control register).

Resolution of the timer is programmed via the 3 LSBs of the alarm control register (see Fig.11, Alarm and timer Interrupt logic diagram).

7.8 Event counter mode

Event counter mode is selected by bits 4 and 5 which are logic 1, 0 in the control/status register. The event counter mode is used to count pulses externally applied to the oscillator input (OSCO left open-circuit).

The event counter stores up to 6 digits of data, which are stored as 6 hexadecimal values located in locations 1, 2, and 3. Thus, up to 1 million events may be recorded.

An event counter alarm occurs when the event counter registers match the value programmed in locations 9, A, and B, and the event alarm is enabled (bits 4 and 5 which are logic 0, 1 in the alarm control register). In this event, the alarm flag (bit 1 of the control/status register) is set. The inverted value of this flag can be transferred to the interrupt pin (pin 7) by setting the alarm interrupt enable in the alarm control register. In this mode, the timer (location 07) increments once for every one, one-hundred, ten thousand, or 1 million events, depending on the value programmed in bits 0, 1 and 2 of the alarm control register. In all other events, the timer functions are as in the clock mode.

7.9 Interrupt output

The conditions for activating the open-drain n-channel interrupt output $\overline{INT}$ (active LOW) are determined by appropriate programming of the alarm control register. These conditions are clock alarm, timer alarm, timer overflow, and event counter alarm. An interrupt occurs when the alarm flag or the timer flag is set, and the corresponding interrupt is enabled. In all events, the interrupt is cleared only by software resetting of the flag which initiated the interrupt.

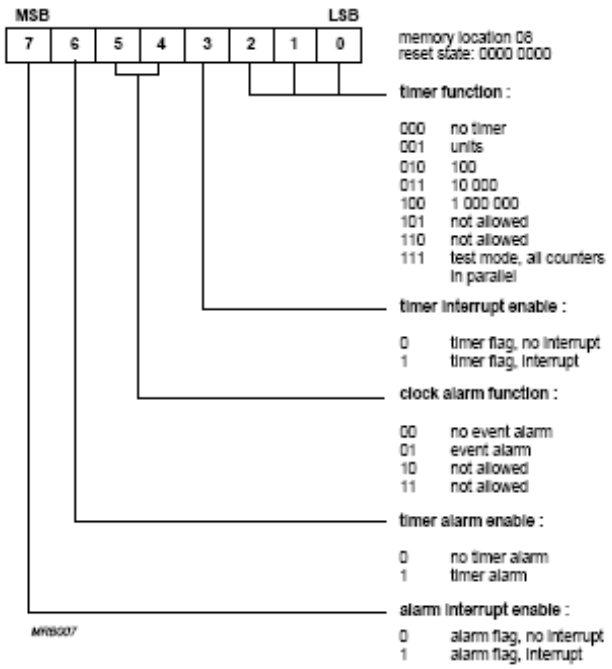


Fig.10 Alarm control register, event-counter mode.

In the clock mode, if the alarm enable is not activated (alarm enable bit of control/status register is logic 0), the interrupt output toggles at 1 Hz with a 50% duty cycle (may be used for calibration). This is the default power-on state of the device. The OFF voltage of the interrupt output may exceed the supply voltage, up to a maximum of 6.0 V. A logic diagram of the interrupt output is shown in Fig.11.

7.10 Oscillator and divider

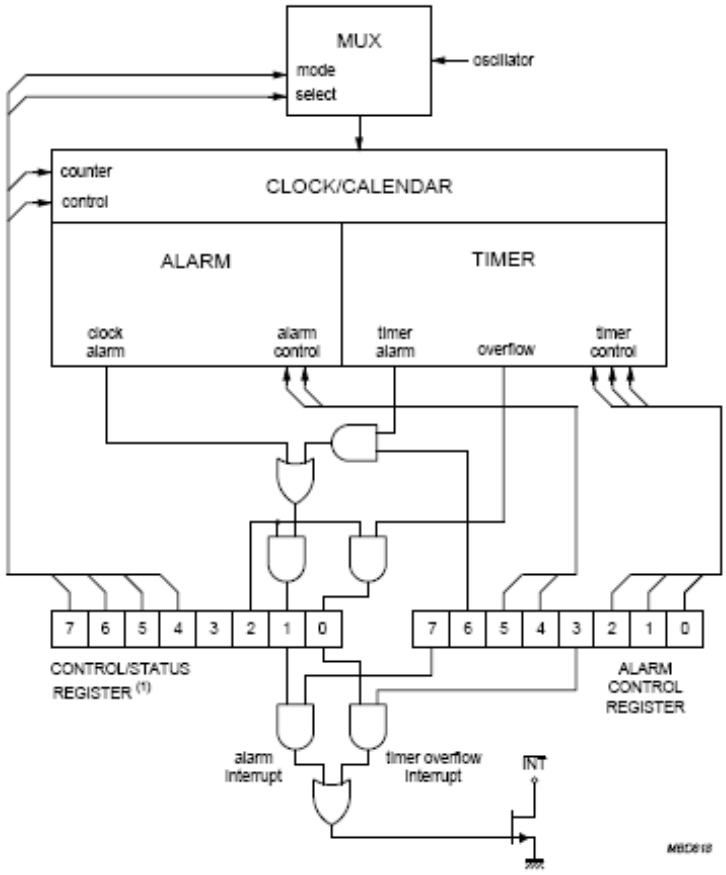
A 32.768 kHz quartz crystal has to be connected to OSCI (pin 1) and OSCO (pin 2). A trimmer capacitor between OSCI and V_{DD} is used for tuning the oscillator (see quartz frequency adjustment). A 100 Hz clock signal is derived from the quartz oscillator for the clock counters.

In the 50 Hz clock mode or event-counter mode the oscillator is disabled and the oscillator input is switched to a high impedance state. This allows the user to feed the 50 Hz reference frequency or an external high speed event signal into the input OSCI.

7.11 Initialization

When power-up occurs the I^2C-bus interface, the control/status register and all clock counters are reset. The device starts time-keeping in the 32.768 kHz clock mode with the 24 h format on the first of January at 0.00.00: 00. A 1 Hz square wave with 50% duty cycle appears at the interrupt output pin (starts HIGH).

It is recommended to set the stop counting flag of the control/status register before loading the actual time into the counters. Loading of illegal states may lead to a temporary clock malfunction.



(1) If the alarm enable bit of the control/status register is reset (logic 0), a 1 Hz signal can be observed on the interrupt pin $\overline{INT}$.

Fig.11 Alarm and timer interrupt logic diagram.

8 CHARACTERISTICS OF THE I²C-BUS

The I²C-bus is for bidirectional, two-line communication between different ICs or modules. The two lines are a serial data line (SDA) and a serial clock line (SCL). Both lines must be connected to a positive supply via a pull-up resistor. Data transfer may be initiated only when the bus is not busy.

8.1 Bit transfer (see Fig.12)

One data bit is transferred during each clock pulse. The data on the SDA line must remain stable during the HIGH period of the clock pulse as changes in the data line at this time will be interpreted as a control signal.

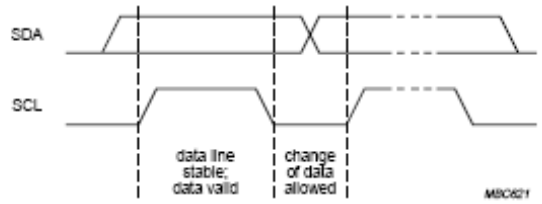


Fig.12 Bit transfer.

8.2 Start and stop conditions (see Fig.13)

Both data and clock lines remain HIGH when the bus is not busy. A HIGH-to-LOW transition of the data line, while the clock is HIGH is defined as the start condition (S). A LOW-to-HIGH transition of the data line while the clock is HIGH is defined as the stop condition (P).

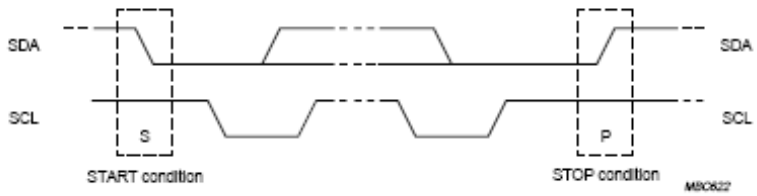


Fig.13 Definition of start and stop conditions.

8.3 System configuration (see Fig.14)

A device generating a message is a 'transmitter', a device receiving a message is the 'receiver'. The device that controls the message is the 'master' and the devices which are controlled by the master are the 'slaves'.

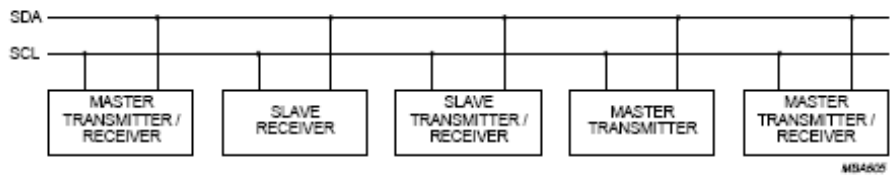


Fig.14 System configuration.

8.4 Acknowledge (see Fig.15)

The number of data bytes transferred between the start and stop conditions from transmitter to receiver is unlimited. Each byte of eight bits is followed by an acknowledge bit. The acknowledge bit is a HIGH level signal put on the bus by the transmitter during which time the master generates an extra acknowledge related clock pulse. A slave receiver which is addressed must generate an acknowledge after the reception of each byte. Also a master receiver must generate an acknowledge after the reception of each byte that has been clocked out of the slave transmitter.

The device that acknowledges must pull down the SDA line during the acknowledge clock pulse, so that the SDA line is stable LOW during the HIGH period of the acknowledge related clock pulse (set-up and hold times must be taken into consideration). A master receiver must signal an end of data to the transmitter by not generating an acknowledge on the last byte that has been clocked out of the slave. In this event the transmitter must leave the data line HIGH to enable the master to generate a stop condition.

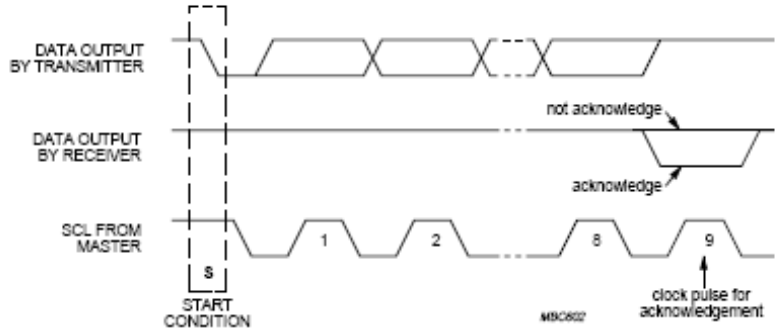


Fig.15 Acknowledgment on the I^2C-bus.

9 I²C-BUS PROTOCOL

9.1 Addressing

Before any data is transmitted on the I²C-bus, the device which should respond is addressed first. The addressing is always carried out with the first byte transmitted after the start procedure.

The clock/calendar acts as a slave receiver or slave transmitter. Therefore the clock signal SCL is only an input signal, but the data signal SDA is a bidirectional line.

The clock/calendar slave address is shown in Fig.16. Bit A0 corresponds to hardware address pin A0. Connecting this pin to V_{DD} or V_{SS} allows the device to have one of two different addresses.

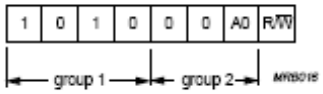


Fig.16 Slave address.

9.2 Clock/calendar READ/WRITE cycles

The I²C-bus configuration for the different PCF8583 READ and WRITE cycles is shown in Figs 17, 18 and 19.

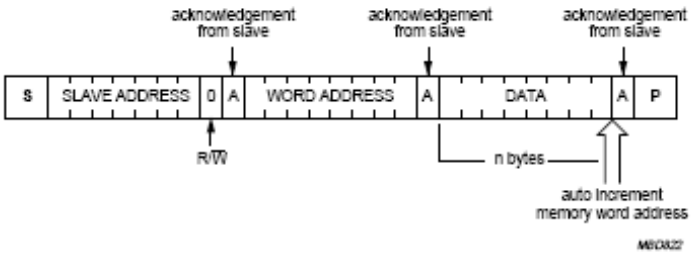


Fig.17 Master transmits to slave receiver (WRITE) mode.

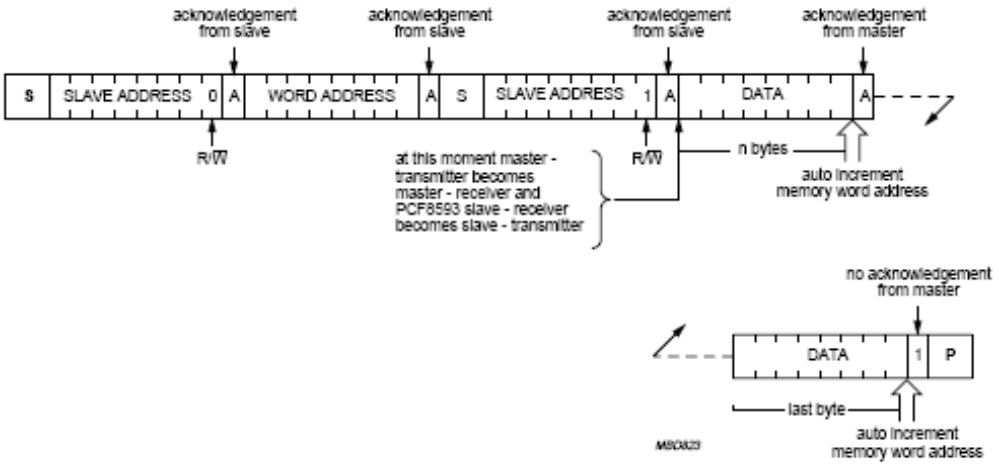


Fig.18 Master reads after setting word address (write word address; READ data).

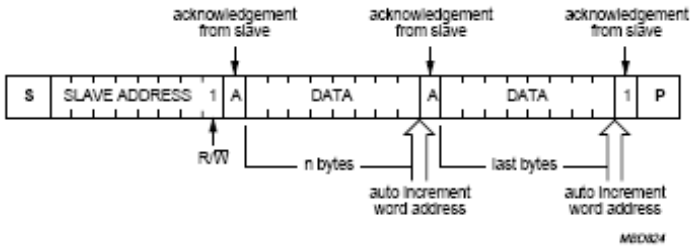


Fig.19 Master reads slave immediately after first byte (READ mode).

10 LIMITING VALUES

In accordance with the Absolute Maximum Rating System (IEC 134).

SYMBOL	PARAMETER	MIN.	MAX.	UNIT
V_{DD}	supply voltage (pin 8)	–0.8	+7.0	V
I_{DD}	supply current (pin 8)	–	50	mA
I_{SS}	supply current (pin 4)	–	50	mA
V_I	input voltage	–0.8	V_{DD} + 0.8	V
I_I	DC input current	–	10	mA
I_O	DC output current	–	10	mA
P_{tot}	total power dissipation per package	–	300	mW
P_O	power dissipation per output	–	50	mW
T_{amb}	operating ambient temperature	–40	+85	°C
T_{stg}	storage temperature	–65	+150	°C

11 HANDLING

Inputs and outputs are protected against electrostatic charge in normal handling. However, to be totally safe, it is desirable to take normal precautions appropriate to handling MOS devices. Advice can be found in Data Handbook IC12 under *"Handling MOS Devices"*.

12 DC CHARACTERISTICS

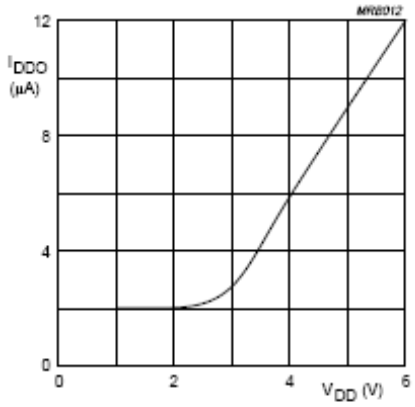
V_{DD} = 2.5 to 6.0 V; V_{SS} = 0 V; T_{amb} = –40 to +85 °C unless otherwise specified.

SYMBOL	PARAMETER	CONDITIONS	MIN.	TYP.(1)	MAX.	UNIT
V_{DD}	supply voltage (operating mode)	I^2C-bus active	2.5	–	6.0	V
		I^2C-bus inactive	1.0	–	6.0	V
V_{DDosc}	supply voltage (quartz oscillator)	T_{amb} = 0 to 70 °C; note 2	1.0	–	6.0	V
I_{DD}	supply current (operating mode)	f_{SCL} = 100 kHz; clock mode; note 3	–	–	200	µA
I_{DDO}	supply current (clock mode)	see Fig.20				
		f_{SCL} = 0 Hz; V_{DD} = 5 V	–	10	50	µA
		f_{SCL} = 0 Hz; V_{DD} = 1 V	–	2	10	µA
I_{DDR}	data retention	f_{OSCI} = 0 Hz; V_{DD} = 1 V				
		T_{amb} = –40 to + 85 °C	–	–	5	µA
		T_{amb} = –25 to + 70 °C	–	–	2	µA
V_{EN}	I^2C-bus enable level	note 4	1.5	1.9	2.3	V
SDA						
V_{IL}	LOW level input voltage	note 5	–0.8	–	$0.3V_{DD}$	V
V_{IH}	HIGH level input voltage	note 5	$0.7V_{DD}$	–	V_{DD} +0.8	V
I_{OL}	LOW level output current	V_{OL} = 0.4 V	3	–	–	mA
I_{LI}	input leakage current	V_I = V_{DD} or V_{SS}	–1	–	+1	µA
C_I	input capacitance	note 6	–	–	7	pF

SYMBOL	PARAMETER	CONDITIONS	MIN.	TYP.(1)	MAX.	UNIT
A0; OSCI						
I_{LI}	input leakage current	$V_I = V_{DD}$ or V_{SS}	–250	–	+250	nA
$\overline{INT}$						
I_{OL}	LOW level output current	V_{OL} = 0.4 V	3	–	–	mA
I_{LI}	input leakage current	$V_I = V_{DD}$ or V_{SS}	–1	–	+1	µA
SCL						
C_I	input capacitance	note 6	–	–	7	pF
I_{LI}	input leakage current	$V_I = V_{DD}$ or V_{SS}	–1	–	+1	µA

Notes

1. Typical values measured at T_{amb} = 25 °C.
2. When powering-up the device, V_{DD} must exceed 1.5 V until stable operation of the oscillator is established.
3. Event counter mode: supply current dependant upon input frequency.
4. The I^2C-bus logic is disabled if $V_{DD} < V_{EN}$.
5. When the voltages are above or below the supply voltages V_{DD} or V_{SS}, an input current may flow; this current must not exceed ±0.5 mA.
6. Tested on sample basis.



f_{SCL} = 32 kHz; T_{amb} = 25 °C.

Fig.20 Typical supply current in clock mode as a function of supply voltage.

13 AC CHARACTERISTICS

V_{DD} = 2.5 to 6.0 V; V_{SS} = 0 V; T_{amb} = –40 to +85 °C; unless otherwise specified.

SYMBOL	PARAMETER	CONDITIONS	MIN.	TYP.	MAX.	UNIT
Oscillator						
C_{osc}	integrated oscillator capacitance		–	40	–	pF
Δf_{osc}	oscillator stability	for ΔV_{DD} = 100 mV; T_{amb} = 25 °C; V_{DD} = 1.5 V	–	2×10^{-7}	–	
f_I	input frequency	note 1	–	–	1	MHz
Quartz crystal parameters (f = 32.768 kHz)						
R_S	series resistance		–	–	40	kΩ
C_L	parallel load capacitance		–	10	–	pF
C_T	trimmer capacitance		5	–	25	pF
I^2C-bus timing (see Fig.21; notes 2 and 3)						
f_{SCL}	SCL clock frequency		–	–	100	kHz
t_{SP}	tolerable spike width on bus		–	–	100	ns
t_{BUF}	bus free time		4.7	–	–	µs
$t_{SU;STA}$	START condition set-up time		4.7	–	–	µs
$t_{HD;STA}$	START condition hold time		4.0	–	–	µs
t_{LOW}	SCL LOW time		4.7	–	–	µs
t_{HIGH}	SCL HIGH time		4.0	–	–	µs
t_r	SCL and SDA rise time		–	–	1.0	µs
t_f	SCL and SDA fall time		–	–	0.3	µs
$t_{SU;DAT}$	data set-up time		250	–	–	ns
$t_{HD;DAT}$	data hold time		0	–	–	ns
$t_{VD;DAT}$	SCL LOW to data out valid		–	–	3.4	µs
$t_{SU;STO}$	STOP condition set-up time		4.0	–	–	µs

Notes

1. Event counter mode only.
2. All timing values are valid within the operating supply voltage and ambient temperature range and reference to V_{IL} and V_{IH} with an input voltage swing of V_{SS} to V_{DD}.
3. A detailed description of the I^2C-bus specification, with applications, is given in brochure *"The I^2C-bus and how to use it"*. This brochure may be ordered using the code 9398 393 40011.

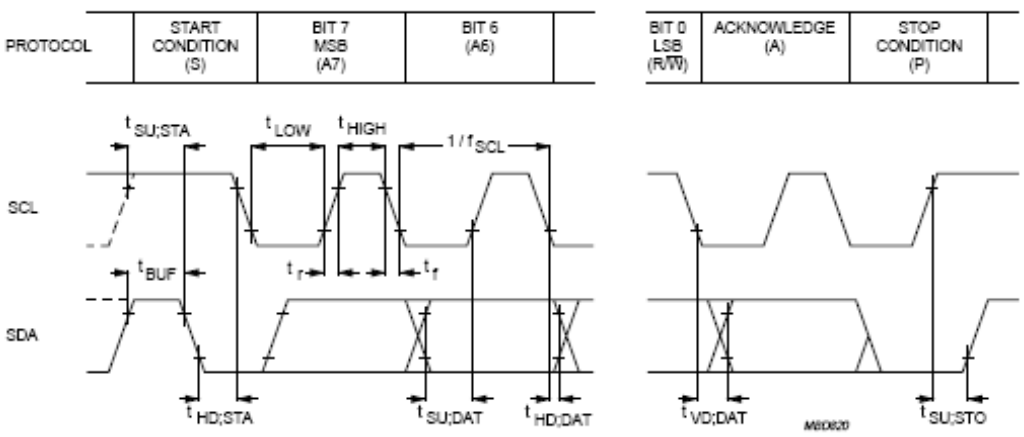


Fig.21 I^2C-bus timing diagram; rise and fall times refer to V_{IL} and V_{IH}.

14 APPLICATION INFORMATION

14.1 Quartz frequency adjustment

14.1.1 METHOD 1: FIXED OSCI CAPACITOR

By evaluating the average capacitance necessary for the application layout a fixed capacitor can be used.
The frequency is best measured via the 1 Hz signal available after power-on at the interrupt output (pin 7).
The frequency tolerance depends on the quartz crystal tolerance, the capacitor tolerance and the device-to-device tolerance (on average $\pm 5 \times 10^{-6}$).
Average deviations of ±5 minutes per year can be achieved.

14.1.2 METHOD 2: OSCI TRIMMER

Using the alarm function (via the I^2C-bus) a signal faster than 1 Hz can be generated at the interrupt output for fast setting of a trimmer.

Procedure:

- Power-on
- Initialization (alarm functions).

Routine:

- Set clock to time T and set alarm to time T + dT
- At time T + dT (interrupt) repeat routine.

14.1.3 METHOD 3:

Direct measurement of OSC out (accounting for test probe capacitance).

The PCF8583 slave address has a fixed combination 1010 as group 1.

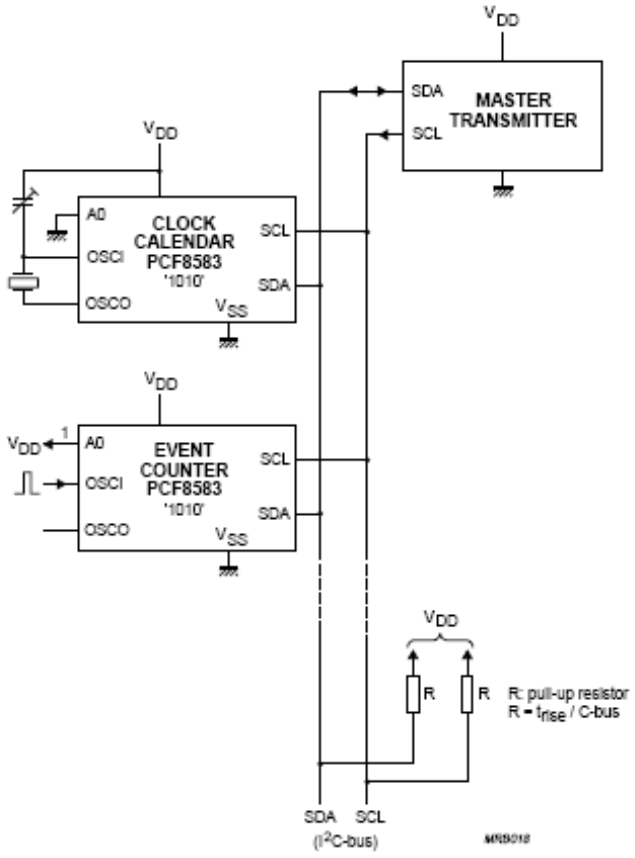


Fig.22 Application diagram.

Features

- High-performance, Low-power AVR® 8-bit Microcontroller
- Advanced RISC Architecture
  - 130 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
- Nonvolatile Program and Data Memories
  - 8K Bytes of In-System Self-Programmable Flash
    Endurance: 10,000 Write/Erase Cycles
  - Optional Boot Code Section with Independent Lock Bits
    In-System Programming by On-chip Boot Program
    True Read-While-Write Operation
  - 512 Bytes EEPROM
    Endurance: 100,000 Write/Erase Cycles
  - 1K Byte Internal SRAM
  - Programming Lock for Software Security
- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescaler, one Compare Mode
  - One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Three PWM Channels
  - 8-channel ADC in TQFP and QFN/MLF package
    Eight Channels 10-bit Accuracy
  - 6-channel ADC in PDIP package
    Eight Channels 10-bit Accuracy
  - 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
  - Five Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, and Standby
- I/O and Packages
  - 23 Programmable I/O Lines
  - 28-lead PDIP, 32-lead TQFP, and 32-pad QFN/MLF
- Operating Voltages
  - 2.7 - 5.5V (ATmega8L)
  - 4.5 - 5.5V (ATmega8)
- Speed Grades
  - 0 - 8 MHz (ATmega8L)
  - 0 - 16 MHz (ATmega8)
- Power Consumption at 4 Mhz, 3V, 25°C
  - Active: 3.6 mA
  - Idle Mode: 1.0 mA
  - Power-down Mode: 0.5 µA

8-bit AVR® with 8K Bytes In-System Programmable Flash

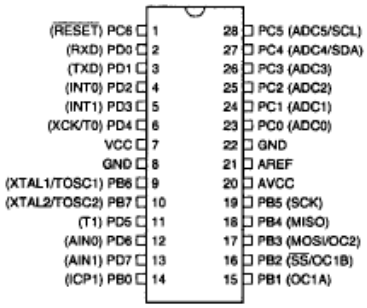
ATmega8
ATmega8L

Summary

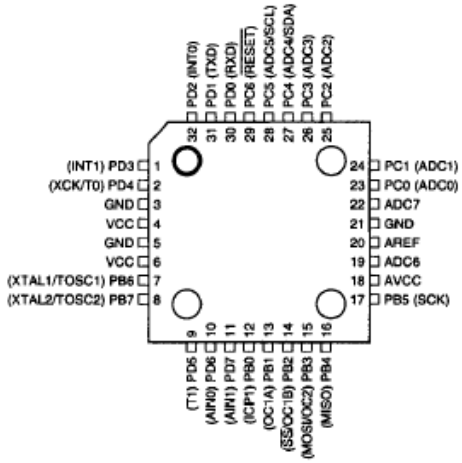
2486PS–AVR–02/06

Pin Configurations

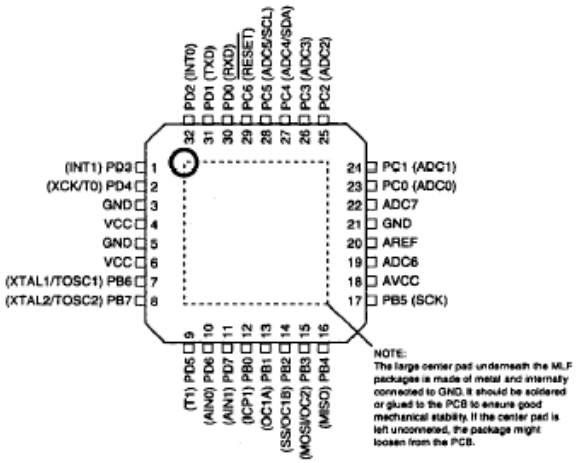
PDIP



TQFP Top View



MLF Top View

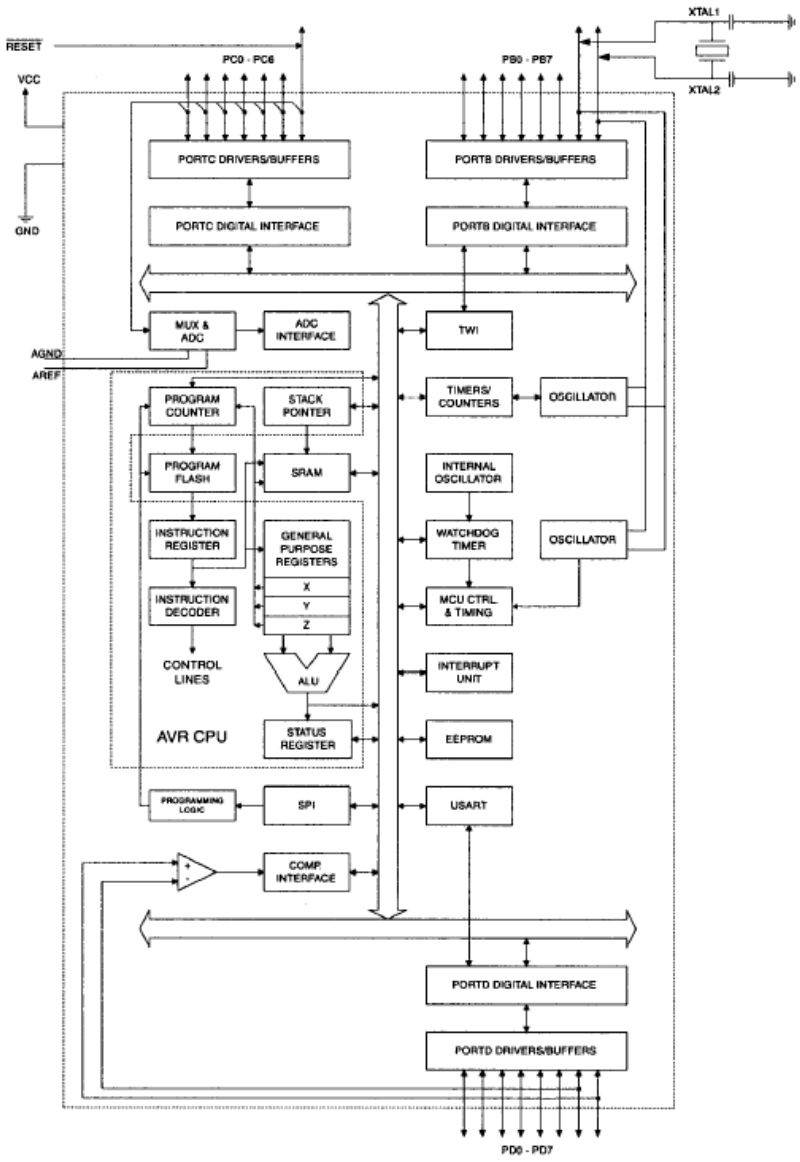


Overview

The ATmega8 is a low-power CMOS 8-bit microcontroller based on the AVR RISC architecture. By executing powerful instructions in a single clock cycle, the ATmega8 achieves throughputs approaching 1 MIPS per MHz, allowing the system designer to optimize power consumption versus processing speed.

Block Diagram

Figure 1. Block Diagram



The 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 ATmega8 provides the following features: 8K bytes of In-System Programmable Flash with Read-While-Write capabilities, 512 bytes of EEPROM, 1K byte of SRAM, 23 general purpose I/O lines, 32 general purpose working registers, three flexible Timer/Counters with compare modes, internal and external interrupts, a serial programmable USART, a byte oriented Two-wire Serial Interface, a 6-channel ADC (eight channels in TQFP and QFN/MLF packages) with 10-bit accuracy, a programmable Watchdog Timer with Internal Oscillator, an SPI serial port, and five software selectable power saving modes. The Idle mode stops the CPU while allowing the SRAM, Timer/Counters, SPI port, and interrupt system to continue functioning. The Power-down mode saves the register contents but freezes the Oscillator, disabling all other chip functions until the next Interrupt or Hardware Reset. In Power-save mode, the asynchronous timer continues to run, allowing the user to maintain a timer base while the rest of the device is sleeping. The ADC Noise Reduction mode stops the CPU and all I/O modules except asynchronous timer and ADC, to minimize switching noise during ADC conversions. In Standby mode, the crystal/resonator Oscillator is running while the rest of the device is sleeping. This allows very fast start-up combined with low-power consumption.

The device is manufactured using Atmel's high density non-volatile memory technology. The Flash Program memory can be reprogrammed In-System through an SPI serial interface, by a conventional non-volatile 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 ATmega8 is a powerful microcontroller that provides a highly-flexible and cost-effective solution to many embedded control applications.

The ATmega8 AVR 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.

Typical values contained in this datasheet are based on simulations and characterization of other AVR microcontrollers manufactured on the same process technology. Min and Max values will be available after the device is characterized.

Pin Descriptions

VCC	Digital supply voltage.
GND	Ground.
Port B (PB7..PB0) XTAL1/XTAL2/TOSC1/TOSC2	Port B is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port B output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port B pins that are externally pulled low will source current if the pull-up resistors are activated. The Port B pins are tri-stated when a reset condition becomes active, even if the clock is not running. Depending on the clock selection fuse settings, PB6 can be used as input to the inverting Oscillator amplifier and input to the internal clock operating circuit. Depending on the clock selection fuse settings, PB7 can be used as output from the inverting Oscillator amplifier. If the Internal Calibrated RC Oscillator is used as chip clock source, PB7..6 is used as TOSC2..1 input for the Asynchronous Timer/Counter2 if the AS2 bit in ASSR is set. The various special features of Port B are elaborated in "Alternate Functions of Port B" on page 56 and "System Clock and Clock Options" on page 23.
Port C (PC5..PC0)	Port C is an 7-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.
PC6/$\overline{\text{RESET}}$	If the RSTDISBL Fuse is programmed, PC6 is used as an I/O pin. Note that the electrical characteristics of PC6 differ from those of the other pins of Port C. If the RSTDISBL Fuse is unprogrammed, PC6 is used as a Reset input. A low level on this pin for longer than the minimum pulse length will generate a Reset, even if the clock is not running. The minimum pulse length is given in Table 15 on page 36. Shorter pulses are not guaranteed to generate a Reset. The various special features of Port C are elaborated on page 59.
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 ATmega8 as listed on page 61.
$\overline{\text{RESET}}$	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 36. Shorter pulses are not guaranteed to generate a reset.
AV_{CC}	AV_{CC} is the supply voltage pin for the A/D Converter, Port C (3..0), and ADC (7..6). It should be externally connected to V_{CC}, even if the ADC is not used. If the ADC is used, it should be connected to V_{CC} through a low-pass filter. Note that Port C (5..4) use digital supply voltage, V_{CC}.
AREF	AREF is the analog reference pin for the A/D Converter.
ADC7..6 (TQFP and QFN/MLF Package Only)	In the TQFP and QFN/MLF package, ADC7..6 serve as analog inputs to the A/D converter. These pins are powered from the analog supply and serve as 10-bit ADC channels.

Resources

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

Register Summary

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
0x3F (0x5F)	SREG	I	T	H	S	V	N	Z	C	9
0x3E (0x5E)	SPH	–	–	–	–	–	SP10	SP9	SP8	11
0x3D (0x5D)	SPL	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	11
0x3C (0x5C)	Reserved									
0x3B (0x5B)	GICR	INT1	INT0	–	–	–	–	IVSEL	IVCE	47, 65
0x3A (0x5A)	GIFR	INTF1	INTF0	–	–	–	–	–	–	66
0x39 (0x59)	TIMSK	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	–	TOIE0	70, 100, 120
0x38 (0x58)	TIFR	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	–	TOV0	71, 101, 120
0x37 (0x57)	SPMCR	SPMIE	RWWSB	–	RWWSRE	BLBSET	PGWRT	PGERS	SPMEN	211
0x36 (0x56)	TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE	169
0x35 (0x55)	MCUCR	SE	SM2	SM1	SM0	ISC11	ISC10	ISC01	ISC00	31, 64
0x34 (0x54)	MCUCSR	–	–	–	–	WDRF	BORF	EXTRF	PORF	39
0x33 (0x53)	TCCR0	–	–	–	–	–	CS02	CS01	CS00	70
0x32 (0x52)	TCNT0	Timer/Counter0 (8 Bits)								70
0x31 (0x51)	OSCCAL	Oscillator Calibration Register								29
0x30 (0x50)	SFIOR	–	–	–	–	ACME	PUD	PSR2	PSR10	56, 73, 121, 191
0x2F (0x4F)	TCCR1A	COM1A1	COM1A0	COM1B1	COM1B0	FOC1A	FOC1B	WGM11	WGM10	95
0x2E (0x4E)	TCCR1B	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	98
0x2D (0x4D)	TCNT1H	Timer/Counter1 – Counter Register High byte								99
0x2C (0x4C)	TCNT1L	Timer/Counter1 – Counter Register Low byte								99
0x2B (0x4B)	OCR1AH	Timer/Counter1 – Output Compare Register A High byte								99
0x2A (0x4A)	OCR1AL	Timer/Counter1 – Output Compare Register A Low byte								99
0x29 (0x49)	OCR1BH	Timer/Counter1 – Output Compare Register B High byte								99
0x28 (0x48)	OCR1BL	Timer/Counter1 – Output Compare Register B Low byte								99
0x27 (0x47)	ICR1H	Timer/Counter1 – Input Capture Register High byte								100
0x26 (0x46)	ICR1L	Timer/Counter1 – Input Capture Register Low byte								100
0x25 (0x45)	TCCR2	FOC2	WGM20	COM21	COM20	WGM21	CS22	CS21	CS20	115
0x24 (0x44)	TCNT2	Timer/Counter2 (8 Bits)								117
0x23 (0x43)	OCR2	Timer/Counter2 Output Compare Register								117
0x22 (0x42)	ASSR	–	–	–	–	AS2	TCN2UB	OCR2UB	TCR2UB	117
0x21 (0x41)	WDTCR	–	–	–	WDCE	WDE	WDP2	WDP1	WDP0	41
0x20[1] (0x40)[1]	UBRRH	URSEL	–	–	–	UBRR[11:8]				156
	UCSRC	URSEL	UMSEL	UPM1	UPM0	USBS	UCSZ1	UCSZ0	UCPOL	154
0x1F (0x3F)	EEARH	–	–	–	–	–	–	–	EEAR8	18
0x1E (0x3E)	EEARL	EEAR7	EEAR6	EEAR5	EEAR4	EEAR3	EEAR2	EEAR1	EEAR0	18
0x1D (0x3D)	EEDR	EEPROM Data Register								18
0x1C (0x3C)	EECR	–	–	–	–	EERIE	EEMWE	EEWE	EERE	18
0x1B (0x3B)	Reserved									
0x1A (0x3A)	Reserved									
0x19 (0x39)	Reserved									
0x18 (0x38)	PORTB	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	63
0x17 (0x37)	DDRB	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	63
0x16 (0x36)	PINB	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	63
0x15 (0x35)	PORTC	–	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0	63
0x14 (0x34)	DDRC	–	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0	63
0x13 (0x33)	PINC	–	PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0	63
0x12 (0x32)	PORTD	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	63
0x11 (0x31)	DDRD	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	63
0x10 (0x30)	PIND	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	63
0x0F (0x2F)	SPDR	SPI Data Register								129
0x0E (0x2E)	SPSR	SPIF	WCOL	–	–	–	–	–	SPI2X	129
0x0D (0x2D)	SPCR	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	127
0x0C (0x2C)	UDR	USART I/O Data Register								151
0x0B (0x2B)	UCSRA	RXC	TXC	UDRE	FE	DOR	PE	U2X	MPCM	152
0x0A (0x2A)	UCSRB	RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8	153
0x09 (0x29)	UBRRL	USART Baud Rate Register Low byte								156
0x08 (0x28)	ACSR	ACD	ACBG	ACO	ACI	ACIE	ACIC	ACIS1	ACIS0	192
0x07 (0x27)	ADMUX	REFS1	REFS0	ADLAR	–	MUX3	MUX2	MUX1	MUX0	203
0x06 (0x26)	ADCSRA	ADEN	ADSC	ADFR	ADIF	ADIE	ADPS2	ADPS1	ADPS0	205
0x05 (0x25)	ADCH	ADC Data Register High byte								206
0x04 (0x24)	ADCL	ADC Data Register Low byte								206
0x03 (0x23)	TWDR	Two-wire Serial Interface Data Register								171
0x02 (0x22)	TWAR	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE	171
0x01 (0x21)	TWSR	TWS7	TWS6	TWS5	TWS4	TWS3	–	TWPS1	TWPS0	171
0x00 (0x20)	TWBR	Two-wire Serial Interface Bit Rate Register								169

Notes:
1. Refer to the USART description for details on how to access UBRRH and UCSRC.
2. For compatibility with future devices, reserved bits should be written to zero if accessed. Reserved I/O memory addresses should never be written.
3. 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 0x00 to 0x1F 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 ← 0xFF – Rd	Z,C,N,V	1
NEG	Rd	Two's Complement	Rd ← 0x00 – 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 • (0xFF - 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 ← 0xFF	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
RCALL	k	Relative Subroutine Call	PC ← PC + k + 1	None	3
ICALL		Indirect Call to (Z)	PC ← Z	None	3
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**

Instruction Set Summary (Continued)

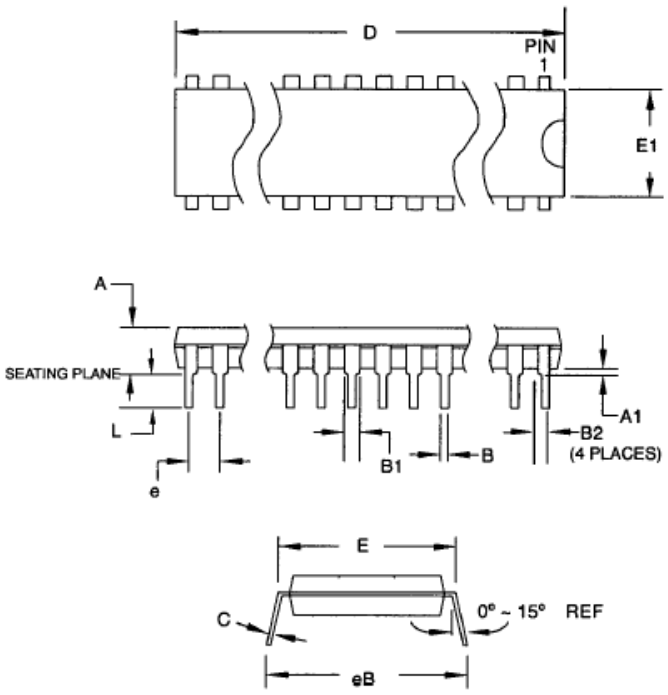
Mnemonics	Operands	Description	Operation	Flags	#Clocks
BRIE	k	Branch if Interrupt Enabled	if (I = 1) then PC ← PC + k + 1	None	1 / 2
BRID	k	Branch if Interrupt Disabled	if (I = 0) then PC ← PC + k + 1	None	1 / 2
DATA TRANSFER INSTRUCTIONS					
MOV	Rd, Rr	Move Between Registers	Rd ← Rr	None	1
MOVW	Rd, 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
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

Ordering Information

Speed (MHz)	Power Supply	Ordering Code	Package[1]	Operation Range
8	2.7 - 5.5	ATmega8L-8AC ATmega8L-8PC ATmega8L-8MC	32A 28P3 32M1-A	Commercial (0°C to 70°C)
		ATmega8L-8AI ATmega8L-8AU[2] ATmega8L-8PI ATmega8L-8PU[2] ATmega8L-8MI ATmega8L-8MU[2]	32A 32A 28P3 28P3 32M1-A 32M1-A	Industrial (-40°C to 85°C)
16	4.5 - 5.5	ATmega8-16AC ATmega8-16PC ATmega8-16MC	32A 28P3 32M1-A	Commercial (0°C to 70°C)
		ATmega8-16AI ATmega8-16AU[2] ATmega8-16PI ATmega8-16PU[2] ATmega8-16MI ATmega8-16MU[2]	32A 32A 28P3 28P3 32M1-A 32M1-A	Industrial (-40°C to 85°C)

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 alternative, complies to the European Directive for Restriction of Hazardous Substances (RoHS directive). Also Halide free and fully Green.

28P3



COMMON DIMENSIONS
(Unit of Measure = mm)

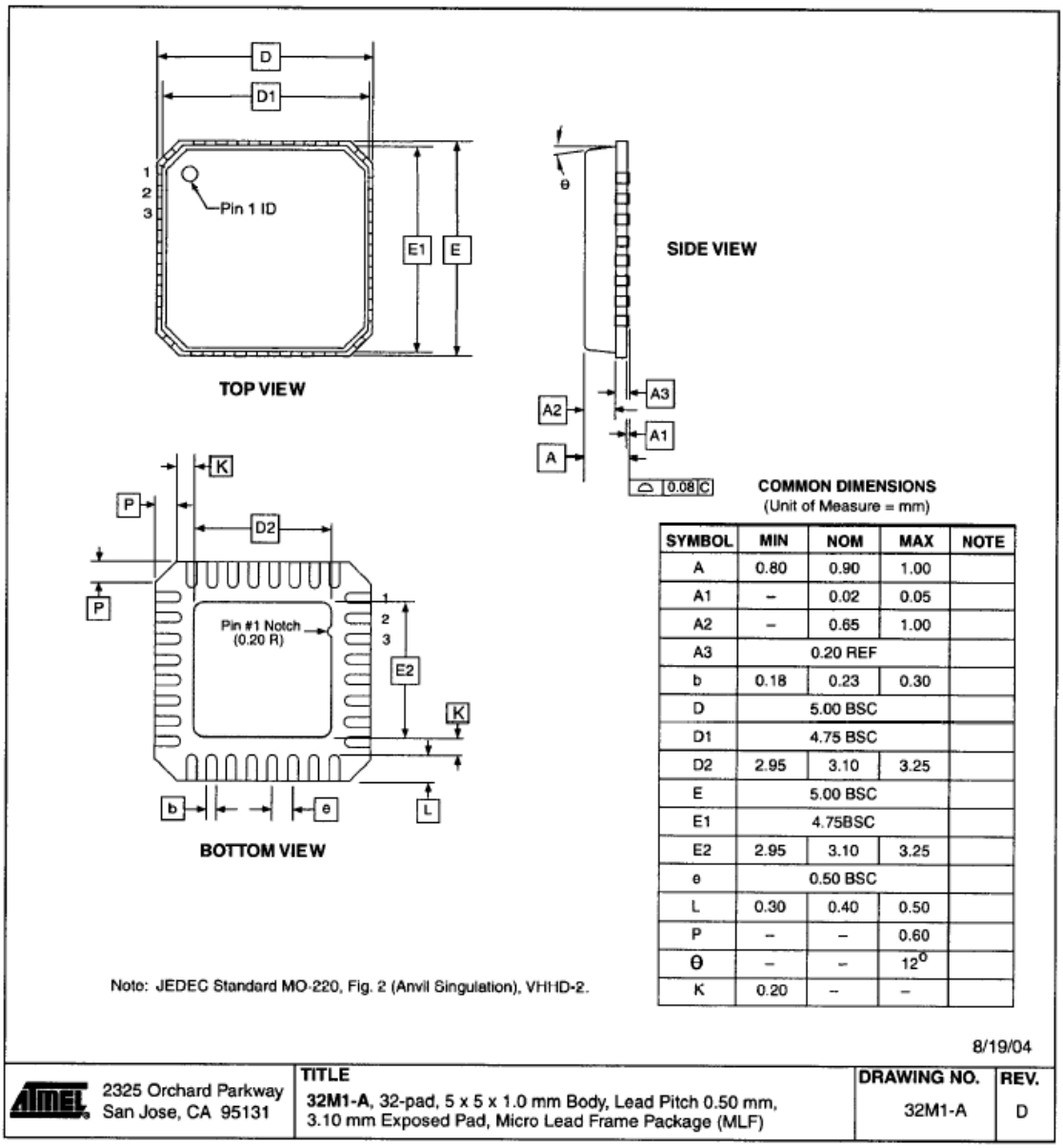
SYMBOL	MIN	NOM	MAX	NOTE
A	–	–	4.5724	
A1	0.508	–	–	
D	34.544	–	34.798	Note 1
E	7.620	–	8.255	
E1	7.112	–	7.493	Note 1
B	0.381	–	0.533	
B1	1.143	–	1.397	
B2	0.762	–	1.143	
L	3.175	–	3.429	
C	0.203	–	0.356	
eB	–	–	10.160	
e	2.540 TYP			

Note: 1. 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	**28P3**, 28-lead (0.300"/7.62 mm Wide) Plastic Dual Inline Package (PDIP)	28P3	B

32M1-A



Note: JEDEC Standard MO-220, Fig. 2 (Anvil Singulation), VHHD-2.

COMMON DIMENSIONS
(Unit of Measure = mm)

SYMBOL	MIN	NOM	MAX	NOTE
A	0.80	0.90	1.00	
A1	–	0.02	0.05	
A2	–	0.65	1.00	
A3		0.20 REF		
b	0.18	0.23	0.30	
D		5.00 BSC		
D1		4.75 BSC		
D2	2.95	3.10	3.25	
E		5.00 BSC		
E1		4.75BSC		
E2	2.95	3.10	3.25	
e		0.50 BSC		
L	0.30	0.40	0.50	
P	–	–	0.60	
θ	–	–	12^{o}	
K	0.20	–	–	

8/19/04

	TITLE	DRAWING NO.	REV.
2325 Orchard Parkway San Jose, CA 95131	**32M1-A**, 32-pad, 5 x 5 x 1.0 mm Body, Lead Pitch 0.50 mm, 3.10 mm Exposed Pad, Micro Lead Frame Package (MLF)	32M1-A	D