

Advanced Embedded Automatic Car Parking System

¹R.S.Harishraghav. ²G.Sri Naga Chaitanya.

^{1,2}B.Tech ,

^{1,2}SreeVidyanikethanEngg.College

ABSTRACT

The objective of paper is to develop a system to indicate the vacant lane. The paper involves a system including infrared transmitter and receiver in every lane and a LED display outside the car parking gate. By using this system these people can automatically know which lane is vacant for parking the car. This paper shows the using of two lanes and each lane consists two slots .At the entrance of each slot IR transmitter and receiver pair is arranged for identifying the vacancy of the vehicle. 5V, 500mA regulated power supply, 7805 three terminal voltage regulator is used for voltage regulation. Full wave bridge rectifier is used to rectify the ac output of secondary of 230/12V step down transformer.

Key Words : LED, IR transmitter, Full wave bridge rectifier

Date of Submission: 30 July 2014



Date of Publication: 15 August 2014

I. INTRODUCTION

An embedded system is a system which is going to do a predefined specified task is the embedded system and is even defined as combination of both software and hardware. All embedded systems are including computers or microprocessors. Some of these computers are however very simple systems as compared with a personal computer In the earliest years of computers in the 1940s, computers were sometimes dedicated to a single task, but were too large to be considered "embedded". Over time however, the concept of programmable controllers developed from a mix of computer technology, solid state devices, and traditional electromechanical sequences. The first recognizably modern embedded system was the Apollo Guidance Computer, developed by Charles Stark Draper at the MIT Instrumentation Laboratory.

EMBEDDED SYSTEM SPECIAL FEATURES:

- ❖ Embedded systems need to be highly reliable. Once in a while, pressing ALT-CTRL-OEL is OK on your desktop, but you cannot afford to reset your embedded system.
- ❖ Some embedded systems have to operate in extreme environmental conditions such as very high temperatures and humidity.
- ❖ Embedded systems that address the consumer market (for exam-ple, electronic toys) are very cost-sensitive: Even a reduction of \$0.1 is lot of cost saving, because thousands or millions systems may be sold.

II. MICROPROCESSOR& MICROCONTROLLER

MICRO PROCESSOR (μ P):

A siliconchip that contains a CPU. In the world of personal computers, the terms microprocessor and CPU are used interchangeably. At the heart of all personal computers and most workstations sits a microprocessor.

Three basic characteristics differentiate microprocessors:

Instruction set: The set of instructions that the microprocessor can execute.

Bandwidth: The number of bits processed in a single instruction.

Clock speed: Given in megahertz (MHz), the clock speed determines how many instructions per second the processor can execute.

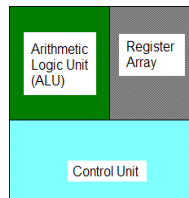


Fig No: 2.1 Three Basic Elements of a Microprocessor

MICRO CONTROLLER (μC): A microcontroller is a small computer on a single integrated circuit containing a processor core, memory, and programmable input/output peripherals. Program memory in the form of NOR flash or OTP ROM is also often included on chip, as well as a typically small amount of RAM. Microcontrollers are designed for embedded applications, in contrast to the microprocessors used in personal computers or other general purpose applications

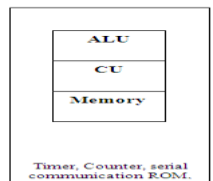


Fig No: 2.2 Block Diagram of Micro Controller (μC)

APPLICATION SPECIFIC INTEGRATED CIRCUIT (ASIC) : ASIC is a combination of digital and analog circuits packed into an IC to achieve the desired control/computation function

ASIC TYPICALLY CONTAINS

- CPU cores for computation and control
- Peripherals to control timing critical functions
- Memories to store data and program
- Analog circuits to provide clocks and interface to the real world which is analog in nature
- I/Os to connect to external components like LEDs, memories, monitors etc.

AT89C51 MICROCONTROLLER:

FEATURES:

- 80C51 based architecture
- 4-Kbytes of on-chip Reprogrammable Flash Memory
- 128 x 8 RAM
- Two 16-bit Timer/Counters
- Full duplex serial channel
- Boolean processor
- Four 8-bit I/O ports, 32 I/O lines
- Memory addressing capability
 - 64K ROM and 64K RAM
- Power save modes:
 - Idle and power-down
- Six interrupt sources
- Most instructions execute in 0.3 μs
- CMOS and TTL compatible
- Maximum speed: 40 MHz @ $V_{cc} = 5\text{V}$
- Industrial temperature available
- Packages available:
 - 40-pin DIP
 - 44-pin PLCC

– 44-pin PQFP

AT89C51 Block Diagram:

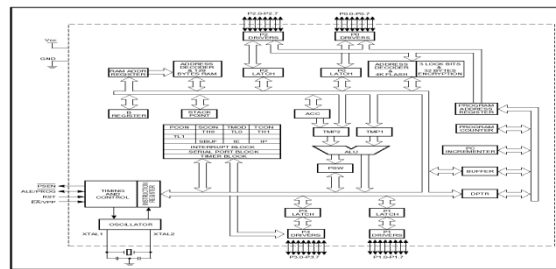


Fig No:2. 3 At89c51 Diagram

PIN DESCRIPTION:

VCC-Supply voltage

GND-Ground

Port 0: Port 0 is an 8-bit open drain bi-directional I/O port. As an output port, each pin can sink eight TTL inputs. When 1s are written to port 0 pins, the pins can be used as high impedance inputs.

Port 1: Port 1 is an 8-bit bi-directional I/O port with internal pull-ups. The port 1 output buffers can sink/source four TTL inputs. When 1s are written to port 1 pins, they are pulled high by the internal pull-ups can be used as inputs. As inputs, Port 1 pins that are externally being pulled low will source current (1) because of the internal pull-ups.

Port 2 : Port 2 is an 8-bit bi-directional I/O port with internal pull-ups. The port 2 output buffers can sink/source four TTL inputs. When 1s are written to port 2 pins, they are pulled high by the internal pull-ups can be used as inputs. As inputs, Port 2 pins that are externally being pulled low will source current because of the internal pull-ups.

Port 3 : Port 3 is an 8-bit bi-directional I/O port with internal pull-ups. The port 3 output buffers can sink/source four TTL inputs. When 1s are written to port 3 pins, they are pulled high by the internal pull-ups can be used as inputs.

Port 3 also receives some control signals for Flash Programming and verification.

Tabular Column: 2.1.PORT PINS

Port pin	Alternate Functions
P3.0	RXD(serial input port)
P3.1	TXD(serial input port)
P3.2	INT0(external interrupt 0)
P3.3	INT1(external interrupt 1)
P3.4	T0(timer 0 external input)
P3.5	T1(timer 1 external input)
P3.6	WR(external data memory write strobe)
P3.7	RD(external data memory read strobe)

III. TIMERS

TIMER : On-chip timing/counting facility has proved the capabilities of the microcontroller for implementing the real time application. These include pulse counting, frequency measurement, pulse width measurement, baud rate generation, etc.

PROGRAMMING 8051 TIMERS: The 8051 has timers: Timer 0 and Timer 1. they can be used either as timers or as event counters. Let us first discuss about the timers' registers and how to program the timers to generate time delays.

BASIC RIGISTERS OF THE TIMER: Both Timer 0 and Timer 1 are 16 bits wide. Since the 8051 has an 8-bit architecture, each 16-bit timer is accessed as two separate registers of low byte and high byte.

TIMER 0 REGISTERS : The 16-bit register of Timer 0 is accessed as low byte and high byte. the low byte register is called TL0(Timer 0 low byte)and the high byte register is referred to as TH0(Timer 0 high byte).These register can be accessed like any other register, such as A,B,R0,R1,R2,etc.for example, the instruction "MOV TL0, #4F"moves the value 4FH into TL0,the low byte of Timer 0.These registers can also be read like any other register.

Mode 1 programming:

The following are the characteristic and operations of mode 1:

- [1]. It is a 16-bit timer: therefore, it allows values of 0000 to FFFFH to be loaded into the timer's registers TL and TH.
- [2]. After TH and TL are loaded with 16-bit initial value, the timer must be started. This is done by "SETB TR0" for Timer 0 and "SET TR1" for Timer 1.
- [3]. After the timer is started, it starts to count up. it counts up until it reaches its limit of FFFFH. When it rolls over from FFFFH to 0000, it sets high a flag bit called TF (timer flag).this timer flag can be monitored. When this timer flag is raised, one option would be to stop the timer with the instructions "CLR TR0" or "CLR TR1" for Timer 0 and Timer 1, respectively. Again, it must be noted that each timer has its own timer flag: TF0 for Timer 0, and TF1 for Timer 1.
- [4]. After the timer reaches its limit its limit and rolls over, in order to repeat the process the registers TH and TL must be reloaded with the original value and TF must be reset to 0

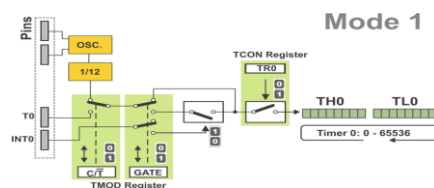


Fig No: 3.1 Tmod Register

Steps to program in mode 1:

- [1]. Load the TMOD value register indicating which timer (Timer 0 or Timer 1) is to be used and which timer mode (0 or 1) is selected.
- [2]. Load register TL and TH with initial count values.
- [3]. Start the timer.
- [4]. Keep monitoring the timer flag (TF) with the "JNB TFx, target" instruction to see if it is raised. Get out of the loop when TF becomes high.
- [5]. Stop the timer.
- [6]. Clear the TF flag for the next round
- [7]. Go back to Step 2 to load TH and TL again.

Finding values to be loaded into the timer: Assuming that we know the amount of timer delay we need, the question is how to find the values needed for the TH, TL registers. To calculate the values to be loaded into the TL and TH registers.

- [1]. Divide the desired time delay by $1.085\mu\text{s}$.
- [2]. Perform $65536-n$, where n is the decimal value we got in step 1.
- [3]. Convert the result of step 2 to hex, where $yyxx$ is the initial hex value to be loaded into the timer's registers.
- [4]. Set $TL=xx$ and $TH=yy$.

Generating a large time delay: Size of the time delay depends on two factors; (a) the crystal frequency and (b) the timer's 16-bit register in mode 1. Both of these factors are beyond the control of the 8051 programmer. We saw earlier that the largest time delay is achieved by making both TH and TL zero. What if that is not enough?

Mode 0: Mode 0 is exactly like mode 1 except that it is a 13-bit timer instead of 16-bit. The 13-bit counter can hold values between 0000 to 1FFFH in TH-TL. Therefore, when the timer reaches its maximum of 1FFFH, it rolls over to 0000, and TF is raised.

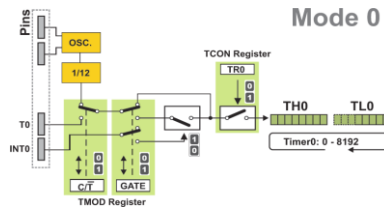


Fig No: 3.2 Modes 0

Mode 2 programming: The following are the characteristics and operations of mode 2.

- [1]. It is 8-bit timer; therefore, it allows only values from 00 to FFH to be loaded into the timer's register TH.
- [2]. After TH is loaded with the 8-bit value, the 8051 gives a copy of it to TL. Then the timer must be started. This is done by the instruction "SETB TR0" for Timer 0 and "SETB TR1" for Timer 1. This is just like mode 1.
- [3]. After the timer is started, it starts to count up by incrementing the TL register. It counts up until it reaches its limit of FFH. When it rolls over from FFH to 00, it sets high the TF (timer flag). If we are using Timer 0, TF0 goes high; if we are using Timer 1, and TF1 is raised.

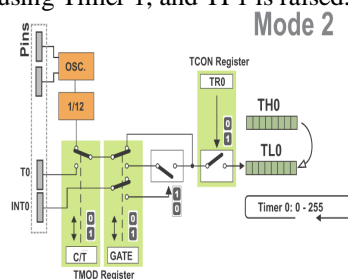


Fig No: 3.3 MODE 2

It must be emphasized that mode 2 is an 8-bit timer. However, it has an auto-reloading capability. In auto-reload TH is loaded with the initial count and a copy of it is given to TL. This reloading leaves TH unchanged, still holding a copy of the original value. This mode has many applications, including setting the baud rate in serial communication.

Steps to program in mode 2

To generate a time delay using the timer's mode 2 take the following steps.

- [1]. Load the TMOD value register indicating which timer (Timer 0 or Timer 1) is to be used, and select the timer mode (mode 2).
- [2]. Load the TH registers with the initial count value.
- [3]. Start the timer
- [4]. Keep monitoring the timer flag (TF) with the "JNB TFx, target" instruction to see whether it is raised. Get out of the loop when TF goes high.
- [5]. Clear the TF flag
- [6]. Go back to step 4, since mode 2 is auto-reload

IV. POWER SUPPLY

BLOCK DIAGRAM

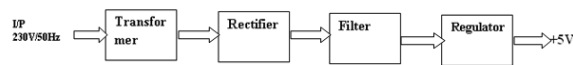


Fig No: 5.1 Power Supply

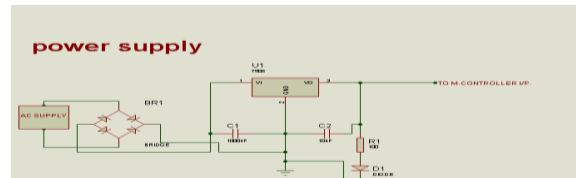


Fig No: 5.1.2 Powr Supply

BASIC PRINCIPLE: A transformer makes use of Faraday's law and the ferromagnetic properties of an iron core to efficiently raise or lower AC voltages. It of course cannot increase power so that if the voltage is raised, the current is proportionally lowered and vice versa.

<p>From Faraday's Law</p> $\frac{V_S}{V_P} = \frac{N_S}{N_P}$	<p>For ideal transformer</p> <p>The voltage ratio is equal to the turns ratio, and power in equals power out.</p>	<p>From conservation of energy</p> $P_P = V_P I_P = V_S I_S = P_S$
---	---	--

SOFTWARE EQUIPMENT

ABOUT SOFTWARE

Software's used are:

- Kiel software for c programming
- Express PCB for lay out design
- Express SCH for schematic design

KIEL SOFTWARE: Installing the Kiel software on a Windows PC

- Insert the CD-ROM in your computer's CD drive
- On most computers, the CD will "auto run", and you will see the Kiel installation menu. If the menu does not appear, manually double click on the Setup icon, in the root directory: you will then see the Keil menu.
- On the Keil menu, please select "Install Evaluation Software". (You will not require a license number to install this software).
- Follow the installation instructions as they appear.

LOADING THE PROJECTS

The example projects for this book are NOT loaded automatically when you install the Keil compiler. These files are stored on the CD in a directory "/Pont". The files are arranged by chapter: for example, the project discussed in Chapter 3 is in the directory "/Pont/Ch03_00-Hello".

Rather than using the projects on the CD (where changes cannot be saved), please copy the files from CD onto an appropriate directory on your hard disk.

Note: you will need to change the file properties after copying: file transferred from the CD will be 'read only'.

Configuring the Simulator

Open the Kiel μ Vision2

Go to Project – Open Project and browse for Hello in Ch03_00 in Pont and open it

SCREEN SHOTS OF HARDWARE

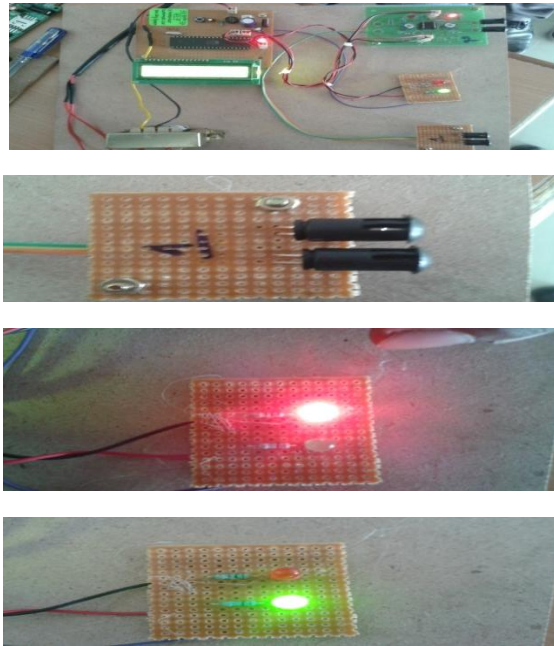


Fig.4.1.Hardware kit



Fig: 4.2. Results display board

- [1]. The 8051 Micro controller and Embedded Systems by Muhammad Ali Mazidi and JaniceGillispie Mazidi
- [2]. The 8051 micro controller Architecture, Programming & Applications by KennethJ.Ayala
- [3]. Fundamentals of Micro processors and Micro computers by B. Ram
- [4]. ElectronicComponents by D.V.Prasad

V. CONCLUSION

**Important factor in traffic areas.
Automated without human being.
16*2 lcd display is used at every floor.
Sensors are used for sensing the car.**

- ✓
- ✓
- ✓
- ✓
- ✓

APPENDIX [Source code]

```

#include<reg52.h>
#define TX_P1 P1^0;
#define RX_P1 P1^1;
#define TX_P2 P2^0;
#define RX_P2 P2^1;
#define TX_P3 P3^0;
#define RX_P3 P3^1;
#define TX_P4 P4^0;
#define RX_P4 P4^1;
#define TX_P5 P5^0;
#define RX_P5 P5^1;
#define TX_P6 P6^0;
#define RX_P6 P6^1;
#define TX_P7 P7^0;
#define RX_P7 P7^1;
#define TX_P8 P8^0;
#define RX_P8 P8^1;
#define TX_P9 P9^0;
#define RX_P9 P9^1;
#define TX_P10 P10^0;
#define RX_P10 P10^1;
#define TX_P11 P11^0;
#define RX_P11 P11^1;
#define TX_P12 P12^0;
#define RX_P12 P12^1;
#define TX_P13 P13^0;
#define RX_P13 P13^1;
#define TX_P14 P14^0;
#define RX_P14 P14^1;
#define TX_P15 P15^0;
#define RX_P15 P15^1;
#define TX_P16 P16^0;
#define RX_P16 P16^1;
#define TX_P17 P17^0;
#define RX_P17 P17^1;
#define TX_P18 P18^0;
#define RX_P18 P18^1;
#define TX_P19 P19^0;
#define RX_P19 P19^1;
#define TX_P20 P20^0;
#define RX_P20 P20^1;
#define TX_P21 P21^0;
#define RX_P21 P21^1;
#define TX_P22 P22^0;
#define RX_P22 P22^1;
#define TX_P23 P23^0;
#define RX_P23 P23^1;
#define TX_P24 P24^0;
#define RX_P24 P24^1;
#define TX_P25 P25^0;
#define RX_P25 P25^1;
#define TX_P26 P26^0;
#define RX_P26 P26^1;
#define TX_P27 P27^0;
#define RX_P27 P27^1;
#define TX_P28 P28^0;
#define RX_P28 P28^1;
#define TX_P29 P29^0;
#define RX_P29 P29^1;
#define TX_P30 P30^0;
#define RX_P30 P30^1;
#define TX_P31 P31^0;
#define RX_P31 P31^1;
#define TX_P32 P32^0;
#define RX_P32 P32^1;
#define TX_P33 P33^0;
#define RX_P33 P33^1;
#define TX_P34 P34^0;
#define RX_P34 P34^1;
#define TX_P35 P35^0;
#define RX_P35 P35^1;
#define TX_P36 P36^0;
#define RX_P36 P36^1;
#define TX_P37 P37^0;
#define RX_P37 P37^1;
#define TX_P38 P38^0;
#define RX_P38 P38^1;
#define TX_P39 P39^0;
#define RX_P39 P39^1;
#define TX_P40 P40^0;
#define RX_P40 P40^1;
#define TX_P41 P41^0;
#define RX_P41 P41^1;
#define TX_P42 P42^0;
#define RX_P42 P42^1;
#define TX_P43 P43^0;
#define RX_P43 P43^1;
#define TX_P44 P44^0;
#define RX_P44 P44^1;
#define TX_P45 P45^0;
#define RX_P45 P45^1;
#define TX_P46 P46^0;
#define RX_P46 P46^1;
#define TX_P47 P47^0;
#define RX_P47 P47^1;
#define TX_P48 P48^0;
#define RX_P48 P48^1;
#define TX_P49 P49^0;
#define RX_P49 P49^1;
#define TX_P50 P50^0;
#define RX_P50 P50^1;
#define TX_P51 P51^0;
#define RX_P51 P51^1;
#define TX_P52 P52^0;
#define RX_P52 P52^1;
#define TX_P53 P53^0;
#define RX_P53 P53^1;
#define TX_P54 P54^0;
#define RX_P54 P54^1;
#define TX_P55 P55^0;
#define RX_P55 P55^1;
#define TX_P56 P56^0;
#define RX_P56 P56^1;
#define TX_P57 P57^0;
#define RX_P57 P57^1;
#define TX_P58 P58^0;
#define RX_P58 P58^1;
#define TX_P59 P59^0;
#define RX_P59 P59^1;
#define TX_P60 P60^0;
#define RX_P60 P60^1;
#define TX_P61 P61^0;
#define RX_P61 P61^1;
#define TX_P62 P62^0;
#define RX_P62 P62^1;
#define TX_P63 P63^0;
#define RX_P63 P63^1;
#define TX_P64 P64^0;
#define RX_P64 P64^1;
#define TX_P65 P65^0;
#define RX_P65 P65^1;
#define TX_P66 P66^0;
#define RX_P66 P66^1;
#define TX_P67 P67^0;
#define RX_P67 P67^1;
#define TX_P68 P68^0;
#define RX_P68 P68^1;
#define TX_P69 P69^0;
#define RX_P69 P69^1;
#define TX_P70 P70^0;
#define RX_P70 P70^1;
#define TX_P71 P71^0;
#define RX_P71 P71^1;
#define TX_P72 P72^0;
#define RX_P72 P72^1;
#define TX_P73 P73^0;
#define RX_P73 P73^1;
#define TX_P74 P74^0;
#define RX_P74 P74^1;
#define TX_P75 P75^0;
#define RX_P75 P75^1;
#define TX_P76 P76^0;
#define RX_P76 P76^1;
#define TX_P77 P77^0;
#define RX_P77 P77^1;
#define TX_P78 P78^0;
#define RX_P78 P78^1;
#define TX_P79 P79^0;
#define RX_P79 P79^1;
#define TX_P80 P80^0;
#define RX_P80 P80^1;
#define TX_P81 P81^0;
#define RX_P81 P81^1;
#define TX_P82 P82^0;
#define RX_P82 P82^1;
#define TX_P83 P83^0;
#define RX_P83 P83^1;
#define TX_P84 P84^0;
#define RX_P84 P84^1;
#define TX_P85 P85^0;
#define RX_P85 P85^1;
#define TX_P86 P86^0;
#define RX_P86 P86^1;
#define TX_P87 P87^0;
#define RX_P87 P87^1;
#define TX_P88 P88^0;
#define RX_P88 P88^1;
#define TX_P89 P89^0;
#define RX_P89 P89^1;
#define TX_P90 P90^0;
#define RX_P90 P90^1;
#define TX_P91 P91^0;
#define RX_P91 P91^1;
#define TX_P92 P92^0;
#define RX_P92 P92^1;
#define TX_P93 P93^0;
#define RX_P93 P93^1;
#define TX_P94 P94^0;
#define RX_P94 P94^1;
#define TX_P95 P95^0;
#define RX_P95 P95^1;
#define TX_P96 P96^0;
#define RX_P96 P96^1;
#define TX_P97 P97^0;
#define RX_P97 P97^1;
#define TX_P98 P98^0;
#define RX_P98 P98^1;
#define TX_P99 P99^0;
#define RX_P99 P99^1;
#define TX_P100 P100^0;
#define RX_P100 P100^1;

```

REFERENCES ON THE WEB

- [1]. www.national.com
- [2]. www.atmel.com
- [3]. www.microsoftsearch.com
- [4]. www.geocities.com



R.S.Harish raghav is from tirupati ,andhrapradesh ,received his B.Tech degree from JNTUA,Ananthapur, interested in Instrumentation and Embedded Systems.



G.Sri Naga Chaitanya is from tirupati ,andhrapradesh ,received his B.Tech degree from JNTUA,Ananthapur, interested in Instrumentation and Embedded Systems and VLSI.