CZECH TECHNICAL UNIVERSITY IN PRAGUE
Faculty of Nuclear Sciences and Physical Engineering
Department of Physical Electronics

Bachelor's degree project

The Electronic Guard of Physically Handicapped Persons

Author: Adam Novozámský
Supervisor: Eng. Jaroslav Pavel
Consultant: Eng. Josef Voltr, C Sc.
Academic year: 2007/2008

Herein I would like to thank my leader Eng. Jaroslav Pavel for his expert guiding, my American friend Libby Merritt for her corrections of my English, and my family for helping during my studies as well.

I declare that the submitted work was elaborated by me and I have mentioned all used documents and literature.

31. 1. 2008, Prague Adam Novozámský

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Zde bude vložen onen list papíru, kde je zadání mé bakalářské práce...

Introduction

Czech

Název práce: Elektronický hlídač tělesně postižených osob
Autor: Adam Novozámský
Obor: Inženýrská informatika
Druh práce: Bakalářská práce
Vedoucí práce: Ing. Jaroslav Pavel, Katedra fyzikální elektroniky, Fakulta jaderná a fyzikálně inženýrská, České vysoké učení technické v Praze
Konzultant: Ing. Josef Voltr, CSc., Katedra fyzikální elektroniky Fakulta jaderná a fyzikálně inženýrská, České vysoké učení technické v Praze

Abstrakt: Cílem této bakalářská práce je návrh a realizace elektronického hlídače postižených osob. Zařízení bude sloužit ke kontrole a přivolání pomoci, jestliže pacient vypadl z lůžka nebo invalidního vozíku. V tomto dokumentu je popsáno kompletní hardwarové a softwarové řešení a její realizace.
Kontrola je zajištěna optickým prvkem s infračerveným detektorem, který předává informace do PIC mikrokontroléru. K přivolání pomoci je použit mobilní telefon. Komunikaci mezi PICem a mobilním telefonem zprostředkovává HIN232, který převádí úrovně napětí.

Klíčová slova: GSM alarm, PIC, AT command, PPL

English

Title: The Electronic Guard of Physically Handicapped Persons
Author: Adam Novozámský

Abstract: The thesis deals with a design and construction of the electronic guard for physically handicapped persons. The final device will be instrumental to control and to call assistance, if the patient fell out from a clinical bed or a wheelchair. This document describes complete hardware and software solution and the execution.
The control is guaranteed via an optical element with infrared detector. It hands over information to the PIC micro controller. The cell phone is used for calling help. The communication between PIC and cell phone supplies the HIN232 that transfers the levels of voltages.

Key words: GSM alarm, PIC, AT command, PPL

Contents

List of Figures

List of Tables

List of Equation

Conventions in Writing

We used the special text conventions for better understanding of the project documentation. These text conventions are described in the following **[Table 1]**.

Description	Meaning	Example
Bold Consolas in square brackets	Reference to other chapters	**[3.2.1.1]**
	Reference to the figure, table, etc.	**[Figure 3]**
Bold Consolas superscript in square brackets	Referenced document	[D1]
	Referenced software	[S1]
Plain Consolas	Code in C language	`void main()`
Bold Italic Consolas	File name	***File.hex***

Table 1: Document Conventions

If the reference document is typed in a chapter title, the whole chapter is partial undertaken from this referenced document.

1 Goals of the Project

This project terminates my course of bachelor's studies at the Department of Physical Electronics at the Faculty of Nuclear Sciences and Physical Engineering, at the Czech Technical University in Prague. My study program "Instruments and Informatics" offers a complex education in programming knowledge electronics. Last year we designed some projects in electronics and I was interested in working with microcontrollers.

I was working as a personal assistant by the handicapped persons during the summer. And I got to know how difficult it was to call somebody for help, when a handicapped person fell down from a clinical bed or a wheelchair. So I started to think about using the microcontrollers in this case. I hope it will be helpful.

2 Task

Topic:

The Electronic Guard of Physically Handicapped Persons

Elaboration instructions:

I. Get acquainted with architecture PIC micro controllers, AT commands for GSM modules, and a phase-locked loop (PLL) electronic control system
II. Design optimal solution for monitoring physically handicapped person by means of optoelectronic detector, applicable micro controller, and GSM module
III. Accomplish realization of hardware and software this device mentions above

3 Project Solution

3.1 Solution Introduction

We naturally divided our solution to the hardware and software part. The hardware explains the preference, feature, and usage of main components. The PICmicro® MCU **[3.2.1]** was chosen like microcomputer for its various advantages. The cell phone **[3.2.4]** serves as a device for calling help. And the interface between PIC and cell phone is the gadget HIN232 **[3.2.2]**.

When all components were assembled, the printed circuit board was designed by the EAGLE Light Edition.

The software solution is described in the section following the part of hardware. The execution program was written in C language for its code transparency. After long searching, I chose the C compiler from the company mikroElektronika®**[si]**.

3.2 Hardware Design

We can hand out the hardware design to the five subsections.

- The first part deals with the selection the optimal microcontroller and its specification.
- In the second part we speak about the HIN232, which is a converter of the signal voltage.
- The third part contains the description about the optical element.
- In the following part we describe the way to use the communication between a cell phone and a microcontroller; and why we used the cell phone.
- In the fifth part we describe the used power supply.
- This last partition connects the previous parts and it displays the whole solution. Here is also the short description about the EAGLE, the software for the design at the printed circuit board.

3.2.1 PIC – Microchip Microcontroller

Microcontrollers can be found in any products these days. For example, a modern washing machine in our house consists of a timer, button and LED that contains a microcontroller. All modern cars contain a microcontroller. To understand how a microcontroller works, we need to understand digital electronics. We can make a microcontroller work by writing programming code using C programming language. By learning C programming and using it to program the microcontroller to do and react to what we wish, everything becomes so simple.

A microcontroller is a small computer and it can only perform simple tasks.

The main part of microcontroller:

- Processor that executes programs. Processor execute program digitally. All instruction given to the processor should be in digital form.
- Program Memory to store the program that has been compiled successfully by the programmer.
- RAM (random-access memory) to store "variables."

- IO Port to connect sensor, keypad, LED, Relay and so on.
- Timer to count the time to execute some process.

There are a lot of kind of microcontrollers, and many families, which is suitable. At the school we worked in the electronic practical with the PICmicro® MCU and it was the main reason way I chose the microcontroller from this company. The trading policy of this company is really friendly to students, because every month you can order its microcontrollers absolutely free; also without paying any postage (they send it from USA). It is a good example for the other companies.

PICmicro® MCUs are quickly replacing computers when it comes to programming robotic devices. These microcontrollers are small and can be programmed to carry out a number of tasks and are ideal for school and industrial projects. A simple program is written using a computer; it is then downloaded to a microcontroller which, in turn, can control a robotic device.

3.2.1.1 PIC16F877A Specification

Our project has been developed with the most frequent and common PIC16F877A, although it has more features than so small project requires. The reason was because I wanted to widen this application in the future.

This part explains some basic information about PICmicro® MCU used for this project. Additional information may be found in the PIC16F877A datasheet [D2], which may be downloaded from the Microchip website. This datasheet should be considered a complementary document to this project, and it is highly recommended reading for a better understanding of the device architecture and operation of the PICs and its peripherals.

The PIC16F877A is a single device which is brilliant and useful. This microcontroller is very easy to be assembled, program and also the price is very cheap. It cost less than 10 dollars. The good thing is that a single unit can be purchased at that 10 dollar price. It is unlike some other Integrated Circuit that must be bought at a minimum order quantity such as 1000 units or 2000 units or else you won't be able to purchase it.

Additional basic components that you need to make this IC (Integrated Circuit) work is just a 5V power supply adapter, a 20MHz crystal oscillator and 2 units of 22pF capacitors (C2 and C1 in the **[Figure 1]**). The blocking capacitors (C4 and C5 in the **[Figure 1]**) are not necessary for a simple circuit.

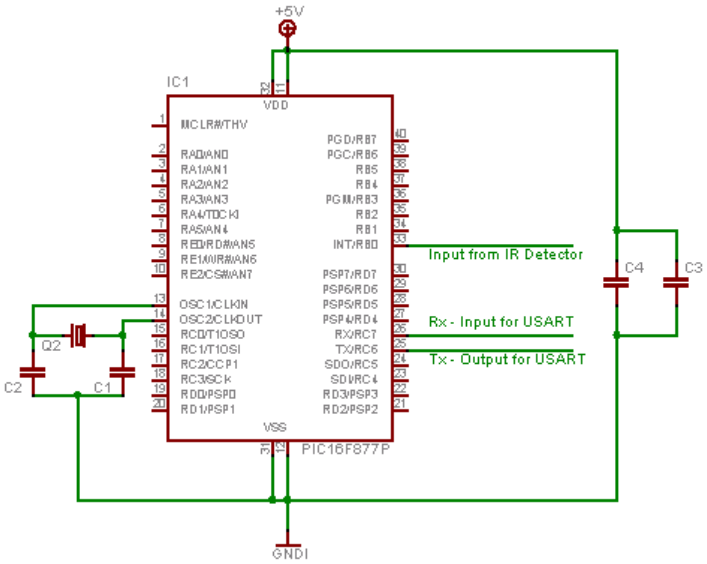


Figure 1: PIC circuit

PIC16F877A is a small piece of semiconductor integrated circuits. The package type of the integrated circuits is DIP package. DIP stand for Dual Inline Package for semiconductor IC. This package is very easy to be soldered onto the strip board. However using a DIP socket is much easier so that this chip can be plugged and removed from the development board. One unit of PIC16F877A microcontroller can be programmed and erased about

10,000 times. Therefore it is very good for new product development phase. The erasing time is almost unnoticeable because once new program are loaded into the PIC, the old program will automatically be erased.

RAM	368
EEPROM	256 bytes
Flash Program Memory	8k words
Operating Frequency	DC to 20MHz
Pins	40
I/O port	Port A,B,C,D,E

Table 2: Specification for PIC16F877A

PIC16F877A is already made with 368 bytes of Random Access Memory (RAM) inside it. Any temporary variable storage that we wrote in our program will be stored inside the RAM. Using this microcontroller you don't need to buy any external RAM memory.

256 bytes of EEPROM are also available inside this microcontroller. This is very useful to store information such as a PIN Number, Serial Number, etc. Using EEPROM is very important because data stored inside EEPROM will be retained when power supply is turned off. RAM did not store data permanently. Data inside RAM is not retained when power supply is turn off.

The size of program code that can be stored is about 8k words inside PIC16F877A ROM. 1 word size is 14 bits.

The crystal oscillator speed that can be connected to the PIC microcontroller range from DC to 20Mhz. We use the frequency 20Mhz. Our crystal oscillator is connected with about 22pF capacitor.

There are 5 input/output ports on PIC microcontroller namely port A, port B, port C, port D, and port E. Each port has a different function. Most of them can be used as I/O port.

There are 40 pins on PIC16F877A. Most of them can be used as an I/O pin, but others are already for specific functions. These are the pin functions:

1. MCLR - to reset the PIC
2. RA0 - port A pin 0
3. RA1 - port A pin 1
4. RA2 - port A pin 2
5. RA3 - port A pin 3
6. RA4 - port A pin 4
7. RA5 - port A pin 5
8. RE0 - port E pin 0
9. RE1 - port E pin 1
10. RE2 - port E pin 2
11. VDD - power supply
12. VSS - ground
13. OSC1 - connect to osc.
14. OSC2 - connect to osc.
15. RC0 - port C pin 0
16. RC1 - port C pin 1
17. RC2 - port C pin 2
18. RC3 - port C pin 3
19. RD0 - port D pin 0
20. RD1 - port D pin 1
21. RD2 - port D pin 2
22. RD3 - port D pin 3
23. RC4 - port C pin 4
24. RC5 - port C pin 5
25. RC6 - port C pin 6
26. RC7 - port C pin 7
27. RD4 - port D pin 4
28. RD5 - port D pin 5
29. RD6 - port D pin 6
30. RD7 - port D pin 7
31. VSS - ground
32. VDD - power supply
33. RB0 - port B pin 0
34. RB1 - port B pin 1
35. RB2 - port B pin 2
36. RB3 - port B pin 3
37. RB4 - port B pin 4
38. RB5 - port B pin 5
39. RB6 - port B pin 6
40. RB7 - port B pin 7

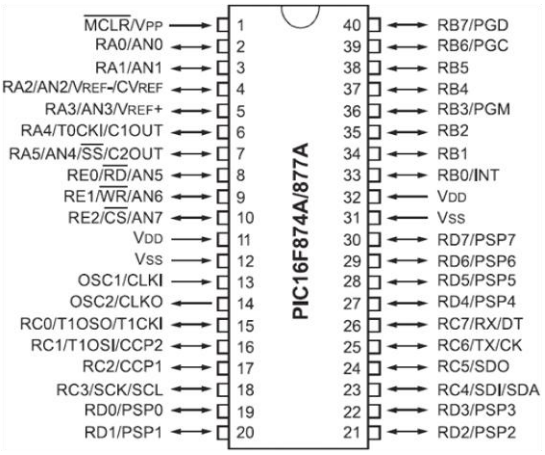


Figure 2: Pin Diagrams

So many applications can be done by the utilizing all of these pins:

- LCD – connect to Port B pin
- LED – connect to any pin declared as output
- Relay and Motor - connect to any pin declared as output
- External EEPROM – connect to I2C interface pin – RC3 and RC4 (SCL and SDA)
- LDR, Potentiometer and sensor – connect to analogue input pin such as RA0
- GSM modem dial up modem – connect to RC6 and RC7 – the serial communication interface using RS232 protocol

3.2.1.1.1 Addressable universal Synchronous Asynchronous Receiver Transmitter (USART)

The USART module is one of the two serial I/O modules. It is sometimes called the Serial Communications Interface or SCI. The configuration of pins RC6/TX and RC7/RX for USART communication is setup on bit SPEN (RCSTA<7>) and on bits TRISC<7:6>.

The USART can be configured in the following modes:

- Asynchronous (full duplex) – It can be configured as a full duplex asynchronous system that can communicate with peripheral devices such as CRT terminals, personal computers (PC) etc.
- Synchronous (half duplex) – The other choice is to configure the USART as a half duplex synchronous system that can communicate with peripheral devices such as A/D or D/A integrated circuits, serial EEPROMs etc. We have two ways and means to set up the synchronous mode – Master or Slave.

Synchronous operation uses a clock and data line while there is no separate clock accompanying the data for Asynchronous transmission. Since there is no clock signal in asynchronous operation, one pin can be used for transmission and another pin can be used for reception. Transmission and reception can be independently enabled. Both can occur at the same time — this is known as a full duplex operation.

In this project we will deal exclusively with asynchronous operation. The most common use of the USART in asynchronous mode is to communicate to a PC serial port using the RS-232 protocol. The cell phone has the RS-232 interface, so this communication is useful also in our solution.

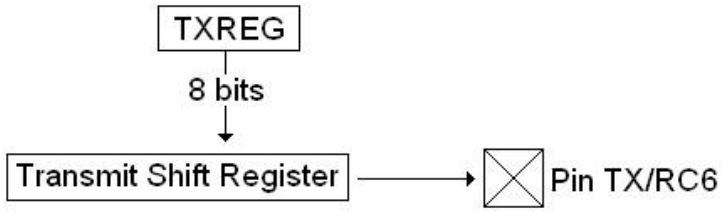


Figure 3: Simplified transmission block diagram for 8bit

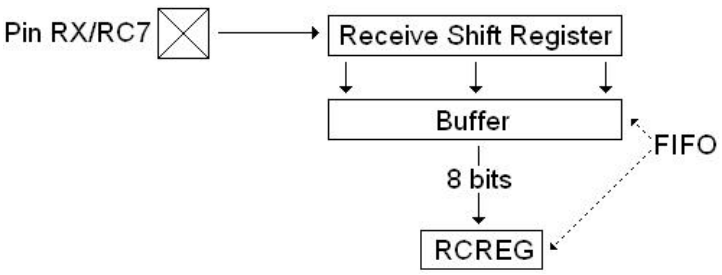


Figure 4: Simplified reception block diagram for 8bit

The USART can be configured to transmit, resp. receive eight or nine data bits by the TX9 bit in the TXSTA register, resp. by the RX9 bit in the RCSTA register. In our solution we will deal exclusively with eight data bits.

Once data has been written to TXREG, the bits are moved into the transmit shift register. From there they are clocked out onto the TX pin preceded by a start bit and followed by a stop bit. The use of a separate transmit shift register allows new data to be written to the TXREG register while the previous

data is still being transmitted. This allows the maximum throughput to be achieved.

After the detection of a start bit, the bits of serial data are shifted from the RX pin into the receive shift register one bit at a time. After the last bit has been shifted in, the stop bit is checked and the data is moved into the buffer which passes the data through to the RCREG register if it is empty. The buffer and RCREG register to form a two element FIFO. The use of a separate receive shift register and a FIFO buffer allows time for the software running on the PICmicro® MCU to read out the received data before an overrun error occurs. It is possible to have received two bytes and be busy receiving a third byte before the data in the RCREG register is read.

The USART outputs/inputs logic level signals on the TX and RX pins of the PICmicro® MCU. The signal is high when no transmission or reception is in progress and goes low when the transmission starts. This low going transition is used by the receiver to synchronize the incoming data. The signal stays low for the duration of the start bit and is followed by the data bits, least significant bit first. In the case of an eight-bit transfer, there are eight data bits and the last data bit is followed by the stop bit which is high. The transmission therefore ends with the pin high. After the stop bit has completed, the start bit of the next transmission can occur as shown by the dotted lines.

There are several things to note about this waveform, which represents the signal on the TX or RX pins of the microcontroller. The start bit is a zero and the stop bit is a one. The data is sent from the least significant bit first so the bit pattern looks backwards in comparison to the way it appears when written as a binary number. The data is not inverted even though RS-232 uses negative voltages to represent a logic one. Generally, when using the USART for RS-232 communications, the signals must be inverted and level shifted through a transceiver chip of some sort.

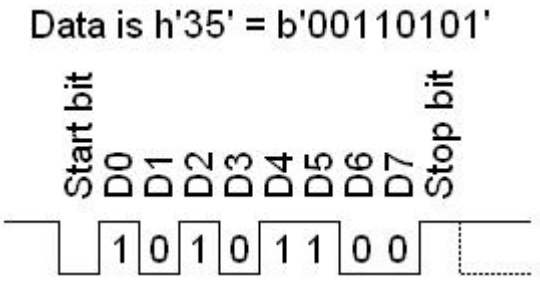


Figure 5: Asynchronous 8 bit waveform example

The signals on the USART pins of the microcontroller use logic levels. This means that for a five volt supply, the signals will be close to five volts when they are high and close to ground when they are low. When communicating with other logic devices, these signals can be used directly. In our application we connected the cell phone and the PICmicro® MCU, which both requires different voltage levels to be used. The Microchip recommends converting the signals to the required levels its TC232 device. But we chose the HIN232 from the company Intersil. This device will be discussed in more detail later in the document.

Because, we write our software via C compiler, it sets up all register for us. And we will describe the use the USART further in the software section. To summarize, here is the list of registers.

There are several registers used to control the USART:

- *The **SPBRG** register allows the baud rate to be set.*
- *The **TXSTA** and **RCSTA** registers are used to control transmission and reception but there are some overlapping functions and both registers are always used.*
- *The **TXREG** and **RCREG** registers are used write data to be transmitted and to read the received data.*
- *The **PIR1** and **PIE1** registers contain the interrupt flag bits and enable bits to allow the USART to generate interrupts. Interrupts are often used when the PICmicro is busy executing code and data needs to be transmitted or received in the background.*

The interrupt flags are not only used for interrupts but can also be read during normal operation to determine whether data has been received or can be transmitted.

3.2.2 HIN232 [D3]

The circuit of the HIN232 is illustrated in the **[Figure 6]**. The HIN232 family of RS-232 transmitters/receivers are powered by a single +5V power supply (except HIN231 and HIN239), feature low power consumption, and meet all ElA RS-232C and V.28 specifications. The circuit is divided into three sections: the charge pump, transmitter, and receiver.

3.2.2.1 Charge Pump

The charge pump contains two sections: the voltage doubler and the voltage inverter. Each section is driven by a two phases, 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 VCC. During phase two, the voltage on C1 is added to VCC, producing a signal across C3 equal to twice VCC. During phase one, C2 is also charged to 2VCC, and then during phase two, it is inverted with respect to ground and to produce a signal across C4 equal to -2VCC. The charge pump accepts input voltages up to 5.5V.

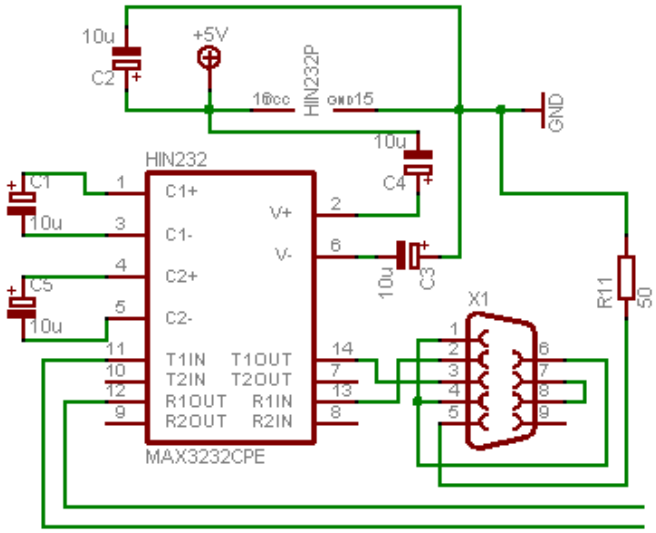


Figure 6: Hin232 in the schema

The output impedance of the voltage doubler section (V+) is approximately 200Ω, and the output impedance of the voltage inverter section (V-) is approximately 450Ω. A typical application uses 1µF capacitors for C1-C4, but we chose 10µF. However, the value is not critical. Increasing the values of C1 and C2 will lower the output impedance of the voltage doubler and inverter, increasing the values of the reservoir capacitors, C3 and C4, lowers the ripple on the V+ and V- supplies.

3.2.2.2 Transmitters

The transmitters are TTL/CMOS compatible inverters which translate the inputs to RS-232 outputs. The input logic threshold is about 26% of VCC, or 1.3V for VCC = 5V. Logic 1 at the input results in a voltage between -5V and V- at the output, and logic 0 results in a voltage between +5V and (V±0.6V). Each transmitter input has an internal 400kΩ pull-up 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 specifications of ±5V minimum with the worst case conditions of: all transmitters driving 3kΩ minimum load impedance, VCC = 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 VCC = 0V.

3.2.2.3 Receivers

The receiver inputs accept up to ±30V while presenting the required 3kΩ to 7kΩ input impedance even if the power is off (VCC = 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 specifications. The receiver output is 0V to VCC. 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.

3.2.3 Optical Element

The application of an infra red optical element for controlling if something or somebody is on the place is really good election. First and foremost is a contactless solution, which does not go behold. And it is independent on the sunshine intensity.

3.2.3.1 Schema Description

The base of the circuit is two infra red LED diodes (light-emitting diodes) D1, D2, and the infra red phototransistor T1. These two elements have to be designed in the optical feed forward. If something or someone is between the LEDs and phototransistor, the IR signal is interrupt, and the T1 does not detect anything. But the signal is detected, if the IR ray has freeway to T1. The base of the detector is the tone decoder LM567. The circuit contains an integrated oscillator, which frequency is comparing with frequency on input in circuit phase hinge. In the case the frequency on input and output are the same, the output is turned over to the low level. This change is detected in the pin INT/RB0 of the microchip.

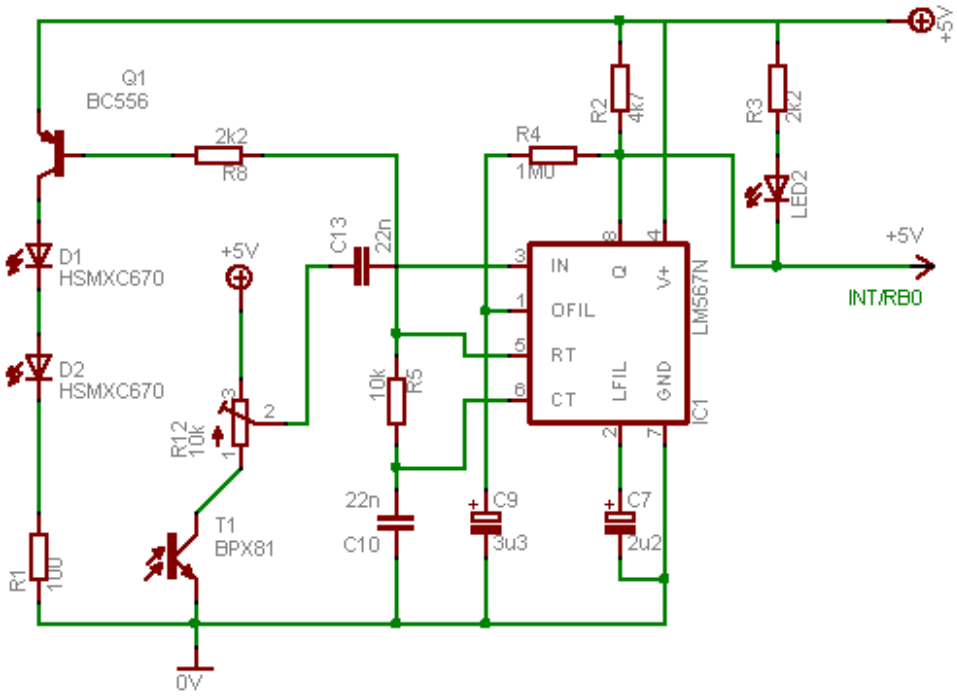


Figure 7: The schema of optical element

The detector is not allowed to be unaffected by ambient light, so we used the modulation of signal with the frequency over 4kHz. The output of oscillator is on the pin number 5 and its frequency determines the resistor R3 between the pin 5 and pin 6, and the capacitor C2.

$$f_o \cong \frac{1}{1.1 \cdot R_3 \cdot C_2} = \frac{1}{1.1 \cdot 10^4 \cdot 22 \cdot 10^{-9}} \cong 4{,}1\ kHz$$

Equation 1: The frequency modulation

The transistor Q1 is switched by the signal from oscillator. In the collector circuit Q1 are connected the infra red LED diodes in the series with a limiting resistor R1. If the beaming from the LED diodes lands on the phototransistor T1, the collector is shows the signal with the same frequency like the signal transmitted by the LEDs D1 and D2. From the traveler of the rotary trimming resistor R6 is leaded this signal through the capacitor C1 to the LM567 on the pin 3(IN). The capacitor C1 separates the direct component. The output on the pin 8 (Q) is turned over by the sufficient signal level and the indication diode LED1 is putt on the light. The sensitivity we can set via the rotary trimming resistor R6.

3.2.3.2 LM567 [D1]

The LM567 is general purpose tone decoders designed to provide a saturated transistor switch to ground when an input signal is present within the pass band. The circuit consists of I and Q detector driven by a voltage controlled oscillator which determines the center frequency of the decoder. External components are used to independently set center frequency, bandwidth and output delay.

Features:

- 20 to 1 frequency range with an external resistor
- Logic compatible output with 100 mA current sinking capability
- Bandwidth adjustable from 0 to 14%
- High rejection of out of band signals and noise
- Immunity to false signals
- Highly stable center frequency
- Center frequency adjustable from 0.01 Hz to 500 kHz

Applications:

- Touch tone decoding
- Precision oscillator
- Frequency monitoring and control
- Wide band FSK demodulation
- Ultrasonic controls
- Carrier current remote controls
- Communications paging decoders

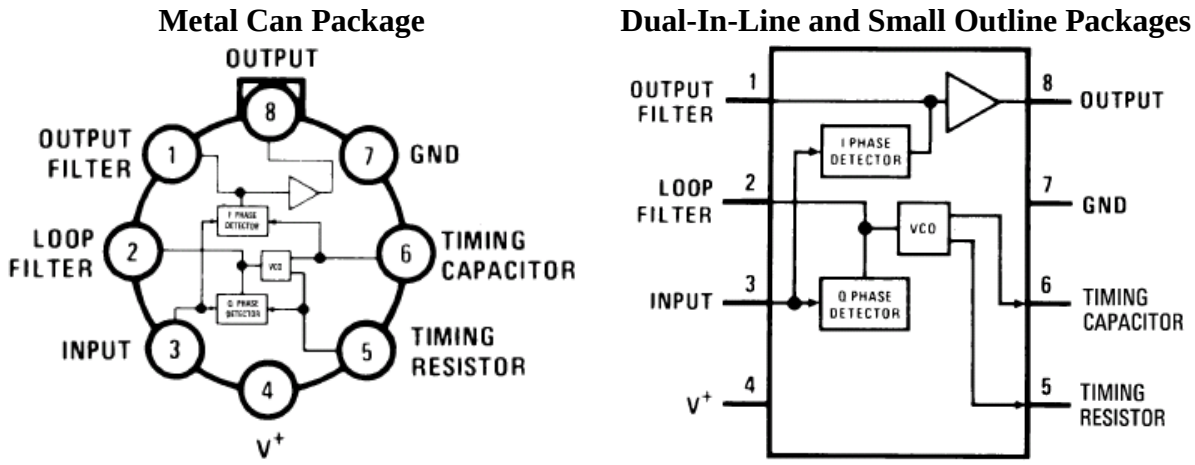


Figure 8: LM567 top view

3.2.4 The Cell Phone

There are many reasons why I chose the GSM module for calling help. The most important was the signal coverage, which is almost everywhere nowadays. The operating agency for this project was Vodafone, and its signal coverage is depicted in the **[Figure 9]**.

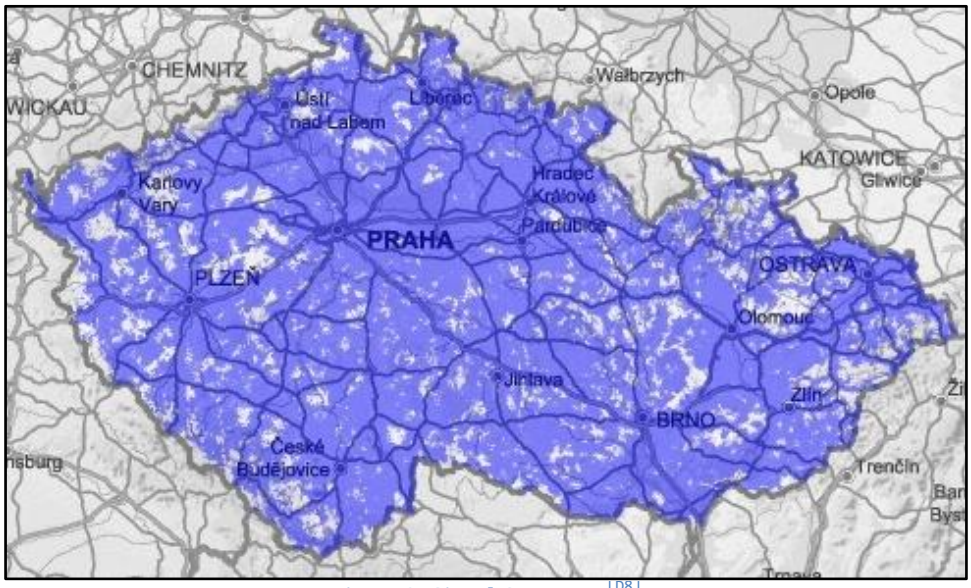


Figure 9: Signal Coverage [D8]

When we sought the GSM module for our project we were interested in its price. A cell phone was cheaper than the classic GSM module. There are many types in the bazaar. We needed the cell phone with possibility of the serial communication.

This project is compatible with an all cellular phones observant the Hayes-Standard Commands. The Siemens c35i matched our demands. We use this cellular phone only for calling help, so there is no reason why we should write more about this device.

The basic info we need is in the **[Table 3]**.

Figure 10: Siemens c35

Weight	110g
Dimensions	118 x 46 x 21 mm
Talk time	Up to 5 hrs
Stand by time	Up to 180 hrs
Dual Band	GSM 900 and GSM 1800

Table 3: Features of c35i

3.2.4.1 AT Command Set [D6]

Remote control operation of the GSM mobile telephone runs via a serial interface (data cable of infrared connection), where AT+C commands according to ETSI GSM 07.07 and GSM 07.05 specification as well as several manufacturer specific AT commands are available. Commands should begin with the character string „AT" and end with"<CR>" (= 0x0D). The input of a command is acknowledged by the display of "OK" or "ERROR". A command currently in process is interrupted by each additional character entered. This means that you should not enter the next command until you have received the acknowledgment. Otherwise the current command is interrupted.

The commands supported (Hayes-Standard Commands) are listed in the following **[Table 4]**.

Command	Function
A/	Repeat last command
AT...	Prefix for all other commands
ATA	Accept call
ATD<str>;	Dial the dialing string <str> with the voice utility Valid dial modifiers: "T" (tone dialing), "P" (pulse dialing) is ignored. The character ";" is important, for this tells the phone that the call should be set up with the voice utility. Otherwise an attempt is made to set up a data call, which the phone immediately acknowledges with "ERROR". The dial command responds with OK to the user right after starting a void call. Other behavior like *# sequences in the dial command and also data calls remain unchanged.
ATD<n>;	The telephone book is selected with the command at+cpbs (or at^spbs).
ATD<mem><n>;	Dial the telephone number from the telephone book <mem> location number <n>
ATDL	Dial last telephone number
ATE0	Deactivate command echo
ATE1	Activate command echo
ATH[0]	Separate connection
ATQ0	Display acknowledgments
ATQ1	Suppress acknowledgments
ATV0	Output acknowledgments as numbers

Command	Function
ATV1	Output acknowledgments as text
AT&F[0]	Reset to factory profile
ATZ	Set to default configuration
AT+GCAP	Output the capabilities list

Table 4: The Hayes-standard commands

3.2.4.1.1 Acknowledgments for Normal Data Communication

After we send the AT command to the mobile phone, we receive the acknowledgments from the cell phone. This confirmation is important for recognizing errors in the communication between the cell phone and the microcontroller, or if the command was executed.

So we have to check the acknowledgments after sending every command. The following **[Table 5]** describes the meaning of all acknowledgments.

Response	Numeric	Meaning
OK	0	Command executed, no errors
RING	2	Ring detected
NO CARRIER	3	Link not established or disconnected
ERROR	4	Invalid command or command line too long
NO DIALTONE	6	No dial tone, dialing impossible, wrong mode
BUSY	7	Remote station busy

Table 5: The cell phone acknowledgments

3.2.5 Power Supply

The power supply circuit is solved by means of the stabilizer. The diode D2 (1N4004) is serving like polarity reversal protection. The capacitor C16, C12, and C11 filters the power supply voltage. The power supply is connected via the screw cable box on the pins X2-5 and X2-6, and the voltage has to be about 9V.

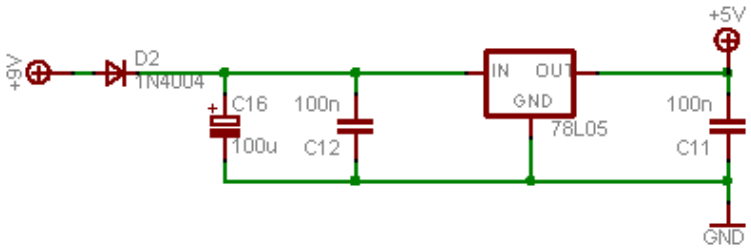


Figure 11: Power Supply

The cell phone has own supplying from the battery inside the phone.

3.2.5.1 L7805CV [D2]

The L7800 series of three-terminal positive regulators is available in TO-220, TO-220FP, TO-3 and D2PAK packages and several fixed output voltages, making it useful in a wide range of applications. These regulators can provide local on-card regulation, eliminating the

distribution problems associated with single point regulation. Each type employs internal current limiting, thermal shut-down and safe area protection, making it essentially indestructible. If adequate heat sinking is provided, they can deliver over 1A output current. Although designed primarily as fixed voltage regulators, these devices can be used with external components to obtain adjustable voltage and currents.

3.2.6 Project Schematics and Printed Circuit Board

Herein in this part we present the whole schema of our project. The circuit was designed in the Eagle [D1] software. The schema is in the **[Figure 11]** and the printed circuit board is in the **[Figure 12]**.

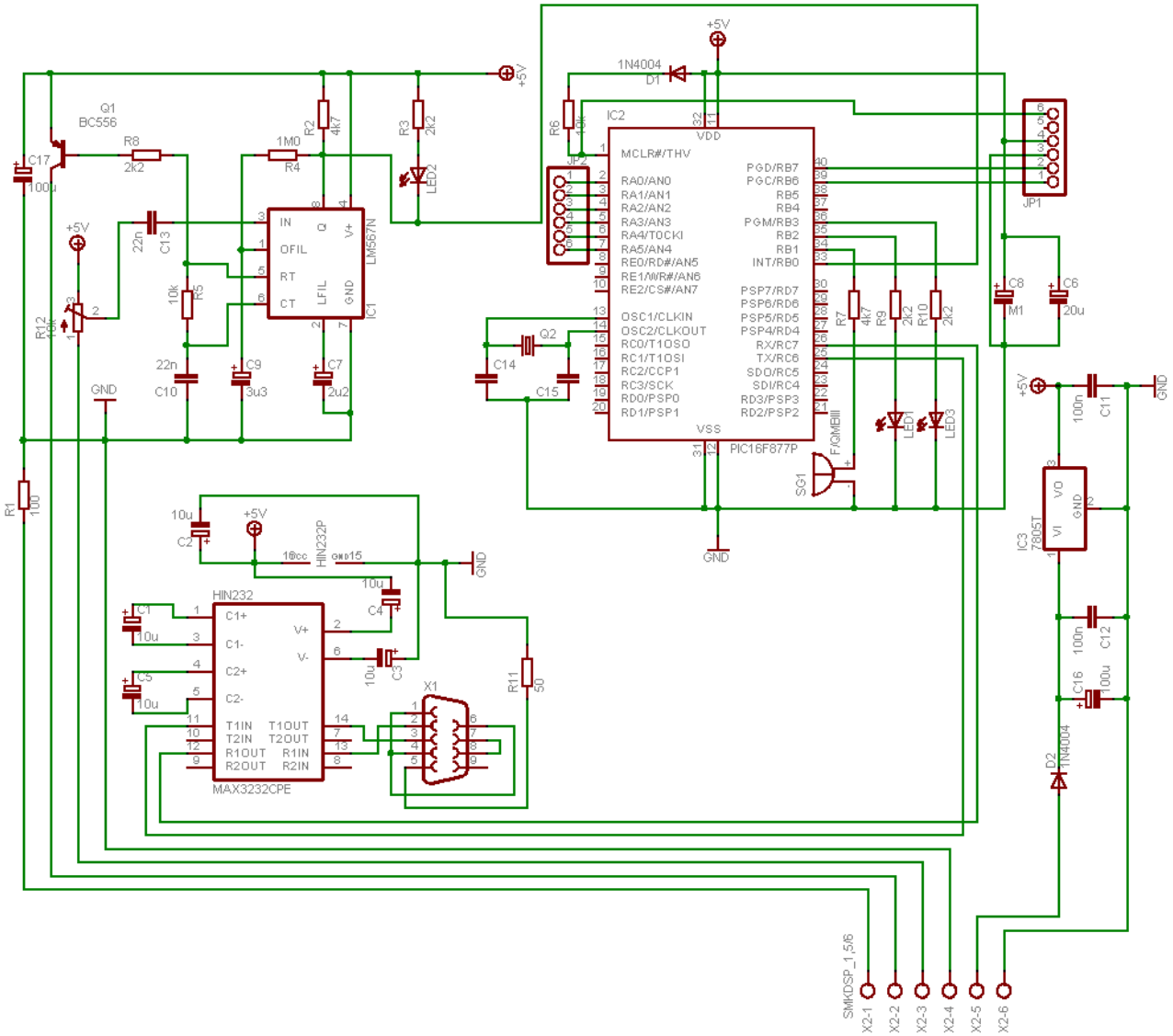
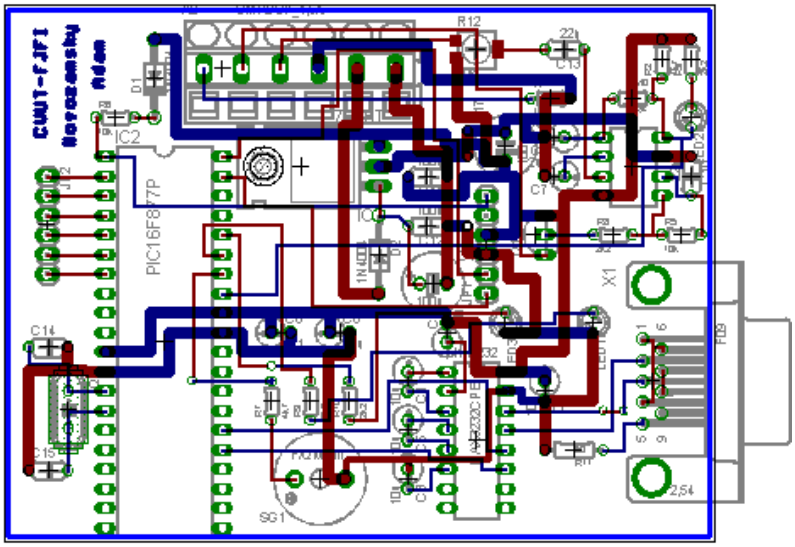


Figure 12: The schema of the whole project



The Printed circuit board was manufactured in Mělník by the company PRINTED s. r. o.
For more information about this company, prices and about other manufacture possibilities look at their web site:
http://www.printed.cz/

Figure 13: Printed Circuit Board

3.2.6.1 EAGLE

The name EAGLE stands for Easily Applicable Graphical Layout Editor. It is PCB software which is a powerful tool for designing printed circuit boards (PCBs). EAGLE is very commonly used by private electronics enthusiasts, because there is a very usable free demo version for nonprofit use and is also available in English. EAGLE has released versions for Microsoft Windows, Linux, and Mac OS X.

The free ECADs from some companies are crippled so that they won't save or won't print. The only limitations of boards made with the EAGLE demo are: 2 copper layers; a maximum size of 80mm x 100mm (½ Eurocard)(~3in x ~4in). The demo version of the schematic editor module can only create single-sheet schematics.

EAGLE provides a schematic editor for designing circuit diagrams and a tightly integrated PCB layout editor, which automatically starts off with all of the components required by the schematic. Components are manually arranged on the board, with the help of colored lines showing the eventual connections between pins that are required by the schematic to aid in finding a placement that will allow the most efficient track layout.

It also provides a good autorouter, which once the components have been placed will attempt to automatically find an optimal track layout to make the electrical connections. It does not always manage to find a way of routing all the signals, although it permits manual routing of critical paths such as power and high frequency lines before letting the autorouter handle the other connections. The .brd files that EAGLE uses to store board layouts are accepted by many PCB production houses.

More info about EAGLE you find in the EAGLE tutorial [D4].

3.3 Software Design

At the beginning of this project we were looking for the compiler for C language which is be able to translate the C code to the assembler code and then to the machine code. We can find a lot of compilers with this description, but for higher rounds of PIC microcontroller only. After long journey across the Internet we found the mikroC[S1] compiler supportive also the PIC16 family. This software is a strong instrument for development application on the PIC microcontroller and its IDE is really friendly. So we chose it for our work and we used the freeware version.

In the mokroElektronika development environment is it possible to program in Assembler or in C. Considering the complexity of the project, it was decided to use C as programming language. This offers a much easier further development and understanding of the code.

3.3.1 IDE mikroC

The sales manager of mikroElektronika David Plecas allowed me to use all documentation from mikroC help [D1] in my Bachelor's degree project. So this part about mikroC was undertaken and overworked from its documentation.

3.3.1.1 Overview

MikroC is a powerful feature and rich development tool for PICmicros. It is designed to provide the easiest possible solution for developing applications for embedded systems, without compromising performance or control.

The mikroC allows quickly develop and deploy complex application. The C source code is written in the highly advanced Code Editor. The libraries are included for a dramatic speed up in the development of: data acquisition, memory, displays, conversions, communications etc. The Code Explorer shows the program structure, variables, and function. We can generate commented, human-readable assembly, and standard HEX compatible with all programmers. The Debugger is included. The IDE provides detailed reports and graphs on code statistics, assembly listing, calling tree etc. And last but not least, in mikroC are plenty of examples to

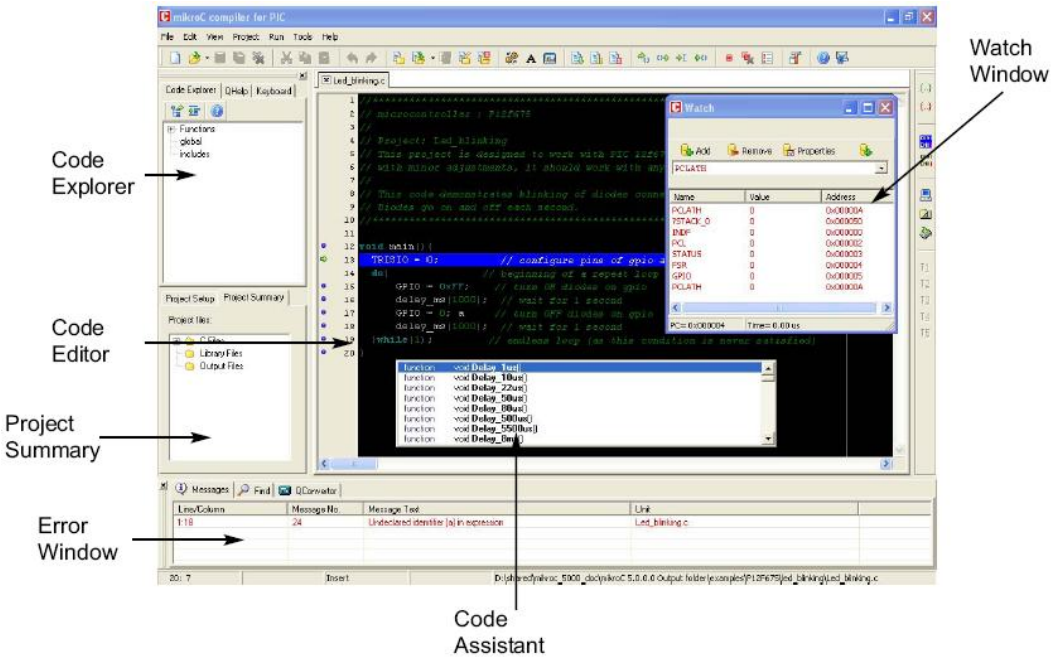


Figure 14: mikroC IDE

expand, develop, and use as building bricks in projects.

- **The Code Editor** is an advanced text editor fashioned to satisfy all the needs of programmer. General code editing is the same as working with any standard text-editor, including familiar Copy, Paste, and Undo actions, common for Windows environment. Other advanced editor features are adjustable syntax highlighting, code assistant, code templates (auto complete), auto correct for common types, bookmarks, and go-to line.

- **The Code Explorer** gives us a clear view of every declared item in the source code. We can jump to a declaration of any item by clicking it, or by clicking the find declaration icon.

- **The source-level Debugger** is an integral component of mikroC development environment. It is designed to simulate operations of Microchip Technology's PICmicros and to assist users in debugging software written for these devices. The Debugger simulates program flow and execution of instruction lines, but does not fully emulate PIC device behavior: it does not update timers, interrupt flags, etc.

- **The Watch Window** allows us to monitor program items while running our program. It displays variables and special function registers of PIC MCU, their address and values. Values are updated as we go through the simulation. We have here also the stopwatch window for displaying the current count the cycles/time since the last Debugger action. It measures the execution time (number of cycles) from the moment the Debugger is started, and can be reset at any time. The other window displays us the map of PIC's RAM, with recently changed items colored red.

- **The Error Window** is located under the message tab, and displays location and type of errors compiler has encountered. The compiler reports also the warnings, but these do not affect the output.

- After successful compilation, we can review **Statistics** of our code. There are six tab windows:
    - ***Memory Usage Window*** provides overview of RAM and ROM memory usage in form of histogram.
    - ***Procedures (Graph) Window*** displays functions in form of histogram, according to their memory allotment.
    - ***Procedures (Location) Window*** displays how functions are distributed in microcontroller's memory.
    - ***Procedures (Details) Window*** displays complete call tree, along with details for each function: size, start and end address, calling frequency, return type, etc.
    - ***RAM Window*** summarizes all GPR and SFR registers and their addresses. Also displays symbolic names of variables and their addresses.
    - ***ROM Window*** lists op-code and their addresses in form of human readable hex code.

- **Integrated Tools**
    - mikroC includes the ***USART*** (Universal Synchronous Asynchronous Receiver Transmitter) ***Communication Terminal*** for RS232 communication.
    - ***ASCII Chart*** is a handy tool, particularly useful when working with LCD display.

- ***7 Segment Display Decoder*** is a convenient visual panel which returns decimal/hex value for any viable combination we would like to display on 7seg.
- ***EEPROM Editor*** allows us to easily manage EEPROM of PIC microcontroller.

3.3.1.2 Using USART in mikroC

We used in our project the USART communication and the mikroC USART Library provides comfortable work with the asynchronous (full duplex) mode. We can easily communicate with other devices via RS232 protocol by use the functions listed below.

Library Routines:

- `Usart_Init`
- `Usart_Data_Ready`
- `Usart_Read`
- `Usart_Write`

3.3.1.2.1 Usart_Init

- **Prototype** `void Usart_Init(const unsigned long baud_rate);`
- **Returns** Nothing
- **Description** It initializes the hardware USART module with the desired baud rate supported by the used device. If the baud rate is unsupported, compiler will report an error.
- **Requires** `Usart_Init` needs to be called before using other functions from USART Library.
- **Example** This will initialize hardware USART and establish the communication at 2400bps: `Usart_Init(2400);`

3.3.1.2.2 Usart_Data_Ready

- **Prototype** `unsigned short Usart_Data_Ready(void);`
- **Returns** Function returns 1 if data is ready or 0 if there is no data.
- **Description** Use the function to test if data in receive buffer is ready for reading.
- **Requires** USART HW module must be initialized and communication established before using this function. See `Usart_Init`.
- **Example** If data is ready, read it:

```
int receive;
...
if (Usart_Data_Ready()) receive = Usart_Read;
```

3.3.1.2.3 Usart_Read

- **Prototype** `unsigned short Usart_Read(void);`
- **Returns** Returns the received byte. If byte is not received, returns 0.
- **Description** Function receives a byte via USART. Use the function Usart_Data_Ready to test if data is ready first.
- **Requires** USART HW module must be initialized and communication established before using this function. See `Usart_Init`.

- **Example** If data is ready, read it:

```
int receive;
...
if (Usart_Data_Ready()) receive = Usart_Read();
```

3.3.1.2.4 Usart_Write

- **Prototype** `void Usart_Write(unsigned short data);`
- **Returns** Nothing
- **Description** Function transmits a byte (data) via USART.
- **Requires** USART HW module must be initialized and communication established before using this function. See `Usart_Init`.
- **Example**

```
int chunk = 0x1E;
Usart_Write(chunk);     /* send chunk via USART */
```

3.3.2 Code Description

We can divide the code into two main parts. The first ensures the initialization of the PIC and the cell mobile. If the communication between the cell phone and the PIC does not work, we display it on the red diode. If it is alright, we show it on the blue diode.

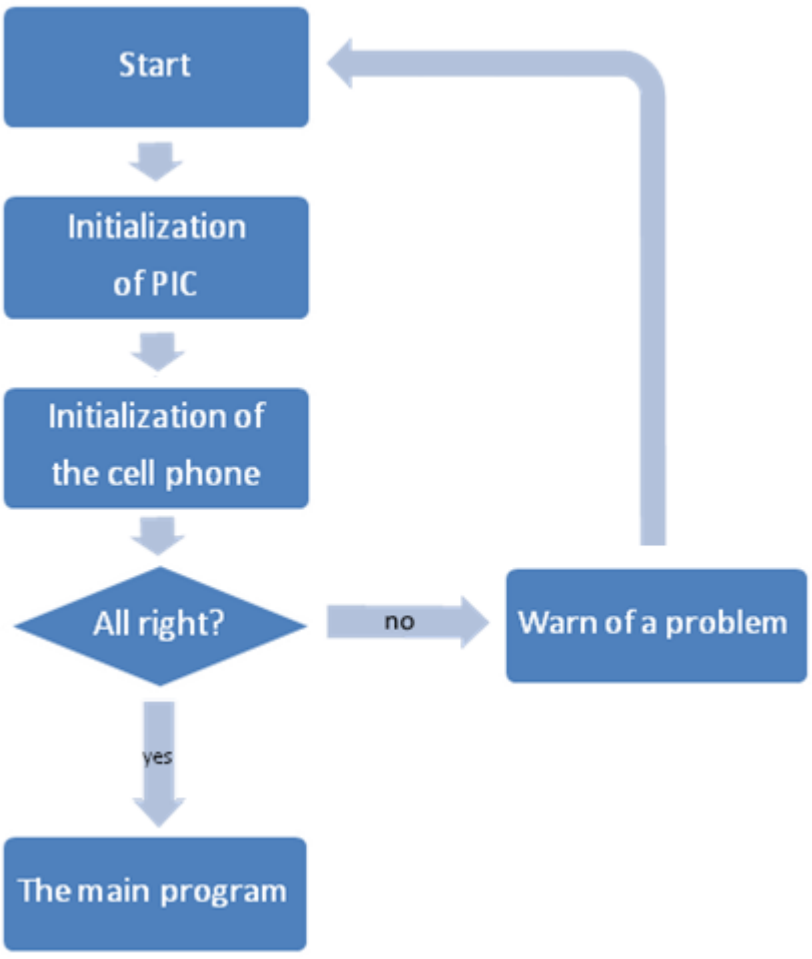


Figure 15: Code description - the Initialization

The Initialization of the PIC is really easy in the IDE mikroC, because we choose the PIC family, the frequency of the used crystal and other configuration at the beginning by the creation of our project. So in this part we set up all pins of PIC. Actually, we use only the PORTB and two pins on the PORTC. The first pin on the PORTB we use like an input from the Optical element. The other thee pins we use like output for the control diodes. We do not need to set up the pins RC7/RX and RC6/TX for USART communication, because the IDE will do it in behalf of us, when we call the library function Usart_Init() as described above.

All pins without using we leave on the low value as the inputs.

3.3.2.1 The main program

We can describe the main program by means of a never-ending cycle. The cycle has three basic parts. At the beginning we should say that the *assistant* is the first phone number and the *friend* is one of the first four phone number in the SIM Card. The Sleeping mode means that the PIC is going to the sleep. It awakes with the switching off and on the device.

Then the main cycle looks out:

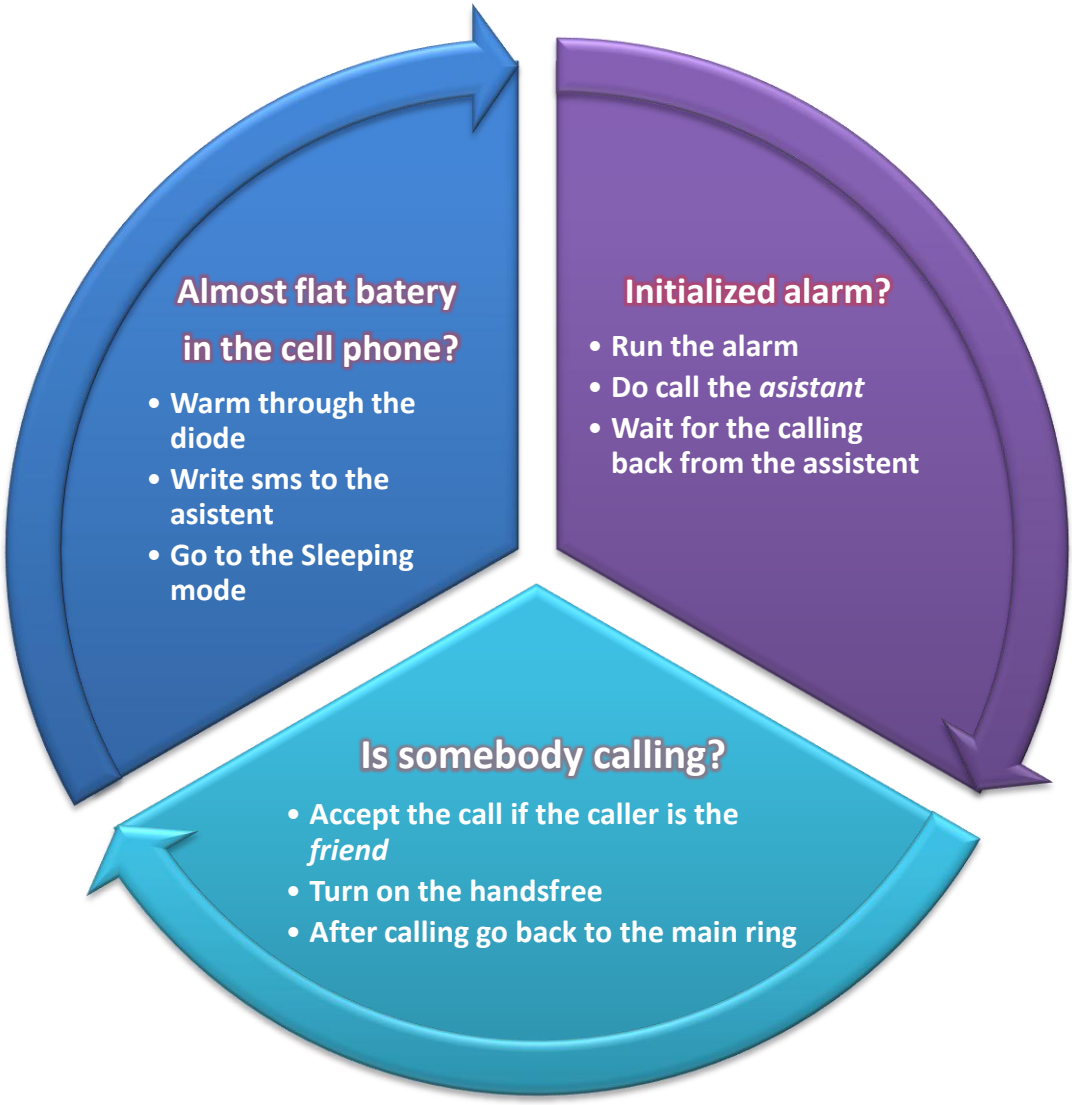


Figure 16: The main program

- In the first part the program is checking the value on the pin RB0, where is the output from the optical element. If the value is high, the alarm is not initialized and the program is checking this value 10 seconds more and if it stays on high the program will return to the cycle.

But if the program will find the low value on the pin RB0 in this 10 seconds, it will warn via the blinking diode and sounds from the buzzer about 10 seconds. If the alarm will be inactivated via the optical element within these 10 seconds, the alarm will turn off and the main program will return to the cycle.

After the alarming through the blinking diode, the program will send the AT command to the cell phone for calling help. This call takes about 10 seconds. On closing the call the program

is waiting for the call back. If the assistant calls back, the hands-free is switch on. After the calling, the program does not go to the main cycle, but is waiting for the other call from the assistant.

- The second part deals with monitoring incoming calls. It accepts only a call of the *friends*. Other calls are turning down. If the call is accept the program switch on the hands-free and after calling it goes to the main cycle.
- The last part consists of the control the battery charge. If the battery is almost flat the program will inform the *assistant* about this situation through the sort message system (SMS). And then it gets across the PIC processor to the Sleeping mode, because due to the discharge the battery in the cell phone the device is not be able function more.

The compete code in C++ and also in assembler is on the enclosed CD.

4 Results

With reference to my task **[2]**, I got acquainted with PIC microcontrollers, AT commands for GSM modules, and a phase-locked look (PLL) electronic control system. Then I chose suitable types of microcontroller, the GSM module and design the whole schema with the PLL. When my schema was working on the developing board, I make the printed circuit board of this device and I start to work on the software part. Finally I can write I fulfill the goals of this project **[2]** without any problems or difficulties and the whole device is ready for real using.

5 Future

This work is not planed to be extended in the future, although the selected microcontroller and the whole project have more possibilities of how to use this device; for example, the usage of this alarm for controlling a cottage, house, or car.
So I leave this project free for other students and I agree if somebody wants to extract and extend my work.
In the next year I am planning to cooperate with the company AREM PRO. My work will be to design the device for controlling if the cutting beam passed through the material. And I will use the knowledge about the PIC and USART communication from my project this year.

6 References

6.1 Documents

[D1] mikroElektronika, mikroC Users Manual [online], C compiler for PIC microcontrollers, c2006
URL: < http://www.mikroe.com/pdf/mikroc/mikroc_manual.pdf>
[D2] Microchip Technology, 39582b.book [online], Data Sheet, c2003
URL: <http://ww1.microchip.com/downloads/en/DeviceDoc/39582b.pdf>
[D3] Intersil Corporation, HIN231, HIN232, HIN236, HIN237, HIN238, HIN239, HIN240, HIN241 [online], +5V Powered RS-232 Transmitters/Receivers, c2000
URL: <http://www.intersil.com/data/fn/fn3138.pdf>
[D4] Richard Hammerl@CHICKEN, Tutorial EAGLE 4.1 -- 2nd edition [online], c2004
URL: <ftp://ftp.cadsoft.de/eagle/program/4.16r2/tutorial-eng.pdf>
[D5] National Semiconductor, LM567/LM567C Tone Decoder, c2004
URL: < http://www.national.com/ds/LM/LM567C.pdf>
[D6] S35i C35i M35i ATC Command Set Ver01, c2000
URL: < http://bramo.ic.cz/soubory/atc35.zip>
[D7] ST Microelectronics, L7805CV datasheet, c2004
URL: <http://www.datasheetcatalog.org/datasheet/stmicroelectronics/2143.pdf>
[D8] Copyright © 2008 Vodafone Czech Republic a.s., map of signal coverage
URL: <http://www.vodafone.cz/consumer/switch/network/check/map.htm>

6.2 Software

[S1] mikroC, mikroElektronika C compiler for Microchip PIC microcontrollers, version7.0.0.3, c2002-2005
URL: <http://www.mikroe.com/en/compilers/mikroc/pic/>
This software is owned by mikroElektronika Associates and is protected by copyright law and international copyright treaty. Therefore, you must treat this Software like any other copyrighted material (e.g., a book).
[S2] EAGLE(Easily Applicable Graphical Layout Editor), Version 4.16r2 for Windows, Light Edition, Copyright (c) 1988-2006 CadSoft, All rights reserved worldwide
URL: <http:// www.cadsoft.de/freeware.htm>