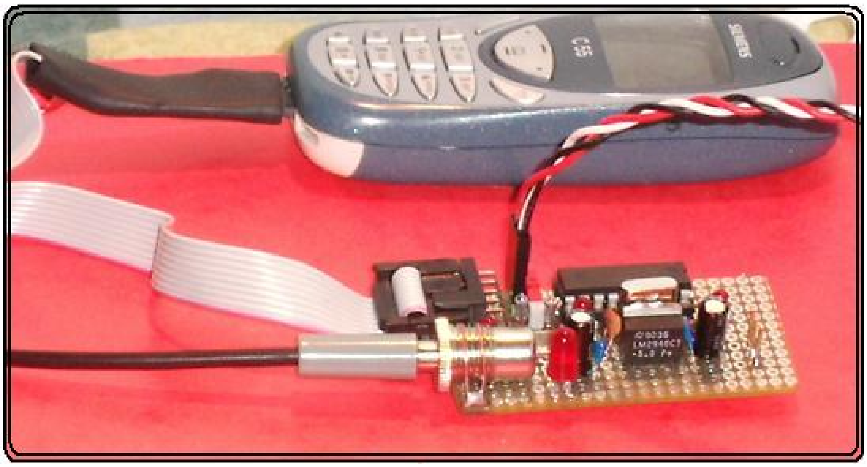
Janis Alnis

How to build your own GSM alarm



Janis.Alnis@mpq.mpg.de

2006

Janis Alnis

How to build your own GSM alarm

This 33 page e-book is an instruction how to get started on developing your own GSM alarm module to be attached to very common *Siemens* or other GSM mobile phones via headset connector.

Application areas include car, house, garage alarms, unattended monitoring of temperature, fault acknowledgement. The GSM alarm can be used in opposite direction as a world-wide remote control to control electrical processes somewhere far away, like switch on heating in home before you arrive, or control your scientific experiment while away from the lab.

It is a great collection of experience that will save you a lot of time to get acquainted with this interesting new field of hobby electronics. It will teach you the AT commands used to control the mobile phone and how to build a small control board based on PIC16C84 that sends commands to the Siemens C55 mobile phone to call any given number (for example your second mobile phone) when a sensor (switch, motion detector) is activated.

Experience in building electronic circuits and programming is required at high school student level.

The author of this e-book Dr. Janis Anis is doing research in Laser Physics works in the Nobel Prize winner laboratory. He also reads lecture courses for students, and has 20 years experience in hobby and ham-radio electronics. Last two years he is interested in this new exciting area of GSM phone based interfaces.

E-BOOK publishing, (2006)

The author is not responsible for any possible mistakes in the information supplied in this book.

CONTENTS

Appendix

Introduction

Mobile phones are developing very rapidly over the last decade. For electronics hobbyist it is fashionable to learn how to program them and make simple micro-controller based interfaces that can be used as a world-wide remote controllers.

HOW IT WORKS:
Mobile phones can be controlled via the accessory connector. A small control board sends commands to the phone to call a given number (another mobile phone) when the sensor (switch, motion detector) is activated.

Simple application areas include

- Car, garage, house, boat, summer house alarms
- Reboot remote server
- switch ON a VCR recording
- autonomous gate opener after receiving a dial signal
- motion detection sensor
- Unattended monitoring of temperature, fuel level sensors
- Some device fault acknowledgement if device halts and does not reset the watchdog timer
- Wending machine that dials your number when empty

The GSM alarm can be used in opposite direction as a world-wide remote control

- switch on heating in home before you arrive, or
- control your scientific experiment while away from the lab.

Advanced applications

- solar panel weather station
- transfer remote snapshot camera pictures
- radio-controlled model
- GPS tracking
- data logger
- vegetable/ flower production
- chicken production

The applications grouped by communication type

- simply dial the number of other mobile phone – costs nothing if other side does not pick up
- send SMS
- send E-mail

In the first chapter I will explain how to communicate with mobile phone from a PC using AT command set.

Next we will build a data cable communicating between PC and mobile phone. This will allow to practice and see how different commands work.

And, finally, we will build a Microchip PIC16C84 microcontroller based circuit that sends the same commands to the phone as a PC. This circuit will dial the number of your other mobile phone when the sensor input is activated. Of course, you can modify the example according to your requirements.

CHAPTER **1**

Theory of AT command set

Most GSM/UMTS mobile phones that work at 900 MHz, 1800 MHz and 1900MHz.include an internal modem. GSM cellular phones can accept an extension of the AT command set that is similar to one used for computer modems (Hayes modem commands).
Besides SMS management, the extension embraces many functions: mobile identification, address book management, signal strength level, call waiting and forwarding, error report, battery charge monitoring, ringing tones, volume, plus non-standard commands introduced by manufacturers like keyboard emulation. Extended sets are used by some phone manufacturers who allow to modify logos, ringtones and calendar entries.

The standard for GSM cell phones (including the AT modem command set extension) are published on-line by ETSI (European Telecommunication Institute). AT command set for general phone control you can download here 3GPP TS 27.005
http://webapp.etsi.org/key/key.asp?GSMSpecPart1=27&GSMSpecPart2=005
and for SMS and 3GPP TS 27.007
http://webapp.etsi.org/key/key.asp?GSMSpecPart1=27&GSMSpecPart2=007

Then you have to read a lot, but remember that not all commands are a must have, actually most are optional.

For everyday programming it is actually quite enough to use a compilation of the most commonly used AT commands.

Calling commands

AT	Attention
ATD	Dial Command
ATH	Hang Up Call
ATL	Monitor Speaker Loudness
ATM	Monitor Speaker Mode
ATO	Go On-Line
ATP	Set Pulse Dial as Default
ATT	Set Tone Dial as Default
AT+CSTA	Select Type of Address
AT+CRC	Cellular Result Codes

SMS Commands

AT+CPMS	Preferred Message Storage
AT+CMGF	Message Format
AT+CSCA	Service Centre Address
AT+CSMP	Set Text Mode Parameters
AT+CSDH	Show Text Mode Parameters

AT+CSCB	Select Cell Broadcast Message Types
AT+CSAS	Save Settings
AT+CRES	Restore Settings
AT+CNMI	New Message Indications to TE
AT+CMGL	List Messages
AT+CMGR	Read Message
AT+CMGS	Send Message
AT+CMSS	Send Message from Storage
AT+CMGW	Write Message to Memory
AT+CMGD	Delete Message

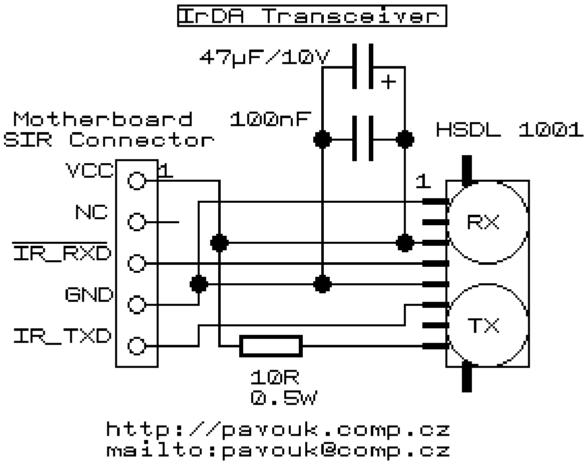
Phone control operations

AT&Y	Select Set as Power up Option
AT+GCAP	Request Complete Capabilities List
AT+GMI	Request Manufacturer Identification
AT+GMM	Request Model Identification
AT+GMR	Request Revision Identification
AT+GSN	Request Product Serial Number Identification
AT+CGMI	Request Manufacturer Identification
AT+CGMM	Request Model Identification
AT+CGMR	Request Revision Identification
AT+CGSN	Request Product Serial Number Identification
AT+CMEE	Report Mobile Equipment Error
AT+CPAS	Phone Activity Status
AT+CPBF	Find Phone Book Entries
AT+CPBR	Read Phone Book Entry
AT+CPBS	Select Phone Book Memory Storage
AT+CPBW	Write Phone Book Entry
AT+CSCS	Select TE Character Set
AT+CSQ	Signal Quality

CHAPTER **2**

Serial communication between PC and a phone

First of all you need a physical connection to the phone, like a data cable, PC Card (PCMCIA), Infrared (IrDA) or Bluetooth. The simplest method is to use IRDA. Here is a simple home-made IRDA adapter
http://pavouk.org/hw/irda.html



Use HyperTerminal which comes with Windows
Start->Accessories->Communications->HyperTerminal
to start a HyperTerminal window.

To send commands use Terminal emulation tools like HyperTerminal (Windows)
http://www.hilgraeve.com/htpe/download.html

Open a serial connection to the phone via the port where the phone is attached. Settings:

Data Rate:	9600 or 19200 baud
Parity:	None
Data Bits:	8
Stop Bits:	1
Software Flow Control:	Off

There exists a special software for communication with Siemens mobile phones. Called SiMoCo. It is a shareware and a trial can be downloaded for example here
http://www.softpedia.com/get/Mobile-Phone-Tools/Siemens/Siemens-Mobile-Control.shtml

SiMoCo can read the phone address book, call the numbers, send SMS. SiMoCo has a window for monitoring the commands sent through the COM port

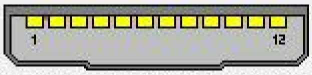
CHAPTER **3**

The Accessory Connector of the Siemens Mobile Phones

We want to build a data cable between PC and phone connected to hands-free accessory connector since we need to practice and be sure we understand the serial communication signals before we build a microprocessor based circuit.

There is a great database of different mobile phone connector schematics on Pinouts.ru
http://pinouts.ru/cgi-bin/view_filt.cgi?text=siemens&lang=ru

Connector for Siemens A35, A36, A40, C25, C35, C45, M35, M35i, M50, ME45, MT50, S25, S35, S45, SL-42, SL45, 3118 cell phones Applicable to A/C/S/M/SL 2x, 3x, 4x, 5x, x25, x35, 3508, s2588.



12 pin Siemens Lumberg cellphone special connector at the mobile phone

Pin	Name	Dir	Description
1	GND	-	Ground
2	SELF-SERVICE	in/out	Recognition/control battery charger
3	LOAD	in	Charging voltage
4	BATTERY	out	Battery (S25 only)
5	DATA OUT (TX)	out	Data sent
6	DATA IN (RX)	in	Data received
7	Z_CLK	-	Clock line for accessory bus. Use as DCD In data operation
8	Z_DATA	-	Data line for accessory bus. Use as CTS in data operation
9	MICG	-	Ground for microphone
10	MIC	in	Microphone input
11	AUD	out	Loudspeaker
12	AUDG	-	Ground for external speaker

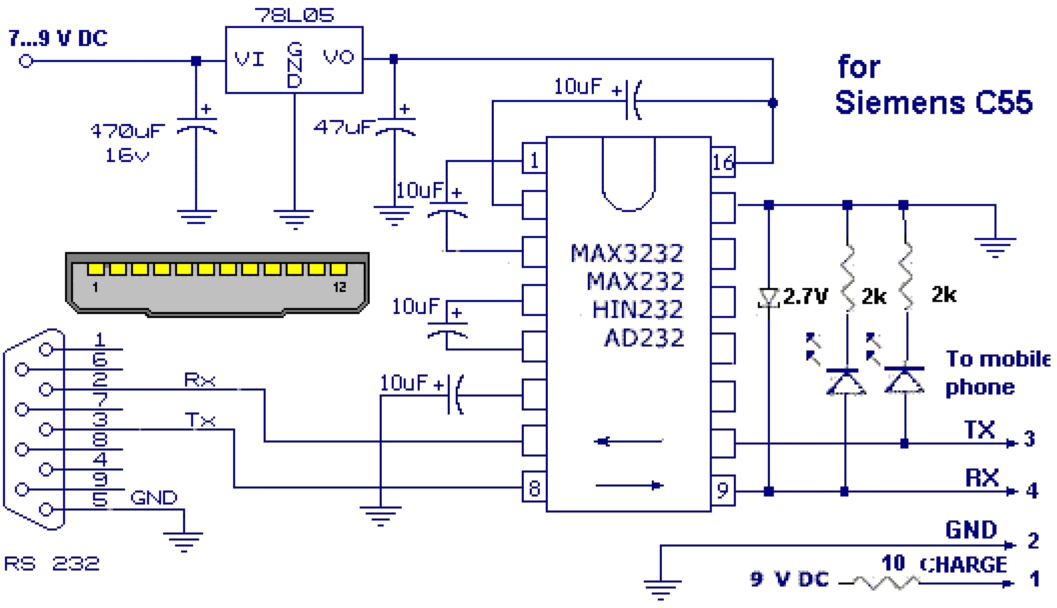
CHAPTER **4**

Data Cable between PC and Siemens Mobile Phone

Now let’s build a data cable that will allow to see the communication signals using oscilloscope or LEDs. I use my Siemens C55 mobile phone, but other models can be used as well. Other manufacturers like Ericsson and Nokia should have a similar interface, but you need to check connector pinouts.

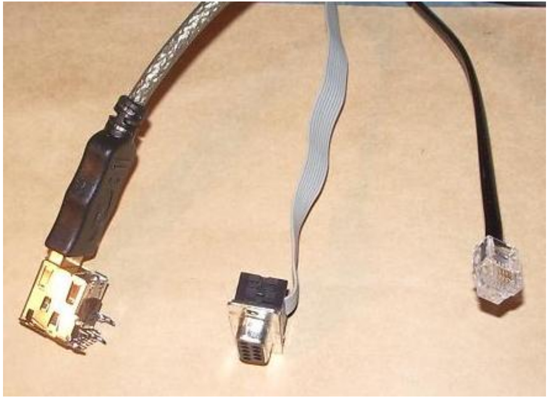
The phone is connected to PC COM port with a data cable. The newer notebook computers do not have COM ports any more and one needs additional USB to COM emulator cable.

This data cable converts RS232 ±3…15 V levels to 0…+3 V levels that are needed for most of the phones. Without a limiting Zener diode the phone locks-up because the voltage supplied by Max232 chip is 5V but internal voltage of the phone is 3V. This is not a damage of phone because the inputs of the phone connector are protected by diodes and causes phone to stop responding until it is switched OFF and ON again.



Siemens phone data connector I built first from a charger cable. It only has 2 pins and it takes some work to gluing in 2 more gold-plated contacts taken from a PC RAM memory connector. A much better solution is to buy headset and use the connector since it has all the 12 pins. Headset is usually cheaper than to buy just a new connector in your electronics shop.

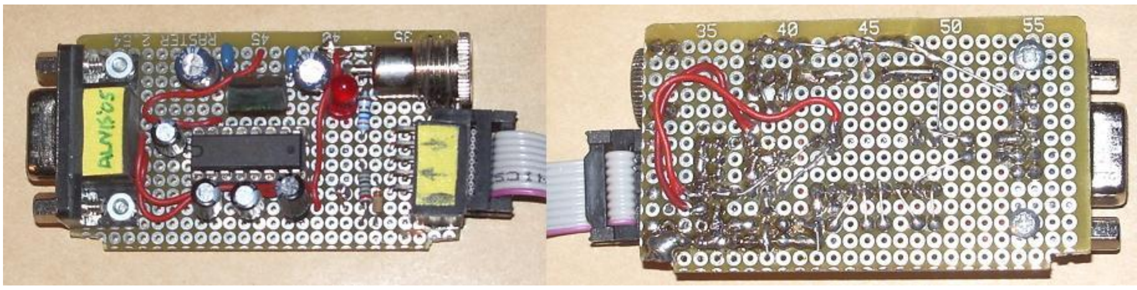
The cable between MAX232 board and phone is used from a 10 pin used RS232 connector connecting slot on the back of PC with the PC motherboard. Also a USB connector or phone line connector can be used.



For continuous operation external power supply can be used or +5 V power from an USB connector necessary for phone charging.

You can make your own program that controls your mobile phone from PC. For example you run long calculations on your PC and at the end you want to receive notice that calculations are finished.

The windows XP does not allow direct sending of data into the address of COM port, so you need some high level programming language like Delphi. On Win 98 and older one can use in principle Qbasic or Pascal.



CHAPTER **5**

Sending your first commands over the data cable

Connecting to a mobile and issuing a few AT commands like dial number is straightforward, but SMS handling can be done after some practice.

After setting **HyperTerminal** to 9600,N,8,1 type the AT command:

AT

And the cell phone should answer:

OK

This is the simplest command to tell the mobile phone to go on attention. It doesn't do anything. However, this is also a means to test if the phone responds on the baud rate and all the serial settings.

Now make a phone call to your mobile phone. The GSM phone will send a

RING

RING

message to PC, when it picks up an incoming call from the network. To accept the call, you may use the *ATA* command.

With that, we have the means to communicate to the phone via serial port. Thus, we could build a system to remotely send or receive data.

Now lets try to call somebody.

ATD123456789;

OK

Please not that *ATD123456789* starts a data call and a *ATD123456789;* starts a voice call. The semicolon on the end of the string is the small difference between a voice call and a data call.

ATZ to hang-up.

The following are some simple AT commands to manage SMS

The *AT+CMGR=1* command reads the SMS message at the index location of 1. Each SMS, when they arrived are stored in indexed memory location of the mobile phone.

For more details look at USBDeveloper homepage

http://www.usbdeveloper.com/GSMPage/gsmpage.htm

With the command *AT+CNMI* you can switch the GSM modem in a mode that you will give you a "String" with every incoming SMS. This command is described in paragraph 3.4.1 of the GSM 07.05 specification. Alternatively, the command *AT+CMGR* can be used to poll the modem for new messages.

For a very good explanation please look at the webpage of Alexander Traud. He describes how to read numbers from phonebook
http://www.traud.de/gsm/numbers.htm
For examples how to read SMS entries
http://www.traud.de/gsm/sms.htm
Reading and sending SMS is more difficult because all the characters are encoded in so called PDU format with fixed length. A good explanation how to read SMS and send a pre-stored SMS is given by Alberto Ricci Bitti, the developer of Tiny Planet GSM phone controller:
http://www.riccibitti.com/tinyplanet/tiny_article.htm

```
AT+CMGF=1
OK
```

This sets the phone to "SMS Text mode" which can be read more easily, but not all phones support it. Value 0 is the PDU mode which is a rather difficult way to organize SMS, but should be used in professional programs for editing the SMS.

```
AT+CMGL="ALL"
…
```

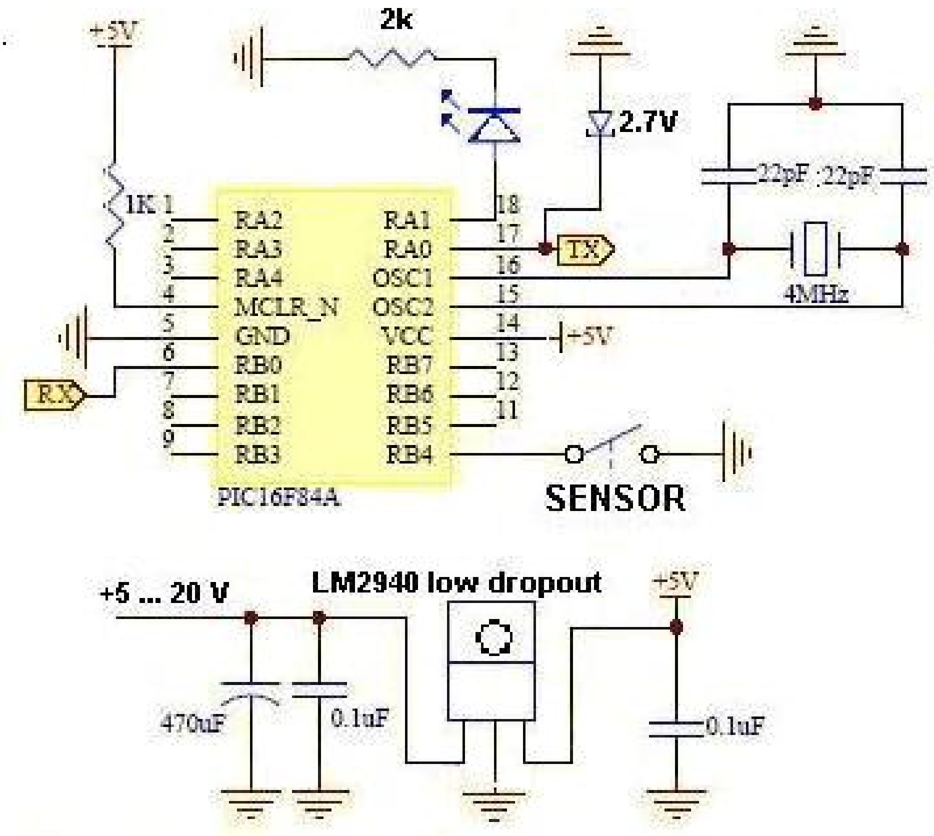
This displays all SMS on the internal phone storage. Some phones have problems with this command sequence.

Using this technique, you can select the commands relevant to your application, simulating manually the intended algorithm before putting it on a microcontroller.

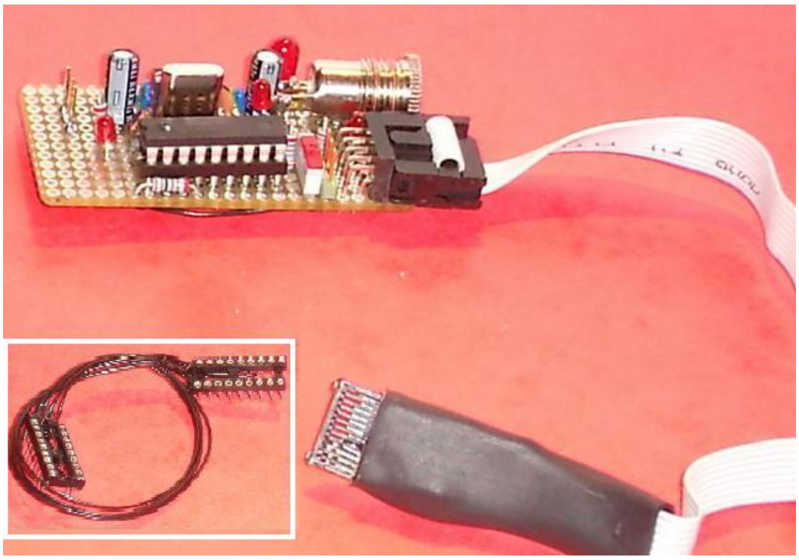
CHAPTER **6**

GSM Alarm Board Construction

Our next step is to construct a small control board attached to phone via accessory connector and based on a single chip processor PIC16C84A or PIC16F84A. Please download the datasheet from Microchip.com http://ww1.microchip.com/downloads/en/DeviceDoc/35007b.pdf
The PiC will be programmed to make a call when the sensor input is activated.



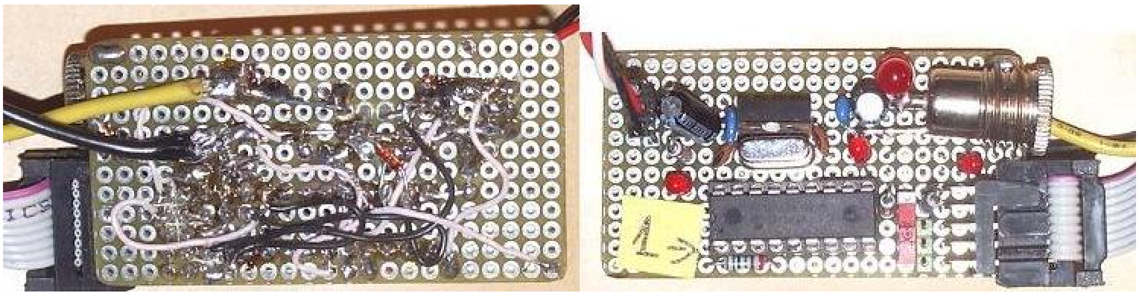
The circuit diagram is simple. The RX, TX and GND signals go to the phone via the cable that we made for the data cable board. There are many free pins that can be programmed for additional functions. It is assembled on a prototyping breadboard allowing to modify the schematics easily.



Note a 5-wire in circuit programming adapter works and greatly saves time on removing the processor from sockets.

Alarm board consumes 33 mA from 5...20 V supply. For battery operation it should be less and PIC should be programmed to enter the sleep mode.
If the sensor has no relay output, but just voltage output, you can build input with an opto-coupler input.

This picture shows the GSM Alarm circuit board from both sides.



Parts cost estimation is as follows:
PIC 5 EUR,
motion detector 10 EUR,
other components 10 EUR,
programmer 10 EUR,
phone + prepaid SIM card for O2 in Germany 50 EUR + 10 EUR/year.

Totally the GSM alarm system costs less than 100 EUR. The commercial systems cost from 200 to 400 EUR.

CHAPTER 7

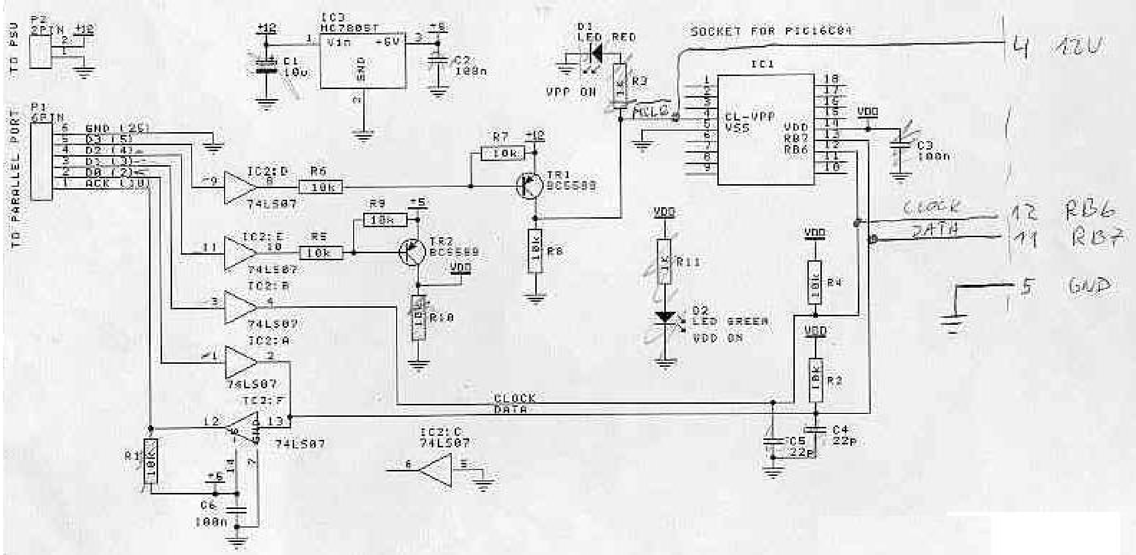
PIC microcontroller programmer

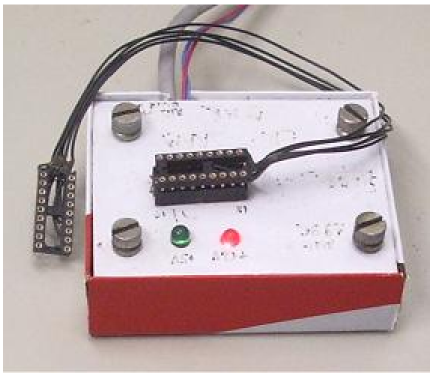
The programm for PIC can be written much more easier in C language than in Assembler. You can download a free compiler CC5X_C_compiler from http://www.bknd.com/

The C-compiler generates a Hex file with data to be put in PIC memory. The Hex file can be opened with a freeware Oshon PIC programmer on Win98. http://www.oshonsoft.com/picprog.html

Win XP does not allow to address the PC parallel port directly, but newer programs that run also on WinXP.

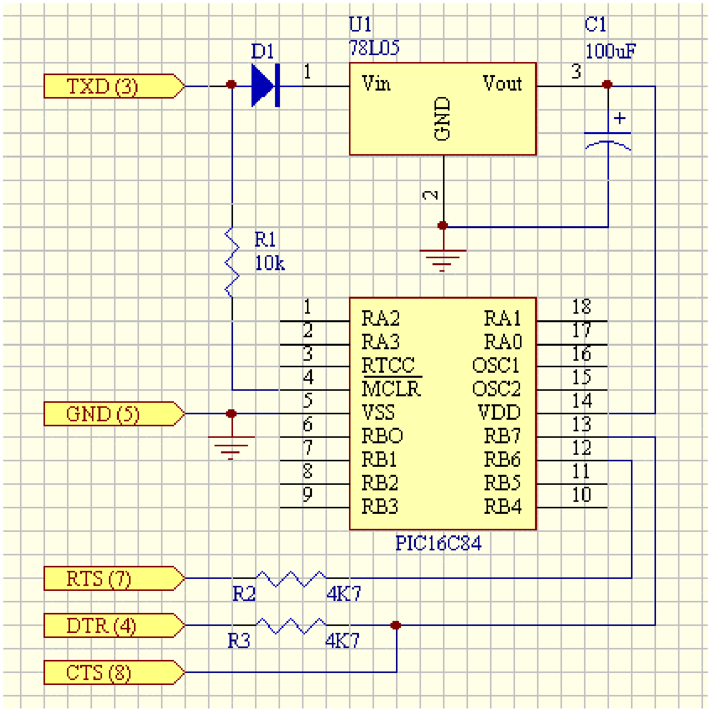
The Oshon PIC programmer software allows to use the classic David Tait's PIC programmer hardware attached to the LPT printer port. My long-living programmer schematic and design is shown below. Default settings used
D2 vdd noninverted
D3 vpp MCLR noinverted
INVERT CLOCK RB6 D1
DOUT RB7 DO
DIN S6





Notice a 5-wire in-circuit-programming adapter that saves time on removing the processor from sockets.

There are many programmer versions around and they are all quite similar because the standard is set by *Microchip*. Like this one http://www.rentron.com/Myke4.htm.



If you are making the first steps also in microcontroller programming my advice is try to make some simple code like blinking LED. The way to test the programmer is to write something and read out the PIC and verify if the code saved on PIC is the same or to erase chip and read out it. It should be empty.

CHAPTER **8**

Software for our GSM Alarm

The code that you program into the Microchip PIC checks the sensor switch and, if it is activated, calls the number +498932905292. After a delay the program will tell phone to hang up. Then it waits approximately 1 minute and, if sensor still activated, will call again.

Your phone will call me in Germany if you don't change the C and HEX code given in the Appendix and I will be glad to hear your circuit works.

You can later modify the program code that it calls the number in the first address in the phonebook.

The code is written in C and supplied in the Appendix. It is a modified serial communication example code available from Microchip.com and USBsolutions.com websites. There are several functions in the code not used in our simple algorithm, but left for you in case you need them when you make your own program.

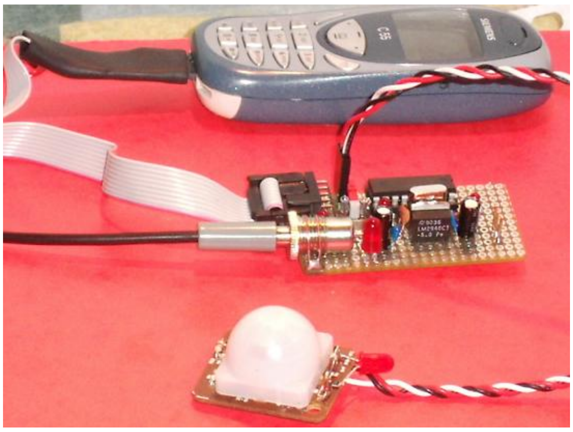
The programmer will need to know the PIC control code. It is **19H** since we run on a quartz oscillator.

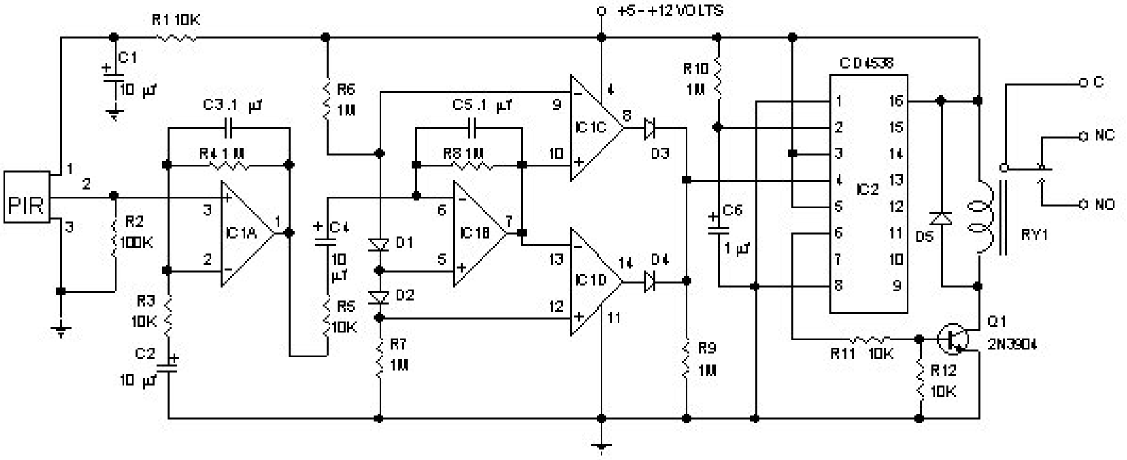
Osc XT 4 MHz
D4 PRotCode 1
D3 Timer 1
D2 WDT 0
D1 XT 0
D0 XT 1

CHAPTER **9**

Motion detector as a sensor

Motion detector (Conrad electronics Germany) current consumption is15 mA. I have made a 5 m long extension cable between the sensor and the phone for facilitating hiding of the phone. Long cable is also useful if sensor picks up transmitter interference from the phone.





IC1 = LM324
IC2 = CD4538

PIR = PIR325
D1 - D5 = 1N914

R10 Kohms X C6 mfd = ON time ms
RY1 - coil voltage rating = V+ voltage

MOTION DETECTOR

Here is a general circuit running on 5 V. It is not worth to make the circuit but better to buy a complete module. The schematic is just for reference to illustrate the working principle of piro-electric motion sensor.

CHAPTER **10**

Autonomous power from Lithium accumulator pack

If you want to monitor activity in your fruit garden where is no electricity please read this chapter.

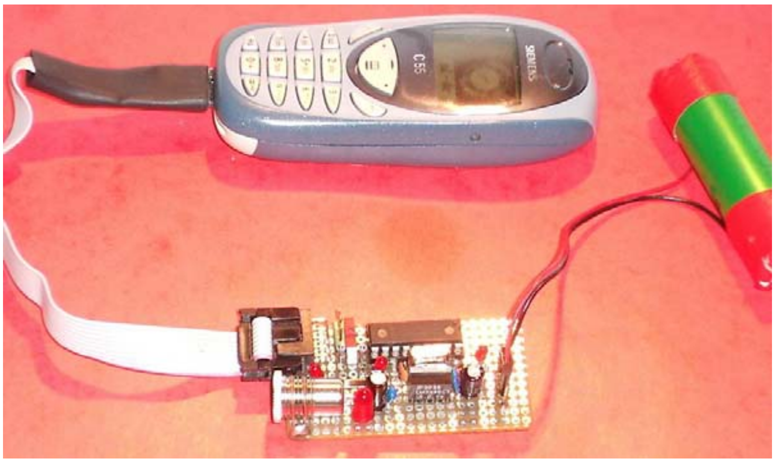
To my surprise, I have observed that the board described in the previous chapter runs also without any power supply, because PIC power line gets charged to 1.5 V from mobile phone serial output line that is on 2.5 V when no data are transmitted. At this low voltage the current consumption is negligible but PIC still works and can send commands to dial number (LEDs however do not emit light at this low voltage). So the alarm still works without battery and one can use a switch or a wire loop as a sensor.

The mobile phone can be made autonomous for longer time or for power failure backup using external battery pack. External battery can be connected to the charging input of phone or less energy wasting option is to solder external Li battery in parallel to the phone battery.

2000 mAh Li cells can be obtained from morally old notebooks and 8 cells can be soldered together in parallel to make amazing 16000 mAh pack. It can keep the system running for 1 month or longer.

The charging of the Li batteries can be done from a phone built in charger if a connector is made parallel to the original phone battery and external pack is connected to it.

The charging of Li rechargeable batteries is also special. It is charged at a constant voltage of 4.2V and the charging current decreases from maximum value to small value when the cell is charged full.



The Li cells can keep their charge for much longer time compared to NiCd or NiMH batteries because the self-discharge rate is low. Li cells do not have a so called “memory effect” that reduces capacity if the cell was not fully discharged.

!!!

Li cells are very sensitive for overvoltage and can get fire if charged over 4.2V per cell. A prlonged short-short will cause heating and fire.
It is recommended to keep the cells in a metal container or in a flower-pot during charging.

The discharge below 3 V is decreasing the cell capacity. If deep discharge has happened, one needs to charge with very small current like 10 mA until the cell voltage reaches 3 V.

If your GSM alarm is autonomous a good way to hide it is to put it in several plastic bags with some silicone grains for moisture absorption like in some medical drug containers and to dig the package under ground. 10 cm works without any problems a few km from base station. If not far from base station you can have system in a metal box and phone will still work. I tried to put phone in a stainless cooking pot with cover and also in a microwave oven with closed doors and still could call it. The sensitivity of the mobile phone receiver is astonishing to be able to violate the Faraday cage shielding effect.

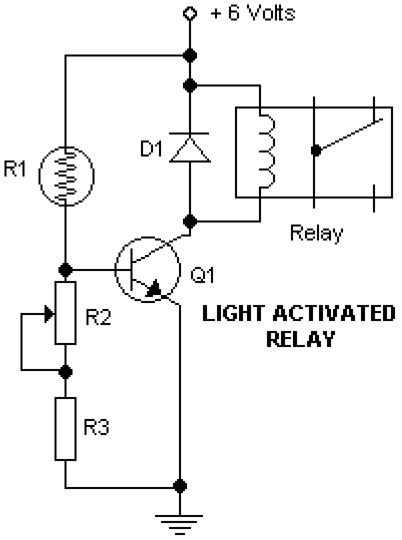
A World-wide Microphone

Sometimes you want to listen if the motor that you switched on is still running. The you connect a headset and switch a phone in auto-answer mode. When you will call the phone will answer and you can hear what is going on.

CHAPTER **11**

Simple GSM remote control

There is a possibility to make a very simple GSM remote control without programming. Use a photodiode on photoresistor placed near the screen and an amplifier to activate a relay that further activates a time relay. The time relay for long delay times should be digital, for example using a PIC microcontroller with nested delay FOR loops in the programme. The possible application is to switch on heater before arriving to home or sauna.

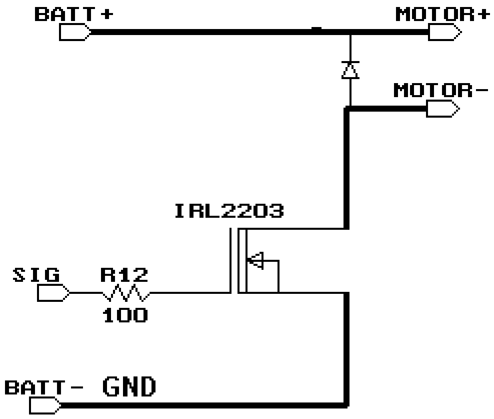


Parts List

D1 = 1N914 diode
Q1 = 2N2222 or
similar NPN
transistor
R1 = photoresistor
R2 = 50K variable
resistor
R3 = 1K
Relay = 5 to 6 volt
relay.

None of the parts of the light sensor are critical. Use potentiometer to adjust the level during light ON phase.

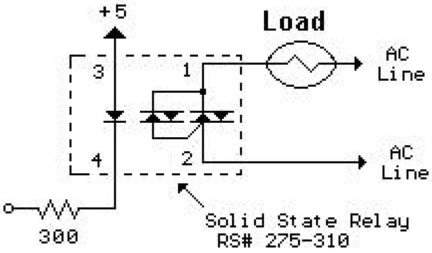
If you will make a GSM remote control you might use one of the following circuit ideas.

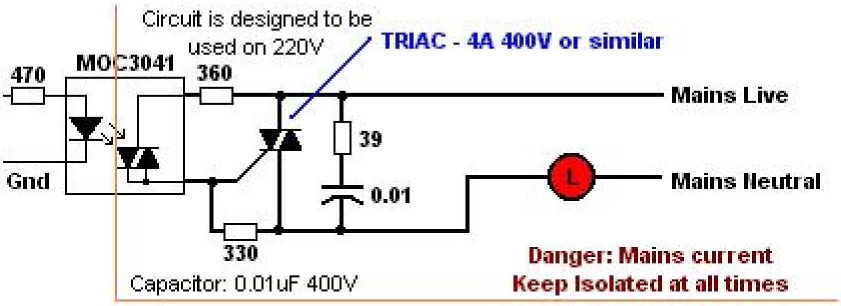


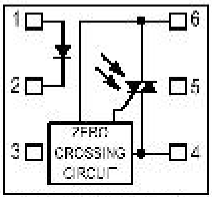
This circuit is used to control high current DC motors in radio-controlled electric model community. Several FETs can be put in parallel. They are special FETs that are steered with TTL logical signals while the load voltage to be controlled can be higher.

IRL2505 with TTL logical input can withstand 55V, 104 A, *R* = 0.008 Ω.
IRL2203 specifications are very little better: logical input, 30V, 116A, 0.007Ω

Below is a nice circuit using a solid state relay, but might be more expensive than just an opto-coupler and a triac.

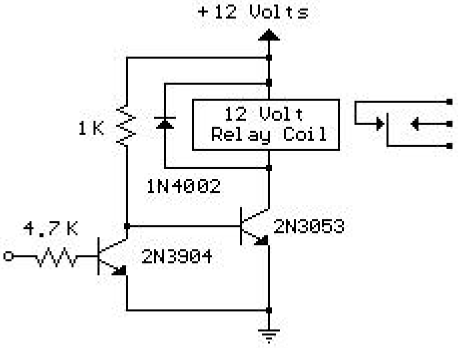






MOC3041 pinouts

This is a classic transistor circuit. One can even omit the input transistor.



CHAPTER **12**

Further Reading - Online Information Overview

The circuit described in this book was developed thanks to several GSM programming enthusiasts who placed a very valuable information online. I will try to make a short guide for you.

Tiny planet
Very good website how to make GSM remote control and SMS decoding
http://www.riccibitti.com/tinyplanet/tiny_intro.htm

Serasidis Vasilis
SMS controlled Relay card
http://www.serasidis.gr/circuits/smscontrol/smscontroller.htm

USB developer
Very good website with the programm similar to one used in this book.
http://www.usbdeveloper.com/Solutions/solutions.htm

Active Experts
They give the set of AT commands
http://www.activexperts.com/activcomport/at/

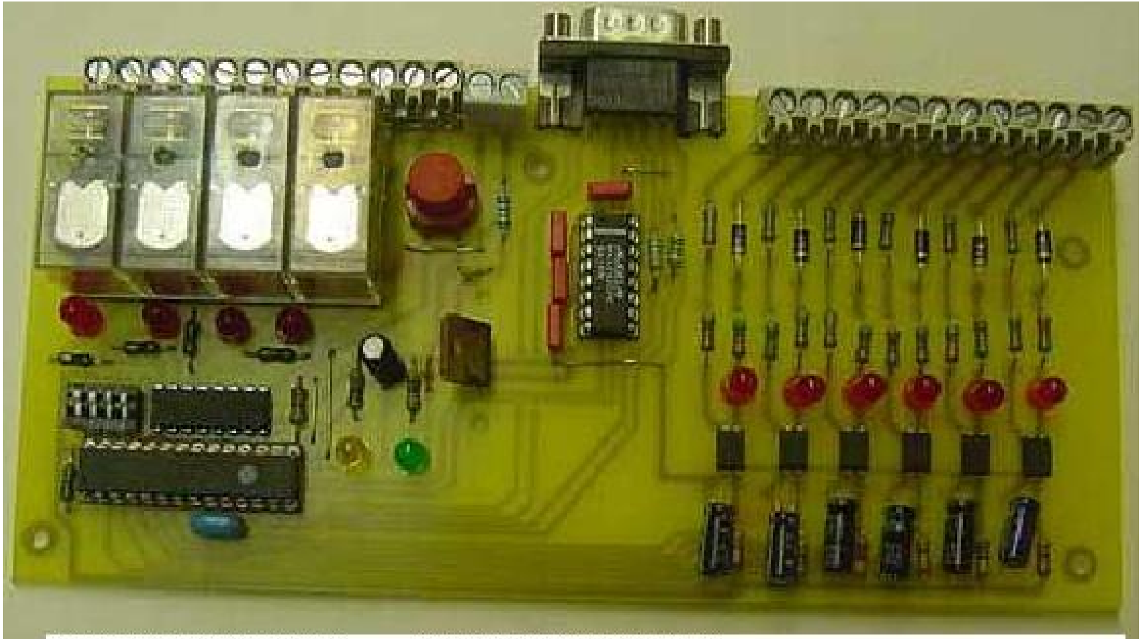
Advanced Wireless Planet
How to send HTML and EMAIL
http://www.gsm-modem.de/gsm-modems.html

SMS-RC
Switch on video recorder VCR for recording TV programme.
http://www.frisnit.com/sms/

Georg Traud Website
Good explanation of AT commands.
http://www.traud.de/gsm/index.html

There are many **commercial systems**, but they are quite expensive:
http://www.hinkel-elektronik.de/shop/7992.html
www.Conrad.com
http://www.dpspro.com/tcs.html

PROBYTE FINLAND relay card and optocoupled inputs -
http://www.probyte.fi/info/gsm.htm#Model%20seclection%20chart



CATALOGUE PICTURE from PROBYTE CONTROL GSM
GSM/SMS-hälytys- ja ohjauslaite

One can search in google words GSM alarm for new information.

Wish you success in your own developments!

Appendix 1

Advanced Example: EMAIL sending application

Source from
http://www.gsm-modem.de/smtp-gprs.html

Let' suppose you want to send with your embedded device an EMAIL by using a SMTP server.

Initial data:
Server to be contacted: smtp.gsm-modem.de
Application Layer Protocol: SMTP (RFC821)
Sender: "JOHN SMITH"<John.Smith@gsm-modem.de>
Receiver: "Receiver"<receiver@gsm-modem.de>
Subject: Email Test
Message body: this message is sent for test Easy GPRS feature. Hello World!

GPRS settings:
APN: internet.gprs
IP of GPRS device: dynamically assigned by the network
DNS: assigned by the network
USERID: Happy User
PASSWORD: EASY GPRS

Checking on the RFC990 the SMTP service we can found that the port 25 is dedicated for SMTP service, therefore our SMTP server will be waiting for incoming connections on that port and we will fix the EASY GPRS port to be contacted on the remote server exactly to 25. Second thing we have to discover is whether the transport protocol has to be TCP or UDP; on the RFC821 we can read that the SMTP Application layer protocol is meant to be on top of TCP/IP protocol, therefore the transport protocol choice will fall on TCP.

Now we have all the information needed to configure our system. With our micro controller we issue to the JOHN SMITH the following AT commands:
AT+CGDCONT = 1,"IP","internet.gprs","0.0.0.0",0,0<cr> (1-GPRS context setting)
AT#USERID = "Happy User"<cr> (2-Authentication setting)
AT#PASSW = "EASY GPRS"<cr> (2-Authentication setting)
AT#SKTSET= 0,25,"smtp.gsm-modem.de"<cr>(3-remote host setting)
For our convenience we store all these parameters with the command:
AT#SKTSAV
Now we can activate the GPRS connection and let the JOHN SMITH module contact the server:
AT#SKTOP<cr>
When we receive the CONNECT indication, then we are exchanging data with the SMTP server program on the remote host machine.Now following the SMTP

protocol we proceed with the HELLO presentation and mail delivery directly over the serial line (in green the data sent by us, in violet the one received from host):

```
220 smtp.gsm-modem.de ESMTP ; Thu, 5 Jun 2003 14:45:11 +0200
HELLO gsm-modem.de<cr><lf>
250 smtp.gsm-modem.de Hello [111.111.111.127], pleased to meet you
MAIL FROM: "JOHN SMITH"<John.Smith@gsm-modem.de><cr><lf>
250 2.1.0 "JOHN SMITH"<John.Smith@gsm-modem.de>... Sender ok
RCPT TO: "Receiver"<receiver@gsm-modem.de><cr><lf>
250 2.1.5 " John.Smith@gsm-modem.de "... Recipient ok
DATA<cr><lf>
354 Enter mail, end with "." on a line by itself
From: "JOHN SMITH"<John.Smith@gsm-modem.de><cr><lf>
To: "Receiver"<receiver@gsm-modem.de><cr><lf>
Subject: Email Test<cr><lf>
This message is sent for testing Easy GPRS feature. Hello World!<cr><lf>
.<cr><lf>
250 2.0.0 h55CjBVI020859 Message accepted for delivery
```

DONE! Easy as EASY GPRS.

Appendix 2

The C program for our circuit board

```
#pragma chip PIC16F84

/*
  Baudrate 9600 baud => 104.167 microsec. per bit
  TMR0 counts each 8th microsec. => 13.02 steps per bit
  ______       _____ _____ _____ _____ _____ _____ _____ _____ _____
        |_____|_____|_____|_____|_____|_____|_____|_____|_____|
         Start bit0  bit1  bit2  bit3  bit4  bit5  bit6  bit7  Stop
*/

#define _4_MHz    /* 4 MHz system clock */
#define BPS9600   /* 9600 bits per sec */

// optional items
#define UseTMR0
#define USE_CONST

#pragma bit RS232_out @ PORTA.0
#pragma bit RS232_in  @ PORTB.0
#pragma bit LED_Green @ PORTA.1
#pragma bit Button0   @ PORTB.4

#ifdef _4_MHz

  #ifdef BPS9600
      #define   TimeStartbit    4    //9600
      #define   BitTimeIncr     13   //9600
  #endif

#endif

unsigned long DelayCount;
char bitCount, limit;
char serialData;
char command; //first 4 bits are state
              // state 0 - expected = 0xAA
              // state 1 - expected = 0x55
              // state 2 - command
              // state 3 - bitwise complement of command

              //MSB 4 bits are the actual command.

#ifdef _4_MHz
 #ifdef UseTMR0
   #define delayStart   limit = TMR0;
   #define delayOneBit      \
     limit += BitTimeIncr;  \
     while (limit != TMR0) \
         ;
 #else
   #define delayStart   /* empty */

  #ifdef BPS9600
```

```
   #define delayOneBit  {      \
        char ti;               \
        ti = 30;               \
        do ; while( --ti > 0); \
        nop();                 \
     }  /* total: 5 + 30*3 - 1 + 1 + 9 \
           = 104 microsec. at 4 MHz */
   #endif

 #endif
#endif

void sendData( char dout)
/* sends one byte */
{
    RS232_out = 0;  /* startbit */
    delayStart
    for (bitCount = 9; ; bitCount--)  {
        delayOneBit
        if (bitCount == 0)
            break;
        Carry = 1;  /* incl. stopbit */
        dout = rr( dout);
        RS232_out = Carry;
    }
}

#define NText 20
const char text[NText] = "ATD+498932905292\r\n";

#define NText2 3
const char text2[NText2] = "ATZ";



const char ResetMobile[5] = "ATZ\r\n"; //reset mobile
const char NoEcho[6] = "ATE0\r\n"; //no echo
const char FwdMsg[13] = "AT+CNMI=3,3\r\n"; //forward all messages to
mobile phone
const char SndMsg[12] = "AT+CMGS=26\r\n"; //send message to mobile
phone, AT+CMGS=%d\r\n...PDU..
const char Dial[8] = "ATD000\r\n"; //Dial number 000
const char text1[5] = "State";
const char SMSMsg[54] =
"0011000881697193280000FF10C3B01C146487E56D50704C4FDBCB";

/*
#define delay12usec  {      \
     char titemp0;               \
     titemp0 = 30;               \
     do ; while( --titemp0 > 0); \
     nop();                 \
  }  /* total: 5 + 30*3 - 1 + 1 + 9 \
        = 104 microsec. at 4 MHz */
*/

#define delay10usec  {      \
     nop();               \
     nop();                 \
     nop();               \
     nop();                 \
     nop();               \
     nop();                 \
     nop();               \
     nop();                 \
     nop();               \
  } /* to create a total of 12 sec, but, adjusted for the looping
delay at 4 MHz */

#define delay12usec  {      \
     nop();               \
     nop();                 \
     nop();               \
     nop();                 \
     nop();               \
     nop();                 \
     nop();               \
     nop();                 \
     nop();               \
     nop();                 \
  } /* to create a total of 12 sec, but, adjusted for the looping
delay at 4 MHz */


#define delay13usec  {      \
     nop();                 \
     nop();               \
     nop();                 \
     nop();               \
     nop();                 \
     nop();               \
     nop();                 \
  }  /* to create a total of 13 sec, but, adjusted for the looping
delay at 4 MHz */


void delay_1millisec(void)
{
      int temp;
      for (temp = 0; temp < 100; temp++)
      {
           delay10usec;
      }
}

void delay_500millisec(void)
{
      int16 temp;
      for (temp = 0; temp < 500; temp++)
      {
            delay_1millisec();
      }
}

void delay_200millisec(void)
{
      int16 temp;
      for (temp = 0; temp < 250; temp++)
      {
            delay_1millisec();
      }
}


const char Digitext[10] = "0123456789";
void print_dec(unsigned char bytedata)
{
char i_temp, j_temp;

      if (bytedata > 100)
      {
            i_temp = bytedata/100;
            sendData(Digitext[i_temp]);

            j_temp = (bytedata/10);
            i_temp =  j_temp % 10;
            sendData(Digitext[i_temp]);

            i_temp = bytedata % 10;
            sendData(Digitext[i_temp]);
      }
      else if (bytedata > 10)
      {
            i_temp = bytedata/10;
            sendData(Digitext[i_temp]);
            i_temp = bytedata % 10;
            sendData(Digitext[i_temp]);
      }
      else
      { //single digit
            sendData(Digitext[bytedata]);
      }
}

const char Digitext[10] = "0123456789";

void send_SMS(void)
{
   int i;
   for (i = 0; i < 12; i++)
   {
      sendData( SndMsg[i] );
   }
   delay_500millisec();
   for (i = 0; i < 54; i++)
   {
      sendData( SMSMsg[i] );
   }
   sendData( 0x1A );
   sendData( '\r' );
   sendData( '\n' );
}

void main( void)
{
   char i;
```

```
    PORTB = 0x07; // xxxx xx11
    TRISB = 0xFB; // Bit 2: US_Enable_N (output), Bit 1: Ultrasonic
Recvd Bit (Input), Bit 0 : Rxd_RS232 (Input)

    PORTA = 0x01; // xxxx xx11
    TRISA = 0xF0; // xxxx xx10

    serialData = 0;

    OPTION = 2;   // prescaler divide by 8, for 9600 bps


   while(1)
   {
      LED_Green = 1;
      delay_500millisec();

for (i = 0; i < NText; i++)  {
        sendData( text[i]);  /* text string */
    }

      delay_500millisec();
      delay_500millisec();
      delay_500millisec();
      delay_500millisec();
      delay_500millisec();
      delay_500millisec();
      delay_500millisec();
      delay_500millisec();
      delay_500millisec();
      delay_500millisec();
      delay_500millisec();
      delay_500millisec();
      delay_500millisec();
      delay_500millisec();

for (i = 0; i < NText2; i++)  {
        sendData( text2[i]);  /* text string */
       }
      LED_Green = 0;
      delay_500millisec();
      while (Button0 == 0)
      {
         LED_Green = 1;
         delay_200millisec();
         LED_Green = 0;
         delay_200millisec();
      }
   }
}
```

Appendix 3
Code to be sent to the PIC programmer

```
:1000000066298D0065300D02031800348A010D0841
:1000100082074134543444342B3434343934383442
:1000200039343334323439343034353432343934 89
:100030003234 0D340A34003400344134543 45A34E8
```