Introduction assignment: Implementation of some polling and interrupt tasks:

To carry out the assignments with the microcontroller you need the following applications:

1. "MPLAB" with the PICC c-compiler installed.
2. "PicProg" is an application to program the microcontroller via the serial port (RS-232).
3. A terminal program "Tera Term" to communicate with the microcontroller (via the same serial port).

C:\Program Files\PICC\Ccsc.chm is a useful help-file for the C-compiler.

Installing the environment:

- Make a new folder on your N-drive.
- Copy the file: Test876.c from: **\\GENE\benb\voorStudenten**) to this folder.
- Copy the folder PicProg from: **\\GENE\benb\voorStudenten**) to your N-drive.

Start the MPLAB IDE. Choose Project à Project Wizzard à Next à Choose device PIC 16F876 à Next à Ö Show all installed toolsuites; choose active toolsuit: CCS C-compiler for PIC12/14/16/18 à location: C:\Program files\Picc\CCSC.exe à Project name: for instance "Polling"; Project Directory: the name of your new folder on your N-drive à Add the file: "Test876.c" à Finish

File "Test876.c"

```
à#include <C:\Program Files\PICC\Devices\16F876.H>
à#include <C:\Program Files\PICC\Drivers\CTYPE.H>

// Inform the compiler the clock frequency is 8 MHz
à#use delay(clock=8000000)

// Setup the RS232 communication
à#use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7,  bits=8)

int main(){
  char in_char;
à   delay_ms(10);           // Initialisation
  printf("Hello World\n\r"); // Outputs a string to RS-232.
  do{                        // Do forever
      in_char = getc() & 0x7F;
      printf(" %c %x\n\r", in_char, in_char);
  }while( TRUE );
  return 0;
}
```

Examine the source code above. The code lines market with à must be written in all your source codes.

Compile (Build all) your project. If successful there are a lot of files in your Project Directory.
Open with "Notepad" the file: Test876.LST and watch the assembly code.

PicProg

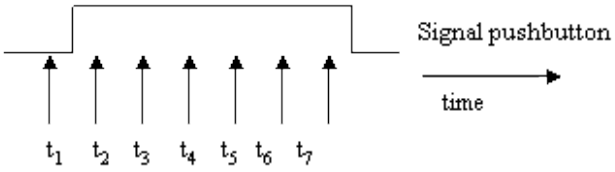
In the file Test876.HEX is the compiled code in hex-format. Load this file into the microcontroller using the application: PicProg. Program the Microcontroller. Push the button RESET and set the switch in the Program-state PGM. **Close** PicProg (to free the RS232 device for another application). Open the terminal program: Tera Term. Choose: Serial.
Set the switch on the microcontroller to mC. Push RESET to start the program.

Use the "Save as" option for the next assignment.

Assignment 1: **polling two pushbuttons**

Connect a pushbutton to pin B1 of the microcontroller and a second pushbutton to pin B2 of the microcontroller. Test the status of these pins frequently and send the message "Pushbutton B1 pressed" or "Pushbutton B2 pressed" to the screen of the PC every time you push the appropriate button. Send a message

once, *independent* the time you push the button. The order of these messages is related to the order you push the buttons.



A hint to beware you for a wrong implementation: the polling routine gives FALSE on t_1 and TRUE on t_2, so the message must be sent on t_3. Also a 'semaphore' must be 'raised' on that moment to prevent sending the message again and again on t_4 to t_7.

You get the state of pin B1 by:

```
short pin_state = 0  //short defines a 1 bit integer type
pin_state = input( PIN_B1 );
```

Use the "Save as" option for the next assignment.

Assignment 2: polling & interrupt
Connect a third pushbutton to pin B0. Extend the 'polling program' by writing an interrupt handler that send the message "Interrupt button pressed" every time you press the button.

```
#INT_EXT                        // the interrupt vector
extern_interrupt_handler(){     // specify the following function is an interrupt
handler
..                              // your code

..
}
```

The initialisation part of the function "main" must call: `ENABLE_INTERRUPTS( GLOBAL );` to makes interrupts possible as well as: `ENABLE_INTERRUPTS( INT_EXT );` to make the specific interrupt (here INT_EXT) possible.
Try to interrupt the message "Pushbutton B1 pressed" by pressing B0 i.e. try to receive a message like: "PushbuttoInterrupt button pressedn B1 pressed".

Assignment 3: Time measurement and fault tolerance.
Push a button during 2 seconds.
Measure the time you push the button with an accuracy of better then 0.5 % and sent the result to the PC.
Make use of the 16 bit timer of the microcontroller. Make an estimation of the accuracy of your implementation. What is the maximum time between two interrupts to achieve this accuracy?
Test your application with the 1 Hz output from the PIDAC-clock module (accuracy > 1: 10^5) cascaded with a four bit counter.
Explain the difference between your measured and your calculated value.
Hint: Make only use of "floats" in the printf-statement.