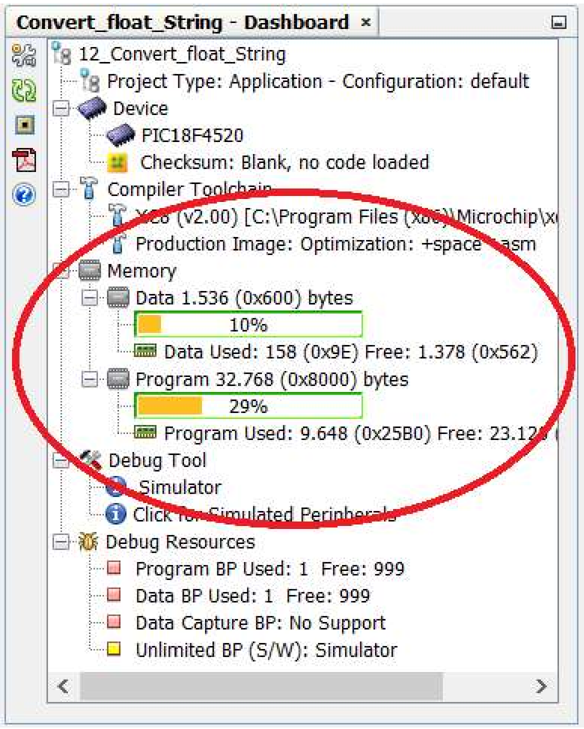
Aufgaben:

I. **Basic questions** (30 minutes, 30 points)
   1. The Embedded Systems Model. Give an example of layers used by you.
   2. Define compiler, linker, interpreter
   3. Byte ordering. Use of bridges

II. **Advanced topics (**15 minutes, 15 points)
   4. Explain the keyboard scan codes principle

III. **For the following code** (45 minutes, 45 points). Some lines are already commented. After successful compilation, one observes that large memory areas (Data SRAM + Program Flash) are used. The code is used to debug an ADC conversion followed by an LCD write. Instead of LCD, the UART console is used. The ADC is 10 bit, referenced at 5 V.
   a. Comment the relevant instructions, the overall source code (10 points).
   b. Explain why the program memory is 29% used and explain how this number can be reduced (10 points).
   c. Find ways to also reduce the data RAM areas (10 points).



```
Convert_float_String - Dashboard ×
12_Convert_float_String
  Project Type: Application - Configuration: default
  Device
    PIC18F4520
    Checksum: Blank, no code loaded
  Compiler Toolchain
    XC8 (v2.00) [C:\Program Files (x86)\Microchip\x
    Production Image: Optimization: +space  asm
  Memory
    Data 1.536 (0x600) bytes
      10%
      Data Used: 158 (0x9E) Free: 1.378 (0x562)
    Program 32.768 (0x8000) bytes
      29%
      Program Used: 9.648 (0x25B0) Free: 23.12
  Debug Tool
    Simulator
    Click for Simulated Peripherals
  Debug Resources
    Program BP Used: 1  Free: 999
    Data BP Used: 1  Free: 999
    Data Capture BP: No Support
    Unlimited BP (S/W): Simulator
```

```c
#include "Definitions.h"
#include <stdlib.h>
#include <stdio.h>
#include <stdarg.h>

void main(void) {
    unsigned int ADC_Result = 100;  //100 = 20.4 degrees Celsius
    float Temp;
    int Temp_digits,Temp_fractional;
    char buf2[3];
    char buf[2];
    Temp = (float)(((ADC_Result*16)/25)*8)/25;
    init_uart();                    // needed for the UART1 Output
                                    // UART1 must be enabled in MPLAB X
                                    // -> Simulator -> UART IO options
    Temp_digits = (int)Temp;
    Temp_fractional = (int)((Temp-(float)Temp_digits)*10.0);

    itoa(buf,Temp_digits,10);
    itoa(buf2, Temp_fractional, 10);
    printf("Temp = %2d",Temp_digits);
    printf("%c",'.');
    printf("%1d\n",Temp_fractional);
    return;
}

char *itoa(char *buf, int val, int base){
      char *cp = buf;
```

```
       if(val < 0) {
              *buf++ = '-';
              val = -val;
       }
       utoa(buf, val, base);
       return cp;
}

char *utoa(char * buf, unsigned val, int base){
       unsigned     v;
       char         c;
       v = val;
       do {
              v /= base;
              buf++;
       } while(v != 0);
       *buf-- = 0;
       do {
              c = val % base;
              val /= base;
              if(c >= 10)
                     c += 'A'-'0'-10;
              c += '0';
              *buf-- = c;
       } while(val != 0);
       return ++buf;
}

void putch(unsigned char data) {      // needed for printf
    while( ! PIR1bits.TXIF)           // wait until the transmitter is ready
        continue;
    TXREG = data;                      // send one character
}

void init_uart(void) {                 // needed for printf
    TXSTAbits.TXEN = 1;                // enable transmitter
    RCSTAbits.SPEN = 1;                // enable serial port
}
```

The `Definitions.h` is as follows. You do not need to comment it.

```
#include <xc.h>

// PIC18F4520 Configuration

// CONFIG1H
#pragma config OSC = XT          // Oscillator Selection bits (XT oscillator)
#pragma config FCMEN = OFF       // (Fail-Safe Clock Monitor disabled)
#pragma config IESO = OFF        // (Oscillator Switchover mode disabled)

// CONFIG2L
#pragma config PWRT = OFF        // Power-up Timer Enable bit (PWRT disabled)
#pragma config BOREN = OFF       // (Brown-out Reset disabled in hardware and software)
#pragma config BORV = 3          // Brown Out Reset Voltage bits (Minimum setting)

// CONFIG2H
#pragma config WDT = OFF         // (WDT disabled (control is placed on the SWDTEN bit))
#pragma config WDTPS = 32768     // Watchdog Timer Postscale Select bits (1:32768)

// CONFIG3H
#pragma config CCP2MX = PORTC    // CCP2 MUX bit (CCP2 input/output is multiplexed with RC1)
#pragma config PBADEN = OFF      // (PORTB<4:0> pins are configured as digital I/O on Reset)
#pragma config LPT1OSC = OFF     // (Timer1 configured for higher power operation)
```

```
#pragma config MCLRE = ON       // (MCLR pin enabled; RE3 input pin disabled)

// CONFIG4L
#pragma config STVREN = ON      // (Stack full/underflow will cause Reset)
#pragma config LVP = ON         // (Single-Supply ICSP enabled)
#pragma config XINST = OFF      // (Instruction set extension and Indexed Addressing mode
                               //  disabled (Legacy mode))

// CONFIG5L
#pragma config CP0 = OFF        // (Block 0 (000800-001FFFh) not code-protected)
#pragma config CP1 = OFF        // (Block 1 (002000-003FFFh) not code-protected)
#pragma config CP2 = OFF        // (Block 2 (004000-005FFFh) not code-protected)
#pragma config CP3 = OFF        // (Block 3 (006000-007FFFh) not code-protected)

// CONFIG5H
#pragma config CPB = OFF        // (Boot block (000000-0007FFh) not code-protected)
#pragma config CPD = OFF        // (Data EEPROM not code-protected)

// CONFIG6L
#pragma config WRT0 = OFF       // (Block 0 (000800-001FFFh) not write-protected)
#pragma config WRT1 = OFF       // (Block 1 (002000-003FFFh) not write-protected)
#pragma config WRT2 = OFF       // (Block 2 (004000-005FFFh) not write-protected)
#pragma config WRT3 = OFF       // (Block 3 (006000-007FFFh) not write-protected)

// CONFIG6H
#pragma config WRTC = OFF       // (Configuration registers (300000-3000FFh) not write-
protected)
#pragma config WRTB = OFF       // (Boot block (000000-0007FFh) not write-protected)
#pragma config WRTD = OFF       // (Data EEPROM not write-protected)

// CONFIG7L
#pragma config EBTR0 = OFF      // Block 0 not protected
#pragma config EBTR1 = OFF      // Block 1 not protected
#pragma config EBTR2 = OFF      // Block 2 not protected
#pragma config EBTR3 = OFF      // Block 3 not protected

// CONFIG7H
#pragma config EBTRB = OFF      // (Boot block not protected

// Prototypes AG
char *itoa(char *buf, int val, int base);
char *utoa(char *buf, unsigned val, int base);
void putch(unsigned char data);
void init_uart(void);

//Defines
#define _XTAL_FREQ 4000000
```