TESTING FOR HEADER RULES COMPLIANCE (Kieras, 2012)

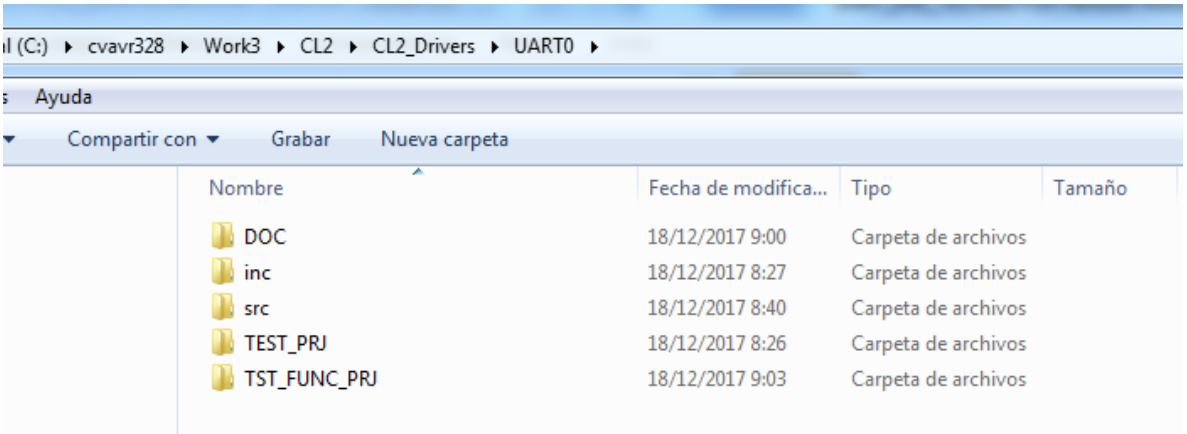
UART0_DR1 MODULE on CVAVR 3 / CL2 – Rev. 18-12-2017 / R.Oliva

Updated with test results 03-01-2018

Reformatted – printout in PDF form

CREATION 18-12-2017 – UART0_DR1 - (Files UART0_DR1.c / .h, formerly within mainfile)
This document in: "C:\cvavr328\Work3\CL2\CL2_Drivers\UART0\DOC\UART_DR1_TESTING FOR HEADER RULES COMPLIANCE_v18-12-2017.docx"

Directory structure proposed:



Rule #1 – OK Groups functional operations with UART0

Rule #2 OK – "Include guards", used here:

```
#ifndef UART0_INCLUDED
#define UART0_INCLUDED
…
#endif
```

Rule #3 All required declarations to use the module appear in the UART0_DR1.h file

Rule #4 OK → .h file only contains declarations, and is included by the .c file

```
.C FILE:
// ***************************************************************
// local functions
// ***************************************************************

NO LOCAL FUNCTIONS IN .C FILE
```

At beginning includes its header: #include "../inc/UART0_DR1.h" (OK)

```
.H FILE:
```

At start of .h file, these are published functions for all-prgrm access:

```
/*************************************************************************
**
**    Functions public to rest of program 18.12.2017
**
*************************************************************************/
```

```
// Interrupt routines
interrupt [USART0_RXC] void usart0_rx_isr(void);
interrupt [USART0_TXC] void usart0_tx_isr(void);

// Alternate getchar() defined with ISR Rx support
char getchar(void);

// ******************************************************************
// ** GetByte() Added for Modbus Inputs - transfer Bytes not chars..**
// ** v1.0 30-05-2012 - Used by MB_Serial() FromModbusTest2()       **
// ******************************************************************
unsigned char GetByte(void);

// Alternate putchar() defined with ISR Tx support
void putchar(char c);

// ******************************************************************
// ** PutByte() Added for Modbus output - transfer Bytes not chars..**
// ** v1.0 30-05-2012 - Used by FinaliseTransmit and ExceptionResp()**
// ******************************************************************
void PutByte(unsigned char txbyte);

// USART0_Init standard 19200,N,8,1 TxRx ISR support
void USART0_Init(void);


// New Functions defined in .C file:

// USART0_Init standard 19200,N,8,1 TxRx ISR support
void USART0_Init(void)
{
  // USART0 initialization - PWRC2
  // Communication Parameters: 8 Data, 1 Stop, No Parity
  // USART0 Receiver: On
  // USART0 Transmitter: On
  // USART0 Mode: Asynchronous
  // USART0 Baud Rate: 19200!
  UCSR0A=0x00;
  UCSR0B=0xD8;
  UCSR0C=0x06;
  UBRR0H=0x00;
  UBRR0L=0x2F;
}

// Slightly modified RX_ISR:
/*****************************************************************************
**
** USART0 Receiver interrupt service routine
** RX_BUF_Size supposed to be diff.(lower) from 256 (24 is default in PWRC2)
*****************************************************************************/

interrupt [USART0_RXC] void usart0_rx_isr(void)
{
 char status,data;
 status=UCSR0A;
 data=UDR0;
 if ((status & (FRAMING_ERROR | PARITY_ERROR | DATA_OVERRUN))==0)
  {
   rx_buffer0[rx_wr_index0++]=data;
   //#if RX_BUFFER_SIZE0 == 256 (commented out 18.12.2017)
   // special case for receiver buffer size=256
   // if (++rx_counter0 == 0)
   //   {
   //#else
   if (rx_wr_index0 == RX_BUFFER_SIZE0) rx_wr_index0=0;
   if (++rx_counter0 == RX_BUFFER_SIZE0)
      {
      rx_counter0=0;
   //#endif
      rx_buffer_overflow0=1;
      }
   }
  }
```

Rule #5 OK → Globally used variables are declared as extern in .h file, and defined in .c file:

```
/*****************************************************************************
**
```

```
**    EXPORTED VARIABLES
**    declared here, but defined in .c file for global access.. 18.12.2017
**
*************************************************************************/

extern char rx_buffer0[RX_BUFFER_SIZE0];

#if RX_BUFFER_SIZE0 <= 256
extern unsigned char rx_wr_index0,rx_rd_index0,rx_counter0;
#else
extern unsigned int rx_wr_index0,rx_rd_index0,rx_counter0;
#endif

// This flag is set on USART0 Receiver buffer overflow
extern bit rx_buffer_overflow0;


extern char tx_buffer0[TX_BUFFER_SIZE0];

#if TX_BUFFER_SIZE0 <= 256
extern unsigned char tx_wr_index0,tx_rd_index0,tx_counter0;
#else
extern unsigned int tx_wr_index0,tx_rd_index0,tx_counter0;
#endif
```

These VARIABLES are memory assigned in the .c file as follows:

```
/*************************************************************************
**
** UART0 Global Variables declared in uart0_dr1.h
** MEMORY IS ASSIGNED
** HERE FOLLOWING RULE #5
**
*************************************************************************/

char rx_buffer0[RX_BUFFER_SIZE0];

#if RX_BUFFER_SIZE0 <= 256
unsigned char rx_wr_index0,rx_rd_index0,rx_counter0;
#else
unsigned int rx_wr_index0,rx_rd_index0,rx_counter0;
#endif

// This flag is set on USART0 Receiver buffer overflow
bit rx_buffer_overflow0;


char tx_buffer0[TX_BUFFER_SIZE0];

#if TX_BUFFER_SIZE0 <= 256
unsigned char tx_wr_index0,tx_rd_index0,tx_counter0;
#else
unsigned int tx_wr_index0,tx_rd_index0,tx_counter0;
#endif
```

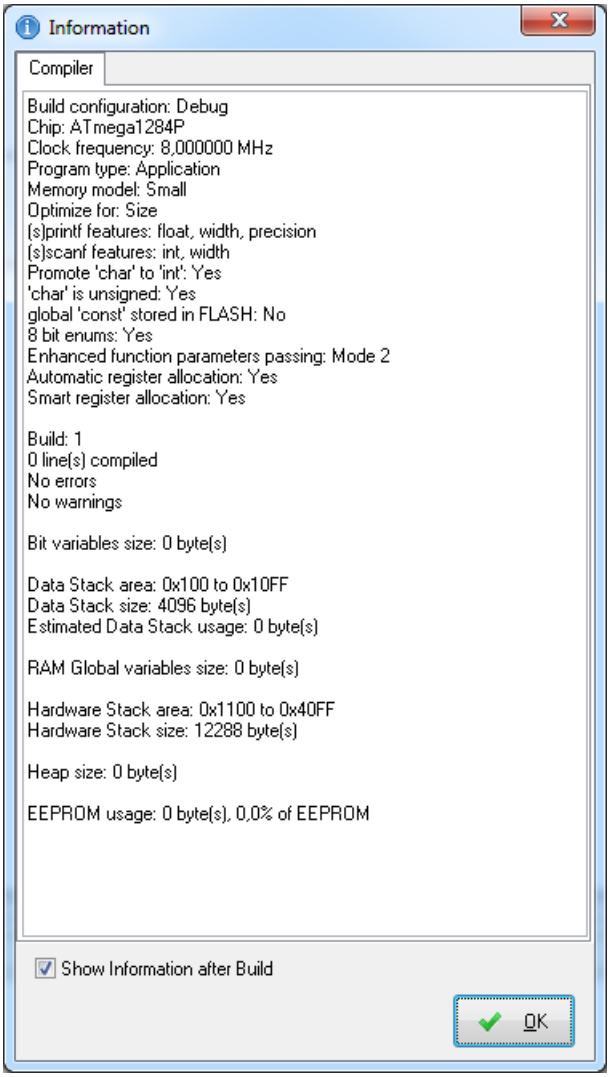
Rule #6 Internal declarations kept out of .h module→ Ok

Rule #7,8 Not Applicable, since no external .h functions are required for this module.

Rule #9 Self compilation :

In CVAVR 3 we need to make a Test_Uart0Dr.prj, including only the file uart0_dr1.c, which at the start executes: #include "../inc/ uart0_dr1.h" – see if it compiles correctly by itself.

PRJ file should be confined it to:
C:\cvavr328\Work3\CL2\CL2_Drivers\. It creates a test_dr.c empty file, which is not used at this stage..

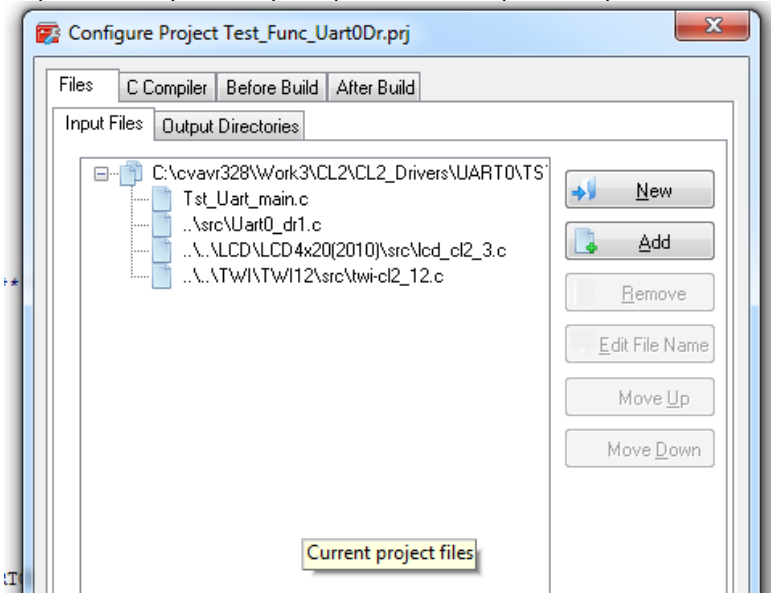


Rule #10 OK: UART0_DR1.c includes at beginning the file UART0_DR1.h, other files not applicable, since no external .h functions are required for this module.

Rule #11 OK no .c files #included.

TESTING:

A) 18.12.2017 NOW TEST NEW UART0_DR1 WITH TST_FUNC_PRJ, C:\cvavr328\Work3\CL2\CL2_Drivers\UART0\TST_FUNC_PRJ\Test_Func_Uart0Dr.prj



SOURCE CODE For Sending only..

C:\cvavr328\Work3\CL2\CL2_Drivers\UART0\TST_FUNC_PRJ\Tst_Uart_main.c

fx Initialize_CL2_simple

```c


#include <mega1284p.h>
#include <delay.h>
#include <string.h>
#include <stdio.h>
#include <stdarg.h>
#include <stdlib.h>
#include <string.h>
#include <io.h>


// Added for LCD- 19-12-2017
#include "..\..\LCD\LCD4x20(2010)\inc\lcd_cl2_3.h"

// Added for TWI - 18-12-2017
#include "..\..\TWI\TWI12\inc\twi-cl2_12.h"

// Testing UART0_DR1 - 18-12-2017
#include "..\inc\Uart0_dr1.h"

// Added for testing -  Initialize_CL2_Simple()
#include "Tst_Uart_main.h"


// PB.0 down --> initialize RTC 19.11.17
#define KBD_LEFT_ARROW          PINB.0

```

CodeVisionAVR - C:\cvavr328\Work3\CL2\CL2_Drivers\UART0\TST_FUNC_PRJ\Test_Func_Uart0Dr.prj

File Edit Search View Project Tools Settings Help

Code Information

Tst_Uart_main.c
- Includes
  - delay.h
  - io.h
  - lcd_cl2_3.h
  - mega1284p.h
  - stdarg.h
  - stdio.h
  - stdlib.h
  - string.h
  - Tst_Uart_main.h
  - twi-cl2_12.h
  - Uart0_dr1.h
- Macros
- Functions
  - Initialize_CL2_simple(void)
  - main(void)

twi-cl2_12.c | twi-cl2_12.h | Uart0_dr1.h | Tst_Uart_main.h | Tst_Uart_main.c | Notes

C:\cvavr328\Work3\CL2\CL2_Drivers\UART0\TST_FUNC_PRJ\Tst_Uart_main.c

Initialize_CL2_simple

```
424     set_LCD_cur(0,0);
425     rtc_set_ckoutfreq(OPTION_CKOUT_01HZ);
426     disp_cstr("CKout_1Hz       ");
427     delay_ms(2000);
428
429
430   printf("Testing UART0 Driver 1:\r\n");
431
432
433
434   delay_ms(2000); //  2sec
435   // ************************************************************************
436
437   while (1)
438         {
439         // Place your code here
440         // 16-5-2017 - Added - for Incorrect Mounting..
441         if(WD_ON_Flag == 0){ #asm("wdr")}
442         OLED = 1;
443         delay_ms(500); //  0.5 blinking
444         OLED = 0;
445         delay_ms(450); // 1s blinking
446         RTC_result = (int8_t)(rtc_get_time(&uRTCHour, &uRTCMin, &uRTCSec));
447         sprintf(s,"Time:%02d:%02d:%02d   ",uRTCHour,uRTCMin,uRTCSec);
448         set_LCD_cur(1,0);
449         disp_str(s);
450         printf("\r\n");
451         puts(s);
452         }
453   }
```

We set up terminal for testing – works ok in transmission:

CodeVisionAVR - C:\cvavr328\Work3\CL2\CL2_Drivers\UART0\TST_FUNC_PRJ\Test_Func_Uart0Dr.prj

File Edit Search View Project Tools Settings Help

Terminal

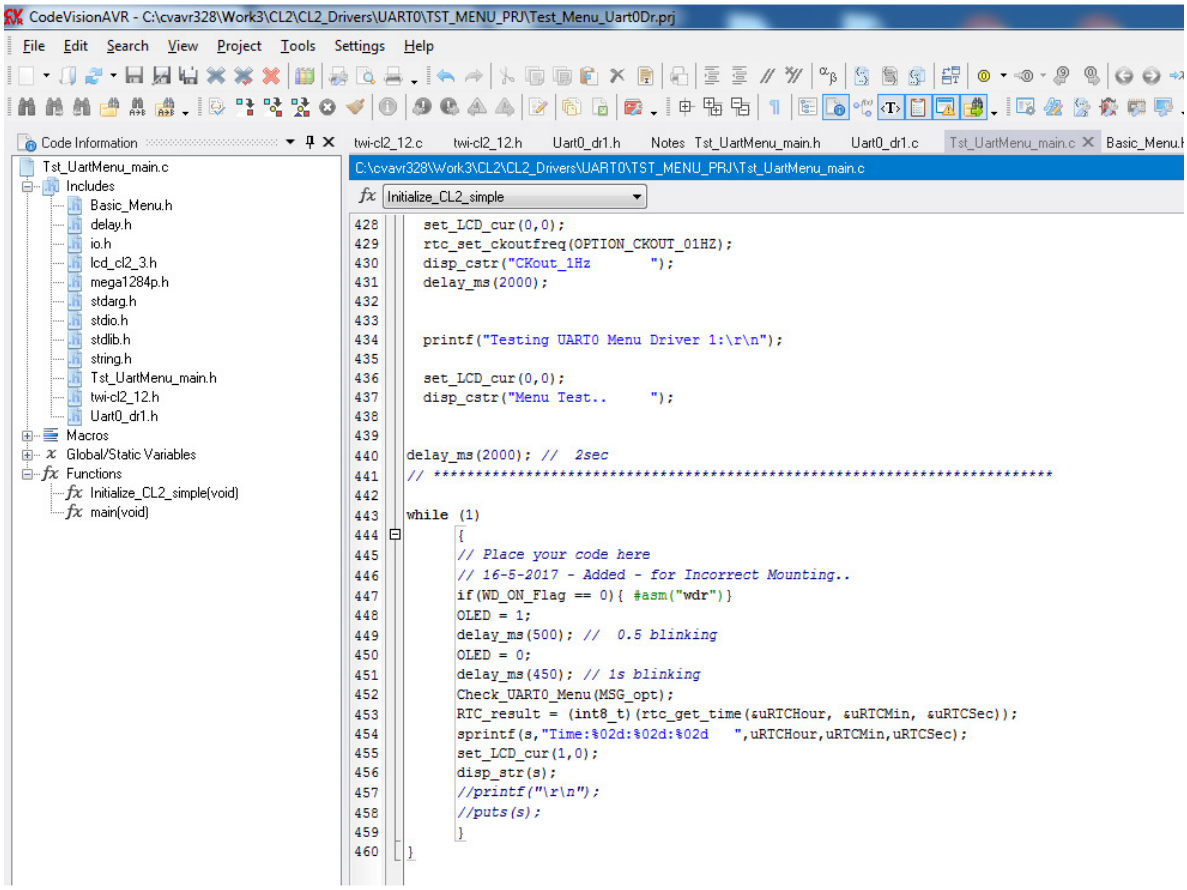
Hex Code:

```
                LCD Access testing..

RTC inicializando..

RTC-OK!

18/12/2017-11:18:42
                    RTC new function ReadTime Access: 0Time:11:18:44
                                                                    Date:18/1
2/2017
       Testing UART0 Driver 1:



Time:11:18:55


Time:11:18:56


Time:11:18:57


Time:11:18:58
```

Seems to work alright!

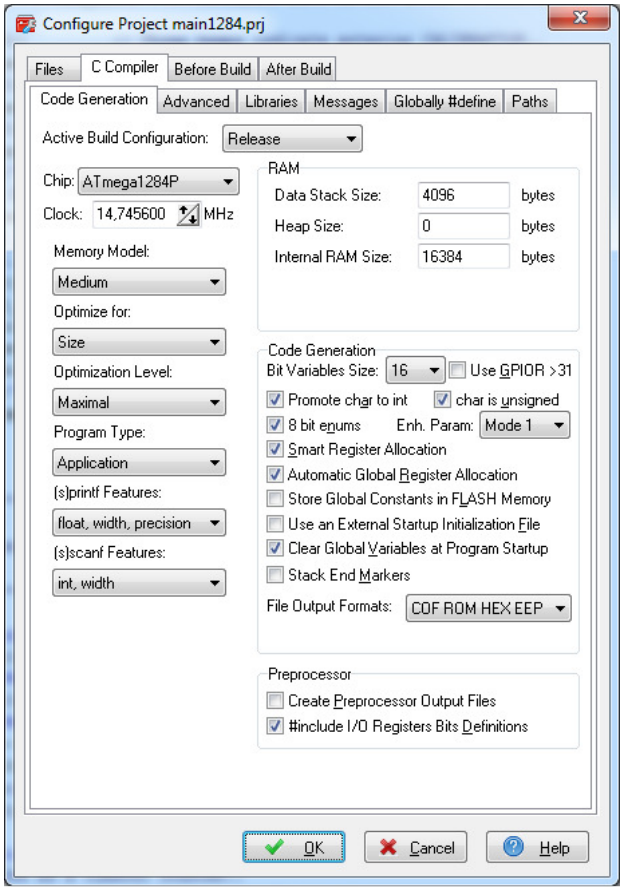
B) 18.12.2017 NOW TEST UART0_DR1 WITH TST_MENU_PRJ, C:\cvavr328\Work3\CL2\CL2_Drivers\UART0\TST_MENU_PRJ\Test_Menu_Uart0Dr.prj

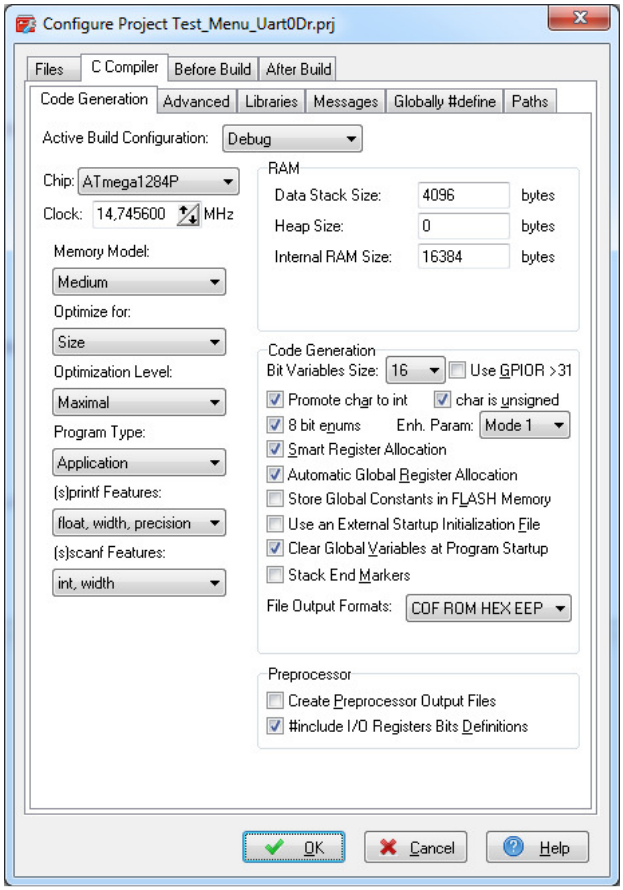
This Project copies (A) and includes a simple Menu System borrowed from PWRC2, to test the reading from terminal abilities of new UART0_dr1.c/.h



New .c/.h test files are:

```
Basic_Menu.c / .h

Problem: Hangs on input of values..

Settings for traditional PWRC2 are shown, and are identical to current ones..:
```





19.12.2017 – Problem traced to:
- Watchdog timer!
On entering the string reading routines, the WD was not petted, and the System resets.
CORRECTED!!

PROBLEM2
We add the External Interrupt on PD.4 routine, fired by the 1Hz oscillator CKOUT.
ORIGINAL IMPLEMENTATION OF rtc_set_ckoutfreq(int8_t option) DON'T WORK!

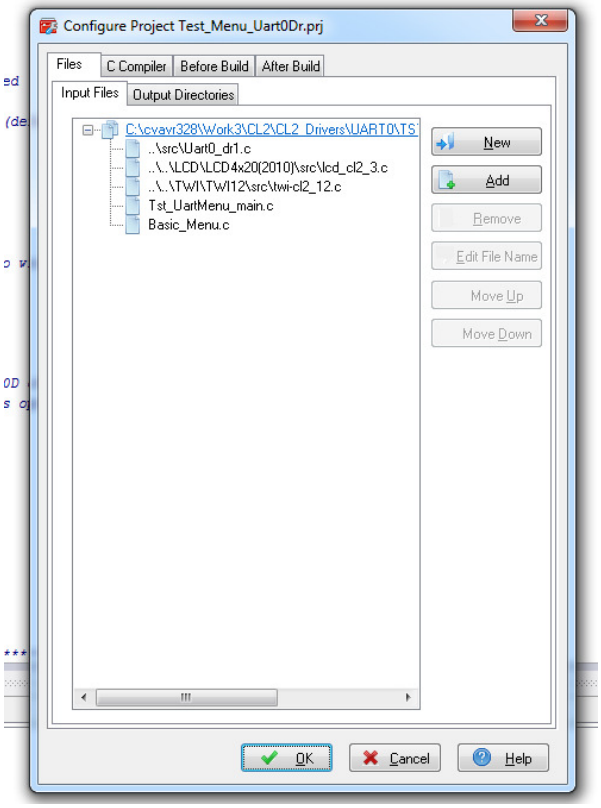
Problems appear since int8_t is typedef'd in stdint.h as "signed char" instead of char..
Requires research or changing the stdint.h file??

SOLVED: Change CKOUTFREQ Routine in twi-CL2_12.c, so that parameter only indicates an option and not the HEX value to be sent, see this modified version.. (OK)

```
/*************************************************************************
** rtc_set_ckoutfreq(int8_t option)
** Start on version 10 - 17.10.2012 PV - CKOUT can be set to 1Hz
** Writes an option paramter that fixes the CLKOUT frequency of the PCF8563
** RTC chip. This frequency can be used to interrupt the CL2Bm1 CPU on PD.4
** (Only level change Interrupts available on this pin..)
**
** Parameters:   option can take following values
** #define OPTION_CKOUT_DIS   0  //0x00
** #define OPTION_CKOUT_32KHZ 1  // 0x80
** #define OPTION_CKOUT_1024HZ 2 // 0x81
** #define OPTION_CKOUT_32HZ   3 // 0x82
** #define OPTION_CKOUT_01HZ   4 // 0x83
**
** Returns: 0 - Data written successfully
**          1 - An error occurred in write..
**          2 - incorrect option value..
** Use - example:
**    rtc_set_ckoutfreq(OPTION_CKOUT_01HZ);
**
*************************************************************************/
```

```
int8_t rtc_set_ckoutfreq(int8_t option){
    char fr_option[2];
    char ret_val;

    fr_option[0] = 0x83;
    printf("\n\rRTC Ckout Option: %03X", option);
    // Check if option is valid..
    switch(option){
          case 0:
             fr_option[0] =  0x00;   // CKOUT Disabled
          case 1:
             fr_option[0] = 0x80;    // CKOUT 32kHz (default on reset)
          case 2:
             fr_option[0] = 0x81;    // CKOUT 1024Hz
          case 3:
             fr_option[0] = 0x82;    // CKOUT 32Hz
          case 4:
             fr_option[0] = 0x83;    // CKOUT 1.0Hz
             ret_val = 0; // option is OK,continue to write value..
             break;
          default:
             ret_val = 2; // Option value incorrect
             return(ret_val);
         }
    // Writes one of the option bytes at position 0x0D of RTC buffer
    // a '1' is written on the MS bit, to enable this option
    // Returns 0 if OK, 1 if write error..
    _FF_cli();
    ret_val = twi_write_RTC(0x0D, 1, fr_option);
    _FF_sei();

    return(ret_val);

}
```

TEST_MENU_Uart0Dr.PRJ PROJECT LOOKS LIKE THIS: (like a visual MAKE) on CVAVR3



As seen: We have a main Tst_UartMenu_main.c file, which contains main(). An annex file called Basic_Menu.c contains the UART0_Menu() routine (from PWRC2, simplified) and auxiliary functions. Also, the Three driver functions (UART0, LCD and TWI-CL2) are called within the PRJ

Final program terminal abilities of new UART0_dr1.c/.h

APPENDIX: As usual, we use the following programming settings: