Programming (Hello World) an Infineon XMC4700 (ARM Cortex M4) Microcontroller.
Using Dave/Eclipse (Code Generator, IDE, Compiler, Linker, Locator, Debugger, Flash Programmer, Utility Tools).

State-of-the-art component-based (= Dave Apps) Embedded Software Engineering/Development.
Introduction:

This is a Hands-On Training / Cookery Book / step-by-step book. It will help inexperienced users to get the XMC4700 RelaxKit up and running.

With this step-by-step book you should be able to get your first useful program in less than 2 hours.

The purpose of this document is to gain know-how of the microcontroller and the tool-chain. Additionally, the "hello world example" can easily be expanded to suit your needs. You can connect either a part of - or your entire application to the XMC4700 RelaxKit. You are also able to benchmark any of your algorithms to find out if the selected microcontroller fulfils all the required functions within the time frame needed.

Note: The style used in this document focuses on working through this material as fast and easily as possible. That means there are full screenshots and dialog-window-screenshots; extensive use of colours and page breaks; and listed source-code is not formatted to ease copy & paste.

Have fun and enjoy the XMC4700 RelaxKit!
XMC4700 RelaxKit:

Note:
For further information, please refer to the XMC4700 RelaxKit home page:
Used/selected microcontroller:

XMC4700 Block Diagramm (Source: Manual/Data Sheet)

Note:
Just by looking at the block diagram, you should be able to get a picture of the microcontroller and to answer some of your initial questions.
“Cookery book“
For your first programming example for the XMC4700 RelaxKit:

<table>
<thead>
<tr>
<th>Chapter/Step</th>
<th>*** Recipes ***</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.)</td>
<td>Dave Installation</td>
</tr>
<tr>
<td></td>
<td>.) Dave Installation</td>
</tr>
<tr>
<td></td>
<td>.) Dave Updates</td>
</tr>
<tr>
<td>1.)</td>
<td>Using Dave Part 1: Transmit Data To A Serial Terminal Program</td>
</tr>
<tr>
<td></td>
<td>.) Microcontroller initialization</td>
</tr>
<tr>
<td></td>
<td>.) Programming of your application (hello world) with the Dave toolchain</td>
</tr>
<tr>
<td></td>
<td>.) Using the debugger</td>
</tr>
<tr>
<td>2.)</td>
<td>Part 2: Receive Data From The Terminal Program</td>
</tr>
<tr>
<td>3.)</td>
<td>Part 3: Ports (LEDs)</td>
</tr>
<tr>
<td>4.)</td>
<td>Part 4: Blinking LEDs (SystemTimer)</td>
</tr>
</tbody>
</table>

Feedback:

<table>
<thead>
<tr>
<th>5.) Feedback</th>
</tr>
</thead>
</table>
0.) Dave Installation:

Install Dave:


Note:
Install all/everything offered.
Do not use blanks/spaces in windows directory names and windows path names!

Result:

Start Dave:

Double click
Workspace Launcher -> Workspace: insert C:\DAVE_4_3_2_Workspace_HelloWorld

Click OK
Update Dave:

Help -> Check for Updates

Help -> Check for DAVE APP Updates

Help -> Install Dave APP/Example/Device Library… -> Dave Site: Work with: select Dave Project Library Manager

Help -> Install Dave APP/Example/Device Library… -> Dave Site: Work with: select Dave APPS Library Manager

Help -> Install Dave APP/Example/Device Library… -> Dave Site: Work with: select Contributed Dave APPS Library Manager

Note:
Select All – and install all/everything

Note:
In case your Update Process is frozen – check your network connection:

Window -> Preferences -> General -> Network Connections -> Active Provider -> change from „Direct“ (= default after installation) to „Native“ (Company-Proxy) -> Apply -> OK
1.) Using Dave Part 1: Transmit Data To The Serial Terminal Program

Create your Dave CE project and select the XMC4700 microcontroller:

Dave … Digital Application virtual/valuable/verbose Engineer
CE … Code Engine/Generator

File
New
click DAVE Project…
Project Name: insert XMC4700_RelaxKit_HelloWorld_UART
Project Type: select Dave CE Project

Click Next
Microcontrollers -> XMC4000 -> XMC4700 Series: select XMC4700-F144F2048

Click Finish
Click Add New APP
Add New APP -> Communication -> General Purpose -> select/click UART

Click Add, click Close
Double click UART UART_0
Note:
Window -> Reset Perspective… restores your IDE windows/perspective/view anytime!

Click Yes
UART_0 Configuration:

UART_0 -> General Settings -> Operating mode: check Full Duplex
UART_0 -> General Settings -> Desired speed[baud]: insert 9600 <ENTER>

Note: Validate each alpha numeric entry by pressing <ENTER>. 
UART_0 -> Advanced Settings -> Protocol Handling -> Transmit mode: check/select Interrupt
UART_0 -> Advanced Settings -> Protocol Handling -> Receive mode: select Direct

UART_0 -> Advanced Settings -> FIFO Settings: click ☑ to untick □ Enable transmit FIFO
UART_0 -> Advanced Settings -> FIFO Settings: click ☑ to untick □ Enable receive FIFO
Note:
To understand the difference between Interrupt und Direct see APP Help:

Right mouse button click at UART-APP: click APP Help
APP Help: see UART Documentation, e.g. Architecture Description, Methods, …
APP Help: see UART Documentation, Methods:
UART_0 -> Interrupt Settings -> Transmit: tick ☑️ End of transmit callback:

Click OK
UART_0 -> Interrupt Settings -> Transmit: ✔ End of transmit callback: insert fEndOfTransmitCallback <ENTER>
UART_0 -> Pin Settings: tick ✓ Enable advanced pin characteristics
UART Pin Connection:

Right mouse button click at UART UART_0 (APP): click Manual Pin Allocator
Manual Pin Allocator -> UART_0 -> Receive Pin: select #108 (P1.4)
Manual Pin Allocator -> UART_0 -> Transmit Pin: select #107 (P1.5)

Click Save, click Close
Evaluation Board
For XMC4000 Family

XMC4700 Relax Lite Kit & XMC4700 Relax Kit for 5V Shields & XMC4700 Relax Kit & XMC4800 Relax EtherCAT Kit
Kit Version 1

Board User's Manual
Revision 1.2, 2016-06-16

Microcontroller
2.5 Debugging and UART-to-USB Communication

The XMC4700/XMC4800 Relax Kit Series-V1 supports debugging via 2 different channels:
- On-board debug probe
- 10-pin Cortex™ Debug Connector (not assembled)

2.5.1 On-board Debug Probe

The on-board debug probe supports Serial Wire Debug (SWD) and UART communication. Both require the installation of Segger’s J-Link Driver which is part of the DAVE™ installation. DAVE™ is a highly efficient development platform for the XMC microcontroller families to simplify and shorten SW development. It can be downloaded at www.infineon.com/dave. The latest Segger J-Link Driver can be downloaded at http://www.segger.com/jlink-software.html. Table 5 shows the pin assignment of the XMC4700/XMC4800 used for debugging and UART communication.

<table>
<thead>
<tr>
<th>Pin Function</th>
<th>Input/Output</th>
<th>XMC Pin</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data pin for Debugging via SWD</td>
<td>I/O</td>
<td>TMS</td>
</tr>
<tr>
<td>Clock pin for Debugging via SWD</td>
<td>O</td>
<td>TCK</td>
</tr>
<tr>
<td>Transmit pin for UART communication (PC_RX)</td>
<td>O</td>
<td>P1.5 (U0C0.DOOUT)</td>
</tr>
<tr>
<td>Receive pin for UART communication (PC_TX)</td>
<td>I</td>
<td>P1.4 (U0C0.DX0B)</td>
</tr>
</tbody>
</table>

2.5.2 Cortex™ Debug Connector (10-pin)

The 10-pin Cortex™ Debug Connector supports Serial Wire Debug (SWD) and Serial Wire Viewer (SWV). The pin assignment of the Cortex™ Debug Connector is shown in Table 6.
Generate your file system:

Click `Generate Code`
Insert your application specific program:

Note:
Dave doesn’t change code in main.c
Change Dave’s Perspective from DAVE CE to DAVE IDE:
-> Click DAVE IDE
Double click MAIN.C:
Double click MAIN.C and delete everything (e.g. Strg+A and DELETE):
Double click MAIN.C and insert Code:

```c
#include <DAVE.h> // Declarations from DAVE Code Generation (includes SFR declaration)
#include <stdio.h>
#include <ctype.h>
#include <string.h>
#include <stdbool.h>

// ***** defines: *****

// ***** Global Variables: *****

// Transmit:
const unsigned char ucArrayMenu[] =
    "\n" "1 ... LEDs ON\n" "2 ... LEDs OFF\n" "3 ... LEDs Toggle\n" "4 ... LEDs Blinking\n" "\n" "your choice: \n"

unsigned int ucStringLength1 = 0;
unsigned int ucStringLength2 = 0;

char mb[200]; // MessageBuffer for sprintf()
unsigned int iMainMenuLoopCounter = 0;

bool bTransmitBusy = true;

// ***** function prototypes: *****

// Transmit:
void fMyPrintf(const unsigned char *p);
void fEndOfTransmitCallback(void);

int main(void)
{
    DAVE_Init(); // Initialization of DAVE APPs

    iMainMenuLoopCounter++;
    sprintf(mb,"\n\n*** MainMenuLoopCounter = %d ***",iMainMenuLoopCounter); // Write formatted data to string mb (MessageBuffer)
    fMyPrintf(mb);
    while (bTransmitBusy);

    fMyPrintf(ucArrayMenu);
    while (bTransmitBusy);
    while(1U)
```
void fEndOfTransmitCallback(void)
{
    bTransmitBusy = false;
}

void fMyPrintf(const unsigned char *p)
{
    unsigned char *helpStartTextPointer = p;
    bTransmitBusy = true; // will be cleared by fEndOfTransmitCallback
    ucStringLength1 = 0;
    ucStringLength2 = strlen(p);

    while(*p)
    {
        ucStringLength1++;
        *p++;
    }

    if(ucStringLength1 == ucStringLength2)
    {  
        UART_Transmit(&UART_0, helpStartTextPointer, ucStringLength2);
    }
}
#include <DAVE.h>  // Declarations from DAVE Code Generation (includes SFR declaration)
#include <stdio.h>
#include <ctype.h>
#include <string.h>
#include <stdbool.h>

// ***** defines: *****

// ***** Global Variables: *****

// Transmit:
const unsigned char ucArrayMenu[] =
    "\n" "1 ... LEDs ON\n" "2 ... LEDs OFF\n" "3 ... LEDs Toggle\n" "4 ... LEDs Blinking\n" "\n" "your choice: ";

unsigned int ucStringLength1 = 0;
unsigned int ucStringLength2 = 0;

char mb[20];  // MessageBuffer for sprintf()
unsigned int MainMenuLoopCounter = 0;
bool bTransmitBusy = true;

// ***** function prototypes: *****

// Transmit:
void fMyPrint(const unsigned char *p);
void fEndOfTransmitCallback(void);
Generate your application program:

Click Rebuild Active Project

```c
#include <DAVE.h> // Declarations from DAVE
#include <stdio.h>
#include <ctype.h>
#include <string.h>
#include <stdbool.h>

// ***** defines: *****

// ***** Global Variables: *****

// Transmit:
const unsigned char ucArrayMenu[] =
    "\n" 
    "1 ... LEDs ON\n"
    "2 ... LEDs OFF\n"
    "3 ... LEDs Toggle\n"
    "4 ... LEDs Blinking\n"
    "\n"
    "your choice: ";

unsigned int ucStringLength1 = 0;
unsigned int ucStringLength2 = 0;

char mb[200]; // MessageBuffer for sprintf()
unsigned int iMainMenuLoopCounter = 0;
```
**Connect** the XMC4700 RelaxKit to the host computer:

**USB Connection:**

**Note:**
A USB driver is installed the first time while connecting the XMC4700 RelaxKit via USB to your host computer.

**Note:**
A default virtual COM Port is generated.
Connecting a serial terminal program (e.g. Docklight):

Start Docklight

Click
Click Project Settings:
Project Settings -> Communication Mode -> Send/Receive on Comm. Channel: select YOUR COM
Project Settings -> COM Port Settings -> Baud Rate: select/check 9600
Check:

Click OK
Configure what you want to send (1, 2, 3, 4, 5) to the XMC4700 RelaxKit:

Double click inside:

Insert 1 <Tab key> 1

Click OK
Repeat for 2, 3, 4, and 5 – and see the result:
Docklight is ready:
Configure/Start/Launch the debugger:

Click Debug
Double click GDB SEGGER J-Link Debugging
Note:

If asked "Do you want to update to the latest firmware version?" → click Yes
Tick ☑️ Remember my decision

Click Yes
Welcome to the Debug perspective:
And start your program:

![Debugger interface](image)

Click Resume (F8)
And see the result:
Terminate the Debugger Connection:

Click Terminate:

```c
// ***** function prototypes: *****

// Transmit:
void fMyPrintf(const unsigned char *p);
void fEndOfTransmitCallback(void);

int main(void)
{
    DAVE_Init(); // Initialization of DAVE APPs
    iMainMenuLoopCounter++;
    printf("\n\n*** MainMenuLoopCounter = %d \n", fMyPrintf(mb);

```

Removing breakpoint @ address 0x080017B4, Size = 2
Reading 64 bytes @ address 0x00000000
Reading 64 bytes @ address 0x00000040
Reading 64 bytes @ address 0x26961789
And go back to the DAVE CE perspective/view:
Part 2: Receive Data From The Terminal Program

Click Add New APP
Add New App -> System: select/click INTERRUPT

Click Add, click Close
Right mouse button click at INTERRUPT INTERRUPT_0 click Rename Instance Label
Insert INT_UART_Receive

Click OK
Double click INTERRUPT INT_UART_Receive
Interrupt Settings - > Interrupt handler: insert INT_UART_Receive_ISR <ENTER>
Dave Signal Connection (UART_0 Receive -> INT_UART_Receive):

Right mouse button click UART UART_0 (APP) -> click HW Signal Connections…

Warnings: x_jq input not connected. Please use of HW Signal Connections dialog to establish a connection.
HW Signal Connections: select event_std_receive, select INT_UART_Receive, select sr_irq

Click Save, click Close
You can see the result (Signal Connection) in the HW Signal Connectivity view:
Change to DAVE IDE Perspective/view:
Double click MAIN.C and insert Code:

```c
// Receive:
uint8_t ucUartReceivedDataGlobal = '\0';
volatile unsigned int uiReceivedCounterGlobal = 0;
unsigned int uiReceivedCounterOLD_Global = 0;
```

```
#include <DAVE.h>  // Declarations from DAVE Code Gen
#include <stdio.h>
#include <ctype.h>
#include <string.h>
#include <stdbool.h>

// ***** defines: *****

// ***** Global Variables: *****

// Transmit:
const unsigned char ucArrayMenu[] =
  "\n"  
  "1 ... LEDs ON\n"  
  "2 ... LEDs OFF\n"  
  "3 ... LEDs Toggle\n"  
  "4 ... LEDs Blinking\n"  
  "\n"  
  "your choice: ";

unsigned int ucStringLength1 = 0;
unsigned int ucStringLength2 = 0;
char mb[200];  // MessageBuffer for sprintf()
unsigned int iMainMenuLoopCounter = 0;
bool bTransmitBusy = true;

// Receive:
uint8_t ucUartReceivedDataGlobal = '\0';
volatile unsigned int uiReceivedCounterGlobal = 0;
unsigned int uiReceivedCounterOLD_Global = 0;

// ***** function prototypes: *****

// Transmit:
```
Double click MAIN.C and insert Code:

```c
if (uiReceivedCounterOLD_Global != uiReceivedCounterGlobal)
{
    switch (ucUartReceivedDataGlobal)
    {
        case '1': fMyPrintf("\n*** -------------------------
---------- > LEDs ON !!! ***\n"), while (bTransmitBusy);
        break;
        case '2': fMyPrintf("\n*** -------------------------
---------- > LEDs OFF !!! ***\n"), while (bTransmitBusy);
        break;
        case '3': fMyPrintf("\n*** -------------------------
---------- > LEDs TOGGLED !!! ***\n"), while (bTransmitBusy);
        break;
        case '4': fMyPrintf("\n*** -------------------------
---------- > LEDs BLINKING !!! ***\n"), while (bTransmitBusy);
        break;
        default : ;
        break;
    }
    uiReceivedCounterOLD_Global = uiReceivedCounterGlobal;
    iMainMenuLoopCounter++;
    sprintf(mb,"\n\n\n*** MainMenuLoopCounter = %d
***",iMainMenuLoopCounter); // Write formatted data to string mb
    fMyPrintf(mb);
    while (bTransmitBusy);
    fMyPrintf(ucArrayMenu);
    while (bTransmitBusy);
}
```
```c
myPrint(strlen(MenuName));  
while (!isReceiveBusy);  
while (0)  
{  
    //*****************************************************************************
    
    if (uiReceivedCounterGlobal != uiReceivedCounterGlobal)  
    {  
        switch (sCurReceiveDataGlobal)  
        {  
            case '1': myPrint("\n***\nLEDs ON \n***\n"); while (!isTransmitBusy);  
            break;  
            case '2': myPrint("\n***\nLEDs OFF \n***\n"); while (!isTransmitBusy);  
            break;  
            case '3': myPrint("\n***\nLEDs TOGGLE \n***\n"); while (!isTransmitBusy);  
            break;  
            case '4': myPrint("\n***\nLEDs BLINKING \n***\n"); while (!isTransmitBusy);  
            break;  
            case '5':  
                break;  
            default:  
                break;  
        }  
        uiReceivedCounterGlobal = uiReceivedCounterGlobal;  

        //MainMenuLoopCounter++;  
        myPrint("\nMainMenuLoopCounter = ", MainMenuLoopCounter); while (!isTransmitBusy);  
        while (!isTransmitBusy);  
        myPrint(MenuName);  
        while (!isTransmitBusy);  
    }  
    //*****************************************************************************
```

Double click MAIN.C and insert Code:

```c
void INT_UART_Receive_ISR (void)
{
    uiReceivedCounterGlobal++;
    UART_Receive(&UART_0, &ucUartReceivedDataGlobal, 1);
}
```
Click Rebuild Active Project
Start the debugger:

Click Debug:
Start the program:

Click **Resume:**
And see the result in the Serial Terminal Program:
Cookery-Book

80
Terminate the Debugger Connection:

Click Terminate:
And go back to the DAVE CE perspective/view:
Part 3: Ports (LEDs)

Click `Add New APP`
Add New APP -> System: select/click DIGITAL_IO

Click Add
Double-Click on the APP to add it to the active project.

Click Add, click Close
See the result:
Rename IOs:

Right mouse button click at DIGITAL_IO DIGITAL_IO_0 (APP): click Rename Instance Label…
Insert LED_P5_P8

Click OK
Right mouse button click at DIGITAL_IO DIGITAL_IO_1 (APP): click Rename Instance Label…
Insert LED_P5_P9

Click OK
And see the result:
Change IOs from Input to Output:

Double click LED_P5_P8: General Settings -> Pin direction: select Input/Output
Double click LED_P5_P9: General Settings -> Pin direction: select Input/Output
Connect IOs to Pins:

Right mouse button click at LED_P5_P8: click Manual Pin Allocator
Pin Number (Port): select #58 (P5.8)

Click Save, click Close
Right mouse button click at LED_P5_P9: click Manual Pin Allocator
Pin Number (Port): select #57 (P5.9)

Click Save, click Close
Note (LEDs):

Evaluation Board
For XMC4000 Family

XMC4700 Relax Lite Kit &
XMC4700 Relax Kit for 5V Shields &
XMC4700 Relax Kit &
XMC4800 Relax EtherCAT Kit
Kit Version 1

Board User's Manual
Revision 1.2, 2016-06-16

Microcontroller
2.4 User Push Buttons and User LEDs

The XMC4700/XMC4800 Relax Kit Series-V1 provides two push buttons and two LEDs. The port pins used can be found in Table 3 and Table 4. These pins are used exclusively for this function and they are not mapped to other devices or connectors.

<table>
<thead>
<tr>
<th>LED</th>
<th>XMC4700/XMC4800 Pin Mapping for User LEDs</th>
</tr>
</thead>
<tbody>
<tr>
<td>LED1</td>
<td>P5.9</td>
</tr>
<tr>
<td>LED2</td>
<td>P5.8</td>
</tr>
</tbody>
</table>
Note:
Instead of the Manual Pin Allocator you can use the Pin Mapping Perspective.
Click Pin Mapping Perspective:
Change from Perspective Pin Mapping to Perpective DAVE IDE:
Generate Code:

Click Generate Code

[Image of a software interface showing project files]

- XMC4700_RelaxKit_HelloWorld_UART
  - Binaries
  - Includes
  - Dave
  - Debug
  - Libraries
  - Startup
  - main.c
    - linker_script.ld
    - solver.bak
Double click MAIN.C and insert Code:

```c
// LEDs Blinking:
#define ON 1
#define OFF 0
```

```c
#include <DAVE.h> // Declarations from DAVE Code Gen

// ***** defines: *****

// LEDs Blinking:
#define ON 1
#define OFF 0

// ***** Global Variables *****

// Transmit:
const unsigned char ucArrayMenu[] = 
"\n" "1 ... LEDs ON\n" "2 ... LEDs OFF\n" "3 ... LEDs Toggle\n" "4 ... LEDs Blinking\n" "\n" "your choice: ";

unsigned int ucStringLength1 = 0;
unsigned int ucStringLength2 = 0;

char mb[200]; // MessageBuffer for sprintf()
unsigned int iMainMenuLoopCounter = 0;

bool bTransmitBusy = true;

// Receive:
uint8_t ucuartReceivedDataGlobal = \'0\';
volatile unsigned int uiReceivedCounterGlobal = 0;
unsigned int uiReceivedCounterOLD_Global = 0;
```
Double click `MAIN.C` and insert Code:

```c
// LEDs Blinking:
bool bLEDsBlinkingStatusGlobal = OFF; // used by the BLINKING INTERRUPT
bool bLEDsOnOffStatusGlobal = OFF; // used by the fLEDs_TOGGLE function
```

```c
#include <DAVE.h>  // Declarations from DAVE Code Generation (includes SF
#include <stdio.h>
#include <ctype.h>
#include <string.h>
#include <stdlib.h>

// ***** defines: *****

// LEDs Blinking:
#define ON 1
#define OFF 0

// ***** Global Variables: *****

// LEDs Blinking:
bool bLEDsBlinkingStatusGlobal = OFF; // used by the BLINKING INTERRUPT
bool bLEDsOnOffStatusGlobal = OFF; // used by the fLEDs_TOGGLE function
```
Double click **MAIN.C** and insert Code:

```c
// LEDs:
void fLEDs_ON (void);
void fLEDs_OFF (void);
void fLEDs_TOGGLE (void);
```
Double click MAIN.C and DELETE the Switch Case Statement and insert the new Switch Case Statement:

```c
switch (ucUartReceivedDataGlobal) {
    case '1': bLEDsBlinkingStatusGlobal = OFF, fLEDs_ON(), fMyPrintf("\n***----------------------------------\n> LEDs ON !!! ***\n") while (bTransmitBusy); break;
    case '2': bLEDsBlinkingStatusGlobal = OFF, fLEDs_OFF(), fMyPrintf("\n***----------------------------------\n> LEDs OFF !!! ***\n") while (bTransmitBusy); break;
    case '3': bLEDsBlinkingStatusGlobal = OFF, fLEDs_TOGGLE(), fMyPrintf("\n***----------------------------------\n> LEDs TOGGLED !!! ***\n") while (bTransmitBusy); break;
    case '4': bLEDsBlinkingStatusGlobal = ON, fMyPrintf("\n***----------------------------------\n> LEDs BLINKING !!! ***\n") while (bTransmitBusy); break;
    case '5': ; break;
    default : ; break;
}
```
```c
if (uiReceivedCounterCLO_Global != uiReceivedCounterGlobal)
{
    switch (uiReceivedDataGlobal)
    {
    case '1': bLEDsBlinkingStatusGlobal = OFF, bLEDs_OK();
        myPrint("\n**** LEDS ON \| \| ***\n"); while (bTransmitBusy);
        break;
    case '2': bLEDsBlinkingStatusGlobal = OFF, bLEDs_OFF();
        myPrint("\n**** LEDS OFF \| \| ***\n"); while (bTransmitBusy);
        break;
    case '3': bLEDsBlinkingStatusGlobal = OFF, bLEDs_TOGGLE();
        myPrint("\n**** LEDS TOGGLED \| \| ***\n"); while (bTransmitBusy);
        break;
    case '4': bLEDsBlinkingStatusGlobal = ON;
        myPrint("\n**** LEDS BLINKING \| \| ***\n"); while (bTransmitBusy);
        break;
    case '5':
        break;
    default:
        break;
    }
    uiReceivedCounterCLO_Global = uiReceivedCounterGlobal;
}
```
Double click MAIN.C and insert Code:

```c
void fLEDs_ON (void)
{
    DIGITAL_IO_SetOutputHigh(LED_P5_P8);
    DIGITAL_IO_SetOutputHigh(LED_P5_P9);
    bLEDsOnOffStatusGlobal = ON;
}

void fLEDs_OFF (void)
{
    DIGITAL_IO_SetOutputLow(LED_P5_P8);
    DIGITAL_IO_SetOutputLow(LED_P5_P9);
    bLEDsOnOffStatusGlobal = OFF;
}

void fLEDs_TOGGLE (void)
{
    if (bLEDsOnOffStatusGlobal == ON)
    {
        fLEDs_OFF();
    }
    else
    {
        fLEDs_ON();
    }
}
```
```c
while(1)
{
    ucStringLength1++;
    *p++;
}
if(ucStringLength1 == ucStringLength2)
{
    UART_Transmit(&UART_0, helpStartTextPointer, ucStringLength2);
}

void fLEDs_ON (void)
{
    DIGITAL_IO_SetOutputHigh(&LED_PS_P8);
    DIGITAL_IO_SetOutputHigh(&LED_PS_P9);
    bLEDsOnOffStatusGlobal = ON;
}

void fLEDs_OFF (void)
{
    DIGITAL_IO_SetOutputLow(&LED_PS_P8);
    DIGITAL_IO_SetOutputLow(&LED_PS_P9);
    bLEDsOnOffStatusGlobal = OFF;
}

void fLEDs_TOGGLE (void)
{
    if (bLEDsOnOffStatusGlobal == ON)
    {
        fLEDs_OFF();
    }
    else
    {
        fLEDs_ON();
    }
```
Note:
DAVE CE Perspective -> IO methods see APP Help:

Right mouse button click at DIGITAL_IO: click APP Help
Note:
IO methods see APP Help:
Note:
IO methods see APP Help:
Click Rebuild Active Project:
Start the debugger:

Click **Debug:**
Start the program:

Click Resume:
And see the result in the Serial Terminal Program AND on the XMC4700 RelaxKit:
e.g.: click 2, click 3:
Terminate the Debugger Connection:

Click **Terminate**:
And go back to the DAVE CE perspective/view:
Part 4: Blinking LEDs (SystemTimer)

Click** Add New APP:
Add New APP -> System: select/click SYSTIMER

Click Add, click Close
Rename SYSTIMER:

Right mouse button click at SYSTIMER SYSTIMER_0: click Rename Instance Label…
Insert SoftwareTimer1ms

click OK
Double click SYSTIMER SoftwareTimer1ms: General Settings: Number of software timers: insert 1 <ENTER>
Generate Code:

Click Generate Code
Change from Perspective DAVE CE to Perspective DAVE IDE:
Double click MAIN.C and insert Code:

```c
// SoftwareTimer:
#define ONESEC 1000000U
```
Double click MAIN.C and insert Code:

```c
// LEDs Blinking:
void fEverySecond (void);
```

```c
unsigned int ucStringLength2 = 0;
char mb[200]; // MessageBuffer for sprintf()
unsigned int iMainMenuLoopCounter = 0;
bool bTransmitBusy = true;

// Receive:
uint8_t ucUARTReceivedDataGlobal = '\0';
volatile unsigned int uiReceivedCounterGlobal = 0;
unsigned int uiReceivedCounterOLD_Global = 0;

// ***** function prototypes: *****

// Transmit:
void fMyPrintf(const unsigned char *p);
void fEndOfTransmitCallback(void);

// LEDs:
void fLEDs_ON (void);
void fLEDs_OFF (void);
void fLEDs_TOGGLE (void);

// LEDs Blinking:
void fEverySecond (void);

int main(void)
{
    DAVE_Init(); // Initialization of DAVE APPs
    
    iMainMenuLoopCounter++;
    sprintf(mb,"\\n\n*** MainMenuLoopCounter = %d ***,iMainMer
    while (bTransmitBusy);
    
    fMyPrintf(mb);
    while (bTransmitBusy);

    fMyPrintf(uArrayMenu);
    while (bTransmitBusy);
}
```
Double click MAIN.C and insert Code:

```c
uint32_t SoftwareTimer;
SYSTIMER_STATUS_t statusSYSTIMER;
```
Double click MAIN.C and insert Code:

```c
SoftwareTimer =
(uint32_t)SYSTIMER_CreateTimer(ONESEC, SYSTIMER_MODE_PERIODIC, (void*)fEverySecond, NULL);
if (SoftwareTimer != 0U)
{
    // Software timer is created successfully
} else
{
    while (1); // Software timer creation is failed
}
statusSYSTIMER = SYSTIMER_StartTimer(SoftwareTimer);
if (statusSYSTIMER == SYSTIMER_STATUS_SUCCESS)
{
    // Software timer is running
} else
{
    while (1); // Error during Software timer start operation
}
```
void fEverySecond (void);

int main(void)
{
    uint32_t SoftwareTimer;
    SYSTIMER_STATUS_t statusSYSTIMER;

    DADEV_Init(); // Initialization of DADEV APPs
    SoftwareTimer = SYSTIMER_CreateTimer(TICK_SEC, SYSTIMER_MODE_PERIODIC,(void*)fEverySecond, NULL);
    if (SoftwareTimer != 0)
    {
        ; // Software timer is created successfully
    }
    else
    {
        while (1); // Software timer creation is failed
    }
    statusSYSTIMER = SYSTIMER_StartTimer(SoftwareTimer);
    if (statusSYSTIMER == SYSTIMER_STATUS_SUCCESS)
    {
        ; // Software timer is running
    }
    else
    {
        while (1); // Error during Software timer start operation
    }
    iMainMenuLoopCounter++;
    printf("\n\nMainMenuLoopCounter = %d \n\n", iMainMenuLoopCounter); // Write formatted data to string
    myPrint(sc); // Print data to serial communication
    while (!TransmitBusy);
    while (!TransmitBusy);
    while (!TransmitBusy);
Double click MAIN.C and insert Code:

```c
void fEverySecond (void)
{
    if (bLEDsBlinkingStatusGlobal == ON)
    {
        fLEDs_TOGGLE();
    }
}
```
```c
void fLEDs_ON (void)
{
  DIGITAL_IO_SetOutputHigh(LED_PS_P8);
  DIGITAL_IO_SetOutputHigh(LED_PS_P9);
  bLEDsOnOffStatusGlobal = ON;
}

void fLEDs_OFF (void)
{
  DIGITAL_IO_SetOutputLow(LED_PS_P8);
  DIGITAL_IO_SetOutputLow(LED_PS_P9);
  bLEDsOnOffStatusGlobal = OFF;
}

void fLEDs_TOGGLE (void)
{
  if (bLEDsOnOffStatusGlobal == ON)
    fLEDs_OFF();
  else
    fLEDs_ON();
}

void fEverySecond (void)
{
  if (bLEDsBlinkingStatusGlobal == ON)
    fLEDs_TOGGLE();
}
```

Click Rebuild Active Project:
Note:
SYSTIMER: methods see APP Help:
DAVE CE Perspective -> right mouse button click at SYSTIMER SoftwareTimer1ms
-> APP Help:
Note:
SYSTIMER: methods see APP Help:

DAVE CE Perspective -> right mouse button click at SYSTIMER SoftwareTimer1ms
-> APP Help -> click Methods:
Start the debugger:

Click Debug:
Start the program:

Click Resume:
And see the result in the **Serial Terminal Program** **AND** on the **XMC4700 RelaxKit**:
Terminate the Debugger Connection:

Click Terminate:
And go back to the DAVE IDE perspective/view:
Save your XMC4700 Hello World project:

File -> Export…
Export -> Select: select/expand Infineon: select/click DAVE Projects

Click Next
Tick ☑ XMC4700 RelaxKit HelloWorld, click Select Archive File, click Browse
Export To Archive File:
Select/Create any Windows Directory of your choice (e.g. C:\XMC4700_Hello-World_UART)
File name: e.g. insert XMC4700_RelaxKit_HelloWorld_UART

Click Save
Click Finish

File - Exit
Conclusion:

In this step-by-step book you have learned how to use the XMC4700 RelaxKit together with the Dave tool chain.

Now you can easily expand your "hello world" program to suit your needs!

You can connect either a part of - or your entire application to the XMC4700 RelaxKit.

You are also able to benchmark any of your algorithms to find out if the selected microcontroller fulfils all the required functions within the time frame needed.

Have fun and enjoy working with the XMC4700 RelaxKit!
5.) Feedback (XMC4700, RelaxKit, Hello World, UART):
Your opinion, suggestions, and/or criticisms

If you have any suggestions please give feedback to:

E-Mail: wilhelm.brezovits@infineon.com

Your suggestions/ideas:

________________________________________________________

________________________________________________________

________________________________________________________

________________________________________________________

________________________________________________________

________________________________________________________

________________________________________________________

________________________________________________________