1. Project title: Using Hall sensor with XMC 4500 Relax Lite Kit

Munteanu Iuana **Sbera Cosmin-Ionut** **Vatavu Bogdan**

iuana_munteanu@yahoo.com **cosmin_07_92@yahoo.com** **b.vatavu@yahoo.com**

2. Abstract

Using XMC 4500 Relax Lite Kit and a hall sensor we created a project to measure the time between two interactions of the hall sensor and a permanent magnet.

3. Introduction, project aims and objectives

The differential Hall Effect sensor TLE4921-5U provides a high sensitivity and a superior stability over temperature and symmetrical thresholds in order to achieve a stable duty cycle. The integrated circuit (based on Hall effect) provides a digital signal output with frequency proportional to the speed of rotation. Unlike other rotational sensors differential Hall ICs are not influenced by radial vibration within the effective air gap of the sensor and require no external signal processing. The Differential Hall Sensor IC detects the motion and position of ferromagnetic and permanent magnet structures by measuring the differential flux density of the magnetic field. To detect ferromagnetic objects the magnetic field must be provided by a back biasing permanent magnet (*south* or *north* pole of the magnet attached to the rear unmarked side of the IC package).

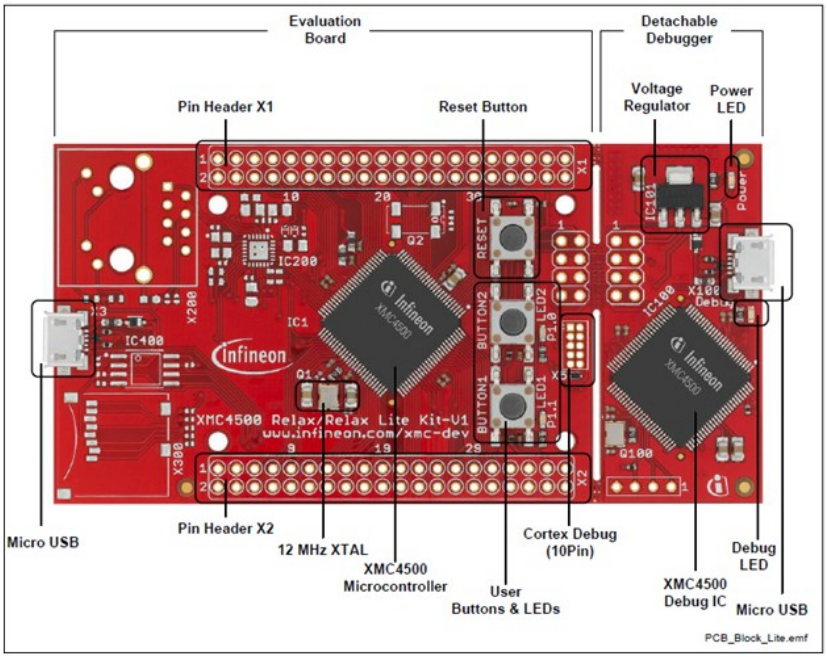
A seven-segment display (SSD), or seven-segment indicator, is a form of electronic display device for displaying decimal numerals that is an alternative to the more complex dot matrix displays. The seven elements of the display can be lit in different combinations to represent the arabic numerals. Often the seven segments are arranged in an oblique (slanted) arrangement, which aids readability. In most applications, the seven segments are of nearly uniform shape and size (usually elongated hexagons, though trapezoids and rectangles can also be used), though in the case of adding machines, the vertical segments are longer and more oddly shaped at the ends in an effort to further enhance readability.

The XMC4500 Relax Kit-V1 and the XMC4500 Relax Lite Kit-V1 are designed to evaluate the capabilities of the XMC4500 Microcontroller and the powerful, free of charge tool chain DAVE™. The XMC4500 Relax Kit extends the feature set with an Ethernet-enabled communication option, e.g. to run an embedded web server. You can store your own HTML web pages on a microSD Card or control the XMC4500 via the web browser on your PC. The XMC4500 Relax Lite Kit-V1 does not support the web server application, because the components for the Ethernet are not assembled. Both boards are marked with "XMC4500 Relax/Relax Lite Kit-V1".

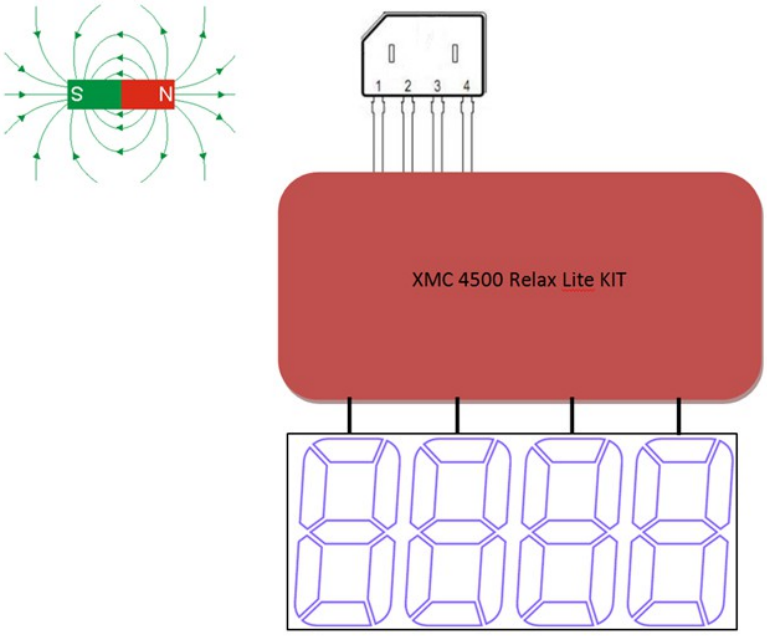
The main building blocks are:

- **XMC4500 Microcontroller in a LQFP100 package**
- **On-board USB debugger realized with a 2nd XMC4500 for serial wire debug**
- **Two 40 pin header X1 and X2**

- **On-board power generation for power supply of the XMC45000 Microcontroller and the debug IC**
- **2 User Buttons and 2 User LEDs**
- **USB Plug**



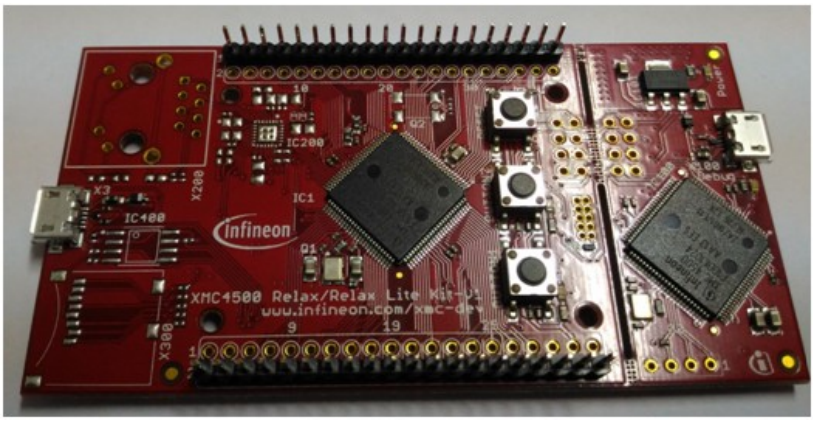
4. System overview



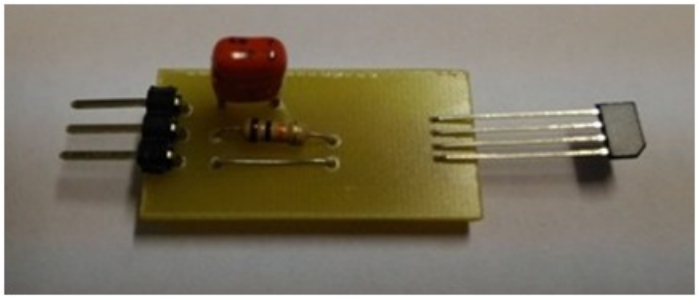
5. Schematics and components

For implementation we used the following components:

- XMC 4500 Relax Lite Kit



- TLE4921-5U Hall sensor



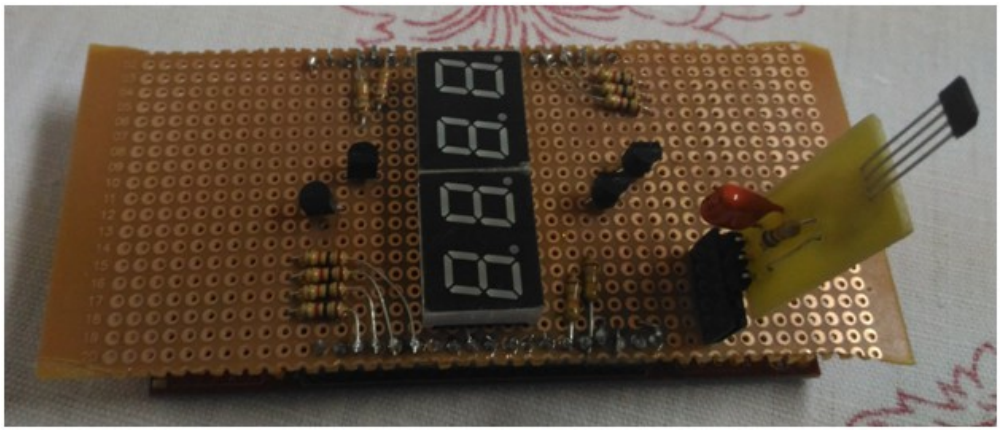
- 4 7-segment digits

- 4 2n3904 transistors

- 4 18Kohm resistors

- 7 1Kohm resistors

In the following image we see the final assemble of the components on a PCB board :



6. Software

DAVE (Infineon) Digital Application Virtual Engineer (DAVE™), a C/C++-language software development and code generation tool for microcontroller applications. DAVE is a standalone system with automatic code generation modules and is suited to develop software drivers for Infineon microcontrollers and aids the developer with automatically created C-level templates and user desired functionalities.

DAVE was developed by Infineon Technologies. Therefore, the automatic code generator supports only Infineon microcontrollers. The user also has to get used to the concept of the Eclipse IDE. The generated code can be also used on other (often non free) development environments from Keil, Tasking and so on.

Dave is a one single tool that has packaged the following components:

- **Eclipse based IDE**
- **Configurable source code libraries known as DAVE APPs**
- **Low level device drivers for various peripherals**
- **A Code Engine for generating source code libraries from APPs required by your application**
- **A free tool chain (GCC ARM) for building your application code**
- **A debugger (GDB ARM) to help download your application binary to the target board and debug it.**
- **A Software Development Kit (SDK) provides development environment for DAVE APPs Here is a pictorial view of what DAVE has to offer.**

The DIGITAL_IO pins we used are shown in the below image

APP Instance Name	APP Pin Name	Pin Number (Port)
D1		
	pin	#51 (P2.1)
D2		
	pin	#96 (P0.6)
D3		
	pin	#40 (P2.15)
D4		
	pin	#100 (P0.2)
HALL_SENSOR		
	pin	#98 (P0.4)
S1		
	pin	#57 (P5.1)
S2		
	pin	#68 (P1.15)
S3		
	pin	#70 (P1.13)
S4		
	pin	#72 (P1.11)
S5		
	pin	#3 (P0.10)
S6		
	pin	#4 (P0.9)
S7		
	pin	#6 (P3.1)

The source code for the project is:

```
#include <DAVE.h>
void stins() {
 DIGITAL_IO_SetOutputLow(&D4);
 DIGITAL_IO_SetOutputLow(&D3);
 DIGITAL_IO_SetOutputLow(&D2);
 DIGITAL_IO_SetOutputLow(&D1);
 DIGITAL_IO_SetOutputLow(&S1);
 DIGITAL_IO_SetOutputLow(&S2);
 DIGITAL_IO_SetOutputLow(&S3);
 DIGITAL_IO_SetOutputLow(&S4);
 DIGITAL_IO_SetOutputLow(&S5);
 DIGITAL_IO_SetOutputLow(&S6);
 DIGITAL_IO_SetOutputLow(&S7);
}
void val0() {
 DIGITAL_IO_SetOutputLow(&S1);
 DIGITAL_IO_SetOutputHigh(&S2);
 DIGITAL_IO_SetOutputHigh(&S3);
 DIGITAL_IO_SetOutputHigh(&S4);
 DIGITAL_IO_SetOutputHigh(&S5);
 DIGITAL_IO_SetOutputHigh(&S6);
 DIGITAL_IO_SetOutputHigh(&S7);
}
void val1() {
 DIGITAL_IO_SetOutputLow(&S1);
 DIGITAL_IO_SetOutputLow(&S2);
 DIGITAL_IO_SetOutputLow(&S3);
 DIGITAL_IO_SetOutputHigh(&S4);
```

```
 DIGITAL_IO_SetOutputHigh(&S5);
 DIGITAL_IO_SetOutputLow(&S6);
 DIGITAL_IO_SetOutputLow(&S7);
}
void val2() {
 DIGITAL_IO_SetOutputHigh(&S1);
 DIGITAL_IO_SetOutputLow(&S2);
 DIGITAL_IO_SetOutputHigh(&S3);
 DIGITAL_IO_SetOutputHigh(&S4);
 DIGITAL_IO_SetOutputLow(&S5);
 DIGITAL_IO_SetOutputHigh(&S6);
 DIGITAL_IO_SetOutputHigh(&S7);
}
void val3() {
 DIGITAL_IO_SetOutputHigh(&S1);
 DIGITAL_IO_SetOutputLow(&S2);
 DIGITAL_IO_SetOutputHigh(&S3);
 DIGITAL_IO_SetOutputHigh(&S4);
 DIGITAL_IO_SetOutputHigh(&S5);
 DIGITAL_IO_SetOutputLow(&S6);
 DIGITAL_IO_SetOutputHigh(&S7);
}
void val4() {
 DIGITAL_IO_SetOutputHigh(&S1);
 DIGITAL_IO_SetOutputHigh(&S2);
 DIGITAL_IO_SetOutputLow(&S3);
 DIGITAL_IO_SetOutputHigh(&S4);
 DIGITAL_IO_SetOutputHigh(&S5);
 DIGITAL_IO_SetOutputLow(&S6);
 DIGITAL_IO_SetOutputLow(&S7);
}
void val5() {
 DIGITAL_IO_SetOutputHigh(&S1);
 DIGITAL_IO_SetOutputHigh(&S2);
 DIGITAL_IO_SetOutputHigh(&S3);
 DIGITAL_IO_SetOutputLow(&S4);
 DIGITAL_IO_SetOutputHigh(&S5);
 DIGITAL_IO_SetOutputLow(&S6);
 DIGITAL_IO_SetOutputHigh(&S7);
}
void val6() {
 DIGITAL_IO_SetOutputHigh(&S1);
 DIGITAL_IO_SetOutputHigh(&S2);
 DIGITAL_IO_SetOutputHigh(&S3);
 DIGITAL_IO_SetOutputLow(&S4);
 DIGITAL_IO_SetOutputHigh(&S5);
```

```
 DIGITAL_IO_SetOutputHigh(&S6);
 DIGITAL_IO_SetOutputHigh(&S7);
}
void val7() {
 DIGITAL_IO_SetOutputLow(&S1);
 DIGITAL_IO_SetOutputLow(&S2);
 DIGITAL_IO_SetOutputHigh(&S3);
 DIGITAL_IO_SetOutputHigh(&S4);
 DIGITAL_IO_SetOutputHigh(&S5);
 DIGITAL_IO_SetOutputLow(&S6);
 DIGITAL_IO_SetOutputLow(&S7);
}
void val8() {
 DIGITAL_IO_SetOutputHigh(&S1);
 DIGITAL_IO_SetOutputHigh(&S2);
 DIGITAL_IO_SetOutputHigh(&S3);
 DIGITAL_IO_SetOutputHigh(&S4);
 DIGITAL_IO_SetOutputHigh(&S5);
 DIGITAL_IO_SetOutputHigh(&S6);
 DIGITAL_IO_SetOutputHigh(&S7);
}
void val9() {
 DIGITAL_IO_SetOutputHigh(&S1);
 DIGITAL_IO_SetOutputHigh(&S2);
 DIGITAL_IO_SetOutputHigh(&S3);
 DIGITAL_IO_SetOutputHigh(&S4);
 DIGITAL_IO_SetOutputHigh(&S5);
 DIGITAL_IO_SetOutputLow(&S6);
 DIGITAL_IO_SetOutputHigh(&S7);
}
void init() {
 DIGITAL_IO_SetOutputHigh(&D4);
 DIGITAL_IO_SetOutputHigh(&D3);
 DIGITAL_IO_SetOutputHigh(&D2);
 DIGITAL_IO_SetOutputHigh(&D1);
 DIGITAL_IO_SetOutputLow(&S1);
 DIGITAL_IO_SetOutputHigh(&S2);
 DIGITAL_IO_SetOutputHigh(&S3);
 DIGITAL_IO_SetOutputHigh(&S4);
 DIGITAL_IO_SetOutputHigh(&S5);
 DIGITAL_IO_SetOutputHigh(&S6);
 DIGITAL_IO_SetOutputHigh(&S7);
}
void afis(int val) {
 switch (val) {
 case 0: val0(); break;
```

```
 case 1: val1(); break;
 case 2: val2(); break;
 case 3: val3(); break;
 case 4: val4(); break;
 case 5: val5(); break;
 case 6: val6(); break;
 case 7: val7(); break;
 case 8: val8(); break;
 case 9: val9(); break;
 default: stins(); break;   }
}
int main(void) {
 int v1 = 0;
 int v2 = 0;
 int v3 = 0;
 int v4 = 0;
 uint32_t pin_status = 0;
 uint32_t t;
 int c, v = 0;
 DAVE_STATUS_t status;
 status = DAVE_Init();
 if (status == DAVE_STATUS_FAILURE) {
    XMC_DEBUG("DAVE APPs initialization failed\n");
    while (1U) {   }
 }
 init();
 while (1U) {
    pin_status = DIGITAL_IO_GetInput(&HALL_SENSOR);
    if (pin_status == 1) {  v++;  }
    if (v > 2 && v % 2 == 1) {
       for (t = 0; t < 0x3fffff; t++) {
          stins();
          DIGITAL_IO_SetOutputHigh(&D4);
          afis(v4);
          for (c = 0; c < 100; c++);
          t = t + 100;
          stins();
          DIGITAL_IO_SetOutputHigh(&D3);
          afis(v3);
          for (c = 0; c < 100; c++);
          t = t + 100;
          stins();
          DIGITAL_IO_SetOutputHigh(&D2);
          afis(v2);
```

```
            for (c = 0; c < 100; c++);
            t = t + 100;
            stins();
            DIGITAL_IO_SetOutputHigh(&D1);
            afis(v1);
            for (c = 0; c < 100; c++);
            t = t + 100;
        }
        v4++;
        if (v4 == 10) {   v3++; v4 = 0; }
        if (v3 == 6) { v2++; v3 = 0; }
        if (v2 == 10) {   v1++; v2 = 0; }
        if (v1 == 6) { v1 = v2 = v3 = v4 = 0;  }
    } else {
        for (t = 0; t < 0x5fffff; t++) {
            stins();
            DIGITAL_IO_SetOutputHigh(&D4);
            afis(v4);
            for (c = 0; c < 100; c++);
            t = t + 100;
            stins();
            DIGITAL_IO_SetOutputHigh(&D3);
            afis(v3);
            for (c = 0; c < 100; c++);
            t = t + 100;
            stins();
            DIGITAL_IO_SetOutputHigh(&D2);
            afis(v2);
            for (c = 0; c < 100; c++);
            t = t + 100;
            stins();
            DIGITAL_IO_SetOutputHigh(&D1);
            afis(v1);
            for (c = 0; c < 100; c++);
            t = t + 100;
        }
        v1 = v2 = v3 = v4 = 0;
    }
 }
}
```

7. Project results & applications

After the power on the project is on init, the value shown on the digits is

The following image shows the time since the magnet passed by the hall

by default 0. **sensor.**

The timer is reset to value 0000 after the second interaction of the magnet with the hall sensor. The circuit is running in a loop which allows us to use the timer multiple times because after the reset we can start again the timer.

8. Reference

http://www.infineon.com/dgdl/Board_Users_Manual_XMC4500_Relax_Kit-V1_R1.2_released.pdf?fileId=db3a30433acf32c9013adf6b97b112f9

http://www.infineon.com/dgdl/Infineon-TLE4921_5U-DS-v01_01-en.pdf?fileId=db3a304319c6f18c011a2a97e3a612d8

http://www.avagotech.com/products/leds-and-displays/7-segment/through-hole/hdsp-n151

http://www.infineon.com/cms/de/product/microcontroller/development-tools-software-and-kits/dave-version-4-free-development-platform-for-code-generation/channel.html?channel=db3a30433580b37101359f8ee6963814