VRS cvičenie 2

# Úloha 1

Spustili sme port A

RCC\_AHBPeriphClockCmd(RCC\_AHBPeriph\_GPIOA, *ENABLE*);

Inicializovali sme pin na porte A číslo 5, na ktorý je pripojená LED3, ako výstup.

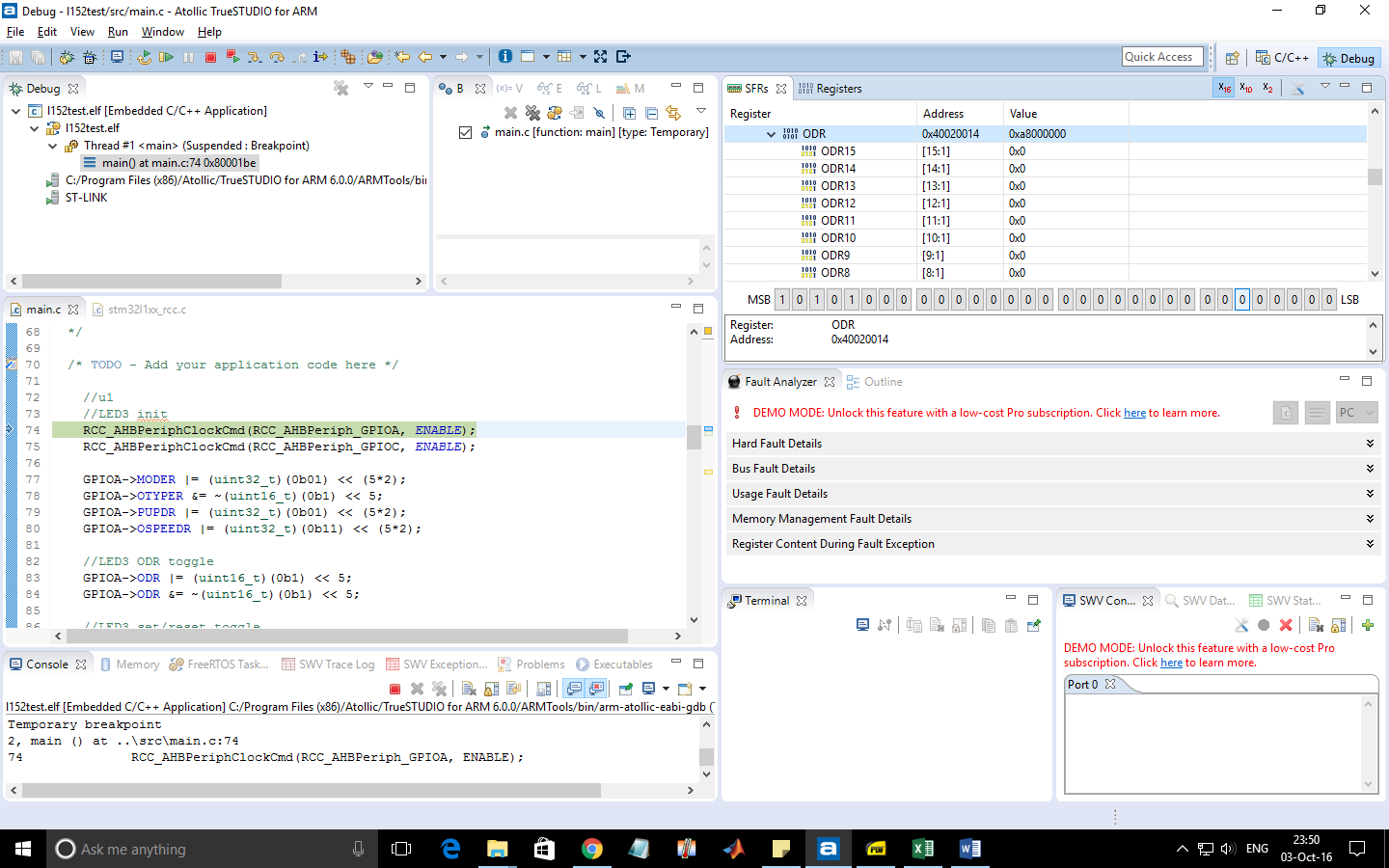
GPIOA->MODER |= (uint32\_t)(0b01) << (5\*2);

GPIOA->OTYPER &= ~(uint16\_t)(0b1) << 5;

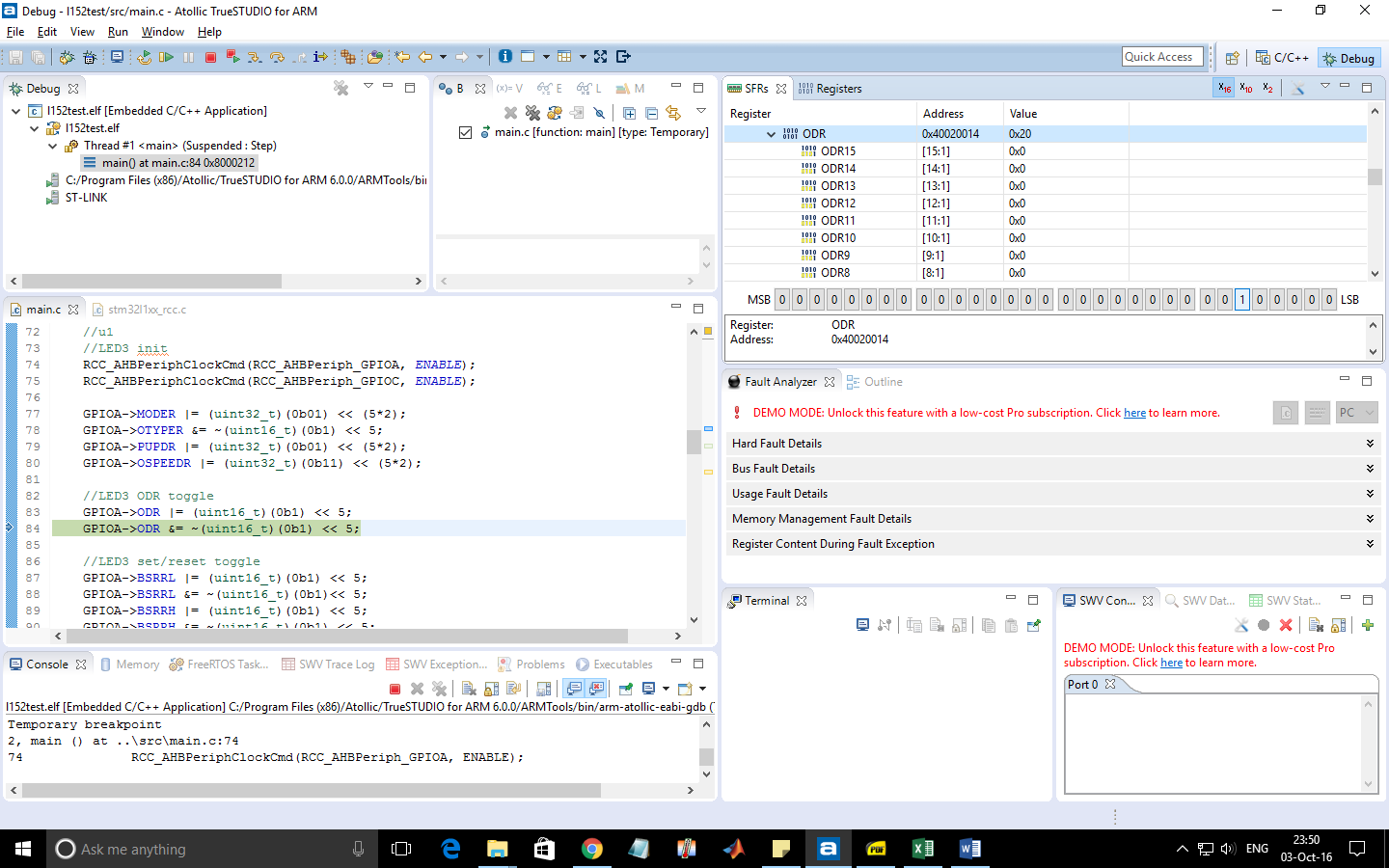
GPIOA->PUPDR |= (uint32\_t)(0b01) << (5\*2);

GPIOA->OSPEEDR |= (uint32\_t)(0b11) << (5\*2);

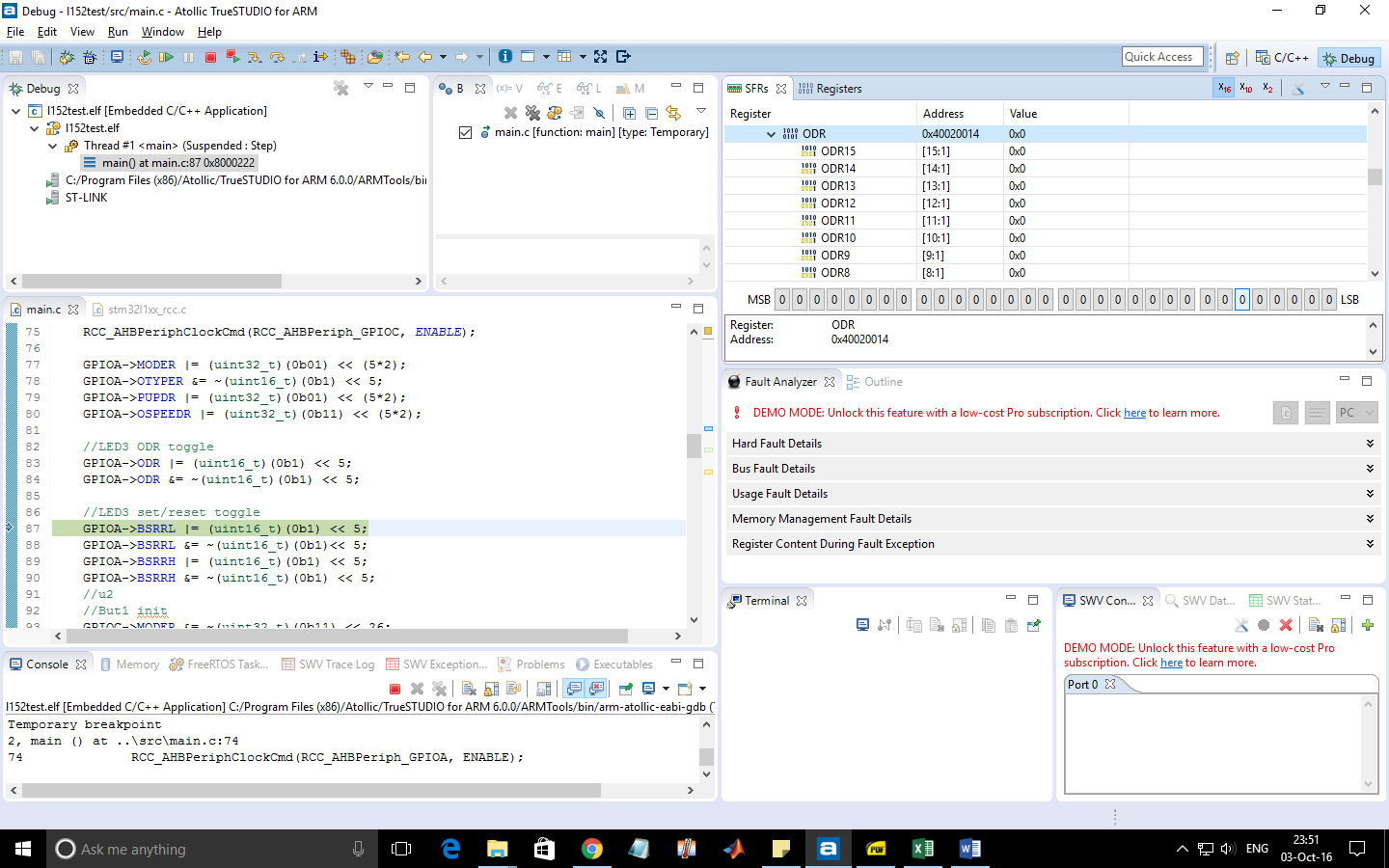
Postupne sme skúšali rozsvietiť LED pomocou registra ODR a BSRR .



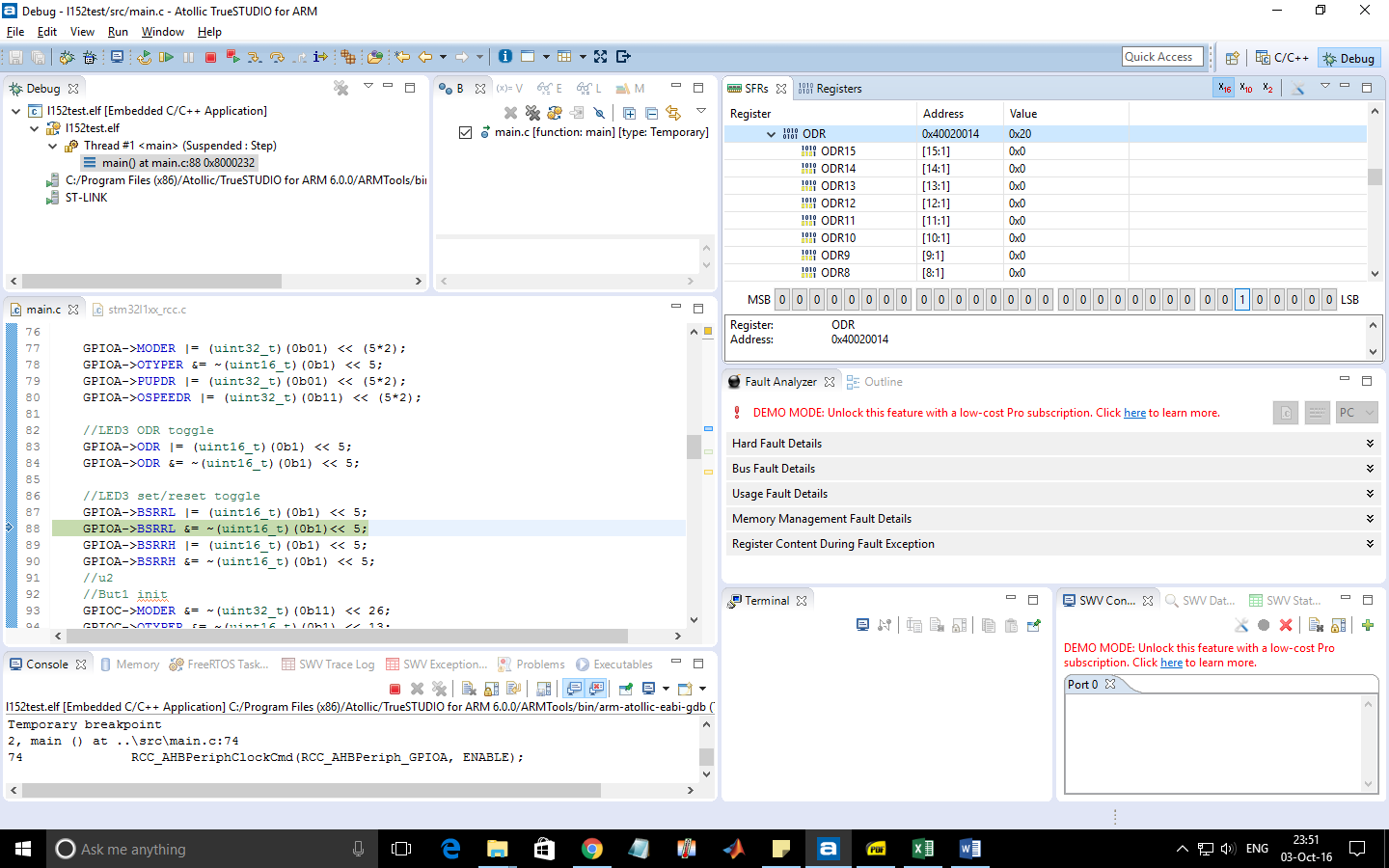
Obrázok 1 Spustenie programu – ODR je nula.



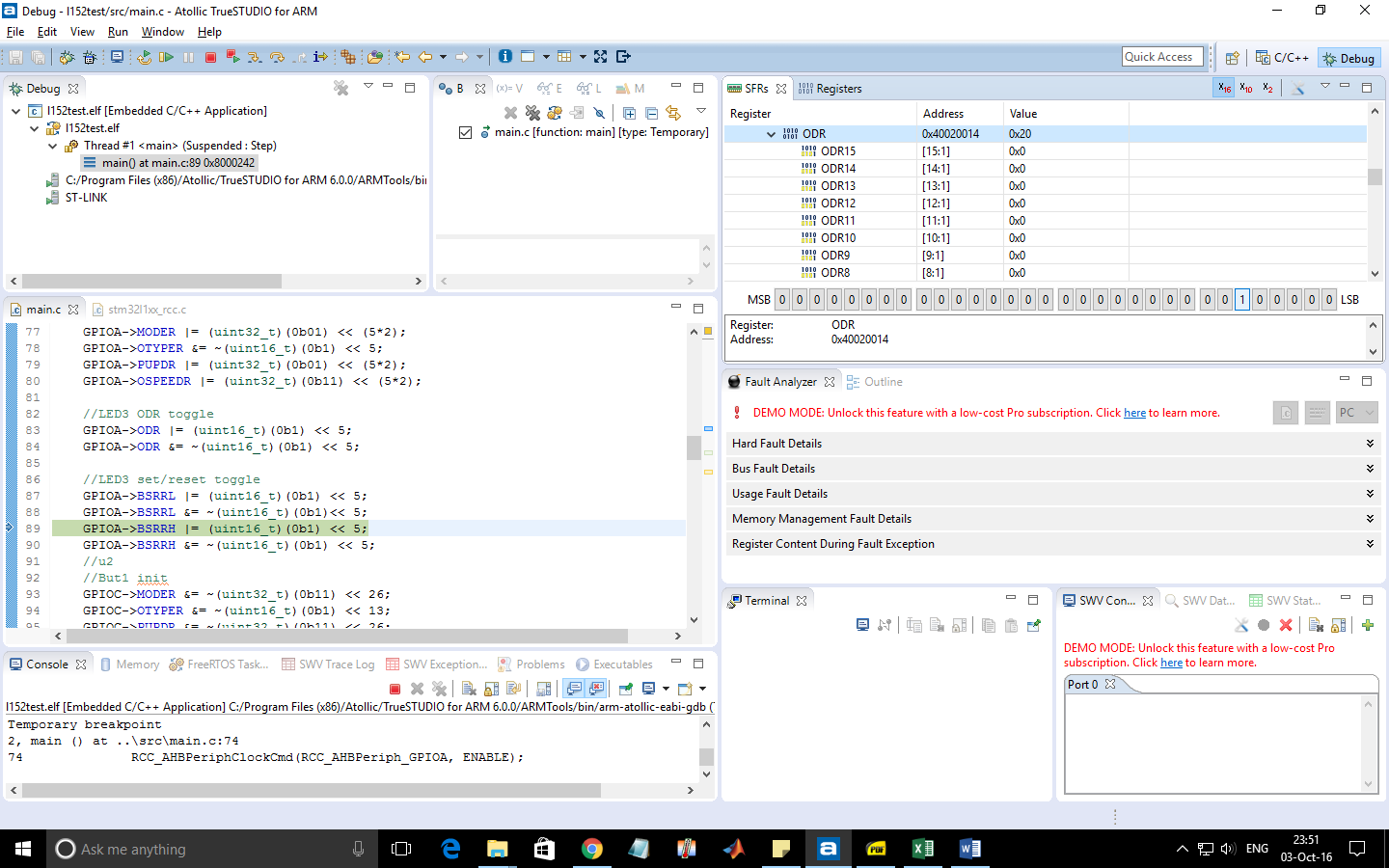
Obrázok 2 Nastavenie ODR na 1. LED zasvieti.



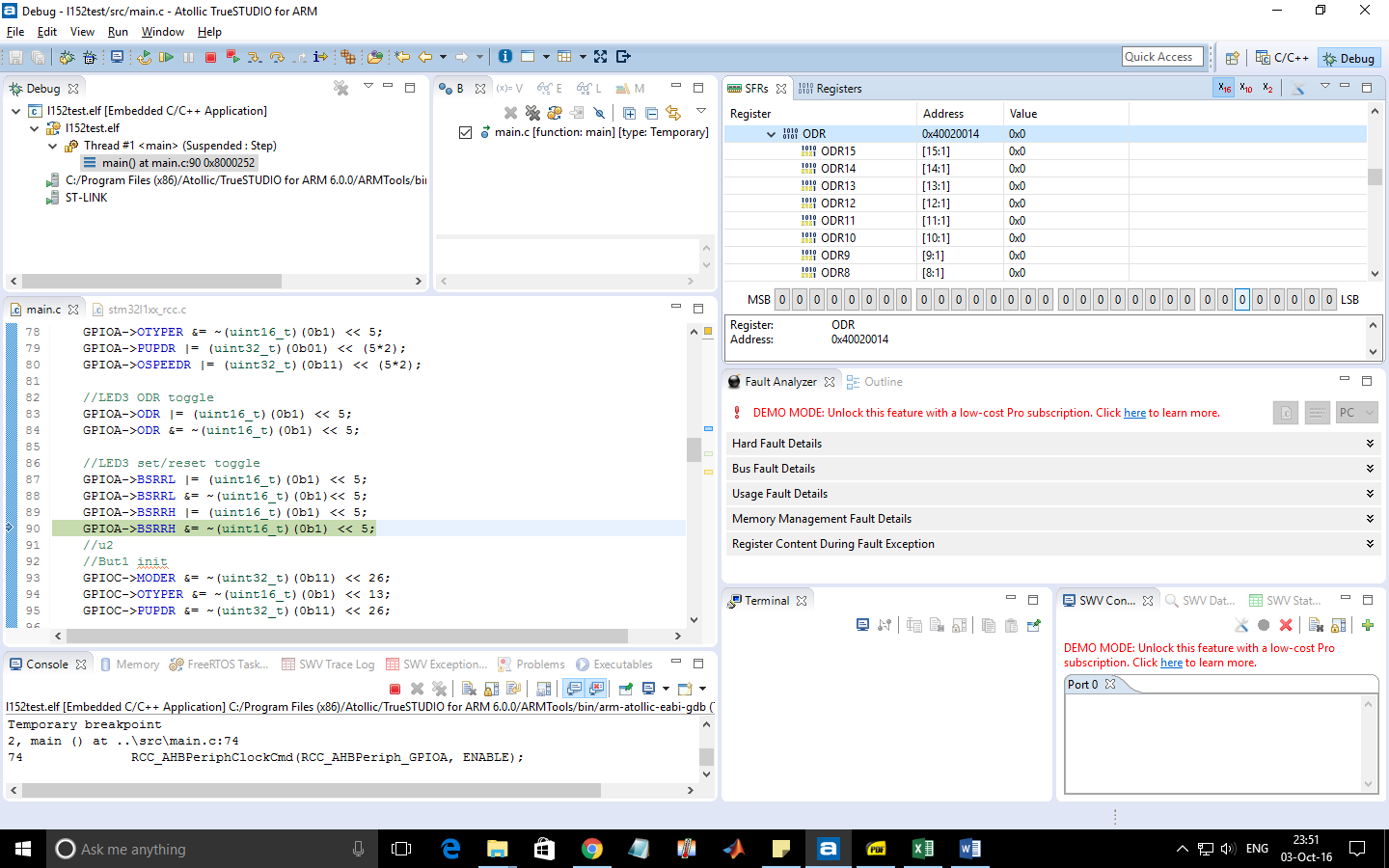
Obrázok 3 Nastavenie ODR na 0. LED zhasne.



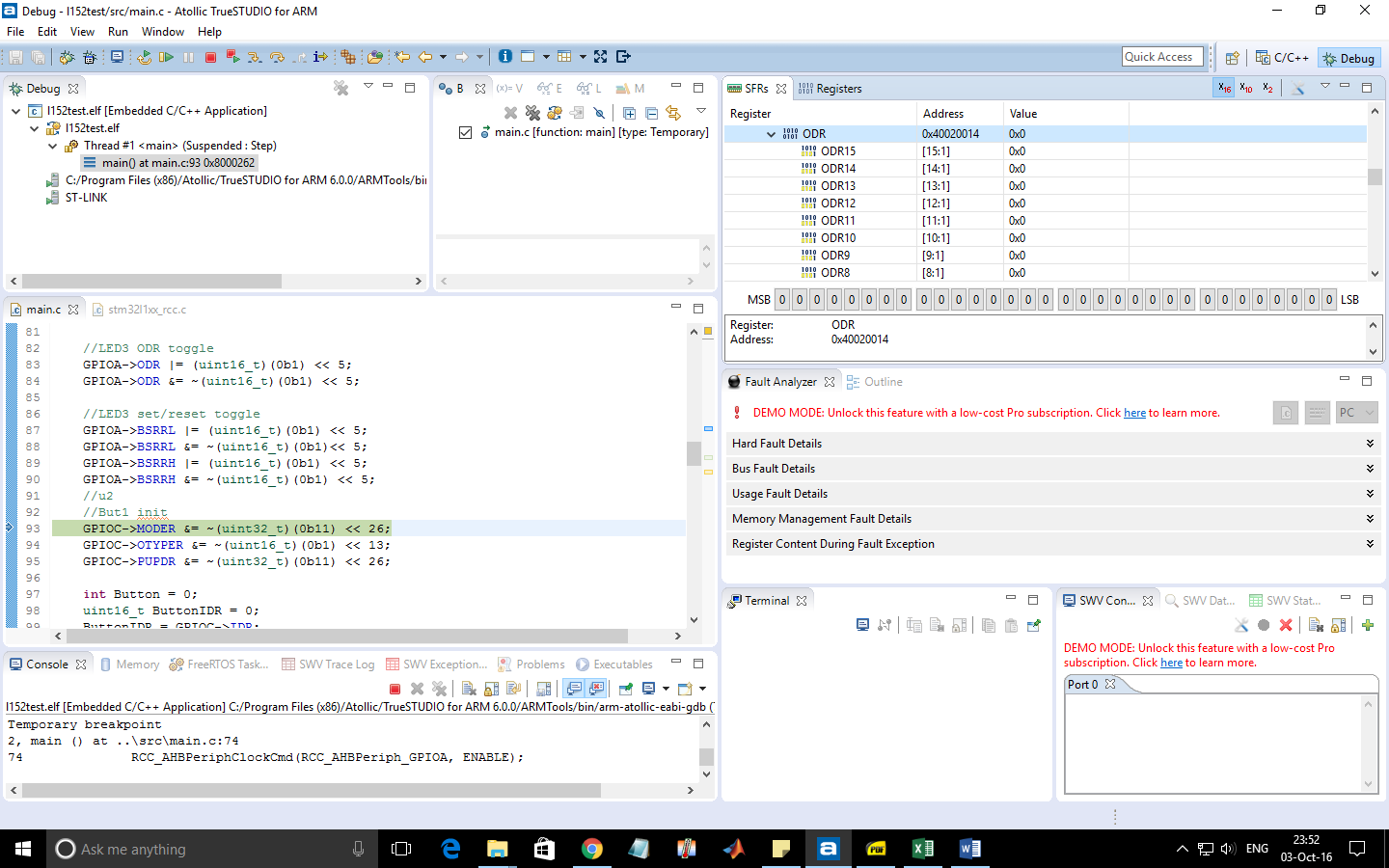
Obrázok 4 Nastavenie registra BSSRL na 1. LED zasvieti.



Obrázok 5 Nastavenie registra BSSRL na 0. LED stale svieti.



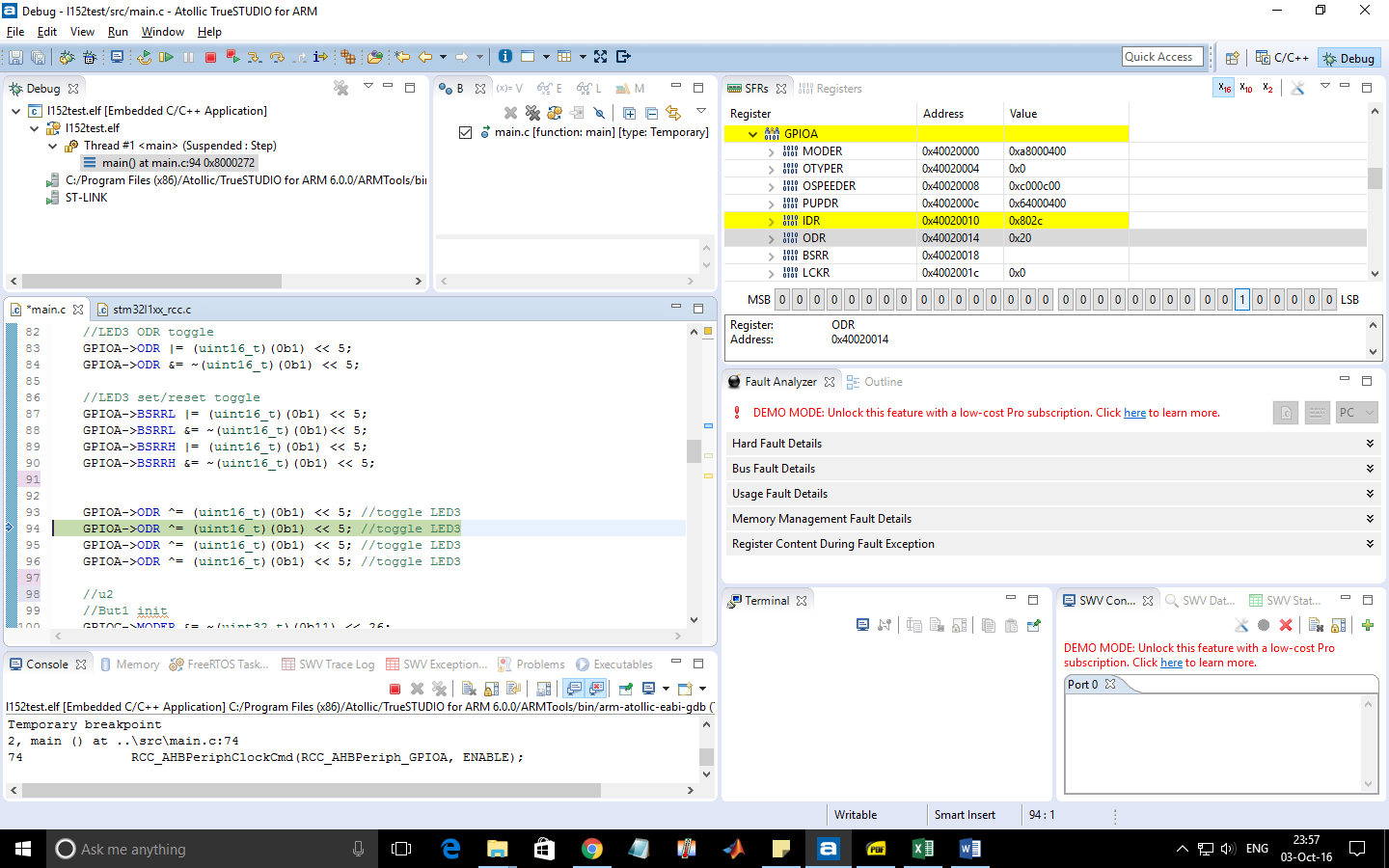
Obrázok 6 Nastavenie registra BSSRH na 1. LED zhasnei.



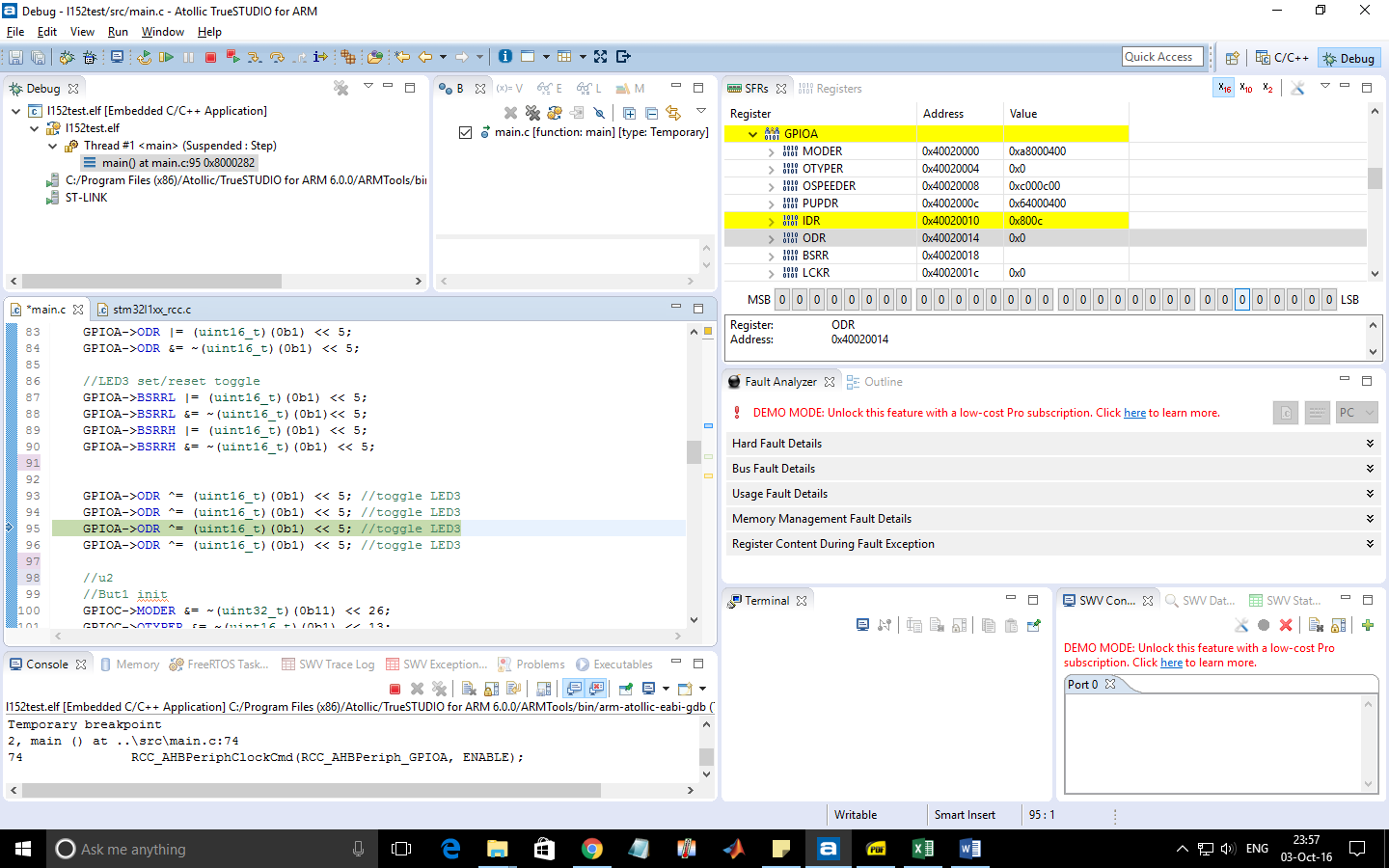
Obrázok 7 Nastavenie registra BSSRL na 0. LED stále nesvieti.

Prepínanie LED pomocou aplikovania funkcie XOR na register ODR.

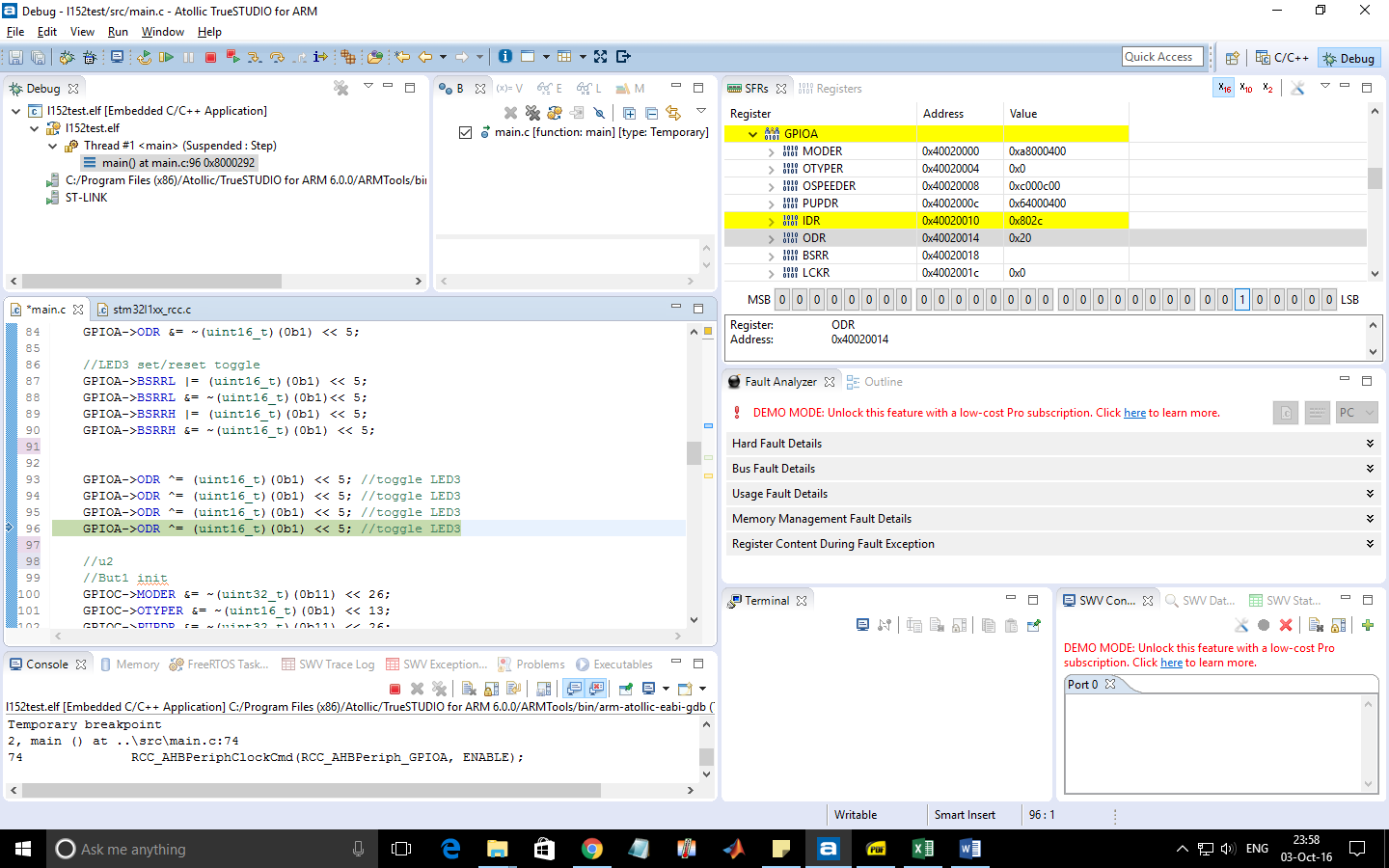
GPIOA->ODR ^= (uint16\_t)(0b1) << 5; //toggle LED3



Obrázok 8 ODR bolo 0 zmení sa na 1. LED sa rozsvieti.



Obrázok 9 ODR bolo 1, zmení sa na 0. LED zhasne.



Obrázok 10 ODR bolo 0, zmení sa na 1, LED ra rozsvieti.

# Úloha 2

Zapli sme port C.

RCC\_AHBPeriphClockCmd(RCC\_AHBPeriph\_GPIOC, *ENABLE*);

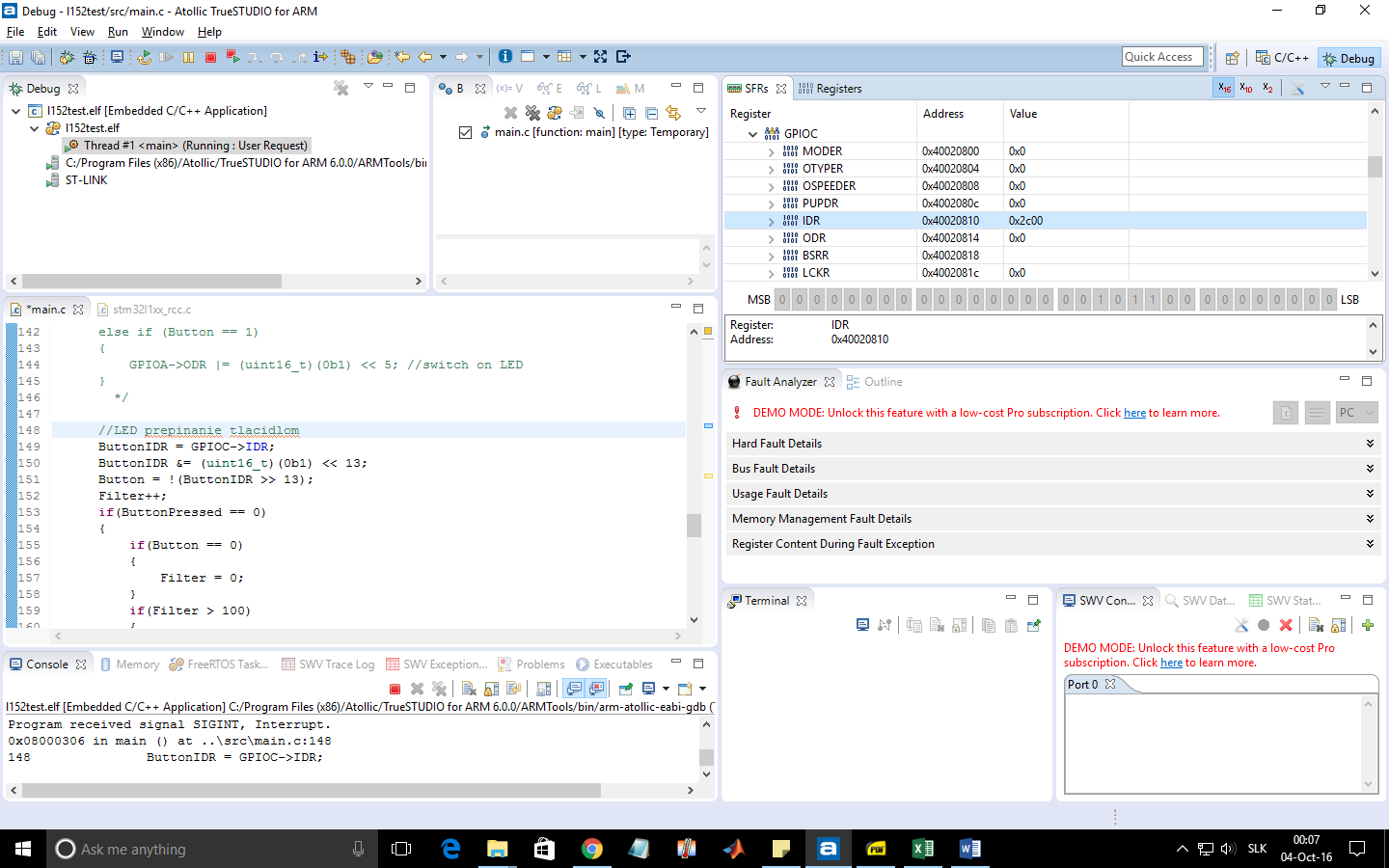
Inicializovali sme pin 13 na porte C pomocou zapísania príslušných hodnôt do registrov, tak aby sme ostatné časti registrov neovplyvnili.

GPIOC->MODER &= ~(uint32\_t)(0b11) << 26;

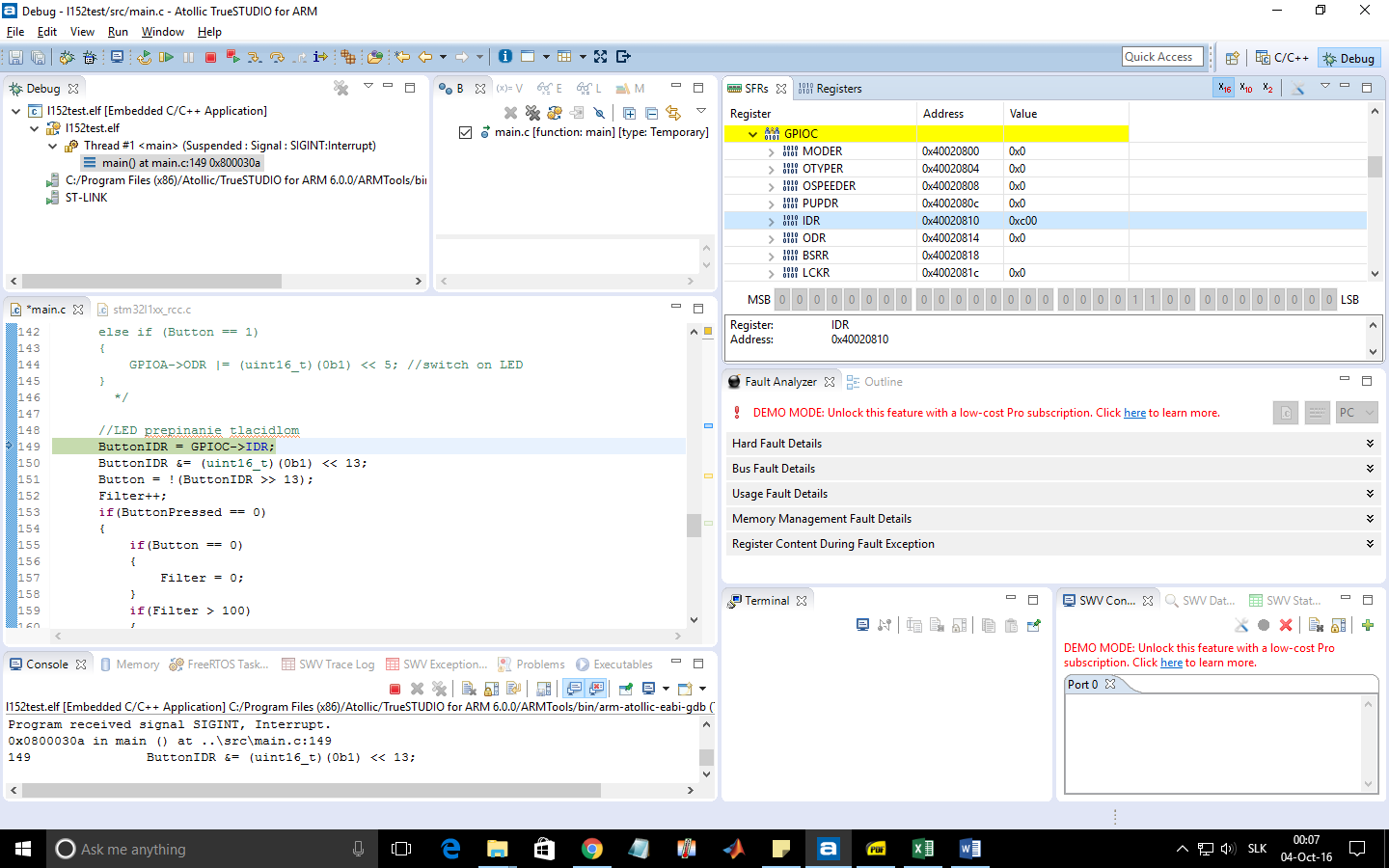
GPIOC->OTYPER &= ~(uint16\_t)(0b1) << 13;

GPIOC->PUPDR &= ~(uint32\_t)(0b11) << 26;

Vyskúšali sme debugovať náš program a pozastavili sme ho pri stlačenom a nestlačenom tlačidle, aby sme si overili hodnotu registra IDS.



Obrázok 11 Pustené tlačítko, BUTTON = 0, IDR je 1.



Obrázok 12 Stlačené tlačítko, BUTTON = 1, IDR je 0.

# Úloha 3

Vytvorili sme program, ktorý striedavo zasvecoval a zhasínal LED3.

pocitadlo++;

**if** (pocitadlo > 1000000)

{

GPIOA->ODR ^= (uint16\_t)(0b1) << 5; //toggle LED3

pocitadlo = 0;

}

Tento podprogram sa nachádza v hlavnej nekonečnej slučke. Počas jej prechodu sa inkrementuje premenná „pocitadlo“, ktorá keď dosiahne hodnotu 1000000 tak sa resetuje a zmení stav LED3.

Vytvorili sme program, ktorý sledoval stlačenie tlačidla, a podľa toho rozsvecoval LED3 aby jej stav zodpovedal stlačeniu tlačidla.

ButtonIDR = GPIOC->IDR;

ButtonIDR &= (uint16\_t)(0b1) << 13;

Button = (**int**)!(ButtonIDR >> 13);

**if** (Button == 0)

{

GPIOA->ODR &= ~(uint16\_t)(0b1) << 5; //switch off LED

}

**else** **if** (Button == 1)

{

GPIOA->ODR |= (uint16\_t)(0b1) << 5; //switch on LED

}

**V tejto úlohe sme prepínali stav LED pomocou zapísania registra ODR. Hodnotu tlačidla sme snímali pomocou čítania registra IDR a pokiaľ tlačidlo bolo stlačené, LED zasvietila, inak nesvietila.**

Vytvorili sme program, ktorý po stlačení Tlačidla zmenil stav LED3 s ohľadom na nedokonalosť tlačidla a teda oscilácie hodnoty IDR pri stlačení.

ButtonIDR = GPIOC->IDR;

ButtonIDR &= (uint16\_t)(0b1) << 13;

Button = !(ButtonIDR >> 13);

Filter++;

**if**(ButtonPressed == 0)

{

**if**(Button == 0)

{

Filter = 0;

}

**if**(Filter > 100)

{

ButtonPressed = 1;

GPIOA->ODR ^= (uint16\_t)(0b1) << 5; //toggle LED3

}

}

**else** **if**(ButtonPressed == 1)

{

**if**(Button == 1)

{

Filter = 0;

}

**if**(Filter > 100)

{

ButtonPressed = 0;

}

}

**V tejto úlohe sme prepínali stav LED z 0 na 1 a naopak len keď sme stlačili tlačidlo pomocou zapísania registra ODR. Pomocou premennej filter sme ošetrili prechodové deje zákmitu tlačidla, ktoré sú spôsobené nedokonalým mechanickým spojením medzi kontaktami tlačidla.**

# Cvičenie 2 pomocou knižnice

## Úloha 1 pomocou knižnice

RCC\_AHBPeriphClockCmd(RCC\_AHBPeriph\_GPIOA, *ENABLE*);

**Spustili sme port A**

GPIO\_InitTypeDef gpioInitStruc;

gpioInitStruc.GPIO\_Mode = *GPIO\_Mode\_OUT*;

gpioInitStruc.GPIO\_OType = *GPIO\_OType\_PP*;

gpioInitStruc.GPIO\_Pin = GPIO\_Pin\_5;

gpioInitStruc.GPIO\_Speed = *GPIO\_Speed\_400KHz*;

**Zapísali sme potrebné parametre do štruktúry pre parametre pinu GPIO\_Pin\_5 ako výstup**

GPIO\_Init(GPIOA, &gpioInitStruc);

**Zapísali sme hodnoty štruktúry do registrov**

GPIO\_WriteBit(GPIOA, GPIO\_Pin\_5, *Bit\_SET*);//zapne led

GPIO\_WriteBit(GPIOA, GPIO\_Pin\_5, *Bit\_RESET*);

**Pomocou funkcie WriteBit sme zapisovali hodnoty na výstup – Bit\_SET = 1, Bit\_RESET = 0**

GPIO\_ToggleBits(GPIOA, GPIO\_Pin\_5);

GPIO\_ToggleBits(GPIOA, GPIO\_Pin\_5);

GPIO\_ToggleBits(GPIOA, GPIO\_Pin\_5);

GPIO\_ToggleBits(GPIOA, GPIO\_Pin\_5);

**Pomocou funkcie ToggleBits (ODR) sme prepisovali hodnotu výstupu z 0 na 1 a naopak**

## Úloha 2 pomocou knižnice

RCC\_AHBPeriphClockCmd(RCC\_AHBPeriph\_GPIOC, *ENABLE*);

**Spustili sme port C**

gpioInitStruc.GPIO\_Mode = *GPIO\_Mode\_IN*;

gpioInitStruc.GPIO\_OType = *GPIO\_OType\_PP*;

gpioInitStruc.GPIO\_Pin = GPIO\_Pin\_13;

gpioInitStruc.GPIO\_PuPd = *GPIO\_PuPd\_NOPULL*;

gpioInitStruc.GPIO\_Speed = *GPIO\_Speed\_40MHz*;

GPIO\_Init(GPIOC, &gpioInitStruc);

**Zapísali sme potrebné parametre do štruktúry pre parametre pinu Pin\_13 ako vstup**

uint8\_t Button = !GPIO\_ReadInputDataBit(GPIOC, GPIO\_Pin\_13);

Button = !GPIO\_ReadInputDataBit(GPIOC, GPIO\_Pin\_13);

**Čítali sme hodnotu tlačidla pomocou funkcie GPIO\_ReadInputDataBit, pre stlačené tlačidlo hodnota bitu = 1, pre pustené tlačidlo hodnota bitu = 0.**

## Úloha 3 pomocou knižnice

**long** pocitadlo = 0;

**int** ButtonPressed = 0;

**int** Filter = 0;  
**while** (1){  
 pocitadlo++;  
 if (pocitadlo > 1000000){  
 GPIO\_ToggleBits(GPIOA, GPIO\_Pin\_5); //toggle LED3  
 pocitadlo = 0;  
 }  
}

**V tejto úlohe sme pomocou funkcie ToggleBits prepínali stav LED z hodnoty 0 na 1 a naopak vždy, keď hodnota pocitadlo dosiahla hodnotu 1000000.**

**while** (1){

Button = !GPIO\_ReadInputDataBit(GPIOC, GPIO\_Pin\_13);

if (Button == 0)

{

GPIO\_WriteBit(GPIOA, GPIO\_Pin\_5, Bit\_RESET); //switch off LED

}

else

{

GPIO\_WriteBit(GPIOA, GPIO\_Pin\_5, Bit\_SET); //switch on LED

}

}

**V tejto úlohe sme prepínali stav LED pomocou funkcie WriteBit – BitRESET = Vypnúť LED, BitSET = Zapnúť LED. Hodnotu tlačidla sme snímali pomocou funkcie !GPIO\_ReadInputDataBit a pokiaľ bolo stlačené, LED zasvietila, inak nesvietila.**

**while** (1){

Button = !GPIO\_ReadInputDataBit(GPIOC, GPIO\_Pin\_13);

Filter++;

**if**(ButtonPressed == 0)

{

**if**(Button == 0)

{

Filter = 0;

}

**if**(Filter > 100)

{

ButtonPressed = 1;

GPIO\_ToggleBits(GPIOA, GPIO\_Pin\_5); //toggle LED3

}

}

**else** **if**(ButtonPressed == 1)

{

**if**(Button == 1)

{

Filter = 0;

}

**if**(Filter > 100)

{

ButtonPressed = 0;

}

}  
}

**V tejto úlohe sme prepínali stav LED z 0 na 1 a naopak len keď sme stlačili tlačidlo pomocou funkcie ToggleBits. Pomocou premennej filter sme ošetrili prechodové deje zákmitu tlačidla, ktoré sú spôsobené nedokonalým mechanickým spojením medzi kontaktami tlačidla.**

# Záver:

V úlohách sme pracovali so vstupmi a výstupmi mikroprocesora, ktoré sme nastavovali v prvých 3 úlohách pomocou priameho zápisu do registrov a v ďalších úlohách sme priame zápisy do registrov nahradili funkciami z knižnice. Zistili sme, že pre nás je rýchlejší a komfortnejší zápis pomocou knižníc, ale na druhej strane, pomocou registrov vieme pristupovať k vstupom a výstupom rýchlejšie.