**Vnorené RS**

Tomáš Kolek

**Obsah**

[**Úloha 1a** 2](#_Toc463695447)

[**Úloha 1b** 7](#_Toc463695448)

[**Úloha 2a** 8](#_Toc463695449)

[**Úloha 2b** 10](#_Toc463695450)

[**Úloha 3.1a** 11](#_Toc463695451)

[**Úloha 3.1b** 13](#_Toc463695452)

[**Úloha 3.2a** 15](#_Toc463695453)

[**Úloha 3.2b** 16](#_Toc463695454)

[**Úloha 3.3a** 18](#_Toc463695455)

[**Úloha 3.3b** 19](#_Toc463695456)

[**Link na GitHub** 21](#_Toc463695457)

# **Úloha 1a**

Nastavíme pin 5 periférie GPIOA podľa zdania:

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

GPIOA->OTYPER &= ~((uint16\_t)(1<<5));

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

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

Zapnutie a vypnutie LED pomocou ODR registra:

GPIOA->ODR |= 0b0000000000100000;

GPIOA->ODR &= ~(0b0000000000100000);

Zapnutie a vypnutie LED pomocou set/reset BSRR registra:

GPIOA->BSRRL |= ((uint16\_t)(1<<5));

GPIOA->BSRRH |= ((uint16\_t)(1<<5));

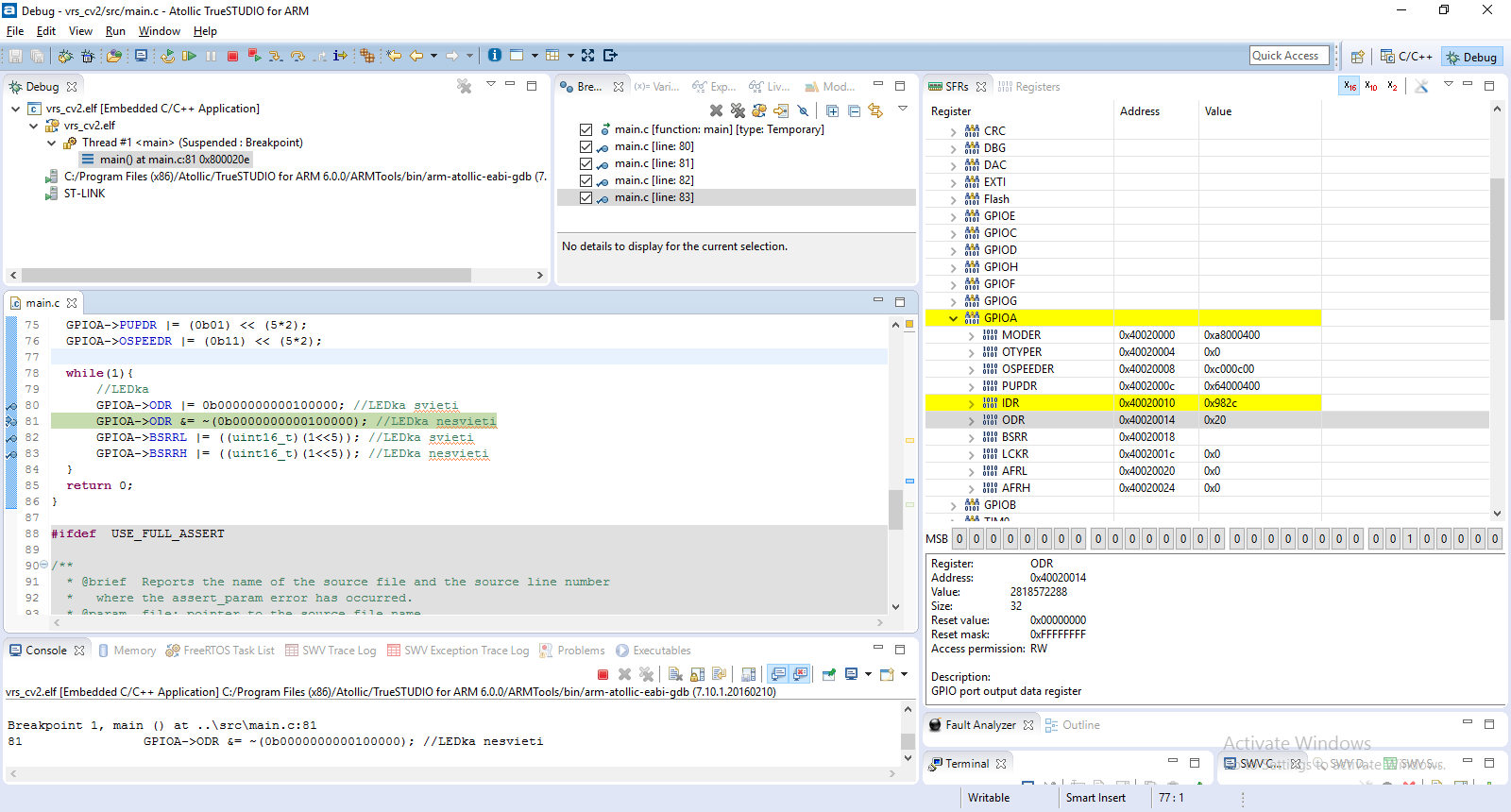
Prepínanie stavu LED pomocou ODR registra a operátora XOR:

GPIOA->ODR ^= (1<<5);

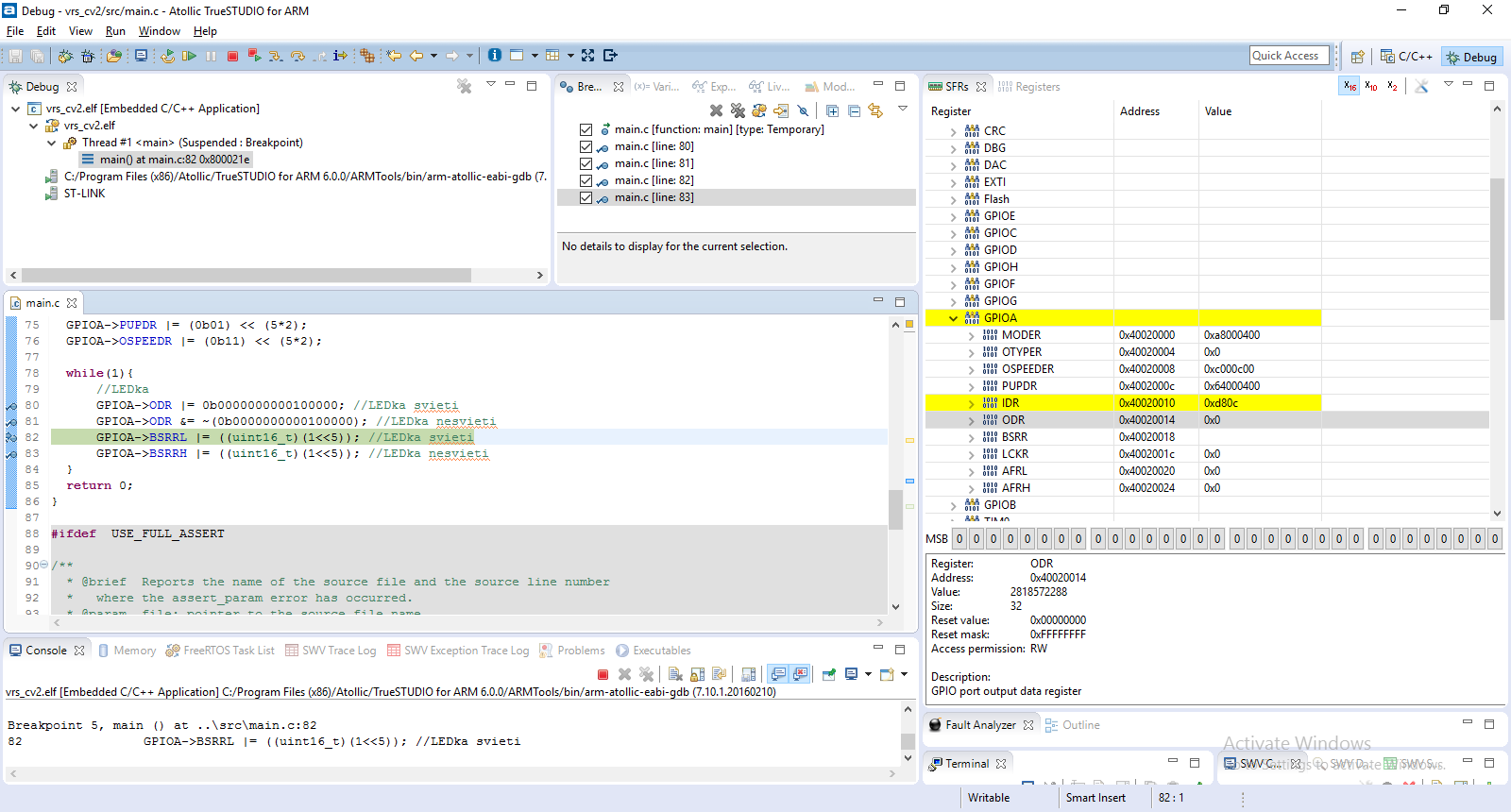
GPIOA->ODR ^= (1<<5);

GPIOA->ODR ^= (1<<5);

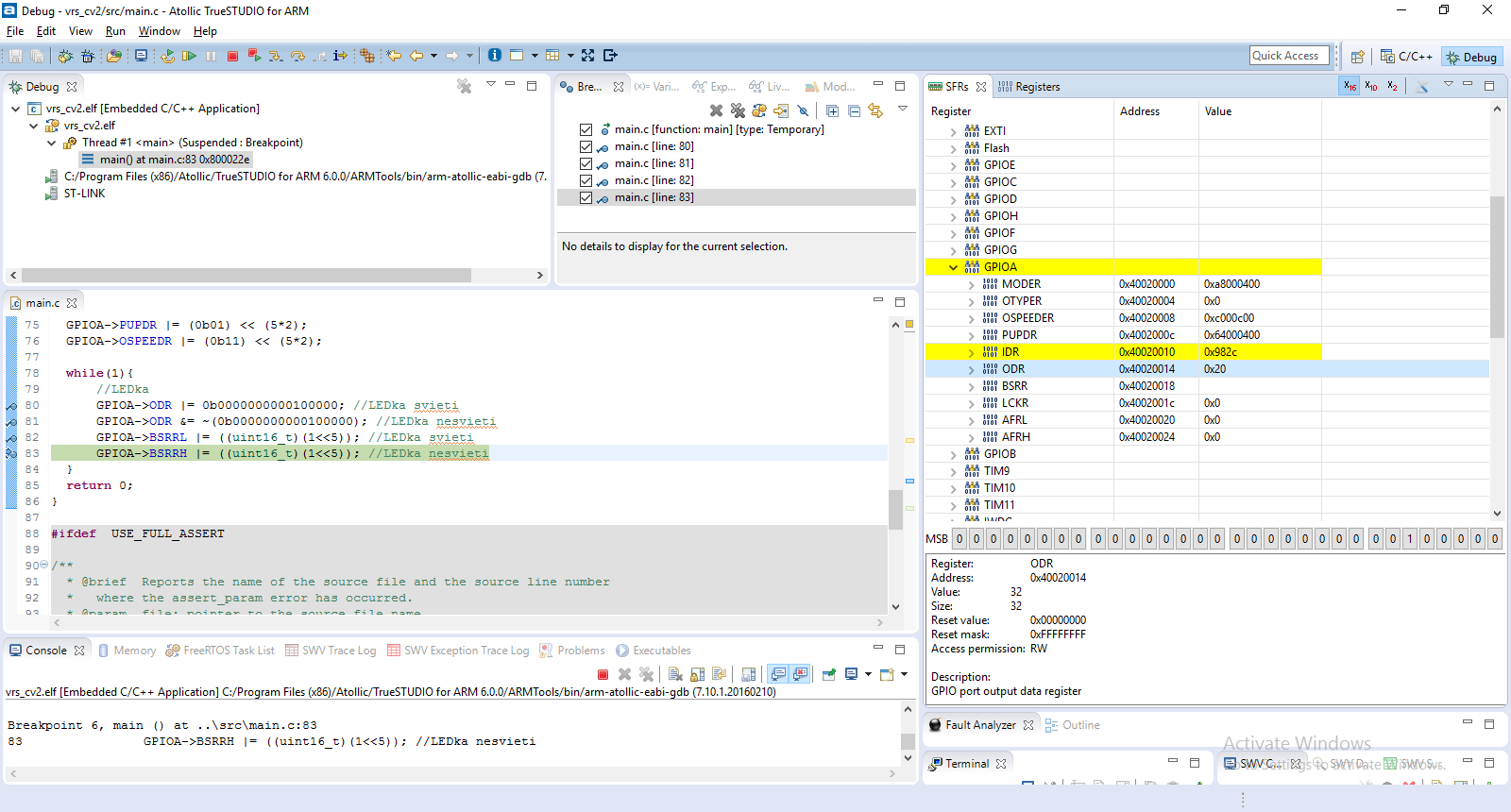
GPIOA->ODR ^= (1<<5);



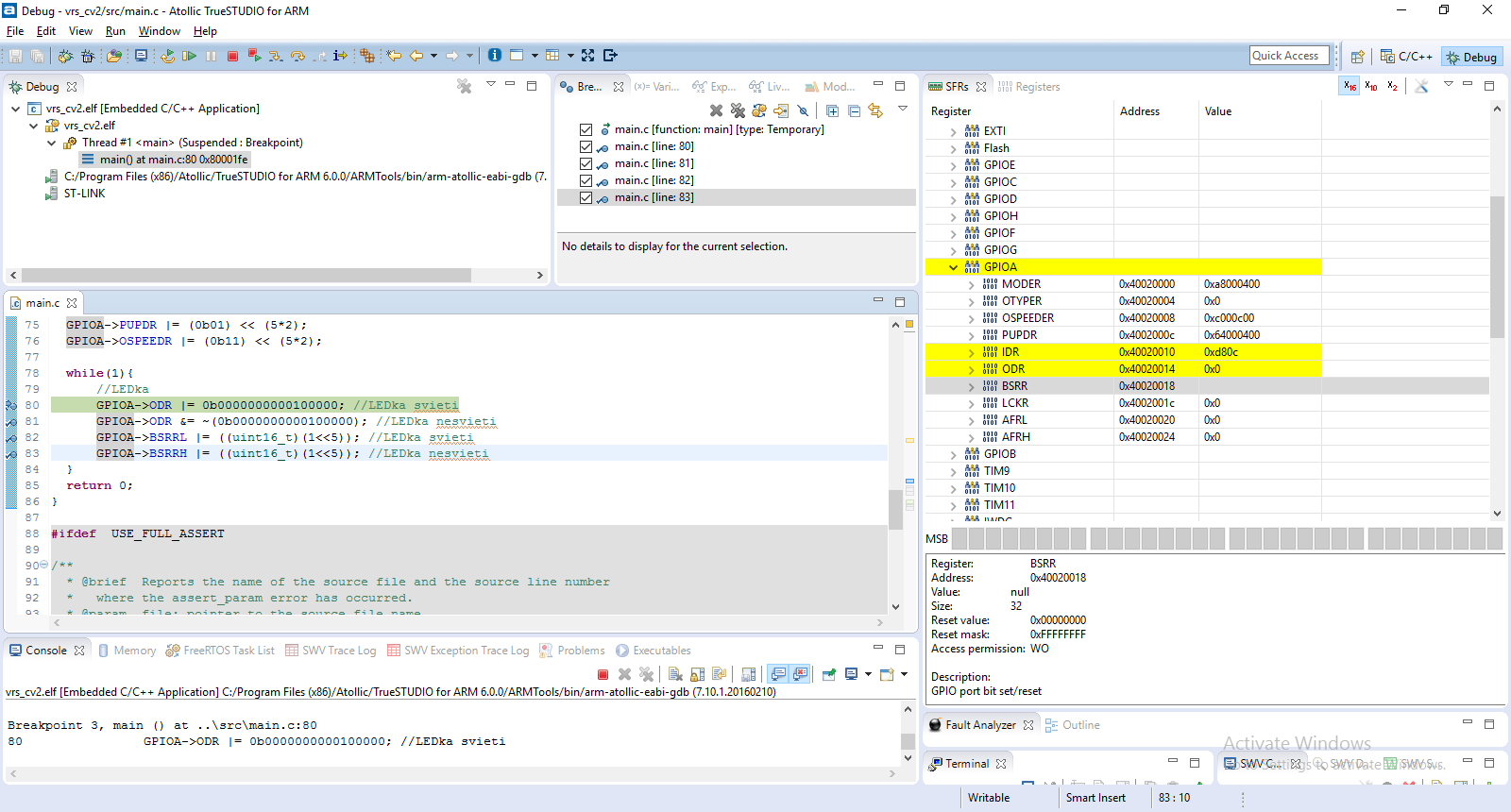
Obr. 1 Zapnutie LED pomocou ODR registra



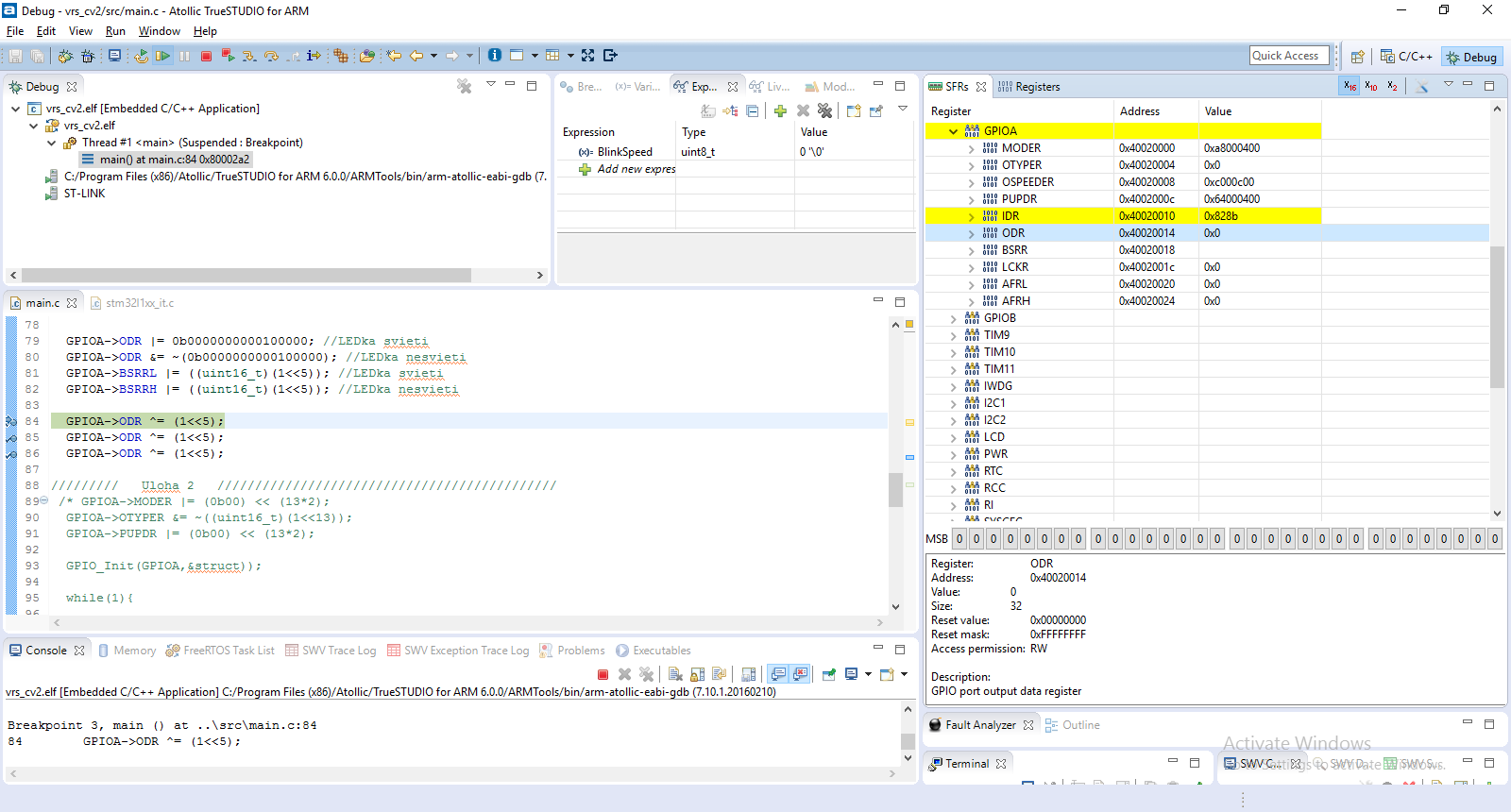
Obr. 2 Vypnutie LED pomocou ODR registra



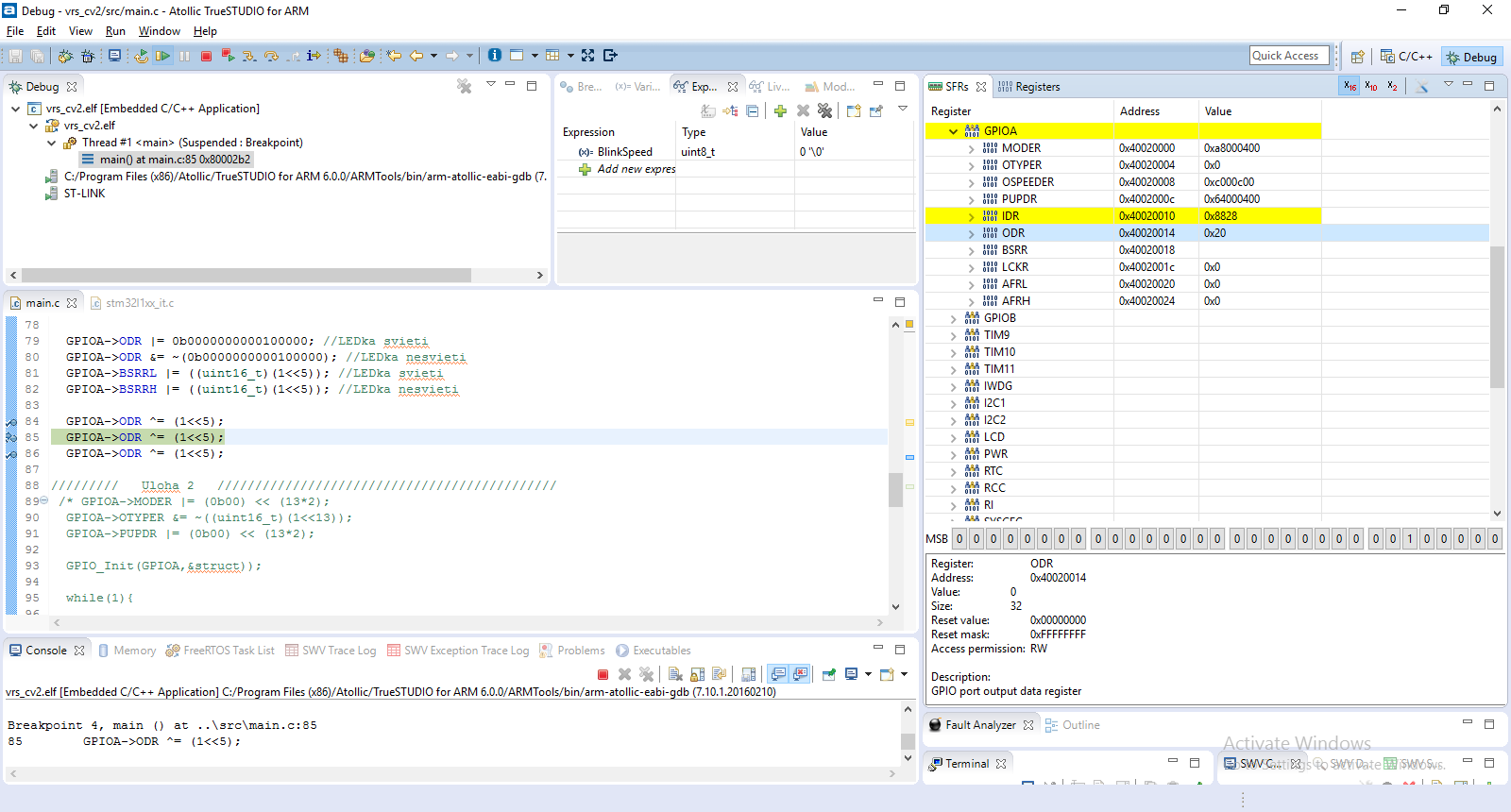
Obr. 3 Zapnutie LED pomocou BSRR registra



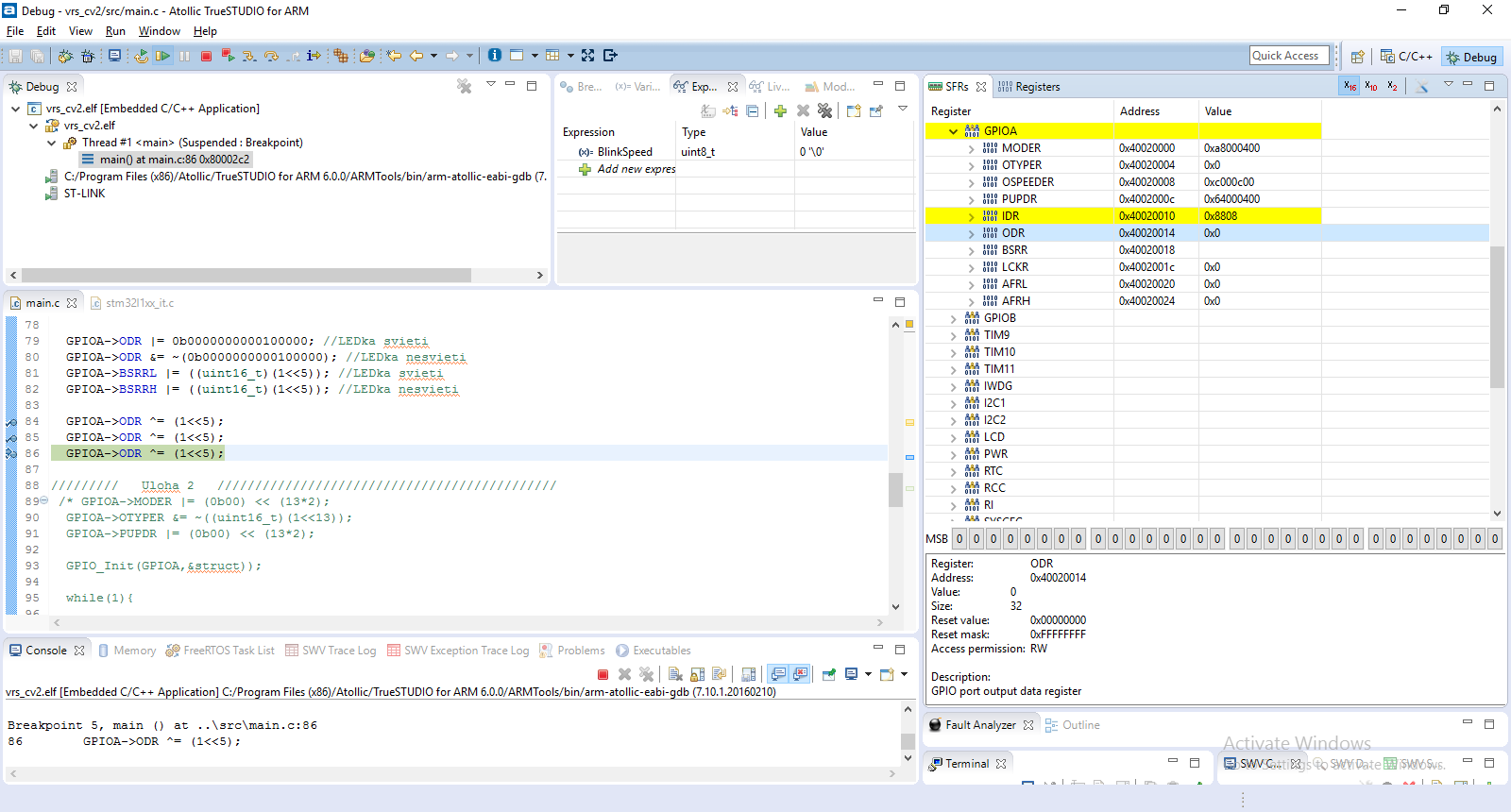
Obr. 4 Vypnutie LED resetovaním BSRR registra



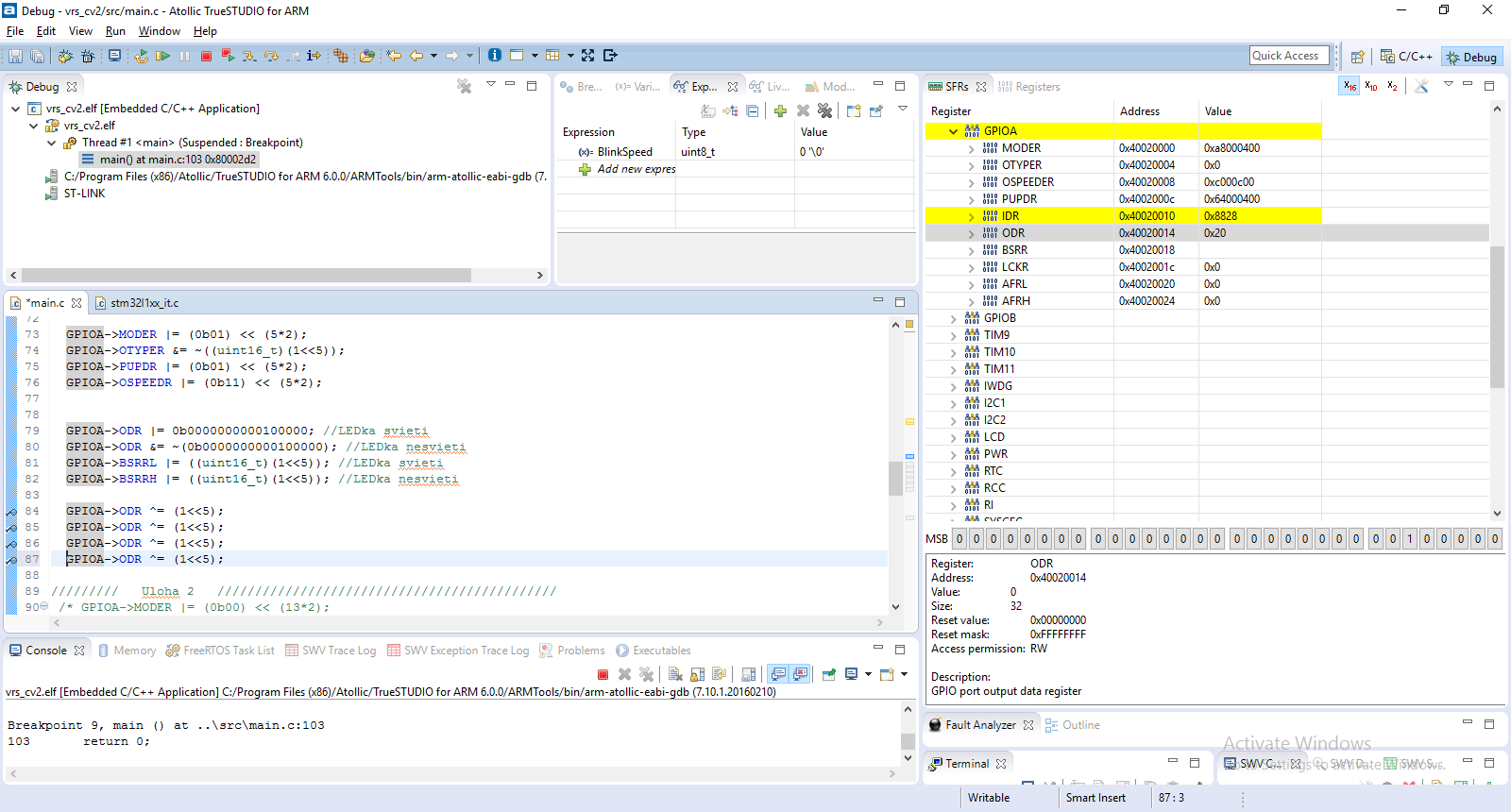
Obr. 5 Prepnutie stavu LED - LED svieti



Obr. 6 Prepnutie stavu LED - LED nesvieti



Obr. 7 Prepnutie stavu LED - LED svieti



Obr. 8 Prepnutie stavu LED - LED nesvieti

# **Úloha 1b**

Toto je **Úloha 1** s použitím knižnice STDPeriphLib.

Definovanie štruktúry:

GPIO\_InitTypeDef struktura;

Naplnenie štruktúry podľa zadania:

struktura.GPIO\_Mode = GPIO\_Mode\_OUT;

struktura.GPIO\_OType = GPIO\_OType\_PP;

struktura.GPIO\_PuPd = GPIO\_PuPd\_UP;

struktura.GPIO\_Pin = GPIO\_Pin\_5;

struktura.GPIO\_Speed = GPIO\_Speed\_40MHz;

Inicializácia štruktúry:

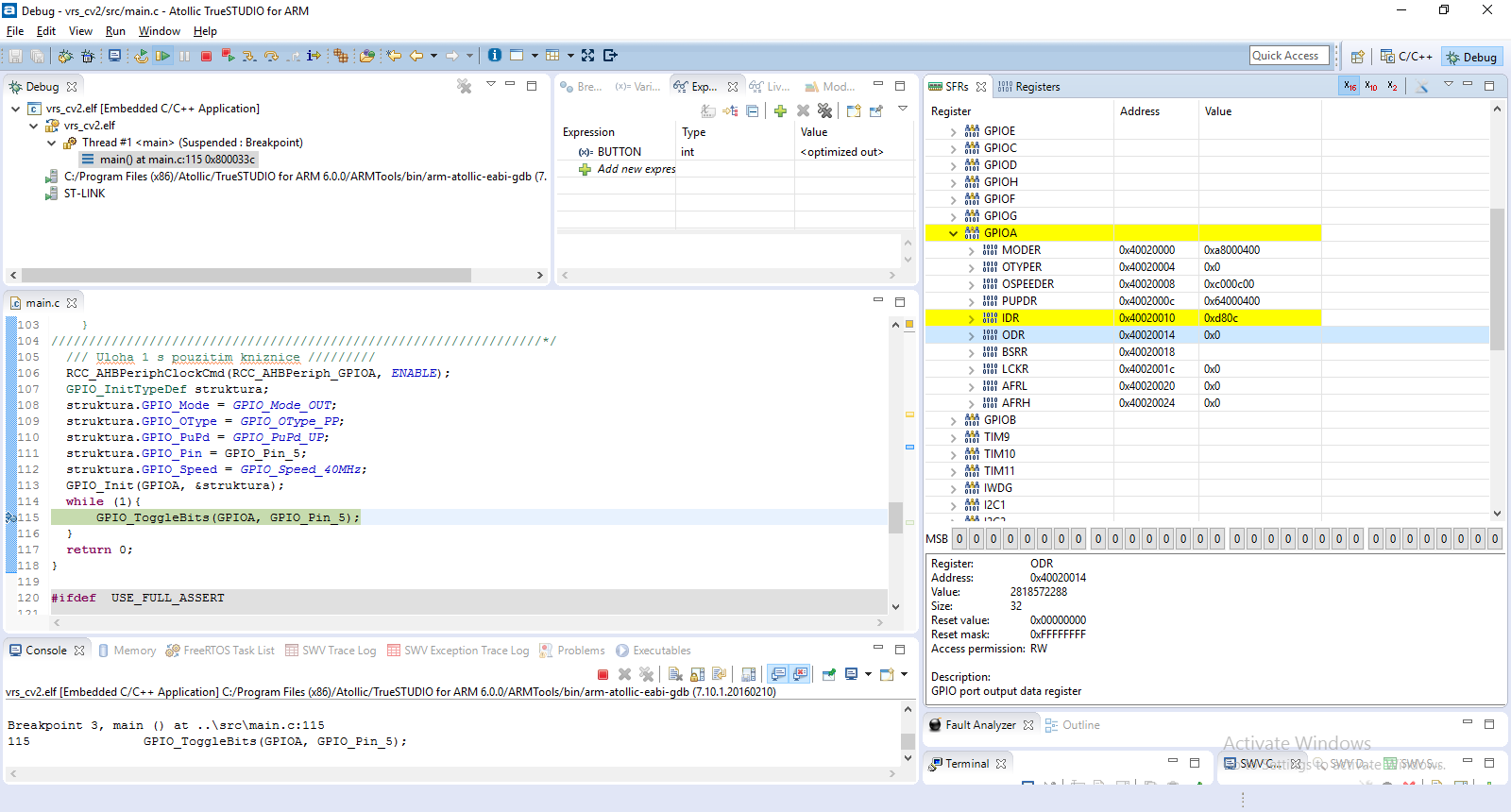
GPIO\_Init(GPIOA, &struktura);

V hlavnej slučke sa prepínajú stavy LED pomocou príkazu *ToggleBits():*

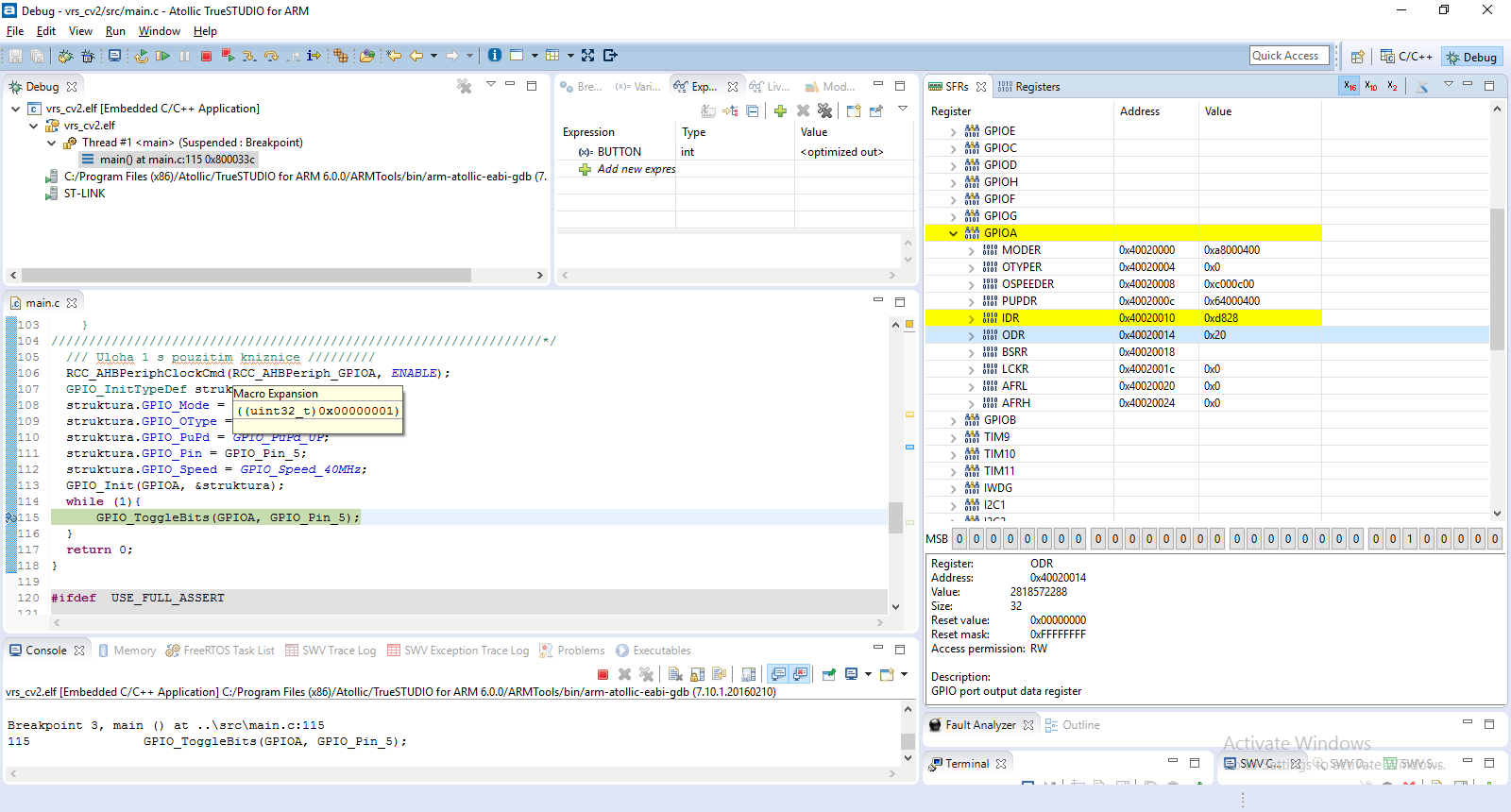
while (1){

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

}



Obr. 9 Prepínanie stavu LED použitím príkazu ToggleBits



Obr. 10 Prepínanie stavu LED použitím príkazu ToggleBits

# **Úloha 2a**

Nastavíme pin 13 periférie GPIOC podľa zadania:

GPIOC->MODER |= (0b00) << (13\*2);

GPIOC->OTYPER &= ~((uint32\_t)(1<<13));

GPIOC->PUPDR |= (0b00) << (13\*2);

V hlavnej slučke porovnávam stav bitu na 13. mieste a podľa toho nastavím hodnotu premennej *BUTTON.* Je tu uplatnená inverzná logika, teda keď je tlačidlo stlačené stav registra je 0 ale premenná je nastavená na 1:

while(1){

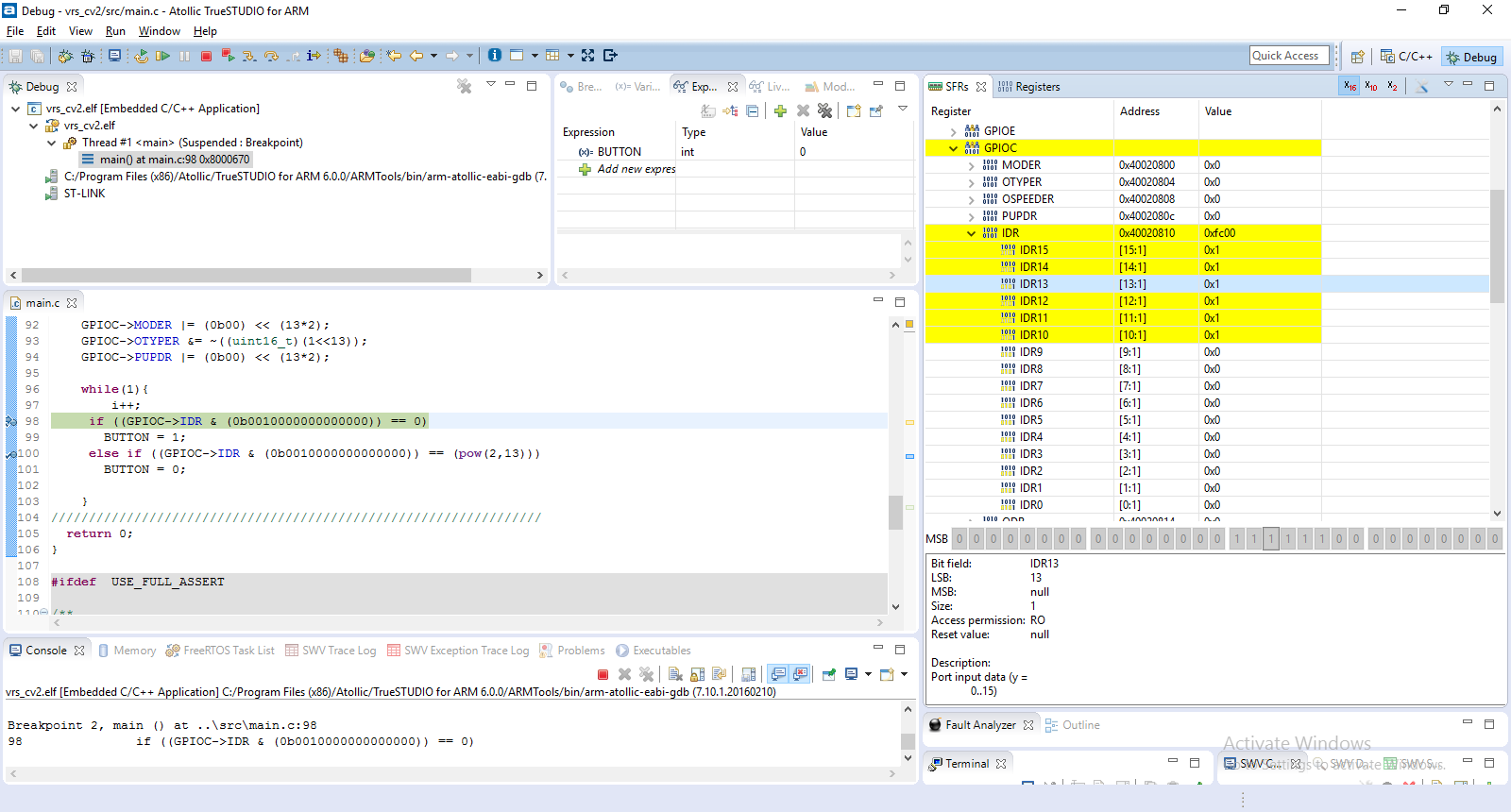
if ((GPIOC->IDR & (0b0010000000000000)) == 0)

BUTTON = 1;

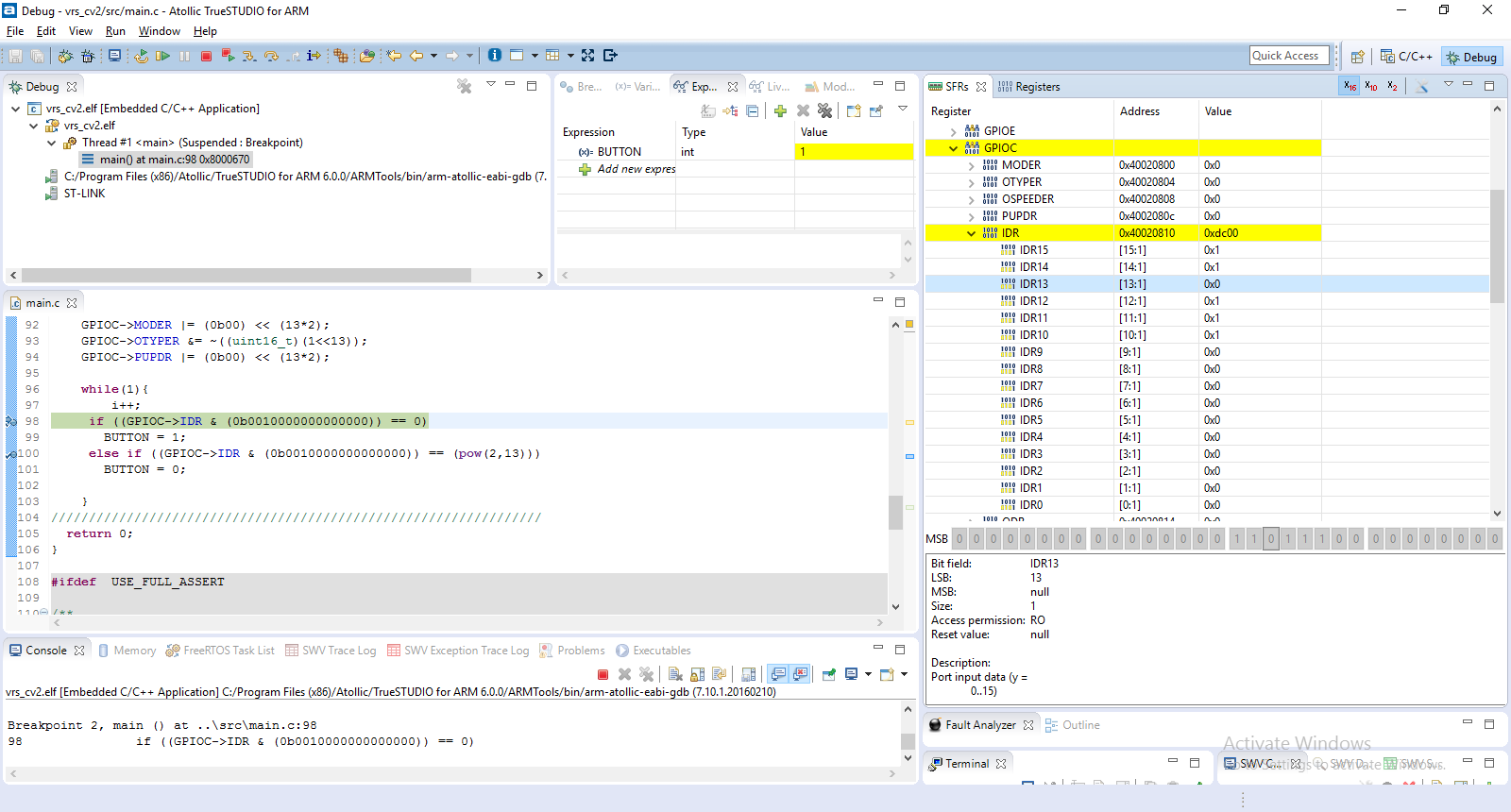
else if ((GPIOC->IDR & (0b0010000000000000)) == (pow(2,13)))

BUTTON = 0;

}



Obr. 11 Tlačidlo je rozopnuté



Obr. 12 Tlačidlo je zopnuté

# **Úloha 2b**

Toto je **Úloha 2** s použitím knižnice STDPeriphLib.

Definovanie štruktúry:

GPIO\_InitTypeDef struktura;

Naplnenie štruktúry podľa zadania:

struktura.GPIO\_Mode = GPIO\_Mode\_IN;

struktura.GPIO\_PuPd = GPIO\_PuPd\_NOPULL;

struktura.GPIO\_OType = GPIO\_OType\_PP;

struktura.GPIO\_Pin = GPIO\_Pin\_13;

Inicializácia štruktúry:

GPIO\_Init(GPIOC, &struktura);

V hlavnej slučke je použitá funkcia *GPIO\_ReadInputDataBit()* na detekciu stlačenia tlačidla:

while(1){

if ((GPIO\_ReadInputDataBit(GPIOC, GPIO\_Pin\_13)) == 0){

BUTTON = 1;

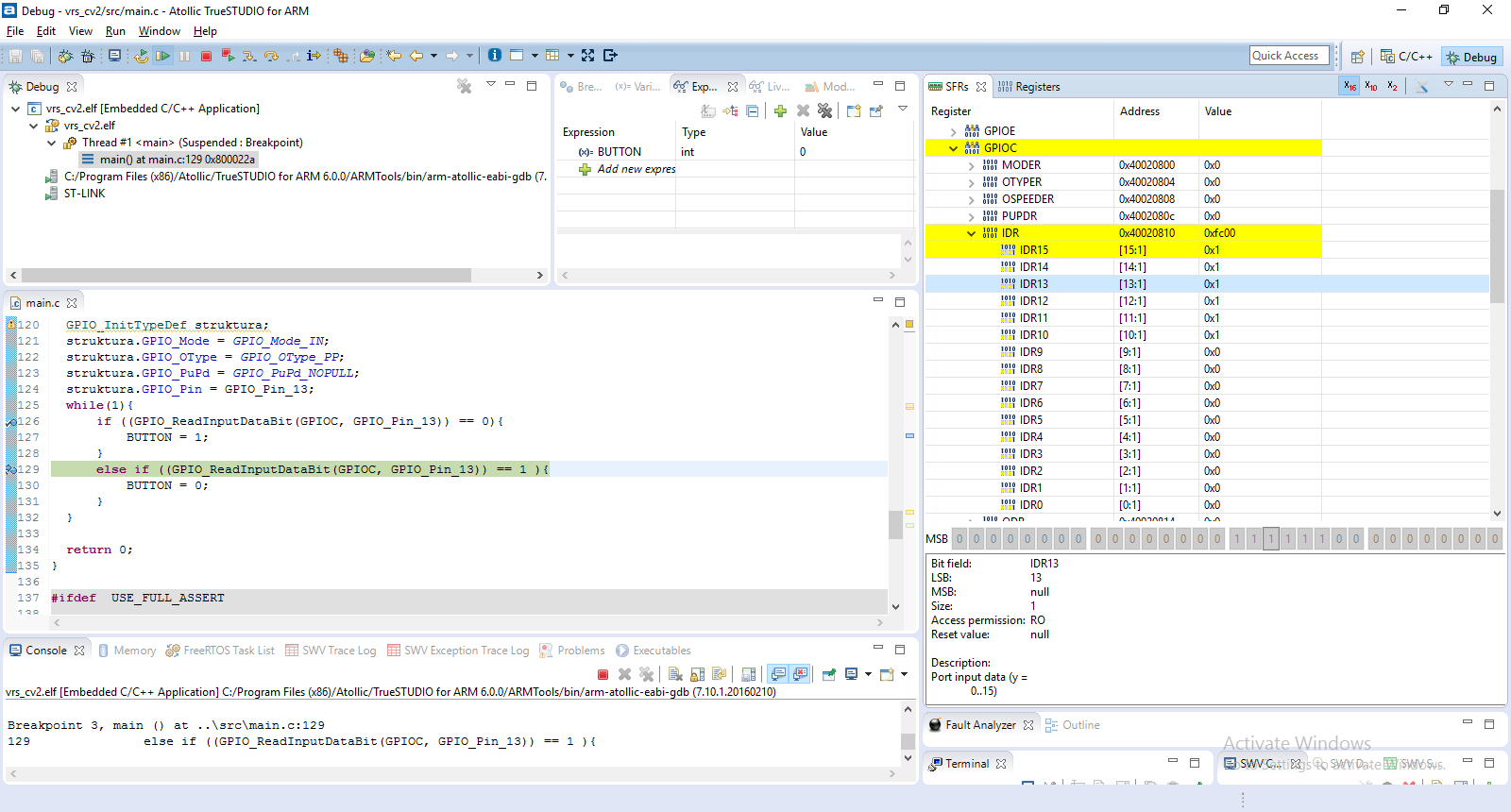
}

else if ((GPIO\_ReadInputDataBit(GPIOC, GPIO\_Pin\_13)) == 1 ){

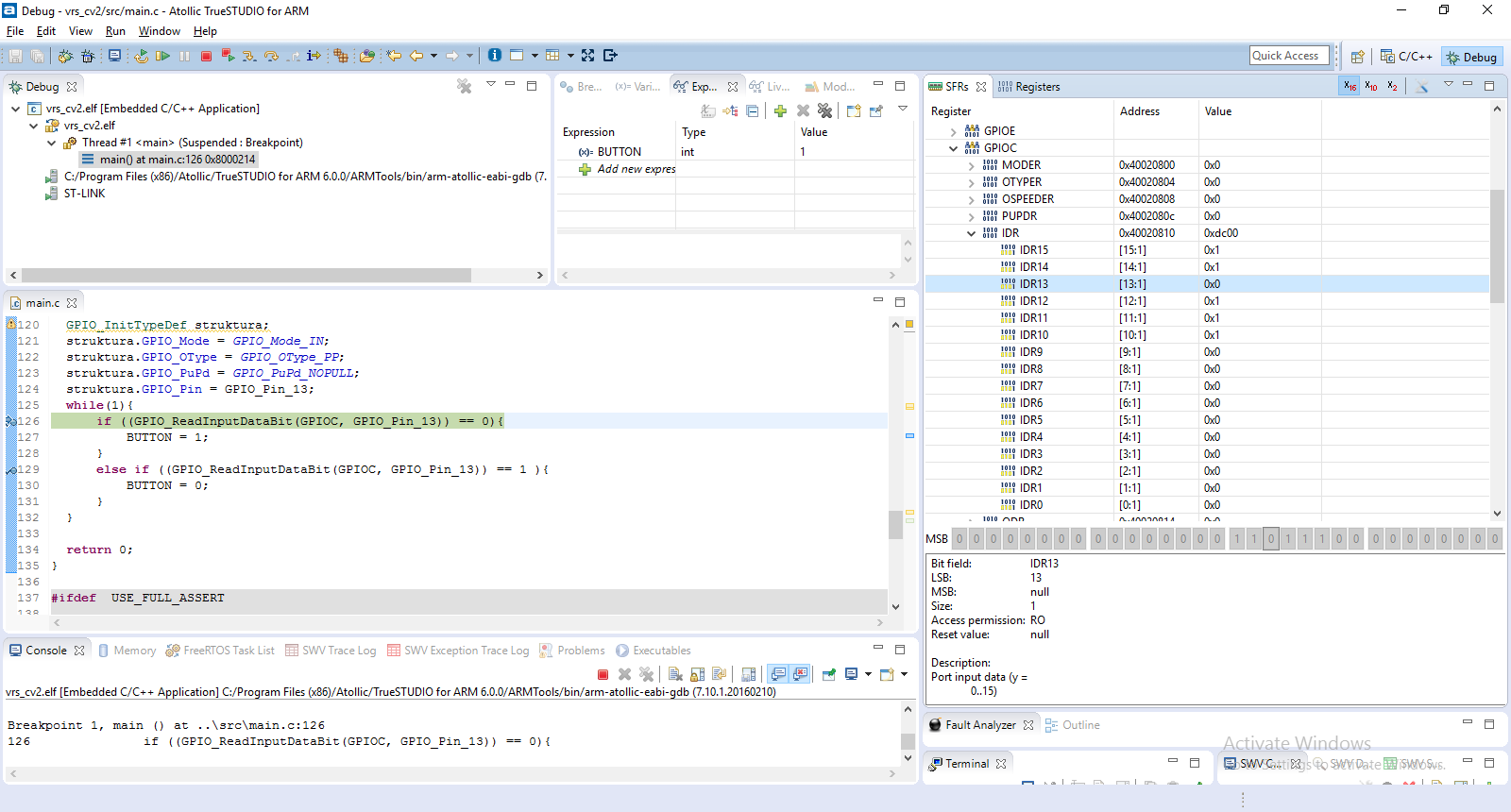
BUTTON = 0;

}

}



Obr. 13 Tlačidlo rozopnuté - hodnota v registri je 1



Obr. 14 Tlačidlo zopnuté - hodnota v registri je 0

# **Úloha 3.1a**

Pred každou úlohou je v kóde ešte nastavenie periférie GPIOA, resp. GPIOC alebo definovanie a inicializácia štruktúry podľa zadania, podobne ako v predchádzajúcich úlohách, preto ju tu už nebudem znova uvádzať ale uvediem rovno hlavnú slučku.

V hlavnej slučke je použitý operátor XOR na prepnutie stavu LEDky. Celý *while* cyklus je spomalený dvoma, resp. štyrmi *for* cyklami:

int i;

int j;

while(1){

i = 1;

j = 1;

GPIOA->ODR ^= (1<<5);

for ( i=1; i<= 500; i++){

for ( j=1; j<= 500; j++){

}

}

GPIOA->ODR ^= (1<<5);

i = 1;

j = 1;

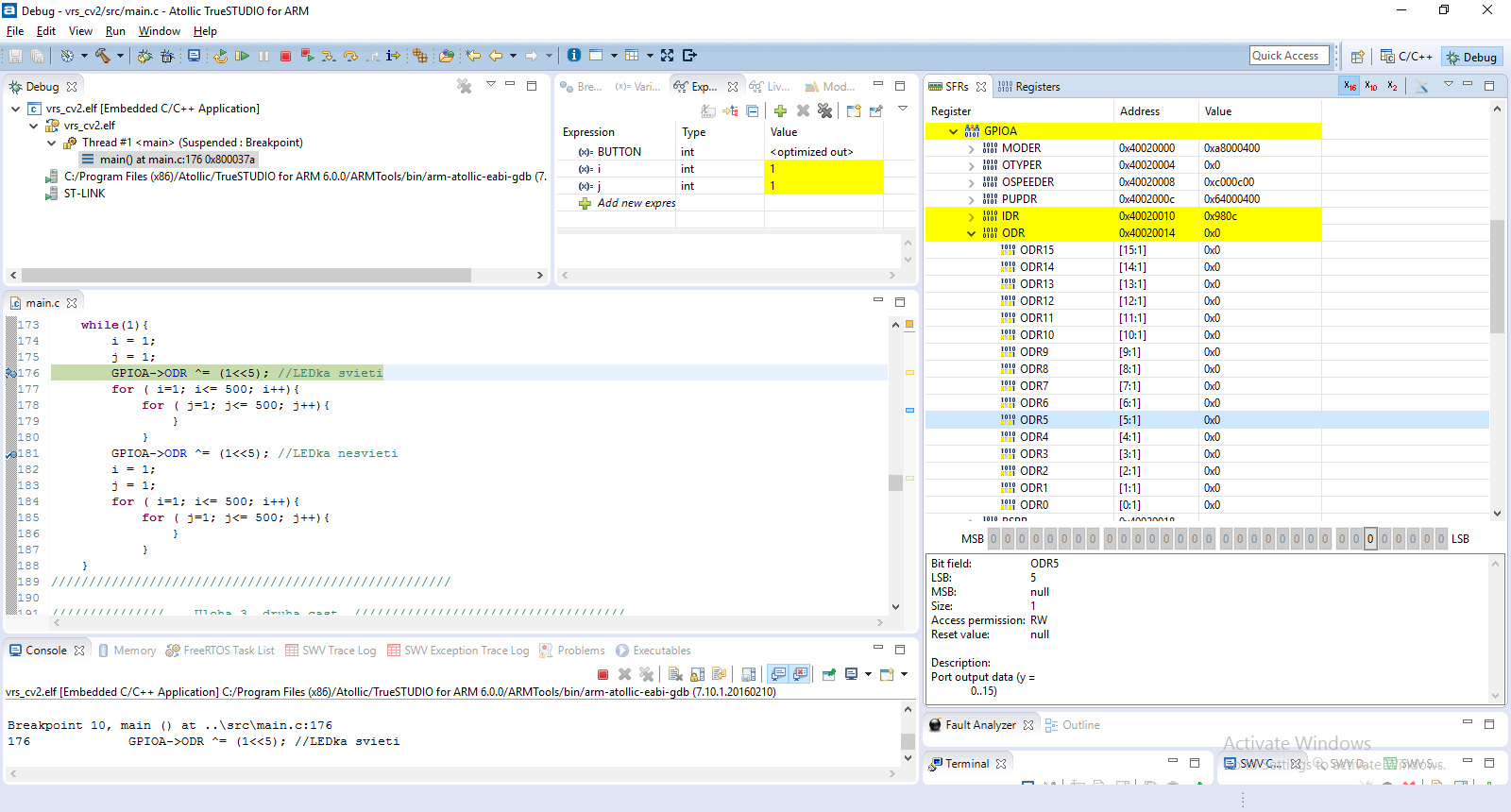
for ( i=1; i<= 500; i++){

for ( j=1; j<= 500; j++){

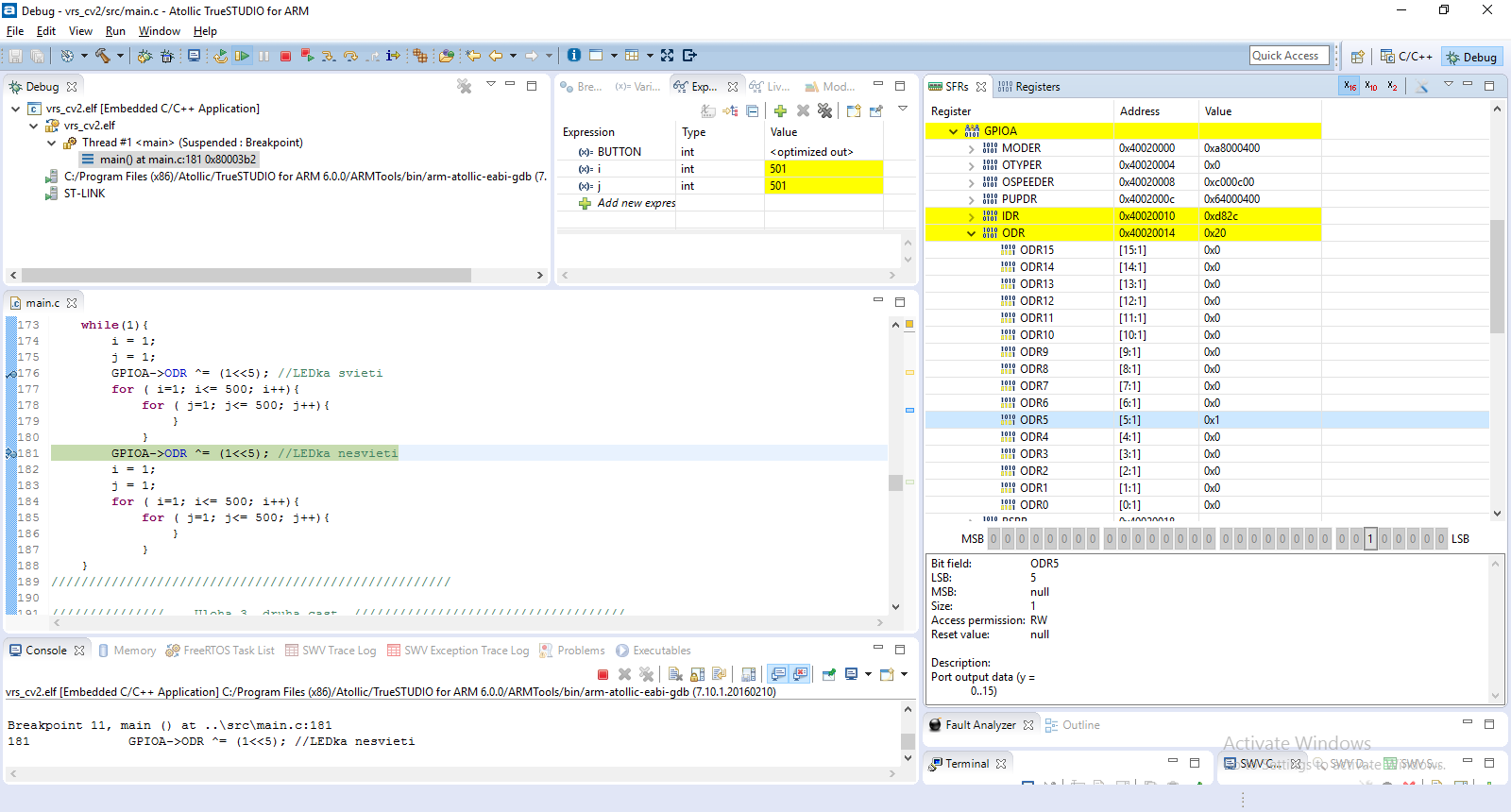
}

}

}



Obr. 15 LEDka nesvieti



Obr. 16 LEDka svieti

# **Úloha 3.1b**

V tejto úlohe je použitá knižnica na prepínanie stavu LEDky a opäť funkcia *GPIO\_ToggleBits().* Na spomalenie je použitý *for* cyklus:

while(1){

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

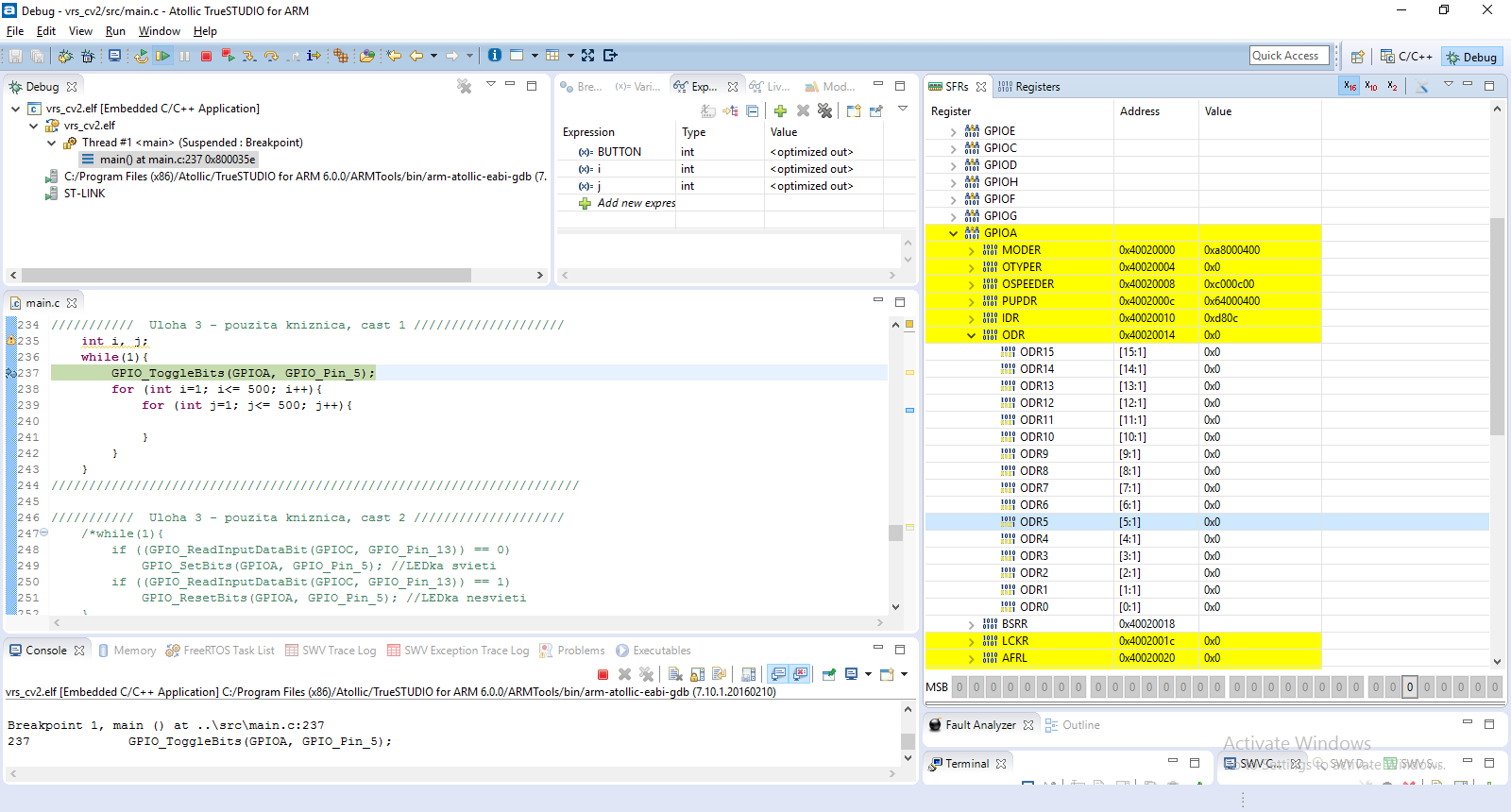
for (int i=1; i<= 500; i++){

for (int j=1; j<= 500; j++){

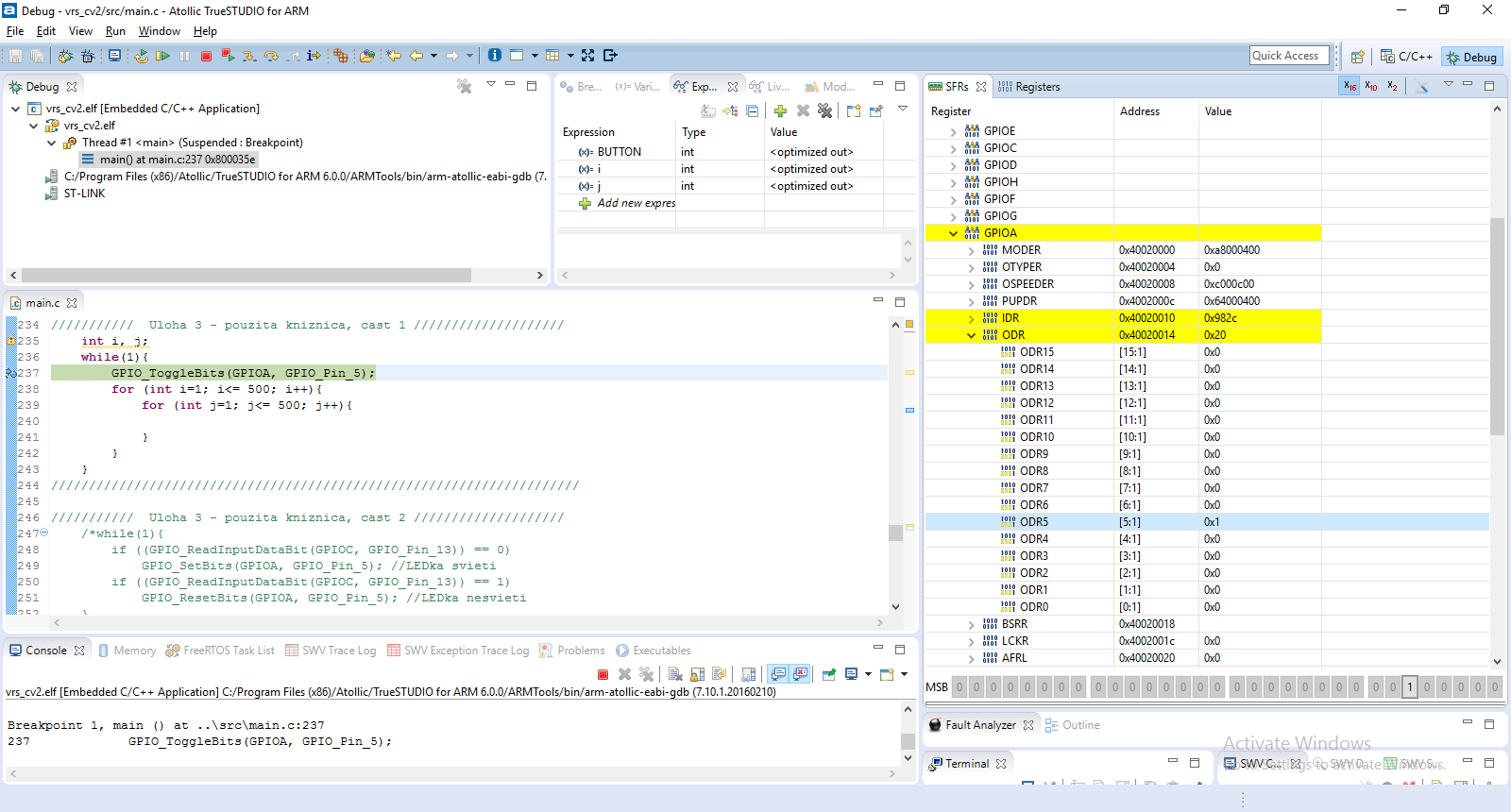
}

}

}



Obr. 17 LEDka nesvieti



Obr. 18 LEDka svieti

# **Úloha 3.2a**

V hlavnej slučke pristupujem do IDR registra, kde sledujem stav tlačidla. Ak je tlačidlo stlačené, teda hodnota v IDR registri je 0, tak pomocou registra ODR zapnem LEDku.

while(1){

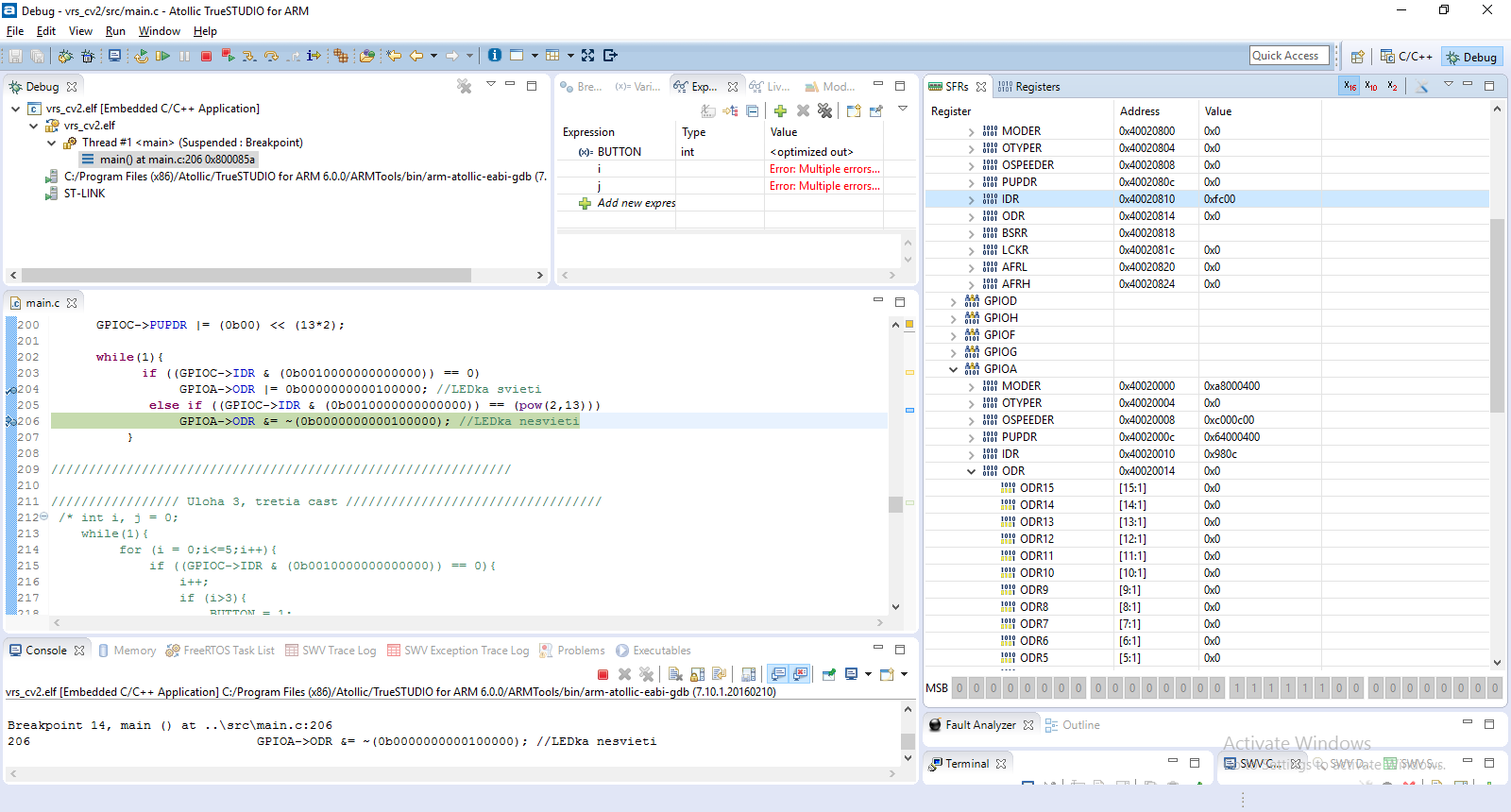
if ((GPIOC->IDR & (0b0010000000000000)) == 0)

GPIOA->ODR |= 0b0000000000100000;

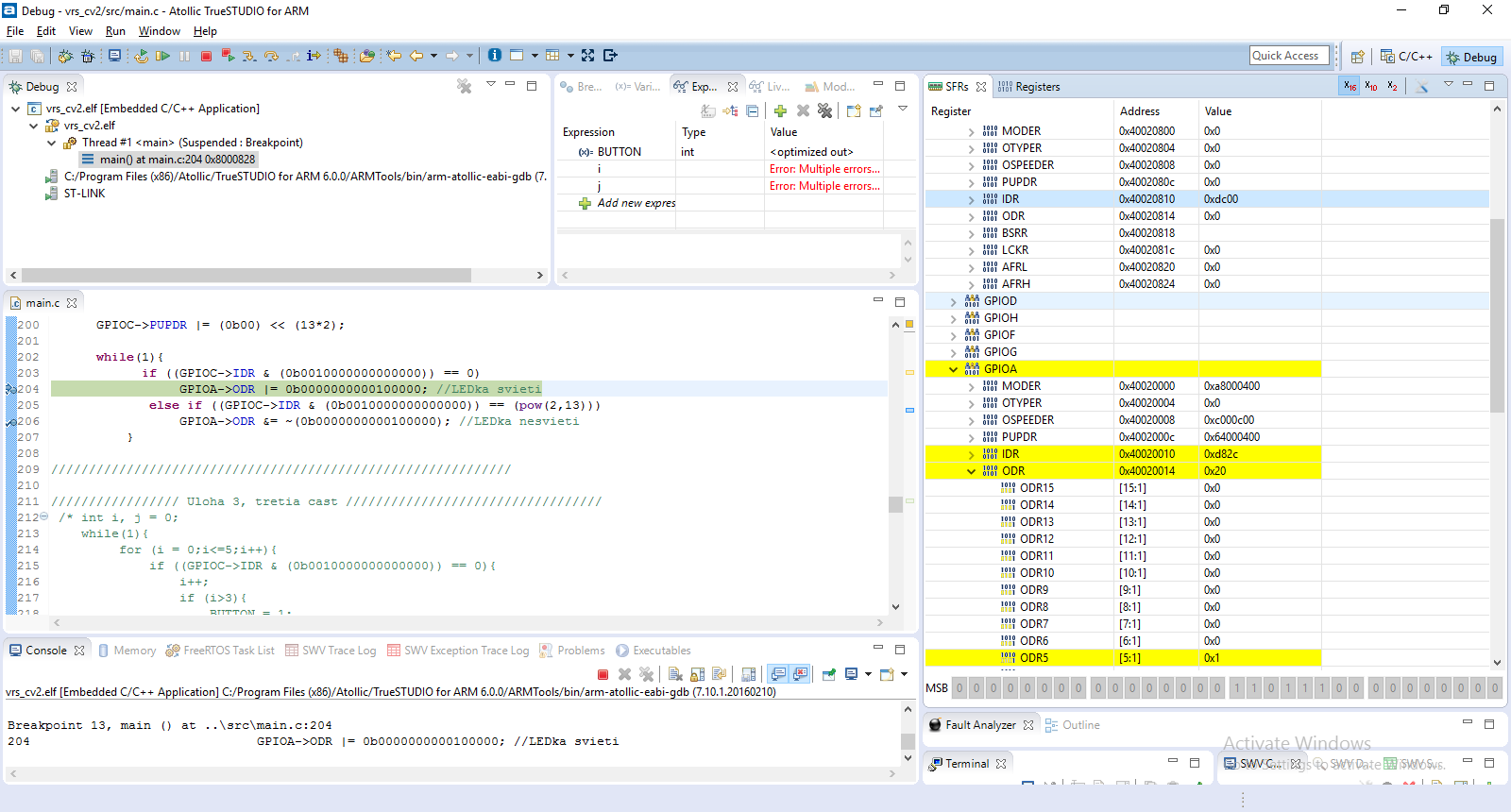
else if ((GPIOC->IDR & (0b0010000000000000)) == (pow(2,13)))

GPIOA->ODR &= ~(0b0000000000100000);

}



Obr. 19 Tlačidlo vypnuté



Obr. 20 Tlačidlo zapnuté - LEDka svieti

# **Úloha 3.2b**

V hlavnej slučke kontrolujem stav pinu 13 na periférii GPIOC pomocou funkcie *GPIO\_ReadInputDataBit()* a pomocou funkcie *GPIO\_SetBits* a *GPIO\_ResetBits()* prepínam stav LEDky. Opäť tu je inverzná logika, teda LEDka svieti keď je hodnota v registri 0:

while(1){

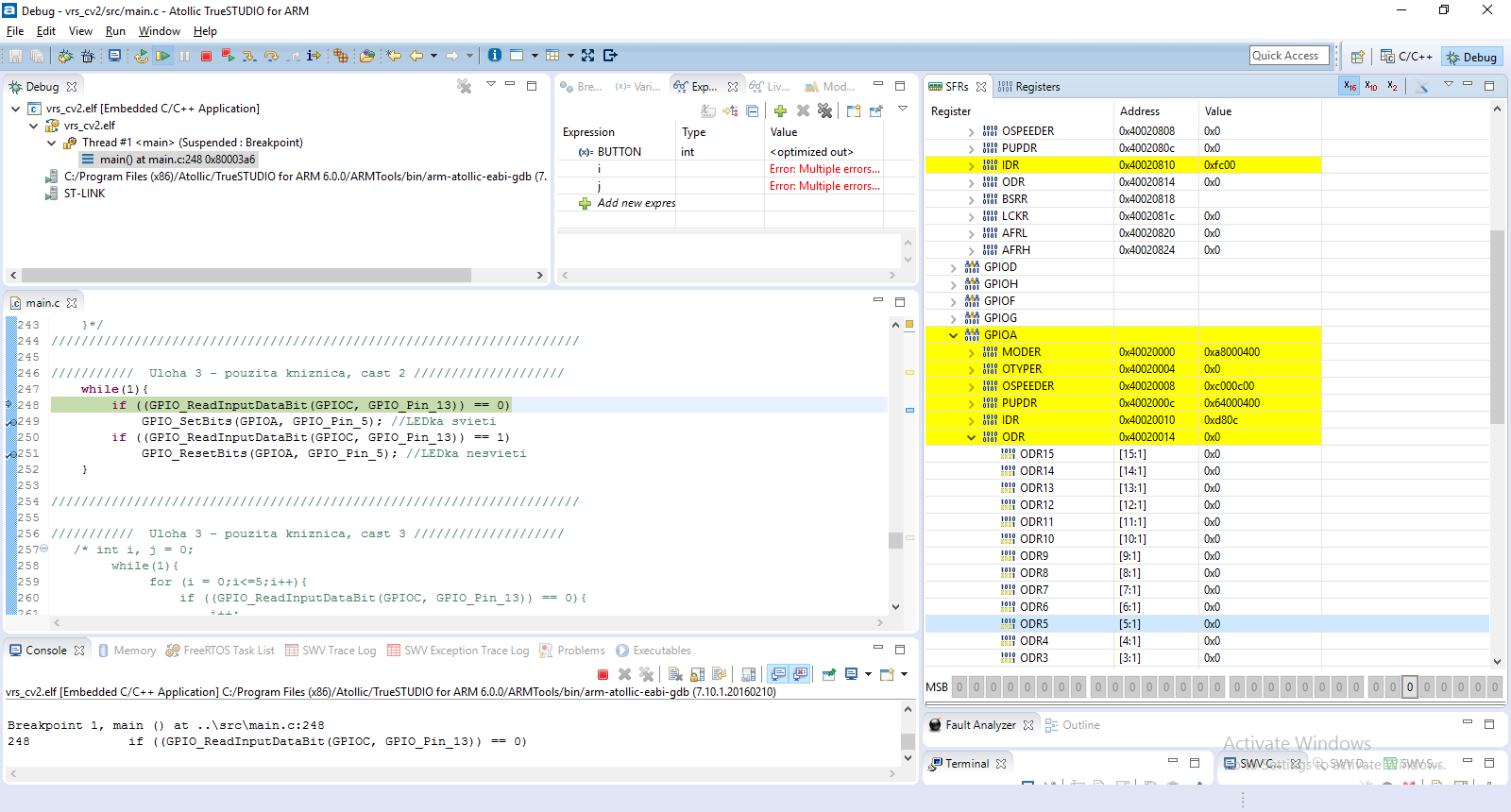
if ((GPIO\_ReadInputDataBit(GPIOC, GPIO\_Pin\_13)) == 0)

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

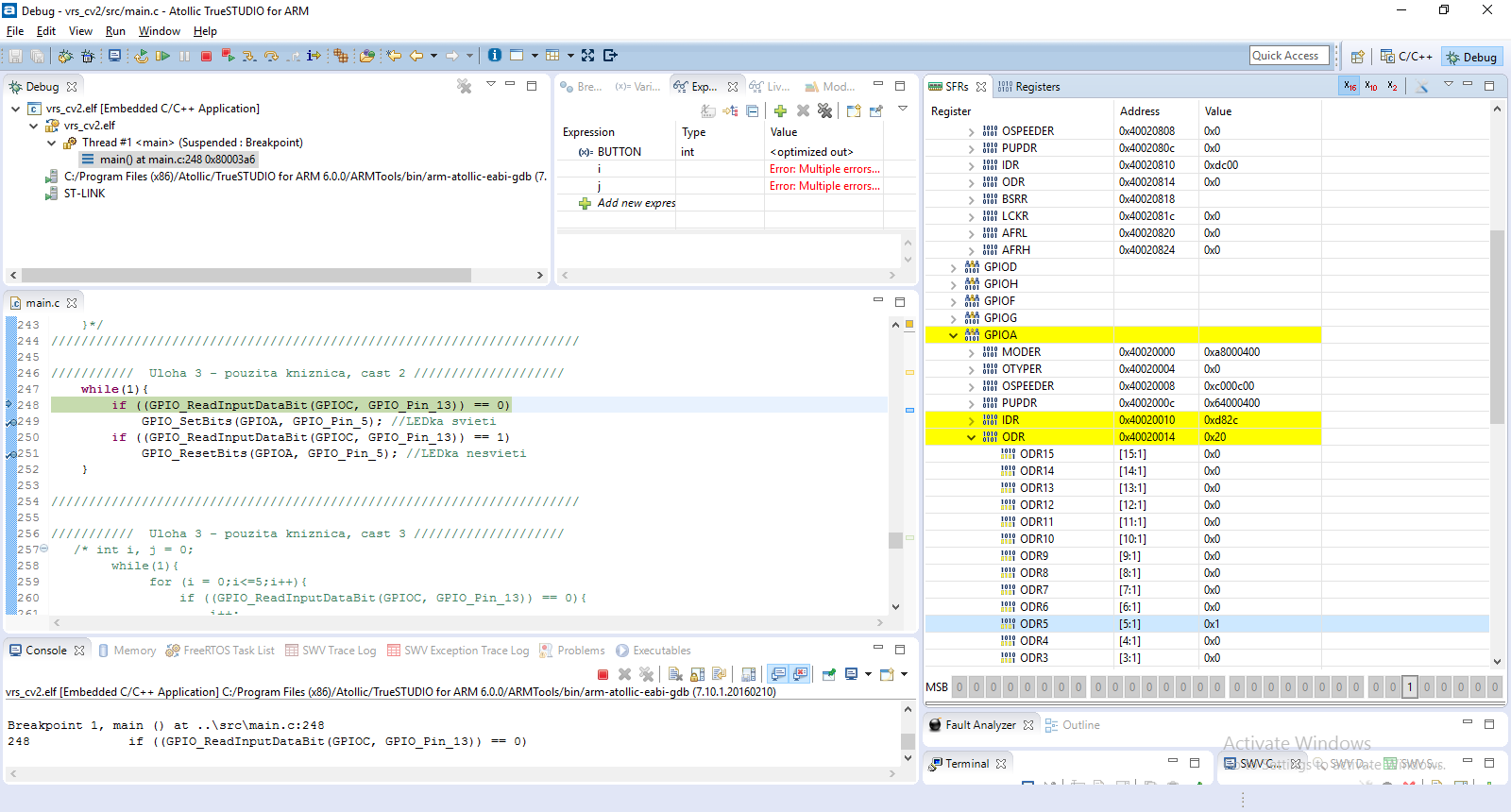
if ((GPIO\_ReadInputDataBit(GPIOC, GPIO\_Pin\_13)) == 1)

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

}



Obr. 21 Tlačidlo vypnuté - LEDka nesvieti



Obr. 22 Tlačidlo zapnuté - LEDka svieti

# **Úloha 3.3a**

V hlavnej slučke sledujem IDR register a vyhodnocujem či bolo stlačené tlačidlo. Toto vyhodnocovanie prebieha vo *for* cykle kvôli prechodovým javom pri stlačení tlačidla. Potom pomocou operátora XOR prepínam stav LEDky. Ak raz bolo stlačené tlačidlo, LEDka zostane svietiť bez toho, aby tlačidlo muselo byť stále zopnuté. Keď sa opäť stlačí tlačidlo, LEDka sa vypne:

int i, j = 0;

while(1){

for (i = 0;i<=5;i++){

if ((GPIOC->IDR & (0b0010000000000000)) == 0){

i++;

if (i>3){

BUTTON = 1;

}

}

}

if (BUTTON == 1){

if ((GPIOC->IDR & (0b0010000000000000)) == (pow(2,13))){

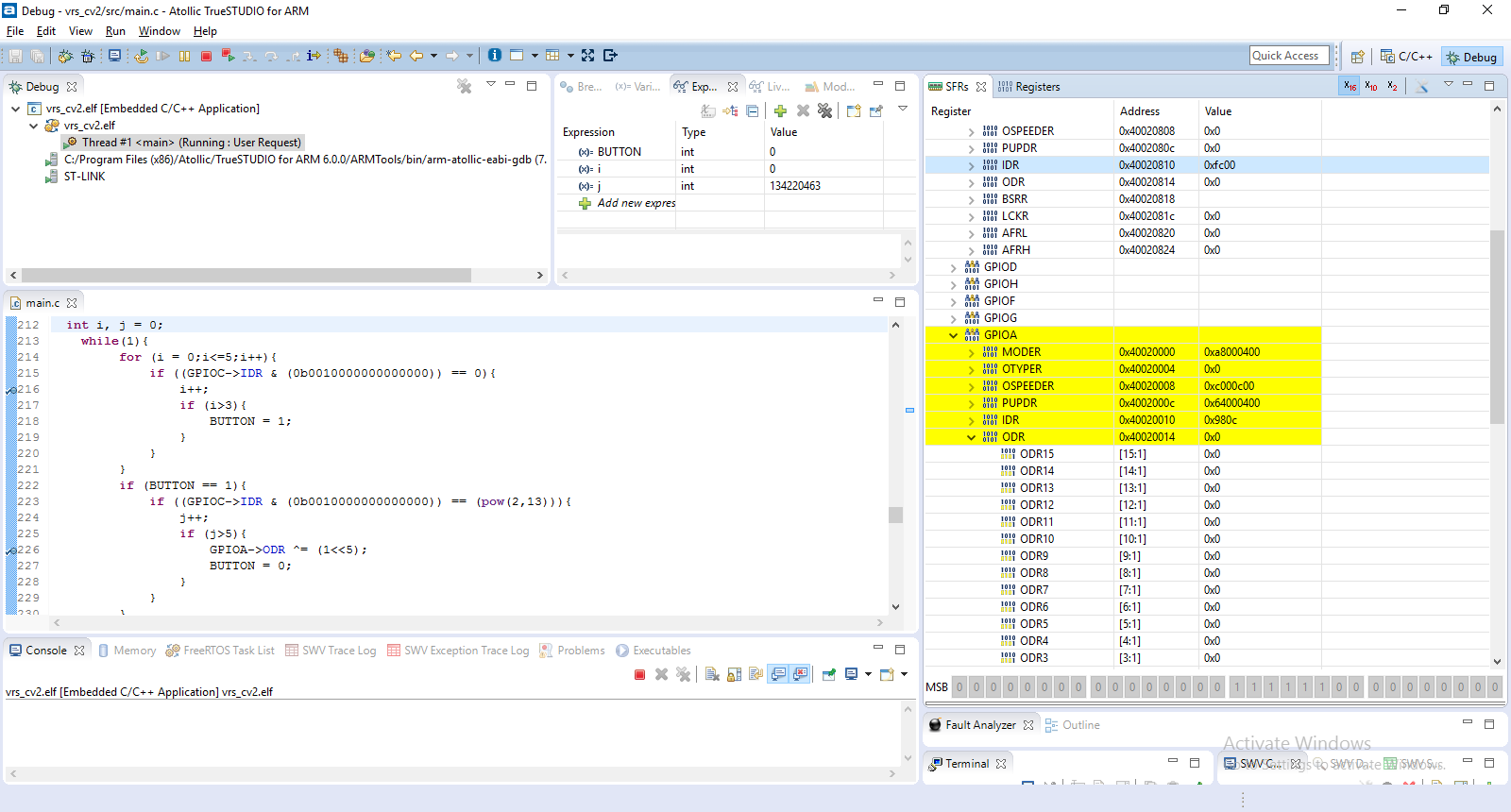
j++;

if (j>5){

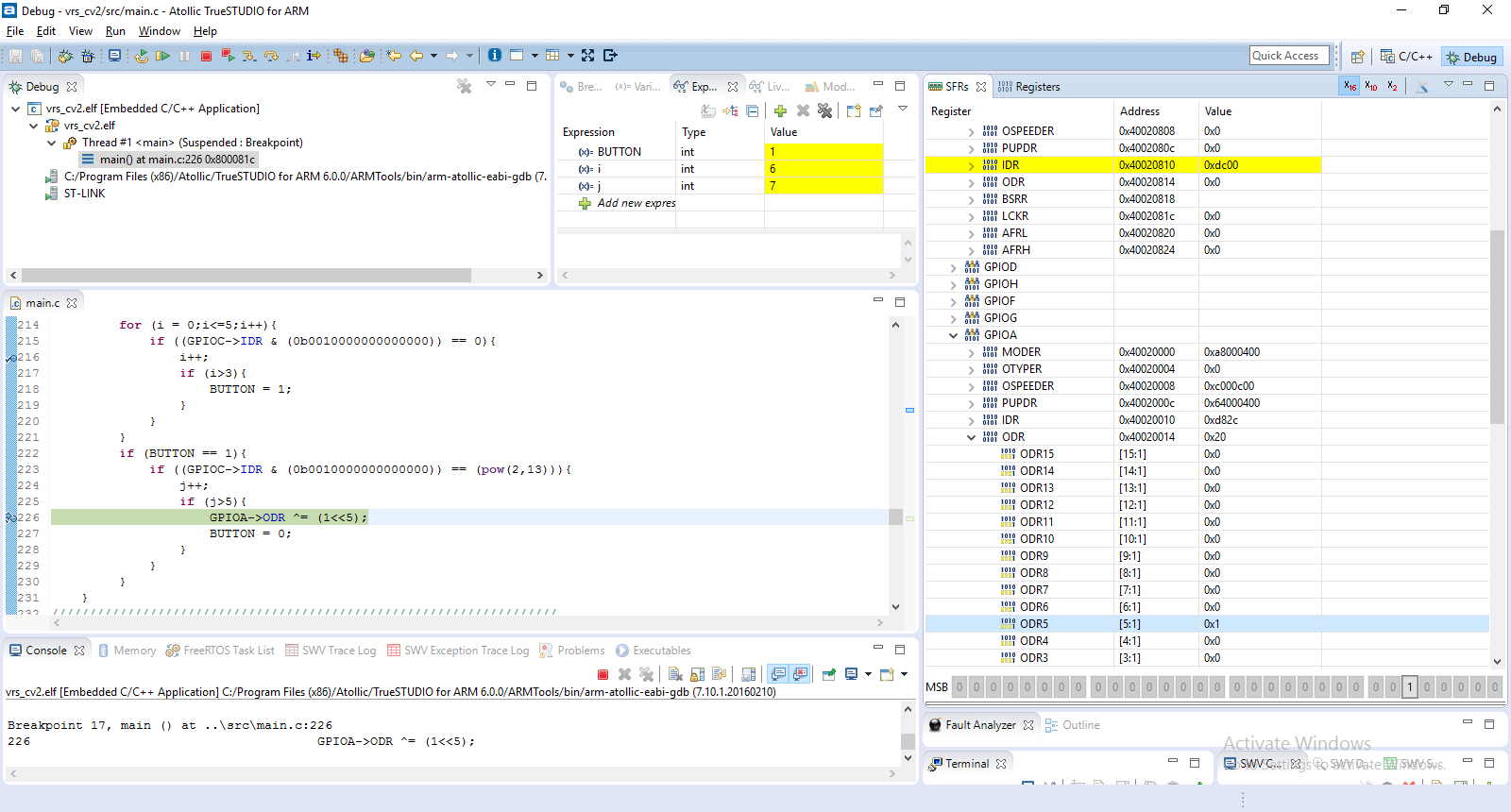
GPIOA->ODR ^= (1<<5);

BUTTON = 0;

}}}}



Obr. 23 Tlačidlo vypnuté - LEDka nesvieti



Obr. 24 Tlačidlo bolo stlačené - LEDka zostala svietiť

# **Úloha 3.3b**

Princíp tejto úlohy rovnaký ako v predchádzajúcej úlohe s tým rozdielom, že je tu použitá funkcia *GPIO\_ReadInputDataBit()* na čítanie stavu z pinu 13 periférie GPIOC a funkcia *GPIO\_ToggleBits()* na nastavenie hodnoty na pin 5 periférie GPIOA:

int i, j = 0;

while(1){

for (i = 0;i<=5;i++){

if ((GPIO\_ReadInputDataBit(GPIOC, GPIO\_Pin\_13)) == 0){

i++;

if (i>3){

BUTTON = 1;

}

}

}

if (BUTTON == 1){

if ((GPIO\_ReadInputDataBit(GPIOC, GPIO\_Pin\_13)) == 1){

j++;

if (j>5){

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

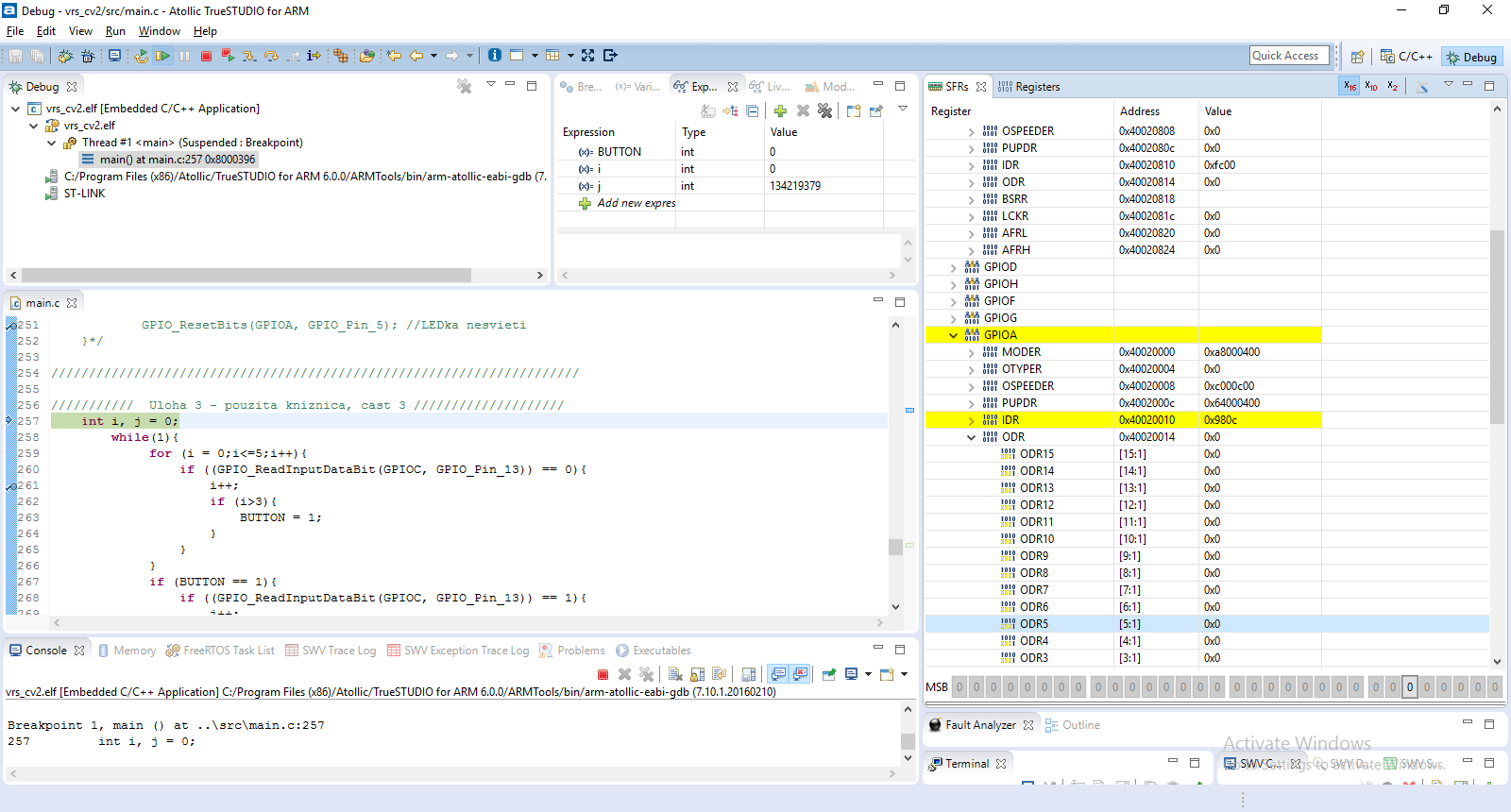
BUTTON = 0;

}

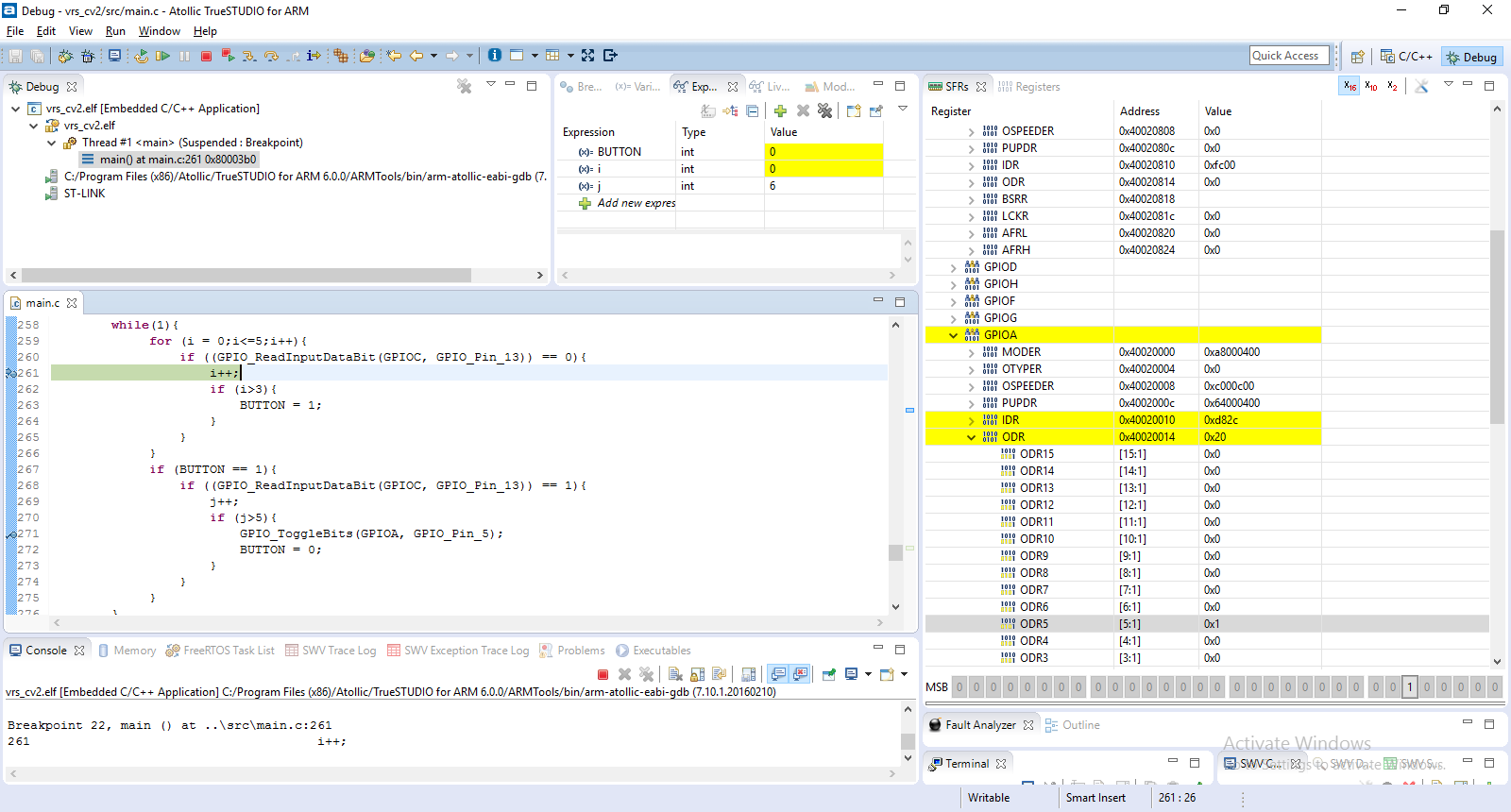
}

}

}



Obr. 25 Tlačidlo vypnuté - LEDka nesvieti



Obr. 26 Tlačidlo bolo stlačené - LEDka zostala svietiť

# **Link na GitHub**

<https://github.com/tomaskolek/vrs_cv2>