BEÁGYAZOTT RENDSZEREK (C)

4. forduló

BOSCH
Invented for life

A kategória támogatója: Robert Bosch Kft.

Ismertető a feladathoz

Fontos információk

A forduló után a megoldások publikálásával együtt iránymutatásként elérhetőek lesznek a **helyezéssel kapcsolatos információk**, látni fogod, hogy a kategóriában a játékosok 20%, 40% vagy 60%-a közé tartozol-e épp.

Felhívjuk figyelmedet, hogy a következő, **5. fordulótól az egyes kategóriák csak a kijelölt napokon lesznek megoldhatóak 7-22 óra között**, érdemes letöltened a naptárat a Kategóriáim menüpontban!

Negyedik forduló

RENDELKEZÉSRE ÁLLÓ IDŐ:

60:00

akarja felrobbantani a kazánt). Egy új TMP126-Q1 hőmérséklet szenzort választott, mely CRC védelemmel képes az adat integritását biztosítani.

Segíts Ájóti Jánosnak a CRC algoritmusok megértésében!

A megoldásban a következő adatlapok lesznek segítségetekre:

TMP126-Q1 adatlap:

https://www.ti.com/lit/ds/symlink/tmp126-q1.pdf

STM32F091RC adatlap:

https://www.st.com/resource/en/datasheet/stm32f091rc.pdf

NUCLEO-F091RC adatlap:

https://www.st.com/resource/en/data_brief/nucleo-f091rc.pdf

HAL API dokumentáció:

https://www.st.com/resource/en/user_manual/dm00122015-description-of-stm32f0-hal-and-lowlayer-drivers-stmicroelectronics.pdf

Kontroller reference manual:

https://www.st.com/resource/en/reference_manual/rm0091-stm32f0x1stm32f0x2stm32f0x8-advanced-armbased-32bit-mcus-stmicroelectronics.pdf

Felhasznált idő: 00:54/60:00 **Elért pontszám:** 0/30

1. feladat 0/5 pont

Melyik kód valósítja meg a szenzornak megfelelő CRC kiszámolását?

Válasz

```
uint16_t crcCalculator(char *ptr, size_t count){
    uint16_t crc = 0xFFFFU;
    size_t j;
    for (j = 0; j < count; ++j)
    {
        size_t i = 8;
        crc ^= (uint16_t)*ptr++ << 8;
        do
        {
            if (crc & 0x8000)
                crc = (crc << 1) ^ 0x1021;
            else
                crc = crc << 1;
        } while (0 < --i);
    }
    return (crc);
}
```

```
uint8_t crcCalculator(char *ptr, size_t count){
    uint8_t crc = 0xFFFFU;
    size_t j;
    for (j = 0; j < count; ++j)
    {
        size_t i = 8;
        crc ^= (uint8_t)*ptr++ << 8;
        do
        {
            if (crc & 0x8000)
                crc = (crc << 1) ^ 0x1021;
            else
                crc = crc << 1;
        } while (0 < --i);
    }
    return (crc);
}
```

```
uint16_t crcCalculator(char *ptr, size_t count){
    uint16_t crc = 0x1021;
    size_t j;
    for (j = 0; j < count; ++j)
    {
        size_t i = 8;
        crc ^= (uint16_t)*ptr++ << 8;
        do
        {
            if (crc & 0x8000)
                crc = (crc << 1) ^ 0xFFFFU;
            else
                crc = crc << 1;
        } while (0 < --i);
    }
    return (crc);
}
```

```
uint8_t crcCalculator(char *ptr, size_t count){
    uint8_t crc = 0x1021;
    size_t j;
    for (j = 0; j < count; ++j)
    {
        size_t i = 8;
        crc ^= (uint8_t)*ptr++ << 8;
        do
        {
            if (crc & 0x8000)
                crc = (crc << 1) ^ 0xFFFFU;
            else
                crc = crc << 1;
        } while (0 < --i);
    }
    return (crc);
}
```

Egyik sem.

Magyarázat

Jánosnak egy CRC-CCITT (0xFFFF)-t kell megvalósítania. Ehhez a *crc* kezdeti értékeként 0xFFFF-t kell megadnia, illetve amikor a *crc* értéke eléri a 0x8000-et, akkor le kell vonnia a CRC polinomot, azaz 0x1021-et. A *crc* változó típusa 16 bites kell hogy legyen.

2. feladat 0/5 pont

Ájóti János rájött, hogy a *for* cikluson belüli *do-while* ciklus kiküszöbölhető egy look-up táblával, ezzel optimalizálva a kód futási idejét. A következőképpen írta át az előző függvényt, melynek hívásakor 0xFFFF *Seed*-et ad át a szenzornak megfelelően.

```
void crcCalculator(const uint8_t DataPtr[], uint32_t Length, uint16_t Seed, uint16_t* const crc)
{
    uint8_t  table_index;
    uint16_t crc_value = Seed;
    uint32_t count;
    for (count = 0; count < Length; count++)
    {
        table_index = ((uint8_t)(crc_value >> 8)) ^ (DataPtr[count]);
        crc_value   = (uint16_t)((crc_value << 8) ^ CrcTable[table_index]);
    }
    *crc = crc_value;
}
```

Melyik look-up tábla használatával tudja János ellenőrizni a szenzortól érkező adatokat?

Válasz

```
static const uint16_t CrcTable[256] =
{
0x0000u, 0x1021u, 0x2042u, 0x3063u, 0x4084u, 0x50a5u, 0x60c6u, 0x70e7u,
0x8108u, 0x9129u, 0xa14au, 0xb16bu, 0xc18cu, 0xd1adu, 0xe1ceu, 0xf1efu,
0x1231u, 0x0210u, 0x3273u, 0x2252u, 0x52b5u, 0x4294u, 0x72f7u, 0x62d6u,
0x9339u, 0x8318u, 0xb37bu, 0xa35au, 0xd3bdu, 0xc39cu, 0xf3ffu, 0xe3deu,
0x2462u, 0x3443u, 0x0420u, 0x1401u, 0x64e6u, 0x74c7u, 0x44a4u, 0x5485u,
0xa56au, 0xb54bu, 0x8528u, 0x9509u, 0xe5eeu, 0xf5cfu, 0xc5acu, 0xd58du,
0x3653u, 0x2672u, 0x1611u, 0x0630u, 0x76d7u, 0x66f6u, 0x5695u, 0x46b4u,
0xb75bu, 0xa77au, 0x9719u, 0x8738u, 0xf7dfu, 0xe7feu, 0xd79du, 0xc7bcu,
0x48c4u, 0x58e5u, 0x6886u, 0x78a7u, 0x0840u, 0x1861u, 0x2802u, 0x3823u,
0xc9ccu, 0xd9edu, 0xe98eu, 0xf9afu, 0x8948u, 0x9969u, 0xa90au, 0xb92bu,
0x5af5u, 0x4ad4u, 0x7ab7u, 0x6a96u, 0x1a71u, 0x0a50u, 0x3a33u, 0x2a12u,
0xdbfdu, 0xcbdcu, 0xfbbfu, 0xeb9eu, 0x9b79u, 0x8b58u, 0xbb3bu, 0xab1au,
0x6ca6u, 0x7c87u, 0x4ce4u, 0x5cc5u, 0x2c22u, 0x3c03u, 0x0c60u, 0x1c41u,
0xedaeu, 0xfd8fu, 0xcdecu, 0xddcdu, 0xad2bu, 0xbd0bu, 0x8d68u, 0x9d49u,
0x7e97u, 0x6eb6u, 0x5ed5u, 0x4ef4u, 0x3e13u, 0x2e32u, 0x1e51u, 0x0e70u,
0xff9fu, 0xefbeu, 0xdfddu, 0xcffcu, 0xbf1bu, 0xaf3au, 0x9f59u, 0x8f78u,
0x9188u, 0x81a9u, 0xb1cau, 0xa1ebu, 0xd10cu, 0xc12du, 0xf14eu, 0xe16fu,
0x1080u, 0x00a1u, 0x30c2u, 0x20e3u, 0x5004u, 0x4025u, 0x7046u, 0x6067u,
0x83b9u, 0x9398u, 0xa3fbu, 0xb3dau, 0xc33du, 0xd31cu, 0xe37fu, 0xf35eu,
0x02b1u, 0x1290u, 0x22f3u, 0x32d2u, 0x4235u, 0x5214u, 0x6277u, 0x7256u,
0xb5eau, 0xa5cbu, 0x95a8u, 0x8589u, 0xf56eu, 0xe54fu, 0xd52cu, 0xc50du,
0x34e2u, 0x24c3u, 0x14a0u, 0x0481u, 0x7466u, 0x6447u, 0x5424u, 0x4405u,
0xa7dbu, 0xb7fau, 0x8799u, 0x97b8u, 0xe75fu, 0xf77eu, 0xc71du, 0xd73cu,
0x26d3u, 0x36f2u, 0x0691u, 0x16b0u, 0x6657u, 0x7676u, 0x4615u, 0x5634u,
0xd94cu, 0xc96du, 0xf90eu, 0xe92fu, 0x99c8u, 0x89e9u, 0xb98au, 0xa9abu,
0x5844u, 0x4865u, 0x7806u, 0x6827u, 0x18c0u, 0x08e1u, 0x3882u, 0x28a3u,
0xcb7du, 0xdb5cu, 0xeb3fu, 0xfb1eu, 0x8bf9u, 0x9bd8u, 0xabbbu, 0xbb9au,
0x4a75u, 0x5a54u, 0x6a37u, 0x7a16u, 0x0af1u, 0x1ad0u, 0x2ab3u, 0x3a92u,
0xfd2eu, 0xed0fu, 0xdd6cu, 0xcd4du, 0xbdaau, 0xad8bu, 0x9de8u, 0x8dc9u,
0x7c26u, 0x6c07u, 0x5c64u, 0x4c45u, 0x3ca2u, 0x2c83u, 0x1ce0u, 0x0cc1u,
0xef1fu, 0xff3eu, 0xcf5du, 0xdf7cu, 0xaf9bu, 0xbfbau, 0x8fd9u, 0x9ff8u,
0x6e17u, 0x7e36u, 0x4e55u, 0x5e74u, 0x2e93u, 0x3eb2u, 0x0ed1u, 0x1ef0u
};
```

```
static const uint16_t CrcTable[255] =
{
0x0000u, 0x1021u, 0x2042u, 0x3063u, 0x4084u, 0x50a5u, 0x60c6u, 0x70e7u,
0x8108u, 0x9129u, 0xa14au, 0xb16bu, 0xc18cu, 0xd1adu, 0xe1ceu, 0xf1efu,
0x1231u, 0x0210u, 0x3273u, 0x2252u, 0x52b5u, 0x4294u, 0x72f7u, 0x62d6u,
0x9339u, 0x8318u, 0xb37bu, 0xa35au, 0xd3bdu, 0xc39cu, 0xf3ffu, 0xe3deu,
0x2462u, 0x3443u, 0x0420u, 0x1401u, 0x64e6u, 0x74c7u, 0x44a4u, 0x5485u,
0xa56au, 0xb54bu, 0x8528u, 0x9509u, 0xe5eeu, 0xf5cfu, 0xc5acu, 0xd58du,
0x3653u, 0x2672u, 0x1611u, 0x0630u, 0x76d7u, 0x66f6u, 0x5695u, 0x46b4u,
0xb75bu, 0xa77au, 0x9719u, 0x8738u, 0xf7dfu, 0xe7feu, 0xd79du, 0xc7bcu,
0x48c4u, 0x58e5u, 0x6886u, 0x78a7u, 0x0840u, 0x1861u, 0x2802u, 0x3823u,
0xc9ccu, 0xd9edu, 0xe98eu, 0xf9afu, 0x8948u, 0x9969u, 0xa90au, 0xb92bu,
0x5af5u, 0x4ad4u, 0x7ab7u, 0x6a96u, 0x1a71u, 0x0a50u, 0x3a33u, 0x2a12u,
0xdbfdu, 0xcbdcu, 0xfbbfu, 0xeb9eu, 0x9b79u, 0x8b58u, 0xbb3bu, 0xab1au,
0x6ca6u, 0x7c87u, 0x4ce4u, 0x5cc5u, 0x2c22u, 0x3c03u, 0x0c60u, 0x1c41u,
```

```
0xedaeu, 0xfd8fu, 0xcdecu, 0xddcdu, 0xad2au, 0xbd0bu, 0x8d68u, 0x9d49u,
0x7e97u, 0x6eb6u, 0x5ed5u, 0x4ef4u, 0x3e13u, 0x2e32u, 0x1e51u, 0x0e70u,
0xff9fu, 0xefbeu, 0xdfddu, 0xcffcu, 0xbf1bu, 0xaf3au, 0x9f59u, 0x8f78u,
0x9188u, 0x81a9u, 0xb1cau, 0xa1ebu, 0xd10cu, 0xc12du, 0xf14eu, 0xe16fu,
0x1080u, 0x00a1u, 0x30c2u, 0x20e3u, 0x5004u, 0x4025u, 0x7046u, 0x6067u,
0x83b9u, 0x9398u, 0xa3fbu, 0xb3dau, 0xc33du, 0xd31cu, 0xe37fu, 0xf35eu,
0x02b1u, 0x1290u, 0x22f3u, 0x32d2u, 0x4235u, 0x5214u, 0x6277u, 0x7256u,
0xb5eau, 0xa5cbu, 0x95a8u, 0x8589u, 0xf56eu, 0xe54fu, 0xd52cu, 0xc50du,
0x34e2u, 0x24c3u, 0x14a0u, 0x0481u, 0x7466u, 0x6447u, 0x5424u, 0x4405u,
0xa7dbu, 0xb7fau, 0x8799u, 0x97b8u, 0xe75fu, 0xf77eu, 0xc71du, 0xd73cu,
0x26d3u, 0x36f2u, 0x0691u, 0x16b0u, 0x6657u, 0x7676u, 0x4615u, 0x5634u,
0xd94cu, 0xc96du, 0xf90eu, 0xe92fu, 0x99c8u, 0x89e9u, 0xb98au, 0xa9abu,
0x5844u, 0x4865u, 0x7806u, 0x6827u, 0x18c0u, 0x08e1u, 0x3882u, 0x28a3u,
0xcb7du, 0xdb5cu, 0xeb3fu, 0xfb1eu, 0x8bf9u, 0x9bd8u, 0xabbbu, 0xbb9au,
0x4a75u, 0x5a54u, 0x6a37u, 0x7a16u, 0x0af1u, 0x1ad0u, 0x2ab3u, 0x3a92u,
0xfd2eu, 0xed0fu, 0xdd6cu, 0xcd4du, 0xbdaau, 0xad8bu, 0x9de8u, 0x8dc9u,
0x7c26u, 0x6c07u, 0x5c64u, 0x4c45u, 0x3ca2u, 0x2c83u, 0x1ce0u, 0x0cc1u,
0xef1fu, 0xff3eu, 0xcf5du, 0xdf7cu, 0xaf9bu, 0xbfbau, 0x8fd9u, 0x9ff8u,
0x6e17u, 0x7e36u, 0x4e55u, 0x5e74u, 0x2e93u, 0x3eb2u, 0x0ed1u
};
```

```
static const uint16_t CrcTable[255] =
{
0x0000u, 0x1021u, 0x2042u, 0x3063u, 0x4084u, 0x50a5u, 0x60c6u, 0x70e7u,
0x8108u, 0x9129u, 0xa14au, 0xb16bu, 0xc18cu, 0xd1adu, 0xe1ceu, 0xf1efu,
0x1231u, 0x0210u, 0x3273u, 0x2252u, 0x52b5u, 0x4294u, 0x72f7u, 0x62d6u,
0x9339u, 0x8318u, 0xb37bu, 0xa35au, 0xd3bdu, 0xc39cu, 0xf3ffu, 0xe3deu,
0x2462u, 0x3443u, 0x0420u, 0x1401u, 0x64e6u, 0x74c7u, 0x44a4u, 0x5485u,
0xa56au, 0xb54bu, 0x8528u, 0x9509u, 0xe5eeu, 0xf5cfu, 0xc5acu, 0xd58du,
0x3653u, 0x2672u, 0x1611u, 0x0630u, 0x76d7u, 0x66f6u, 0x5695u, 0x46b4u,
0xb75bu, 0xa77au, 0x9719u, 0x8738u, 0xf7dfu, 0xe7feu, 0xd79du, 0xc7bcu,
0x48c4u, 0x58e5u, 0x6886u, 0x78a7u, 0x0840u, 0x1861u, 0x2802u, 0x3823u,
0xc9ccu, 0xd9edu, 0xe98eu, 0xf9afu, 0x8948u, 0x9969u, 0xa90au, 0xb92bu,
0x5af5u, 0x4ad4u, 0x7ab7u, 0x6a96u, 0x1a71u, 0x0a50u, 0x3a33u, 0x2a12u,
0xdbfdu, 0xcbdcu, 0xfbbfu, 0xeb9eu, 0x9b79u, 0x8b58u, 0xbb3bu, 0xab1au,
0x6ca6u, 0x7c87u, 0x4ce4u, 0x5cc5u, 0x2c22u, 0x3c03u, 0x0c60u, 0x1c41u,
0xedaeu, 0xfd8fu, 0xcdecu, 0xddcdu, 0xad2bu, 0xbd0bu, 0x8d68u, 0x9d49u,
0x7e97u, 0x6eb6u, 0x5ed5u, 0x4ef4u, 0x3e13u, 0x2e32u, 0x1e51u, 0x0e70u,
0xff9fu, 0xefbeu, 0xdfddu, 0xcffcu, 0xbf1bu, 0xaf3au, 0x9f59u, 0x8f78u,
0x9188u, 0x81a9u, 0xb1cau, 0xa1ebu, 0xd10cu, 0xc12du, 0xf14eu, 0xe16fu,
0x1080u, 0x00a1u, 0x30c2u, 0x20e3u, 0x5004u, 0x4025u, 0x7046u, 0x6067u,
0x83b9u, 0x9398u, 0xa3fbu, 0xb3dau, 0xc33du, 0xd31cu, 0xe37fu, 0xf35eu,
0x02b1u, 0x1290u, 0x22f3u, 0x32d2u, 0x4235u, 0x5214u, 0x6277u, 0x7256u,
0xb5eau, 0xa5cbu, 0x95a8u, 0x8589u, 0xf56eu, 0xe54fu, 0xd52cu, 0xc50du,
0x34e2u, 0x24c3u, 0x14a0u, 0x0481u, 0x7466u, 0x6447u, 0x5424u, 0x4405u,
0xa7dbu, 0xb7fau, 0x8799u, 0x97b8u, 0xe75fu, 0xf77eu, 0xc71du, 0xd73cu,
0x26d3u, 0x36f2u, 0x0691u, 0x16b0u, 0x6657u, 0x7676u, 0x4615u, 0x5634u,
0xd94cu, 0xc96du, 0xf90eu, 0xe92fu, 0x99c8u, 0x89e9u, 0xb98au, 0xa9abu,
0x5844u, 0x4865u, 0x7806u, 0x6827u, 0x18c0u, 0x08e1u, 0x3882u, 0x28a3u,
0xcb7du, 0xdb5cu, 0xeb3fu, 0xfb1eu, 0x8bf9u, 0x9bd8u, 0xabbbu, 0xbb9au,
0x4a75u, 0x5a54u, 0x6a37u, 0x7a16u, 0x0af1u, 0x1ad0u, 0x2ab3u, 0x3a92u,
0xfd2eu, 0xed0fu, 0xdd6cu, 0xcd4du, 0xbdaau, 0xad8bu, 0x9de8u, 0x8dc9u,
0x7c26u, 0x6c07u, 0x5c64u, 0x4c45u, 0x3ca2u, 0x2c83u, 0x1ce0u, 0x0cc1u,
```

```
0xef1fu, 0xff3eu, 0xcf5du, 0xdf7cu, 0xaf9bu, 0xbfbau, 0x8fd9u, 0x9ff8u,
0x6e17u, 0x7e36u, 0x4e55u, 0x5e74u, 0x2e93u, 0x3eb2u, 0x0ed1u
};
```

```
static const uint16_t CrcTable[256] =
{
0x0000u, 0x1021u, 0x2042u, 0x3063u, 0x4084u, 0x50a5u, 0x60c6u, 0x70e7u,
0x8108u, 0x9129u, 0xa14au, 0xb16bu, 0xc18cu, 0xd1adu, 0xe1ceu, 0xf1efu,
0x1231u, 0x0210u, 0x3273u, 0x2252u, 0x52b5u, 0x4294u, 0x72f7u, 0x62d6u,
0x9339u, 0x8318u, 0xb37bu, 0xa35au, 0xd3bdu, 0xc39cu, 0xf3ffu, 0xe3deu,
0x2462u, 0x3443u, 0x0420u, 0x1401u, 0x64e6u, 0x74c7u, 0x44a4u, 0x5485u,
0xa56au, 0xb54bu, 0x8528u, 0x9509u, 0xe5eeu, 0xf5cfu, 0xc5acu, 0xd58du,
0x3653u, 0x2672u, 0x1611u, 0x0630u, 0x76d7u, 0x66f6u, 0x5695u, 0x46b4u,
0xb75bu, 0xa77au, 0x9719u, 0x8738u, 0xf7dfu, 0xe7feu, 0xd79du, 0xc7bcu,
0x48c4u, 0x58e5u, 0x6886u, 0x78a7u, 0x0840u, 0x1861u, 0x2802u, 0x3823u,
0xc9ccu, 0xd9edu, 0xe98eu, 0xf9afu, 0x8948u, 0x9969u, 0xa90au, 0xb92bu,
0x5af5u, 0x4ad4u, 0x7ab7u, 0x6a96u, 0x1a71u, 0x0a50u, 0x3a33u, 0x2a12u,
0xdbfdu, 0xcbdcu, 0xfbbfu, 0xeb9eu, 0x9b79u, 0x8b58u, 0xbb3bu, 0xab1au,
0x6ca6u, 0x7c87u, 0x4ce4u, 0x5cc5u, 0x2c22u, 0x3c03u, 0x0c60u, 0x1c41u,
0xedaeu, 0xfd8fu, 0xcdecu, 0xddcdu, 0xad2au, 0xbd0bu, 0x8d68u, 0x9d49u,
0x7e97u, 0x6eb6u, 0x5ed5u, 0x4ef4u, 0x3e13u, 0x2e32u, 0x1e51u, 0x0e70u,
0xff9fu, 0xefbeu, 0xdfddu, 0xcffcu, 0xbf1bu, 0xaf3au, 0x9f59u, 0x8f78u,
0x9188u, 0x81a9u, 0xb1cau, 0xa1ebu, 0xd10cu, 0xc12du, 0xf14eu, 0xe16fu,
0x1080u, 0x00a1u, 0x30c2u, 0x20e3u, 0x5004u, 0x4025u, 0x7046u, 0x6067u,
0x83b9u, 0x9398u, 0xa3fbu, 0xb3dau, 0xc33du, 0xd31cu, 0xe37fu, 0xf35eu,
0x02b1u, 0x1290u, 0x22f3u, 0x32d2u, 0x4235u, 0x5214u, 0x6277u, 0x7256u,
0xb5eau, 0xa5cbu, 0x95a8u, 0x8689u, 0xf56eu, 0xe54fu, 0xd52cu, 0xc50du,
0x34e2u, 0x24c3u, 0x14a0u, 0x0481u, 0x7466u, 0x6447u, 0x5424u, 0x4405u,
0xa7dbu, 0xb7fau, 0x8799u, 0x97b8u, 0xe75fu, 0xf77eu, 0xc71du, 0xd73cu,
0x26d3u, 0x36f2u, 0x0691u, 0x16b0u, 0x6657u, 0x7676u, 0x4615u, 0x5634u,
0xd94cu, 0xc96du, 0xf90eu, 0xe92fu, 0x99c8u, 0x89e9u, 0xb98au, 0xa9abu,
0x5844u, 0x4865u, 0x7806u, 0x6827u, 0x18c0u, 0x08e1u, 0x3882u, 0x28a3u,
0xcb7du, 0xdb5cu, 0xeb3fu, 0xfb1eu, 0x8bf9u, 0x9bd8u, 0xabbbu, 0xbb9au,
0x4a75u, 0x5a54u, 0x6a37u, 0x7a16u, 0x0af1u, 0x1ad0u, 0x2ab3u, 0x3a92u,
0xfd2eu, 0xed0fu, 0xdd6cu, 0xcd4du, 0xbdaau, 0xad8bu, 0x9de8u, 0x8dc9u,
0x7c26u, 0x6c07u, 0x5c64u, 0x4c45u, 0x3ca2u, 0x2c83u, 0x1ce0u, 0x0cc1u,
0xef1fu, 0xff3eu, 0xcf5du, 0xdf7cu, 0xaf9bu, 0xbfbau, 0x8fd9u, 0x9ff8u,
0x6e17u, 0x7e36u, 0x4e55u, 0x5e74u, 0x2e93u, 0x3eb2u, 0x0ed1u, 0x1ef0u
};
```

Egyik sem.

Magyarázat

Egyik megoldás sem helyes.

A függvény ilyen implementálása mellett, ha a seed értéke 0xFFFF, akkor a look-up tábla kiszámolásához az előző feladatban megadott függvényt *crc = 0* inicializálással kell meghívni a 0-255 számokra.

Ennek eredménye:

```
static const uint16_t CrcTable[256] =
{
0x0000u, 0x1021u, 0x2042u, 0x3063u, 0x4084u, 0x50a5u, 0x60c6u, 0x70e7u,
0x8108u, 0x9129u, 0xa14au, 0xb16bu, 0xc18cu, 0xd1adu, 0xe1ceu, 0xf1efu,
0x1231u, 0x0210u, 0x3273u, 0x2252u, 0x52b5u, 0x4294u, 0x72f7u, 0x62d6u,
0x9339u, 0x8318u, 0xb37bu, 0xa35au, 0xd3bdu, 0xc39cu, 0xf3ffu, 0xe3deu,
0x2462u, 0x3443u, 0x0420u, 0x1401u, 0x64e6u, 0x74c7u, 0x44a4u, 0x5485u,
0xa56au, 0xb54bu, 0x8528u, 0x9509u, 0xe5eeu, 0xf5cfu, 0xc5acu, 0xd58du,
0x3653u, 0x2672u, 0x1611u, 0x0630u, 0x76d7u, 0x66f6u, 0x5695u, 0x46b4u,
0xb75bu, 0xa77au, 0x9719u, 0x8738u, 0xf7dfu, 0xe7feu, 0xd79du, 0xc7bcu,
0x48c4u, 0x58e5u, 0x6886u, 0x78a7u, 0x0840u, 0x1861u, 0x2802u, 0x3823u,
0xc9ccu, 0xd9edu, 0xe98eu, 0xf9afu, 0x8948u, 0x9969u, 0xa90au, 0xb92bu,
0x5af5u, 0x4ad4u, 0x7ab7u, 0x6a96u, 0x1a71u, 0x0a50u, 0x3a33u, 0x2a12u,
0xdbfdu, 0xcbdcu, 0xfbbfu, 0xeb9eu, 0x9b79u, 0x8b58u, 0xbb3bu, 0xab1au,
0x6ca6u, 0x7c87u, 0x4ce4u, 0x5cc5u, 0x2c22u, 0x3c03u, 0x0c60u, 0x1c41u,
0xedaeu, 0xfd8fu, 0xcdecu, 0xddcdu, 0xad2au, 0xbd0bu, 0x8d68u, 0x9d49u,
0x7e97u, 0x6eb6u, 0x5ed5u, 0x4ef4u, 0x3e13u, 0x2e32u, 0x1e51u, 0x0e70u,
0xff9fu, 0xefbeu, 0xdfddu, 0xcffcu, 0xbf1bu, 0xaf3au, 0x9f59u, 0x8f78u,
0x9188u, 0x81a9u, 0xb1cau, 0xa1ebu, 0xd10cu, 0xc12du, 0xf14eu, 0xe16fu,
0x1080u, 0x00a1u, 0x30c2u, 0x20e3u, 0x5004u, 0x4025u, 0x7046u, 0x6067u,
0x83b9u, 0x9398u, 0xa3fbu, 0xb3dau, 0xc33du, 0xd31cu, 0xe37fu, 0xf35eu,
0x02b1u, 0x1290u, 0x22f3u, 0x32d2u, 0x4235u, 0x5214u, 0x6277u, 0x7256u,
0xb5eau, 0xa5cbu, 0x95a8u, 0x8589u, 0xf56eu, 0xe54fu, 0xd52cu, 0xc50du,
0x34e2u, 0x24c3u, 0x14a0u, 0x0481u, 0x7466u, 0x6447u, 0x5424u, 0x4405u,
0xa7dbu, 0xb7fau, 0x8799u, 0x97b8u, 0xe75fu, 0xf77eu, 0xc71du, 0xd73cu,
0x26d3u, 0x36f2u, 0x0691u, 0x16b0u, 0x6657u, 0x7676u, 0x4615u, 0x5634u,
0xd94cu, 0xc96du, 0xf90eu, 0xe92fu, 0x99c8u, 0x89e9u, 0xb98au, 0xa9abu,
0x5844u, 0x4865u, 0x7806u, 0x6827u, 0x18c0u, 0x08e1u, 0x3882u, 0x28a3u,
0xcb7du, 0xdb5cu, 0xeb3fu, 0xfb1eu, 0x8bf9u, 0x9bd8u, 0xabbbu, 0xbb9au,
0x4a75u, 0x5a54u, 0x6a37u, 0x7a16u, 0x0af1u, 0x1ad0u, 0x2ab3u, 0x3a92u,
0xfd2eu, 0xed0fu, 0xdd6cu, 0xcd4du, 0xbdaau, 0xad8bu, 0x9de8u, 0x8dc9u,
0x7c26u, 0x6c07u, 0x5c64u, 0x4c45u, 0x3ca2u, 0x2c83u, 0x1ce0u, 0x0cc1u,
0xef1fu, 0xff3eu, 0xcf5du, 0xdf7cu, 0xaf9bu, 0xbfbau, 0x8fd9u, 0x9ff8u,
0x6e17u, 0x7e36u, 0x4e55u, 0x5e74u, 0x2e93u, 0x3eb2u, 0x0ed1u, 0x1ef0u
};
```

A lehetőségek gyors ellenőrzésére memcmp-t volt érdemes használni.

3. feladat 0/5 pont

Ájóti János olvasgatva a stm32f091rc kontroller adatlapját, arra a következtetésre jut, hogy lehetne hardveres CRC modult is használni.

Melyik konfiguráció segítene Jánosnak a szenzorral való kommunikációban?

Válasz

```c
CRC_HandleTypeDef hcrc;
static void MX_CRC_Init(void)
{
  hcrc.Instance                     = CRC;
  hcrc.Init.DefaultPolynomialUse    = DEFAULT_POLYNOMIAL_DISABLE;
  hcrc.Init.DefaultInitValueUse     = DEFAULT_INIT_VALUE_DISABLE;
  hcrc.Init.GeneratingPolynomial    = 4129;
  hcrc.Init.CRCLength               = CRC_POLYLENGTH_16B;
  hcrc.Init.InitValue               = 65535;
  hcrc.Init.InputDataInversionMode  = CRC_INPUTDATA_INVERSION_NONE;
  hcrc.Init.OutputDataInversionMode = CRC_OUTPUTDATA_INVERSION_DISABLE;
  hcrc.InputDataFormat              = CRC_INPUTDATA_FORMAT_BYTES;
  if (HAL_CRC_Init(&hcrc) != HAL_OK){
    Error_Handler();
  }
}
```

```c
CRC_HandleTypeDef hcrc;
static void MX_CRC_Init(void)
{
  hcrc.Instance                     = CRC;
  hcrc.Init.DefaultPolynomialUse    = DEFAULT_POLYNOMIAL_ENABLE;
  hcrc.Init.DefaultInitValueUse     = DEFAULT_INIT_VALUE_DISABLE;
  hcrc.Init.GeneratingPolynomial    = 4129;
  hcrc.Init.CRCLength               = CRC_POLYLENGTH_16B;
  hcrc.Init.InitValue               = 65535;
  hcrc.Init.InputDataInversionMode  = CRC_INPUTDATA_INVERSION_NONE;
  hcrc.Init.OutputDataInversionMode = CRC_OUTPUTDATA_INVERSION_DISABLE;
  hcrc.InputDataFormat              = CRC_INPUTDATA_FORMAT_BYTES;
  if (HAL_CRC_Init(&hcrc) != HAL_OK){
    Error_Handler();
  }
}
```

```c
CRC_HandleTypeDef hcrc;
static void MX_CRC_Init(void)
{
  hcrc.Instance                     = CRC;
  hcrc.Init.DefaultPolynomialUse    = DEFAULT_POLYNOMIAL_DISABLE;
  hcrc.Init.DefaultInitValueUse     = DEFAULT_INIT_VALUE_DISABLE;
  hcrc.Init.GeneratingPolynomial    = 4129;
  hcrc.Init.CRCLength               = CRC_POLYLENGTH_8B;
  hcrc.Init.InitValue               = 65535;
  hcrc.Init.InputDataInversionMode  = CRC_INPUTDATA_INVERSION_NONE;
  hcrc.Init.OutputDataInversionMode = CRC_OUTPUTDATA_INVERSION_DISABLE;
  hcrc.InputDataFormat              = CRC_INPUTDATA_FORMAT_BYTES;
  if (HAL_CRC_Init(&hcrc) != HAL_OK){
    Error_Handler();
  }
}
```

```c
CRC_HandleTypeDef hcrc;
static void MX_CRC_Init(void)
{
  hcrc.Instance                     = CRC;
  hcrc.Init.DefaultPolynomialUse    = DEFAULT_POLYNOMIAL_ENABLE;
  hcrc.Init.DefaultInitValueUse     = DEFAULT_INIT_VALUE_DISABLE;
  hcrc.Init.GeneratingPolynomial    = 4129;
  hcrc.Init.CRCLength               = CRC_POLYLENGTH_8B;
  hcrc.Init.InitValue               = 65535;
  hcrc.Init.InputDataInversionMode  = CRC_INPUTDATA_INVERSION_NONE;
  hcrc.Init.OutputDataInversionMode = CRC_OUTPUTDATA_INVERSION_DISABLE;
  hcrc.InputDataFormat              = CRC_INPUTDATA_FORMAT_BYTES;
  if (HAL_CRC_Init(&hcrc) != HAL_OK){
    Error_Handler();
  }
}
```

```
CRC_HandleTypeDef hcrc;
static void MX_CRC_Init(void)
{
  hcrc.Instance                     = CRC;
  hcrc.Init.DefaultPolynomialUse    = DEFAULT_POLYNOMIAL_ENABLE;
  hcrc.Init.DefaultInitValueUse     = DEFAULT_INIT_VALUE_DISABLE;
  hcrc.Init.GeneratingPolynomial    = 65535;
  hcrc.Init.CRCLength               = CRC_POLYLENGTH_8B;
  hcrc.Init.InitValue               = 4129;
  hcrc.Init.InputDataInversionMode  = CRC_INPUTDATA_INVERSION_NONE;
  hcrc.Init.OutputDataInversionMode = CRC_OUTPUTDATA_INVERSION_DISABLE;
  hcrc.InputDataFormat              = CRC_INPUTDATA_FORMAT_BYTES;
  if (HAL_CRC_Init(&hcrc) != HAL_OK){
    Error_Handler();
  }
}
```

```
CRC_HandleTypeDef hcrc;
static void MX_CRC_Init(void)
{
  hcrc.Instance                     = CRC;
  hcrc.Init.DefaultPolynomialUse    = DEFAULT_POLYNOMIAL_DISABLE;
  hcrc.Init.DefaultInitValueUse     = DEFAULT_INIT_VALUE_DISABLE;
  hcrc.Init.GeneratingPolynomial    = 65535;
  hcrc.Init.CRCLength               = CRC_POLYLENGTH_8B;
  hcrc.Init.InitValue               = 4129;
  hcrc.Init.InputDataInversionMode  = CRC_INPUTDATA_INVERSION_NONE;
  hcrc.Init.OutputDataInversionMode = CRC_OUTPUTDATA_INVERSION_DISABLE;
  hcrc.InputDataFormat              = CRC_INPUTDATA_FORMAT_BYTES;
  if (HAL_CRC_Init(&hcrc) != HAL_OK){
    Error_Handler();
  }
}
```

```
CRC_HandleTypeDef hcrc;
static void MX_CRC_Init(void)
{
  hcrc.Instance                     = CRC;
  hcrc.Init.DefaultPolynomialUse    = DEFAULT_POLYNOMIAL_DISABLE;
  hcrc.Init.DefaultInitValueUse     = DEFAULT_INIT_VALUE_DISABLE;
  hcrc.Init.GeneratingPolynomial    = 65535;
  hcrc.Init.CRCLength               = CRC_POLYLENGTH_16B;
  hcrc.Init.InitValue               = 4129;
  hcrc.Init.InputDataInversionMode  = CRC_INPUTDATA_INVERSION_NONE;
  hcrc.Init.OutputDataInversionMode = CRC_OUTPUTDATA_INVERSION_DISABLE;
  hcrc.InputDataFormat              = CRC_INPUTDATA_FORMAT_BYTES;
  if (HAL_CRC_Init(&hcrc) != HAL_OK){
    Error_Handler();
  }
}
```

```
CRC_HandleTypeDef hcrc;
static void MX_CRC_Init(void)
{
  hcrc.Instance                     = CRC;
  hcrc.Init.DefaultPolynomialUse    = DEFAULT_POLYNOMIAL_ENABLE;
  hcrc.Init.DefaultInitValueUse     = DEFAULT_INIT_VALUE_DISABLE;
  hcrc.Init.GeneratingPolynomial    = 65535;
  hcrc.Init.CRCLength               = CRC_POLYLENGTH_16B;
  hcrc.Init.InitValue               = 4129;
  hcrc.Init.InputDataInversionMode  = CRC_INPUTDATA_INVERSION_NONE;
  hcrc.Init.OutputDataInversionMode = CRC_OUTPUTDATA_INVERSION_DISABLE;
  hcrc.InputDataFormat              = CRC_INPUTDATA_FORMAT_BYTES;
  if (HAL_CRC_Init(&hcrc) != HAL_OK){
    Error_Handler();
  }
}
```

Magyarázat

A default polinom használatát nem szabad engedélyezni, különben hiába konfiguráljuk be a többi paramétert. A generáló polinom értéke 0x1021, a kezdeti érték 0xFFFF és 16 bites CRC-t várunk a modultól.

4. feladat 0/5 pont

Miután János implementálta a megoldásokat, szeretné ellenőrizni az algoritmusát.

Milyen értéket kéne kapni a "Hello world!" sztringre?

(0XDEAD válasz esetén DEAD-et adjatok meg.)

Válaszok

A helyes válasz:
5476
BD22

Magyarázat

Fontos, hogy a lezáró 0-ra is számoljunk még CRC-t.

Az egyes byte-ok után a következőképpen alakul a CRC értéke.

FFFF -> 283C -> A569 -> 2165 -> FC69 -> DADA -> 9455 -> 984D -> D2F8 ->

4DEA -> DE43 -> 4591 -> BD22 -> 5476

5. feladat 0/10 pont

Most, hogy minden adott a hőmérséklet megfelelő kiolvasásához és ellenőrzéséhez, szeretné ezt egy külön függvényben elvégezni. A TMP126-Q1 /CS lábát a PB6 GPIO-ra köti. Az SPI konfigurációja megfelel a chip által elvártnak, MSB first és 16 bites blokkokban kommunikálunk vele.

Melyik kódrészlet valósítja meg helyesen a kiolvasását és CRC ellenőrzést?

Válaszok

```c
#define SwapTwoBytes(data) \
( (((data) >> 8) & 0x00FF) | (((data) << 8) & 0xFF00) )

void readTempRaw(uint16_t *temp)
{
  uint16_t buff[3] = {0x4500,0,0};
  uint16_t buffSwapped[2];
  uint16_t crc = 0xFFFF;

  HAL_GPIO_WritePin(GPIOB,GPIO_PIN_6,GPIO_PIN_RESET);
  HAL_SPI_Transmit(&hspi1,&buff[0],1,HAL_MAX_DELAY);
  HAL_SPI_Receive(&hspi1,&buff[1],2,HAL_MAX_DELAY);
  HAL_GPIO_WritePin(GPIOB,GPIO_PIN_6,GPIO_PIN_SET);

  buffSwapped[0] = SwapTwoBytes(buff[0]);
  buffSwapped[1] = SwapTwoBytes(buff[1]);

  crcCalculator((uint8_t*)buffSwapped, 4, 0xFFFF, &crc);

  if (crc != buff[2]) {
    /* Hibakezelés */
  }
  else {
    *temp = buff[1];
  }
}
```

```c
#define SwapTwoBytes(data) \
( (((data) >> 8) & 0x00FF) | (((data) << 8) & 0xFF00) )

void readTempRaw(uint16_t *temp)
{
  uint16_t buff[3] = {0x4700,0,0};
  uint16_t buffSwapped[2];
  uint16_t crc = 0xFFFF;

  HAL_GPIO_WritePin(GPIOB,GPIO_PIN_6,GPIO_PIN_RESET);
  HAL_SPI_Transmit(&hspi1,&buff[0],1,HAL_MAX_DELAY);
  HAL_SPI_Receive(&hspi1,&buff[1],2,HAL_MAX_DELAY);
  HAL_GPIO_WritePin(GPIOB,GPIO_PIN_6,GPIO_PIN_SET);

  buffSwapped[0] = SwapTwoBytes(buff[0]);
  buffSwapped[1] = SwapTwoBytes(buff[1]);

  crcCalculator((uint8_t*)buffSwapped, 4, 0xFFFF, &crc);

  if (crc != buff[2]) {
    /* Hibakezelés */
  }
  else {
    *temp = buff[1];
  }
}
```

```c
#define SwapTwoBytes(data) \
( (((data) >> 8) & 0x00FF) | (((data) << 8) & 0xFF00) )

void readTempRaw(uint16_t *temp)
{
  uint16_t buff[3] = {0xC500,0,0};
  uint16_t buffSwapped[2];
  uint16_t crc = 0xFFFF;

  HAL_GPIO_WritePin(GPIOB,GPIO_PIN_6,GPIO_PIN_RESET);
  HAL_SPI_Transmit(&hspi1,&buff[0],1,HAL_MAX_DELAY);
  HAL_SPI_Receive(&hspi1,&buff[1],2,HAL_MAX_DELAY);
  HAL_GPIO_WritePin(GPIOB,GPIO_PIN_6,GPIO_PIN_SET);

  buffSwapped[0] = SwapTwoBytes(buff[0]);
  buffSwapped[1] = SwapTwoBytes(buff[1]);

  crcCalculator((uint8_t*)buffSwapped, 4, 0xFFFF, &crc);

  if (crc != buff[2]) {
    /* Hibakezelés */
  }
  else {
    *temp = buff[1];
  }
}
```

```c
#define SwapTwoBytes(data) \
( (((data) >> 8) & 0x00FF) | (((data) << 8) & 0xFF00) )

void readTempRaw(uint16_t *temp)
{
  uint16_t buff[3] = {0xC700,0,0};
  uint16_t buffSwapped[2];
  uint16_t crc = 0xFFFF;

  HAL_GPIO_WritePin(GPIOB,GPIO_PIN_6,GPIO_PIN_RESET);
  HAL_SPI_Transmit(&hspi1,&buff[0],1,HAL_MAX_DELAY);
  HAL_SPI_Receive(&hspi1,&buff[1],2,HAL_MAX_DELAY);
  HAL_GPIO_WritePin(GPIOB,GPIO_PIN_6,GPIO_PIN_SET);

  buffSwapped[0] = SwapTwoBytes(buff[0]);
  buffSwapped[1] = SwapTwoBytes(buff[1]);

  crcCalculator((uint8_t*)buffSwapped, 4, 0xFFFF, &crc);

  if (crc != buff[2]) {
    /* Hibakezelés */
  }
  else {
    *temp = buff[1];
  }
}
```

```c
#define SwapTwoBytes(data) \
( (((data) >> 8) & 0x00FF) | (((data) << 8) & 0xFF00) )

void readTempRaw(uint16_t *temp)
{
  uint16_t buff[3] = {0x4500,0,0};
  uint16_t buffSwapped[2];
  uint16_t crc = 0xFFFF;

  HAL_GPIO_WritePin(GPIOB,GPIO_PIN_6,GPIO_PIN_RESET);
  HAL_SPI_Transmit(&hspi1,&buff[0],1,HAL_MAX_DELAY);
  HAL_SPI_Receive(&hspi1,&buff[1],2,HAL_MAX_DELAY);
  HAL_GPIO_WritePin(GPIOB,GPIO_PIN_6,GPIO_PIN_SET);

  buffSwapped[0] = SwapTwoBytes(buff[0]);
  buffSwapped[1] = SwapTwoBytes(buff[1]);

  crcCalculator((uint8_t*)buffSwapped, 2, 0xFFFF, &crc);

  if (crc != buff[2]) {
    /* Hibakezelés */
  }
  else {
    *temp = buff[1];
  }
}
```

```c
#define SwapTwoBytes(data) \
( (((data) >> 8) & 0x00FF) | (((data) << 8) & 0xFF00) )

void readTempRaw(uint16_t *temp)
{
  uint16_t buff[3] = {0x4700,0,0};
  uint16_t buffSwapped[2];
  uint16_t crc = 0xFFFF;

  HAL_GPIO_WritePin(GPIOB,GPIO_PIN_6,GPIO_PIN_RESET);
  HAL_SPI_Transmit(&hspi1,&buff[0],1,HAL_MAX_DELAY);
  HAL_SPI_Receive(&hspi1,&buff[1],2,HAL_MAX_DELAY);
  HAL_GPIO_WritePin(GPIOB,GPIO_PIN_6,GPIO_PIN_SET);

  buffSwapped[0] = SwapTwoBytes(buff[0]);
  buffSwapped[1] = SwapTwoBytes(buff[1]);

  crcCalculator((uint8_t*)buffSwapped, 2, 0xFFFF, &crc);

  if (crc != buff[2]) {
    /* Hibakezelés */
  }
  else {
    *temp = buff[1];
  }
}
```

```c
#define SwapTwoBytes(data) \
( (((data) >> 8) & 0x00FF) | (((data) << 8) & 0xFF00) )

void readTempRaw(uint16_t *temp)
{
  uint16_t buff[3] = {0xC500,0,0};
  uint16_t buffSwapped[2];
  uint16_t crc = 0xFFFF;

  HAL_GPIO_WritePin(GPIOB,GPIO_PIN_6,GPIO_PIN_RESET);
  HAL_SPI_Transmit(&hspi1,&buff[0],1,HAL_MAX_DELAY);
  HAL_SPI_Receive(&hspi1,&buff[1],2,HAL_MAX_DELAY);
  HAL_GPIO_WritePin(GPIOB,GPIO_PIN_6,GPIO_PIN_SET);

  buffSwapped[0] = SwapTwoBytes(buff[0]);
  buffSwapped[1] = SwapTwoBytes(buff[1]);

  crcCalculator((uint8_t*)buffSwapped, 2, 0xFFFF, &crc);

  if (crc != buff[2]) {
    /* Hibakezelés */
  }
  else {
    *temp = buff[1];
  }
}
```

```c
#define SwapTwoBytes(data) \
( (((data) >> 8) & 0x00FF) | (((data) << 8) & 0xFF00) )

void readTempRaw(uint16_t *temp)
{
  uint16_t buff[3] = {0xC700,0,0};
  uint16_t buffSwapped[2];
  uint16_t crc = 0xFFFF;

  HAL_GPIO_WritePin(GPIOB,GPIO_PIN_6,GPIO_PIN_RESET);
  HAL_SPI_Transmit(&hspi1,&buff[0],1,HAL_MAX_DELAY);
  HAL_SPI_Receive(&hspi1,&buff[1],2,HAL_MAX_DELAY);
  HAL_GPIO_WritePin(GPIOB,GPIO_PIN_6,GPIO_PIN_SET);

  buffSwapped[0] = SwapTwoBytes(buff[0]);
  buffSwapped[1] = SwapTwoBytes(buff[1]);

  crcCalculator((uint8_t*)buffSwapped, 2, 0xFFFF, &crc);

  if (crc != buff[2]) {
    /* Hibakezelés */
  }
  else {
    *temp = buff[1];
  }
}
```

Egyik sem.

Magyarázat

Mind a Command, mind a kiolvasott hőmérséklet 16 bites érték, ezért 4 bájtra kell CRC-t számolnunk. Az adatlap szerint a Command word 15-ös bitje Don't care, valamint a 9-es Auto increment bit (mivel csak egy data blockot olvasunk, így) bármely értéke helyes.

KÉSZÍTETTE

Megjelenés

Világos