ARM Cortex magú mikrovezérlők

UART gyakorlat

Scherer Balázs

Méréstechnika és
Információs Rendszerek
Tanszék

Tartalom

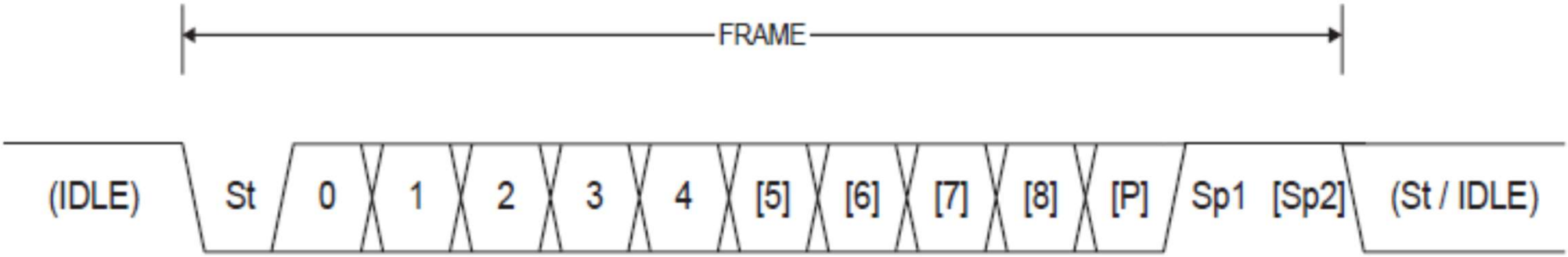
- UART periféria programozás
- Standard C library portolás
- Hasznos tippek

Universal asynchronous receiver/transmitter: UART

- Aszinkron soros átvitel
  - Már a mechanikus telegráf jellegű készülékeknél használtak ilyen átvitelt
  - 1970-es években kezdtek el ilyen aszinkron átvitelt használni
  - 1980 Más-más néven, de a Motorola és az Intel is kihozza saját UART jellegű chipjeit
  - 1990 Első bufferelt UART chipek
  - A mai napig alapfelszerelés az összes mikrovezérlőnél

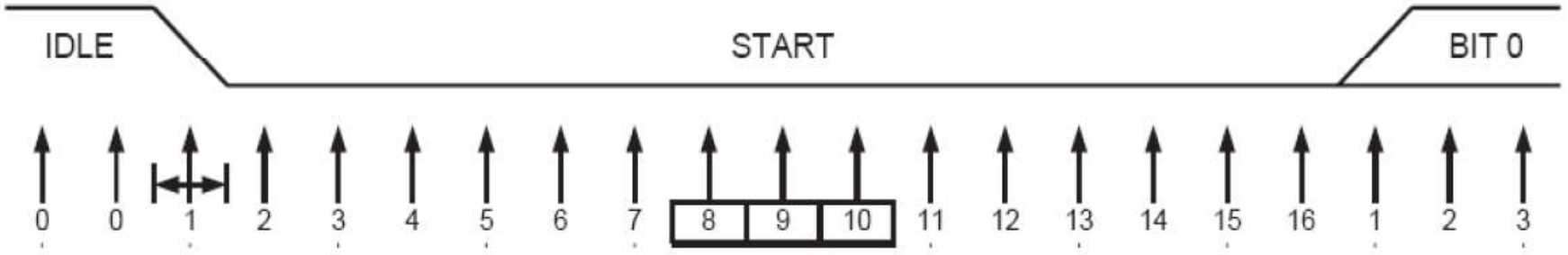
UART *folytatás*

- Egy UART keret
  - Start Bit
  - 5, 6, 7, 8 vagy 9 adat bit
  - Paritás Bit
  - 1, 1.5 vagy 2 Stop Bit
- Szabványos adatsebességek
  - 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200



UART *folytatás*

- UART bit felismerés, vétel.
  - A start bit lefutó élére elinduló mintavételezés
  - A bitszinkronnak az egész kereten keresztül ki kell tartania
  - Óraforrás csúszás problémát okozhat (RC oszcillátor, Hőmérséklet)
    - 2% felett jelentős probléma lehet (gyakorlatban <1%-ot



UART *folytatás*

- Az UART használata
  - **PC terminál, debug**, műszer kapcsolat (Rs232)
  - Ipari kommunikációs hálózat (Rs485)
  - IrDA
  - Autóipari kommunikációs hálózatok
    - LIN (Local Interconnect Network)
  - Modem jellegű illesztésre kommunikációs chipekkel
    - GSM, GPS, GPRS modem
    - ZigBee
    - TCP/IP chip
    - Power Line kommunikáció (külön illesztő chip)
    - USB virtuális soros port (FTDI chipek)

UART felprogramozás az STM32F429 Disco1 kártyán

UART az STM32F429 Disco1 kártyán

- STM32F429 Disco 1 Kártya dokumentáció 6.3.3 fejezet
  - Az ST-LINK/V2-B debug kapcsolat STM32F429I-DISC1 kártyán virtual COM port (VCP) kapcsolatot nyújt az STM32 USART1 (PA9: Rx, PA10: Tx) átirányításával
- A Programozás lépései:
  1. Órajel engedélyezés
  2. Az UART I/O lábainak alternatív funkciójának bekapcsolása
  3. Az USART1 inicializálása: 115200 baud, 8 data bit 0 parity 1 stop bit
  4. Az USART1 anszinkron módba állítása
  5. Az USART1 engedélyezése
  6. Karakterek küldése
  7. A LibC standard Output átirányítása

1. Órajel források engedélyezése

- Két órajel forrás engedélyezése szükséges: Reference manual Table 1. STM32F4xx register boundary addresses
  - A GPIOA órajel forrásának engedélyezése: AHB1 bus
  - Az USART1 órajel forrásának engedélyezése: APB2 bus

- STCube LL könyvtár BUS modul függvényei

1. Órajel források engedélyezése

- Két órajel forrás engedélyezése szükséges: Reference manual Table 1. STM32F4xx register boundary addresses
  - A GPIOA órajel forrásának engedélyezése: AHB1 bus
  - Az USART1 órajel forrásának engedélyezése: APB2 bus

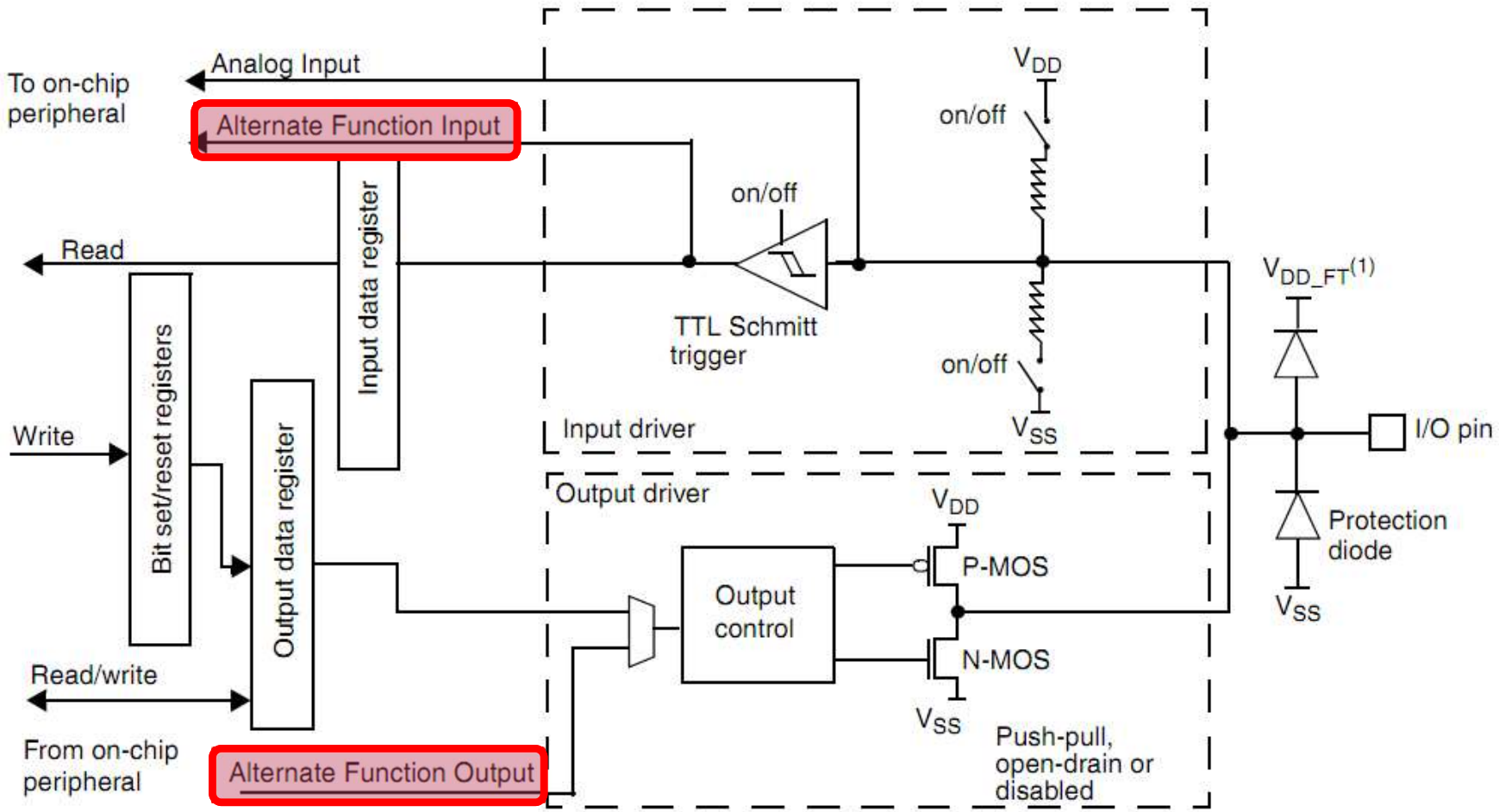
- STCube LL könyvtár BUS modul függvényei

  - LL_AHB1_GRP1_EnableClock(LL_AHB1_GRP1_PERIPH_GPIOA)

  - LL_APB2_GRP1_EnableClock (LL_APB2_GRP1_PERIPH_USART1) ;

2. A GPIOA PA0 láb alternate function beállítása

- Az alternate funkciók beállítása: STM32F429 Datasheet Table 12.



2. A GPIOA PA0 láb alternate function beállítása

- Az alternate funkciók beállítása: STM32F429 Datasheet Table 12.

Table 12. STM32F427xx and STM32F429xx alternate function mapping

Port		AF0	AF1	AF2	AF3	AF4	AF5	AF6	AF7	AF8	AF9	AF10	AF11
		SYS	TIM1/2	TIM3/4/5	TIM8/9/10/11	I2C1/2/3	SPI1/2/3/4/5/6	SPI2/3/SAI1	SPI3/USART1/2/3	USART6/UART4/5/7/8	CAN1/2/TIM12/13/14/LCD	OTG2_HS/OTG1_FS	ETH
Port A	PA0	-	TIM2_CH1/TIM2_ETR	TIM5_CH1	TIM8_ETR	-	-	-	USART2_CTS	UART4_TX	-	-	ETH_MII_CRS
	PA1	-	TIM2_CH2	TIM5_CH2	-	-	-	-	USART2_RTS	UART4_RX	-	-	ETH_MII_RX_CLK/ETH_RMII_REF_CLK
	PA2	-	TIM2_CH3	TIM5_CH3	TIM9_CH1	-	-	-	USART2_TX	-	-	-	ETH_MDIO
	PA3	-	TIM2_CH4	TIM5_CH4	TIM9_CH2	-	-	-	USART2_RX	-	-	OTG_HS_ULPI_D0	ETH_MII_COL
	PA4	-	-	-	-	-	SPI1_NSS	SPI3_NSS/I2S3_WS	USART2_CK	-	-	-	-
	PA5	-	TIM2_CH1/TIM2_ETR	-	TIM8_CH1N	-	SPI1_SCK	-	-	-	-	OTG_HS_ULPI_CK	-
	PA6	-	TIM1_BKIN	TIM3_CH1	TIM8_BKIN	-	SPI1_MISO	-	-	-	TIM13_CH1	-	-
	PA7	-	TIM1_CH1N	TIM3_CH2	TIM8_CH1N	-	SPI1_MOSI	-	-	-	TIM14_CH1	-	ETH_MII_RX_DV/ETH_RMII_CRS_DV
	PA8	MCO1	TIM1_CH1	-	-	I2C3_SCL	-	-	USART1_CK	-	-	OTG_FS_SOF	-
	PA9	-	TIM1_CH2	-	-	I2C3_SMBA	-	-	USART1_TX	-	-	-	-

2. A GPIOA PA0 láb alternate function beállítása

- LL_GPIO_Init függvény példa

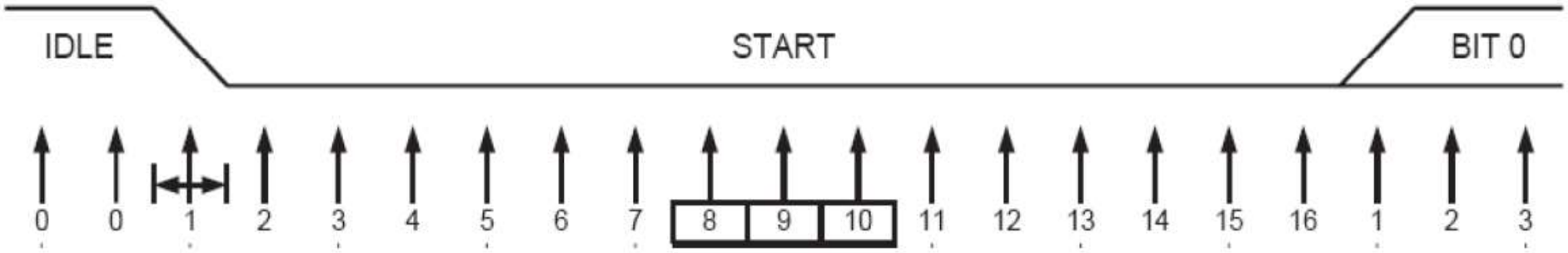
```
GPIO_InitStruct.Mode = LL_GPIO_MODE_ALTERNATE;
GPIO_InitStruct.OutputType = LL_GPIO_OUTPUT_PUSHPULL;
GPIO_InitStruct.Speed = LL_GPIO_SPEED_FREQ_HIGH;
GPIO_InitStruct.Pin = LL_GPIO_PIN_9;
GPIO_InitStruct.Alternate = LL_GPIO_AF_7;
LL_GPIO_Init  (GPIOA,&GPIO_InitStruct);

GPIO_InitStruct.OutputType = LL_GPIO_MODE_INPUT;
GPIO_InitStruct.Pin = LL_GPIO_PIN_10;
LL_GPIO_Init  (GPIOA,&GPIO_InitStruct);
```

3. USART1 inicializáció: 9600 baud, 8 data bit 0 parity 1 stop bit

- A firmware library előnye, hogy nem szükséges kiszámolnunk a Baudrate regiszter előosztóját

- A Baudrate regiszter tartalmazza a periféria órajel leosztásához szükséges értéket, hogy az UART mintavételező frekvenciát előállítsuk
  - Meg kellene állapítani a System Clock értékét, a periféria busz AHB2 órajel osztájának értékét stb.



3. USART1 inicializáció: 9600 baud, 8 data bit 0 parity 1 stop bit

- Firmware Library függvény: USART modul

```
LL_USART_InitTypeDef  USART_InitStruct;

USART_InitStruct.BaudRate = 9600;
USART_InitStruct.DataWidth = LL_USART_DATAWIDTH_8B;
USART_InitStruct.HardwareFlowControl = LL_USART_HWCONTROL_NONE;
USART_InitStruct.OverSampling = LL_USART_OVERSAMPLING_16;
USART_InitStruct.Parity = LL_USART_PARITY_NONE;
USART_InitStruct.StopBits = LL_USART_STOPBITS_1;
USART_InitStruct.TransferDirection = LL_USART_DIRECTION_TX_RX;
LL_USART_Init  (USART1, &USART_InitStruct);
```

4. USART1 aszinkron mód és engedélyezés

- Firmware Library USART modul
  - Advanced configuration services (bonyolult USART sokfajta működési mód, emiatt szükséges)

```
LL_USART_ConfigAsyncMode(USART1);
```

  - Configuration services

```
LL_USART_Enable(USART1);
```

5. Karakterek küldése

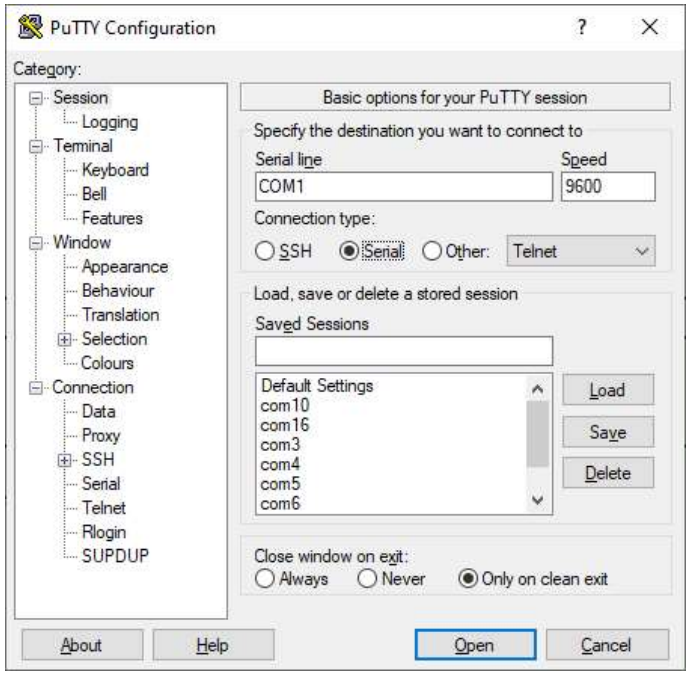
- Firmware Library UART modul
  - Data management

```
LL_USART_TransmitData8  (USART1,'a');
```

Karakterek fogadása a PC-n

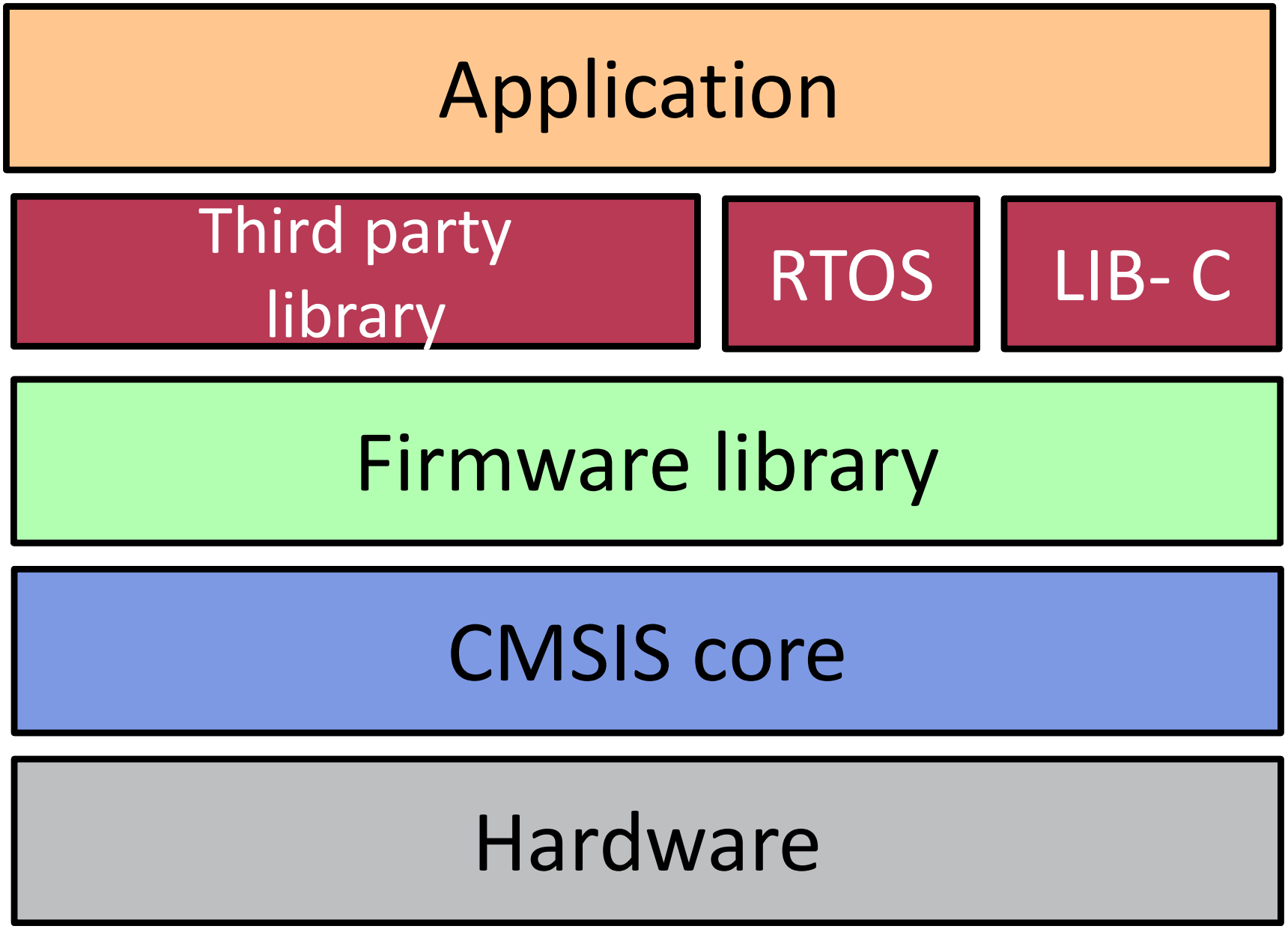
- Virtuális sorosport azonosítása: Win10: DeviceManager / Ports

- Karakterek fogadása: valamilyen terminál program pl. Putty



A LIB-C elhelyezkedése a szoftver architektúrában

UART printf


Application
Third party library
RTOS
LIB- C
Firmware library
CMSIS core
Hardware

UART printf

- Syscall minimal porting
  - iprintf, printf különbségek

```
__attribute__((weak)) int _read(int file, char *ptr, int len)
{
    int DataIdx;

    for (DataIdx = 0; DataIdx < len; DataIdx++)
    {
        *ptr++ = __io_getchar();
    }

return len;
}

__attribute__((weak)) int _write(int file, char *ptr, int len)
{
    int DataIdx;

    for (DataIdx = 0; DataIdx < len; DataIdx++)
    {
        __io_putchar(*ptr++);
    }
    return len;
}
```

UART printf

- IO_putchar átírása
  - Biztosítani kell, hogy minden byte kiíródjon és ne íródjon felül
  - Nincs külön buffer, pl erre is lesz jó a DMA

```
int __io_putchar(int ch)
{
    while(LL_USART_IsActiveFlag_TXE(USART1) == 0);

    LL_USART_TransmitData8  (USART1,ch);


    return 0;
}
```

Néhány trükk 1.

- __DATE__ és __TIME__ szimbólumok
  - A fordítás idejét tartalmazzák
  - Nagyon hasznos a hardware indulásánál kiírni, mert információt ad a szoftver frissességéről, sok problémától megkímélheti az embert.
  - Csak akkor frissül, ha az adott file újra fordul (vigyázni kell, mert ha nem változtatjuk az adott file-t akkor a linkelésnél a régi .o file-ból fog dolgozni a fordító, ezért Re-Build-et érdemes használni)

```
printf(__DATE__ " " __TIME__ "\r\n");
```

Macro Expansion

"Mar 10 2022"

Néhány trükk 2.

- Kiírás színezése
  - A legtöbb terminál képes a fogadására, Putty biztosan
  - Nagyon hasznos a folyamatos debugg logoknál a lényeg kiemelésére
  - Pl: https://stackoverflow.com/questions/4842424/list-of-ansi-color-escape-sequences

```
#define ESCAPE_NORM    "\033[0m"

#define ESCAPE_BLACK   "\033[30m"
#define ESCAPE_RED     "\033[31m"
#define ESCAPE_GREEN   "\033[32m"
#define ESCAPE_YELLOW  "\033[33m"
#define ESCAPE_BLUE    "\033[34m"
#define ESCAPE_MAGENTA "\033[35m"
#define ESCAPE_CYAN    "\033[36m"
#define ESCAPE_WHITE   "\033[37m"
```

```
#define ESCAPE_BBLACK   "\033[90m"
#define ESCAPE_BRED     "\033[91m"
#define ESCAPE_BGREEN   "\033[92m"
#define ESCAPE_BYELLOW  "\033[93m"
#define ESCAPE_BBLUE    "\033[94m"
#define ESCAPE_BMAGENTA "\033[95m"
#define ESCAPE_BCYAN    "\033[96m"
#define ESCAPE_BWHITE   "\033[97m"
```