Hello all,

I have a problem with making USB mass storage device, based on stm32f429 + 4Gb eMMC (this is only small fraction of complete project).

SDIO init works,
eMMC init works,
functions ReadBlock, ReadMultiBlocks, WriteBlock, WriteMultiBlocks works,
bus width = 8 works,
I can read and write to eMMC in firmware, but now I want to implement USB MSD driver.

I used the CubeMX to generate sample code (framework) for USB init and I changed functions STORAGE_GetCapacity_FS, STORAGE_Read_FS, STORAGE_Write_FS in usbd_storage_if.c file.

In the future I would like to use FatFS on eMMC. Before implementing FatFS in firmware, I would first like to be able to use eMMC like an external drive in MS Windows system.

At the moment, when I connect the usb cable to my device and pc, the pc recognizes the external disk, I can see it in Control panel and the size looks ok.
Problem is, that **Windows cannot use the device as it is not properly formatted**.

I tried to format the device through Windows but it fails:

```
DISKPART> format fs=ntfs label=CV

  100 percent completed
DiskPart has encountered an error: The parameter is incorrect.
See the System Event Log for more information.f
```

Here is also part of code which I think could clarify my setup:

SystemClock_Config();

```
void SystemClock_Config(void)
{
    RCC_OscInitTypeDef RCC_OscInitStruct;
    RCC_ClkInitTypeDef RCC_ClkInitStruct;
    RCC_PeriphCLKInitTypeDef PeriphClkInitStruct;

    /* Enable Power Control clock */
    __HAL_RCC_PWR_CLK_ENABLE();

    /* The voltage scaling allows optimizing the power consumption when the device is
     clocked below the maximum system frequency, to update the voltage scaling value
     regarding system frequency refer to product datasheet.  */
    __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE1);

    RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE|RCC_OSCILLATORTYPE_LSE;
    RCC_OscInitStruct.HSEState = RCC_HSE_ON;
    RCC_OscInitStruct.LSEState = RCC_LSE_ON;
    RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
    RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;

    RCC_OscInitStruct.PLL.PLLM = 14;
    RCC_OscInitStruct.PLL.PLLN = 196;
    RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
    RCC_OscInitStruct.PLL.PLLQ = 7;

    if(HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK) while(1);

    RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_SYSCLK|RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
    RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
    RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
    RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV4;
```

```
    RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV4;       // for eMMC bus clock
    HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_5);

    PeriphClkInitStruct.PeriphClockSelection = RCC_PERIPHCLK_RTC;
    PeriphClkInitStruct.RTCClockSelection = RCC_RTCCLKSOURCE_LSE;
    HAL_RCCEx_PeriphCLKConfig(&PeriphClkInitStruct);

    HAL_RCC_MCOConfig(RCC_MCO1, RCC_MCO1SOURCE_HSE, RCC_MCODIV_2);

    HAL_SYSTICK_Config(HAL_RCC_GetHCLKFreq()/1000);

    HAL_SYSTICK_CLKSourceConfig(SYSTICK_CLKSOURCE_HCLK);
}
```

RTC_Init();

```
void RTC_Init(void)
{
    HAL_StatusTypeDef status;

    RTC_TimeTypeDef sTime;
    RTC_DateTypeDef sDate;

    /* Enable LSE crystal */
    __HAL_RCC_LSE_CONFIG(RCC_LSE_ON);

    /* Power clock must be enabled */
    __HAL_RCC_PWR_CLK_ENABLE();

    /* Enable access to backup domain */
    HAL_PWR_EnableBkUpAccess();

    /* RTC Clock Source Selection */
    __HAL_RCC_RTC_CONFIG(RCC_RTCCLKSOURCE_LSE);

    /* Enable the RTC */
    __HAL_RCC_RTC_ENABLE();

    /* Unlock RTC Registers */
    __HAL_RTC_WRITEPROTECTION_DISABLE(&hrtc);

    /**Initialize RTC and set the Time and Date */
    hrtc.Instance = RTC;
    hrtc.Init.HourFormat = RTC_HOURFORMAT_24;
    hrtc.Init.AsynchPrediv = 31;//0;//127;
    hrtc.Init.SynchPrediv = 1023;//31;//255;
    hrtc.Init.OutPut = RTC_OUTPUT_DISABLE;
    hrtc.Init.OutPutPolarity = RTC_OUTPUT_POLARITY_HIGH;
    hrtc.Init.OutPutType = RTC_OUTPUT_TYPE_OPENDRAIN;
    status = HAL_RTC_Init(&hrtc);

    sTime.Hours = 0;
    sTime.Minutes = 0;
    sTime.Seconds = 0;
    sTime.SubSeconds = 0;
    sTime.TimeFormat = RTC_HOURFORMAT_24;
    sTime.DayLightSaving = RTC_DAYLIGHTSAVING_NONE;
    sTime.StoreOperation = RTC_STOREOPERATION_RESET;
    HAL_RTC_SetTime(&hrtc, &sTime, FORMAT_BCD);

    sDate.WeekDay = RTC_WEEKDAY_MONDAY;
    sDate.Month = RTC_MONTH_JANUARY;
    sDate.Date = 1;
    sDate.Year = 0;
    HAL_RTC_SetDate(&hrtc, &sDate, FORMAT_BCD);

    HAL_RTC_WaitForSynchro(&hrtc);
```

```
    /*** setup wakeup timer ***/
    /* Disable Wakeup Counter */
    HAL_RTCEx_DeactivateWakeUpTimer(&hrtc);

    /* Enable the RTC Interrupt */
    /* Peripheral interrupt init*/
    HAL_NVIC_SetPriority(RTC_WKUP_IRQn, 1, 0);
    HAL_NVIC_EnableIRQ(RTC_WKUP_IRQn);

    HAL_RTCEx_SetWakeUpTimer_IT(&hrtc, 0 , RTC_WAKEUPCLOCK_CK_SPRE_16BITS );
}
```

Sdio_Init();

```
void Sdio_Init(void)
{
    GPIO_InitTypeDef  GPIO_InitStruct;

    //GPIOB, GPIOC and GPIOD Periph clock enable
    __HAL_RCC_GPIOB_CLK_ENABLE();
    __HAL_RCC_GPIOC_CLK_ENABLE();
    __HAL_RCC_GPIOD_CLK_ENABLE();

    /*  CLK pin: */
    GPIO_InitStruct.Pin         = GPIO_PIN_12;
    GPIO_InitStruct.Mode        = GPIO_MODE_AF_PP;
    GPIO_InitStruct.Pull        = GPIO_NOPULL;
    GPIO_InitStruct.Speed       = GPIO_SPEED_FAST;
    GPIO_InitStruct.Alternate   = GPIO_AF12_SDIO;
    HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);

    /*  CMD pin: */
    GPIO_InitStruct.Pin         = GPIO_PIN_2;
    GPIO_InitStruct.Mode        = GPIO_MODE_AF_PP;
    GPIO_InitStruct.Pull        = GPIO_PULLUP;
    GPIO_InitStruct.Speed       = GPIO_SPEED_FAST;
    GPIO_InitStruct.Alternate   = GPIO_AF12_SDIO;
    HAL_GPIO_Init(GPIOD, &GPIO_InitStruct);

    /* DATA pins: */
    GPIO_InitStruct.Pin         = GPIO_PIN_6 | GPIO_PIN_7 | GPIO_PIN_8 | GPIO_PIN_9 | GPIO_PIN_10 |
GPIO_PIN_11;
    GPIO_InitStruct.Mode        = GPIO_MODE_AF_PP;
    GPIO_InitStruct.Pull        = GPIO_NOPULL;
    GPIO_InitStruct.Speed       = GPIO_SPEED_FAST;
    GPIO_InitStruct.Alternate   = GPIO_AF12_SDIO;
    HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);

    GPIO_InitStruct.Pin         = GPIO_PIN_8 | GPIO_PIN_9;
    GPIO_InitStruct.Mode        = GPIO_MODE_AF_PP;
    GPIO_InitStruct.Pull        = GPIO_NOPULL;
    GPIO_InitStruct.Speed       = GPIO_SPEED_FAST;
    GPIO_InitStruct.Alternate   = GPIO_AF12_SDIO;
    HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

    /* Enable the SDIO APB2 Clock */
    __HAL_RCC_SDIO_CLK_ENABLE();
    /* Enable the DMA2 Clock */
    __HAL_RCC_DMA2_CLK_ENABLE();

    /* SDIO Reinit */
    __HAL_RCC_SDIO_FORCE_RESET();
    __HAL_RCC_SDIO_RELEASE_RESET();

    /* Power ON Sequence -----------------------------------*/
    /* Configure the SDIO peripheral                        */
    /* SDIO_CK = SDIOCLK / (INIT_CLK_DIV + 2)               */
    /* on STM32F4xx devices, SDIOCLK is fixed to 48MHz      */
    /* SDIO_CK for initialization should not exceed 400 KHz */
```

```
    HSD_Handler.Instance                = SDIO;
    HSD_Handler.Init.ClockEdge          = SDIO_CLOCK_EDGE_RISING;
    HSD_Handler.Init.ClockBypass        = SDIO_CLOCK_BYPASS_DISABLE;
    HSD_Handler.Init.ClockPowerSave     = SDIO_CLOCK_POWER_SAVE_DISABLE;
    HSD_Handler.Init.BusWide            = SDIO_BUS_WIDE_1B;
    HSD_Handler.Init.HardwareFlowControl = SDIO_HARDWARE_FLOW_CONTROL_DISABLE;
    HSD_Handler.Init.ClockDiv           = (uint8_t)0xFF;

    SDIO_Init(SDIO, HSD_Handler.Init);

    SDIO_PowerState_ON(SDIO);
    __SDIO_ENABLE();

    NVIC_Configuration();
}
```

eMMC_Init();

```
void eMMC_Init(void)
{
    uint32_t SdioStatus;
    //__HAL_SD_SDIO_CLEAR_FLAG(SDIO, SDIO_STA_STATBITS);
    __SDIO_CLEAR_FLAG(SDIO, SDIO_STA_STATBITS);
    //CMD0: GO_IDLE_STATE ----------------------------------------------------
    SDIO_CmdInitStructure.Argument              = 0;
    SDIO_CmdInitStructure.CmdIndex              = GO_IDLE_STATE;
    SDIO_CmdInitStructure.Response              = SDIO_RESPONSE_NO;
    SDIO_CmdInitStructure.WaitForInterrupt      = SDIO_WAIT_NO;
    SDIO_CmdInitStructure.CPSM                  = SDIO_CPSM_ENABLE;

    SDIO_SendCommand(SDIO, &SDIO_CmdInitStructure);

    do SdioStatus = SDIO->STA;
        while (!(SdioStatus&SDIO_STA_CMDSENT));

    do
    {
        __SDIO_CLEAR_FLAG(SDIO, SDIO_STA_STATBITS);
        //CMD1: SEND_OP_COND -------------------------------------------------
        SDIO_CmdInitStructure.Argument          = 0x40FF8080;
        SDIO_CmdInitStructure.CmdIndex          = SEND_OP_COND;
        SDIO_CmdInitStructure.Response          = SDIO_RESPONSE_SHORT;
        SDIO_CmdInitStructure.WaitForInterrupt  = SDIO_WAIT_NO;
        SDIO_CmdInitStructure.CPSM              = SDIO_CPSM_ENABLE;

        SDIO_SendCommand(SDIO, &SDIO_CmdInitStructure);

        do SdioStatus = SDIO->STA;
        while (!(SdioStatus&(SDIO_STA_CCRCFAIL|SDIO_STA_CMDREND|SDIO_STA_CTIMEOUT)));
        if (SdioStatus&SDIO_STA_CTIMEOUT)
        {
            LogData.Error   = 1;
            return;
        }
    }
    while (!(SDIO_GetResponse(SDIO_RESP1)&OCR_READY));

    __SDIO_CLEAR_FLAG(SDIO, SDIO_STA_STATBITS);
    //CMD2: ALL_SEND_CID -----------------------------------------------------
    SDIO_CmdInitStructure.Argument              = 0;
    SDIO_CmdInitStructure.CmdIndex              = ALL_SEND_CID;
    SDIO_CmdInitStructure.Response              = SDIO_RESPONSE_LONG;
    SDIO_CmdInitStructure.WaitForInterrupt      = SDIO_WAIT_NO;
    SDIO_CmdInitStructure.CPSM                  = SDIO_CPSM_ENABLE;

    SDIO_SendCommand(SDIO, &SDIO_CmdInitStructure);

    do SdioStatus=SDIO->STA;
    while (!(SdioStatus&(SDIO_STA_CCRCFAIL|SDIO_STA_CMDREND|SDIO_STA_CTIMEOUT)));
```

```
if (SdioStatus&(SDIO_STA_CCRCFAIL|SDIO_STA_CTIMEOUT))
{
    LogData.Error       = 2;
    LogData.SdioStatus  = SdioStatus;
    return;
}

CID_Tab[0] = SDIO_GetResponse(SDIO_RESP1);
CID_Tab[1] = SDIO_GetResponse(SDIO_RESP2);
CID_Tab[2] = SDIO_GetResponse(SDIO_RESP3);
CID_Tab[3] = SDIO_GetResponse(SDIO_RESP4);

__SDIO_CLEAR_FLAG(SDIO, SDIO_STA_STATBITS);

//CMD3: SET_REL_ADDR ----------------------------------------------------
SDIO_CmdInitStructure.Argument              = 0x0001FFFF;   //0x00; //
SDIO_CmdInitStructure.CmdIndex              = SET_REL_ADDR;
SDIO_CmdInitStructure.Response              = SDIO_RESPONSE_SHORT;
SDIO_CmdInitStructure.WaitForInterrupt      = SDIO_WAIT_NO;
SDIO_CmdInitStructure.CPSM                  = SDIO_CPSM_ENABLE;

SDIO_SendCommand(SDIO, &SDIO_CmdInitStructure);

do SdioStatus=SDIO->STA;
while (!(SdioStatus&(SDIO_STA_CCRCFAIL|SDIO_STA_CMDREND|SDIO_STA_CTIMEOUT)));
if (SdioStatus&(SDIO_STA_CCRCFAIL|SDIO_STA_CTIMEOUT))
{
    LogData.Error       = 3;
    LogData.SdioStatus  = SdioStatus;
    return;
}
uint32_t resp1 = SDIO_GetResponse(SDIO_RESP1);

if (resp1 != IDENT_AND_READY)
{
    LogData.Error       = 4;
    return;
}

uint16_t rca = (uint16_t) ( resp1 >> 16);

RCA = rca;

__SDIO_CLEAR_FLAG(SDIO, SDIO_STA_STATBITS);

//CMD9: SEND_CSD ----------------------------------------------------
rca = 0x01;

SDIO_CmdInitStructure.Argument              = (uint32_t)(rca << 16);
SDIO_CmdInitStructure.CmdIndex        = SEND_CSD;
SDIO_CmdInitStructure.Response        = SDIO_RESPONSE_LONG;
SDIO_CmdInitStructure.WaitForInterrupt    = SDIO_WAIT_NO;
SDIO_CmdInitStructure.CPSM          = SDIO_CPSM_ENABLE;

SDIO_SendCommand(SDIO, &SDIO_CmdInitStructure);

do SdioStatus=SDIO->STA;
while (!(SdioStatus&(SDIO_STA_CCRCFAIL|SDIO_STA_CMDREND|SDIO_STA_CTIMEOUT)));


if (SdioStatus&(SDIO_STA_CCRCFAIL|SDIO_STA_CTIMEOUT))
{
    LogData.Error       = 5;
    LogData.SdioStatus=SdioStatus;
    return;
}
```

```
    CSD_Tab[0] = SDIO_GetResponse(SDIO_RESP1);
    CSD_Tab[1] = SDIO_GetResponse(SDIO_RESP2);
    CSD_Tab[2] = SDIO_GetResponse(SDIO_RESP3);
    CSD_Tab[3] = SDIO_GetResponse(SDIO_RESP4);

    __SDIO_CLEAR_FLAG(SDIO, SDIO_STA_STATBITS);

    //CMD7: SEL_DESEL_CARD ------------------------------------------------
    SDIO_CmdInitStructure.Argument              = 0x0001FFFF;
    SDIO_CmdInitStructure.CmdIndex              = SEL_DESEL_CARD;
    SDIO_CmdInitStructure.Response              = SDIO_RESPONSE_SHORT;
    SDIO_CmdInitStructure.WaitForInterrupt      = SDIO_WAIT_NO;
    SDIO_CmdInitStructure.CPSM                  = SDIO_CPSM_ENABLE;

    SDIO_SendCommand(SDIO, &SDIO_CmdInitStructure);

    do SdioStatus = SDIO->STA;
    while (!(SdioStatus&(SDIO_STA_CCRCFAIL|SDIO_STA_CMDREND|SDIO_STA_CTIMEOUT)));
    if (SdioStatus&(SDIO_STA_CCRCFAIL|SDIO_STA_CTIMEOUT))
    {
        LogData.Error       = 7;
        LogData.SdioStatus  = SdioStatus;
        return;
    }

    __SDIO_CLEAR_FLAG(SDIO, SDIO_STA_STATBITS);
    //CMD13: SEND_STATUS --------------------------------------------------
    SDIO_CmdInitStructure.Argument              = 0x0001FFFE;
    SDIO_CmdInitStructure.CmdIndex              = SEND_STATUS;
    SDIO_CmdInitStructure.Response              = SDIO_RESPONSE_SHORT;
    SDIO_CmdInitStructure.WaitForInterrupt      = SDIO_WAIT_NO;
    SDIO_CmdInitStructure.CPSM                  = SDIO_CPSM_ENABLE;

    SDIO_SendCommand(SDIO, &SDIO_CmdInitStructure);

    do SdioStatus = SDIO->STA;
    while (!(SdioStatus&(SDIO_STA_CCRCFAIL|SDIO_STA_CMDREND|SDIO_STA_CTIMEOUT)));
    if (SdioStatus&(SDIO_STA_CCRCFAIL|SDIO_STA_CTIMEOUT))
    {
        LogData.Error       = 8;
        LogData.SdioStatus  = SdioStatus;
        return;
    }
    if (SDIO_GetResponse(SDIO_RESP1)!=TRANSFER_AND_READY)
    {
        LogData.Error       = 9;
        return;
    }

    __SDIO_CLEAR_FLAG(SDIO, SDIO_STA_STATBITS);
    LogData.Error           = 0;
}
```

eMMC_SetBusWidth(8);
Sdio_SetBusWidth(8);
Sdio_SetClock_kHz(24000);
eMMC_GetCardInfo(&SDCardInfo);

MX_USB__DEVICE_Init(void);

```
/* init function */
void MX_USB_DEVICE_Init(void)
{
  /* Init Device Library,Add Supported Class and Start the library*/
  USBD_Init(&hUsbDeviceFS, &FS_Desc, DEVICE_FS);
```

```
  USBD_RegisterClass(&hUsbDeviceFS, &USBD_MSC);

  USBD_MSC_RegisterStorage(&hUsbDeviceFS, &USBD_Storage_Interface_fops_FS);

  USBD_Start(&hUsbDeviceFS);
}
```

STORAGE_GetCapacity_FS:

```
int8_t STORAGE_GetCapacity_FS (uint8_t lun, uint32_t *block_num, uint16_t *block_size)
{
  *block_size = SDCardInfo.CardBlockSize;
  *block_num  = SDCardInfo.CardCapacity / SDCardInfo.CardBlockSize;

  return (USBD_OK);
}
```

STORAGE_Read_FS:

```
int8_t STORAGE_Read_FS (uint8_t lun,  uint8_t *buf,  uint32_t blk_addr,  uint16_t blk_len)
{
     eMMC_ReadMultiBlocks(buf, blk_addr, blk_len);

    return (USBD_OK);
}
```

STORAGE_Write_FS:

```
int8_t STORAGE_Write_FS (uint8_t lun, uint8_t *buf, uint32_t blk_addr, uint16_t blk_len)
{

    eMMC_WriteMultiBlocks(buf, blk_addr, blk_len);

    return (USBD_OK);
}
```

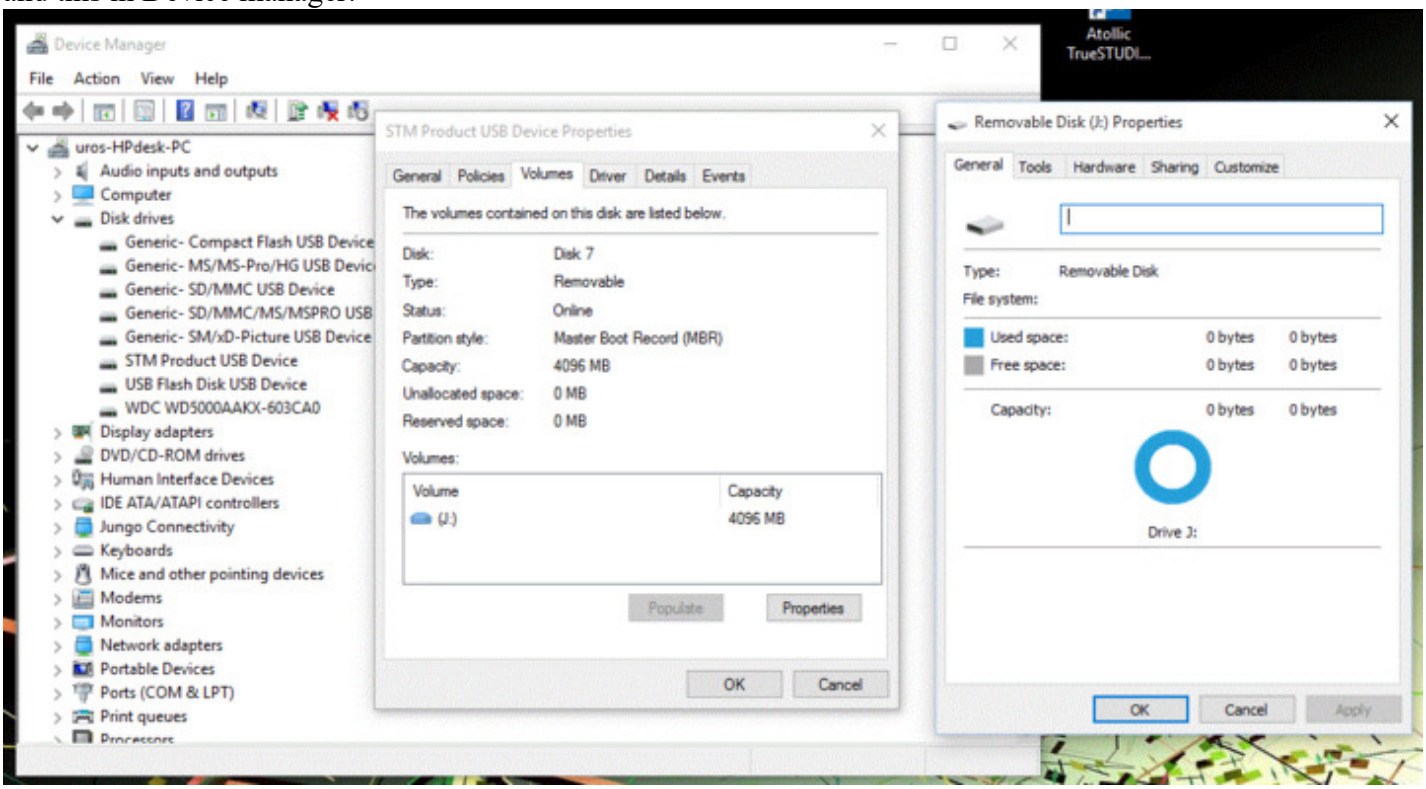
I would be really happy to get any suggestions and if there exists something on this problem on the forum (I could not find it), please redirect me there.

Also let me know, if I posted this in wrong section of forum (this is my first post).

Also, I was able to get this info with DISKPART:

```
DISKPART> list disk

  Disk ###  Status         Size     Free     Dyn  Gpt
  --------  -------------  -------  -------  ---  ---
  Disk 0    Online          465 GB      0 B
  Disk 1    Online          124 MB      0 B
  Disk 2    No Media           0 B      0 B
  Disk 3    No Media           0 B      0 B
  Disk 4    No Media           0 B      0 B
  Disk 5    No Media           0 B      0 B
  Disk 6    No Media           0 B      0 B
* Disk 7    Online         4096 MB      0 B

DISKPART> list volume

  Volume ###  Ltr  Label        Fs     Type        Size     Status     Info
  ----------  ---  -----------  -----  ----------  -------  ---------  --------
  Volume 0     D                       DVD-ROM         0 B  No Media
  Volume 1         System Rese  NTFS   Partition    100 MB  Healthy    System
  Volume 2     C                NTFS   Partition    465 GB  Healthy    Boot
  Volume 3                      NTFS   Partition    450 MB  Healthy    Hidden
  Volume 4     K   0000000108   FAT    Removable    123 MB  Healthy
  Volume 5     E                       Removable       0 B  No Media
  Volume 6     F                       Removable       0 B  No Media
  Volume 7     G                       Removable       0 B  No Media
  Volume 8     H                       Removable       0 B  No Media
  Volume 9     I                       Removable       0 B  No Media
* Volume 10    J                RAW    Removable   4096 MB  Healthy

DISKPART>
```

and this in Device manager:



Best regards, Uros