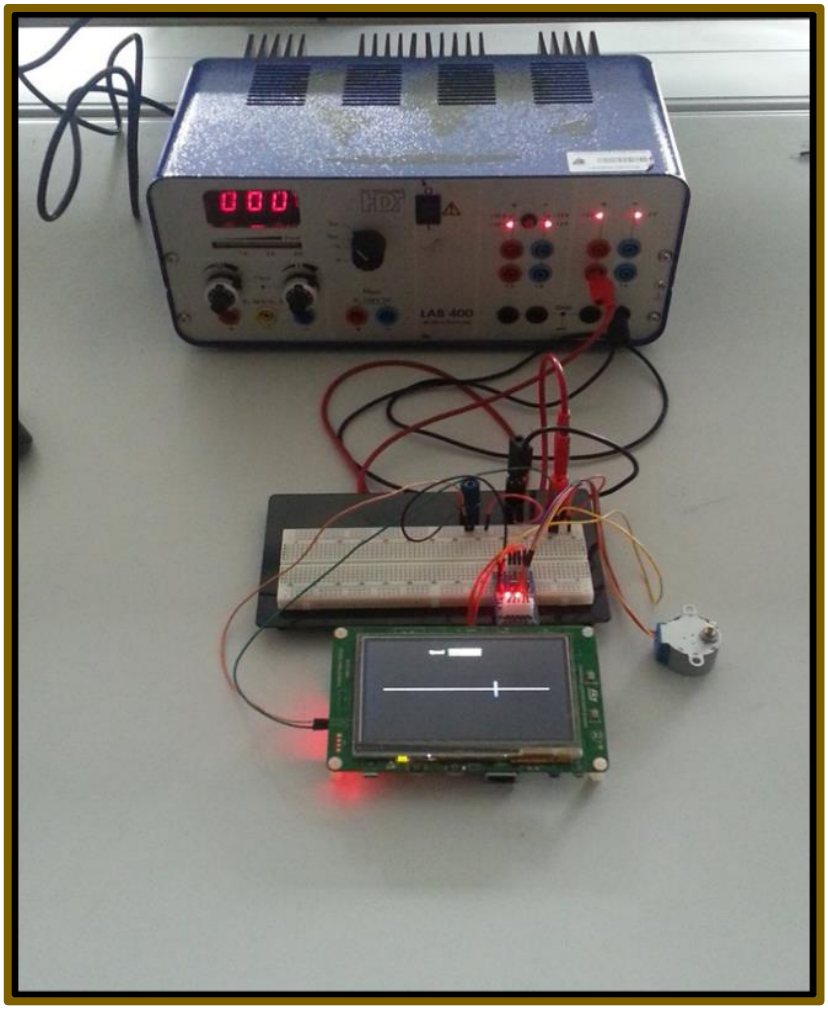
Informationstechnik Labor

SOSE 16

Thema 08: Steuerung eines Schrittmotors mit STM32F74

Prof.J.Walter



Team :

Felipe Fuentes
Khachani Yahya

Matrikelnummer:

39390
43508

Inhalt

Problemstellung:

Mithilfe von Kameraschienen werden bewegte und qualitativ hochwertige Aufnahmen gemacht, dabei wird die Kamera von einem Motor entlang der Schiene bewegt, im Rahmen des Informationstechnik Labors wird mittels einem STM32F746 der Motor entweder nach rechts oder links gedreht, diese ermöglichst wiederum die Translatorische Bewegung der ZaSchie Schiene.

Stand der Technik:

Abbildung 1: Stepper Motor

Abbildung 2: Cinetics Three Axis360 Pro

Im Internet gibt es sämtliche technische Arten der Steuerung eines Schrittmotors entweder Kabellos durch Wlan oder durch ein App wie zum Beispiel das T-Set von Axsy oder mittels jeglicher Mikrocontrollers. In unserem Fall soll die Steuerung der Kameraschiene über einen Mikrocontroller erfolgen, der auf dem Namen STM32F7 von STMicroelectronics hört.

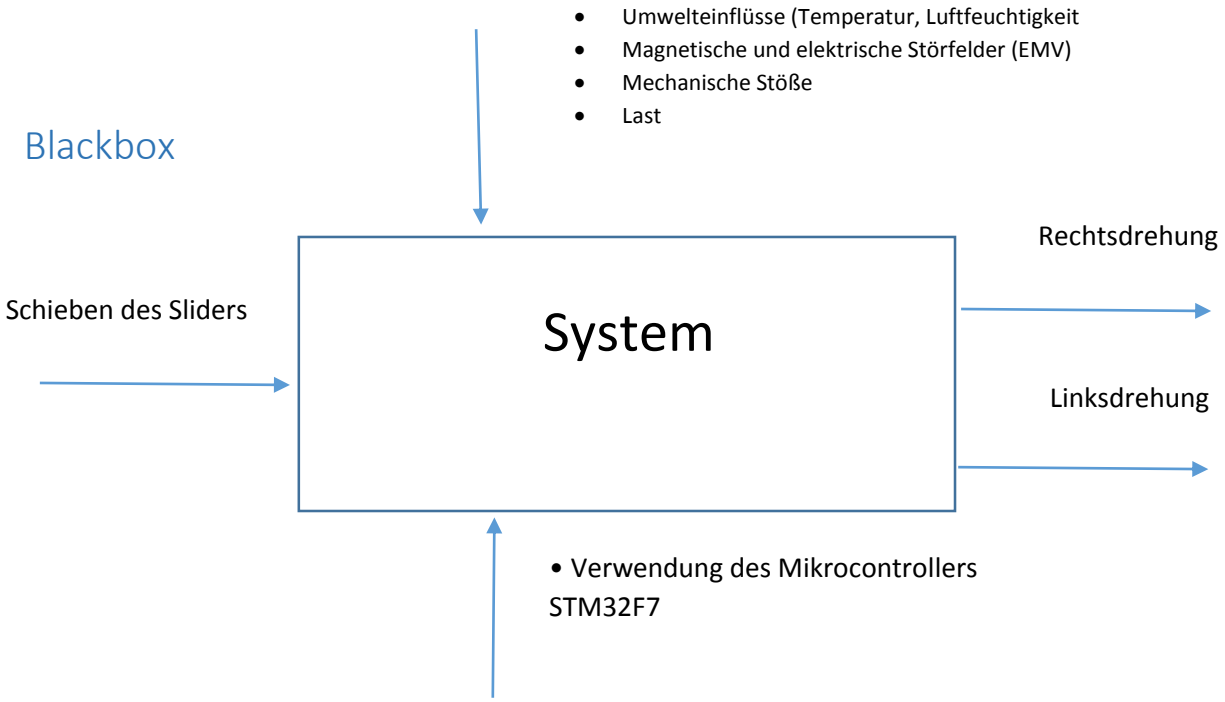
Aufgabenstellung:

Es soll ein C-Code geschrieben werden, mit der ein Schrittmotor über STM32F746 gesteuert werden kann.

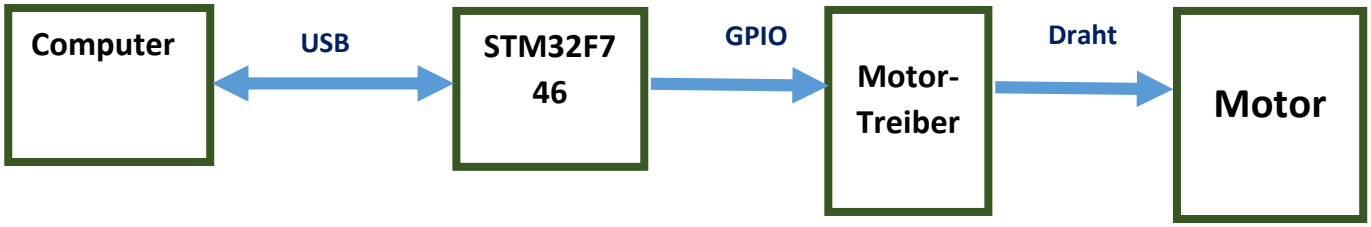
Anforderungsliste:

Hochschule Karlsruhe Technik und Wirtschaft				Anforderungsliste für **Steuerung eines Schrittmotors mit STM32F746**	**Auftrag:** **Formblatt-AnfListe** **Auftragsnummer** SS 2016			
Organisations-Daten		**Prozess-Daten**		**Anforderungen**	**Wert - Daten**			
Nummer	Name	Art	Phase		Mindest-Erfüllung	SOLL-Erfüllung	Ideal-Erfüllung	Einheit
				Physikalisch-Technische Funktion				
F01		F		Maximale einstellbare Frequenz	50	50-100	50 bis 90	Hz
F02		F		Einstellbare Drehrichtung	Links Rechts	Links Rechts	Links Rechts	
F03		W		Spannungsversorgung durch eine Spannungsquelle	5V	5V	5V	
				Technologie				
T01		J/N		Steuerung über GUI				
T02		J/N		STM32F746				
T03		J/N		Schrittmotor-Set S-SPSM- 5V				
				Wirtschaftlichkeit				
W01		F		Kostengünstig				
				Mensch-Produkt-Beziehungen				
M01		F		Graphischer Benutzerschnittstelle „Knopf“				
				Software				
S01		F		Keil µVision				
S02		F		C-Programmiersprache				
Anforderungsarten: J/N - Ja/Nein; F - Forderung; W - Wunsch;				***Konstruktionsphase:*** P - Prinzip; K - Konzept; E - Entwurf; A - Ausarbeitung				
Ersetzt Ausgabe vom **14.04.2016** Version: **6** Name:					**Ausgabe:** **22.04.2016** **Version:** **7** **Blatt 1 von 1**			

Blackbox



Blockschaltbild



Zeitplan

Informationstechnik Labor Fakultät MMT Hochschule Karlsruhe Technik und Wirtschaft		**Zeitplan** für **Titel**							**MTB732** Sommersemester 2016 Datum
Nr.	**Aktivitäten**	2016							**Jahr**
		März			April			Mai	**Monat**
		11	12	13	14	15	16	17	18	**Kalenderwoche**
1	Projektdefinition erstellen	X								
2	Allgemeine Recherche	X	X							
3	Anforderungsliste		X							
4	Blackbox		X							
6	Programmierung der GUI für den STM M32F746			X	X	X	X			
8	Aufbau und testen						X	X		
9	Abschlusspräsentation								X	
		Meilensteine: 1 2							Präsentation: P P	

Abbildung 3: Zeitplan

Stückliste:

Bauteilbezeichnung	Erläuterung
STM32F746	Vorhanden
Schrittmotor-Set S-SPSM- 5V	Vorhanden
Breadboard	Vorhanden

Versuchsaufbau:

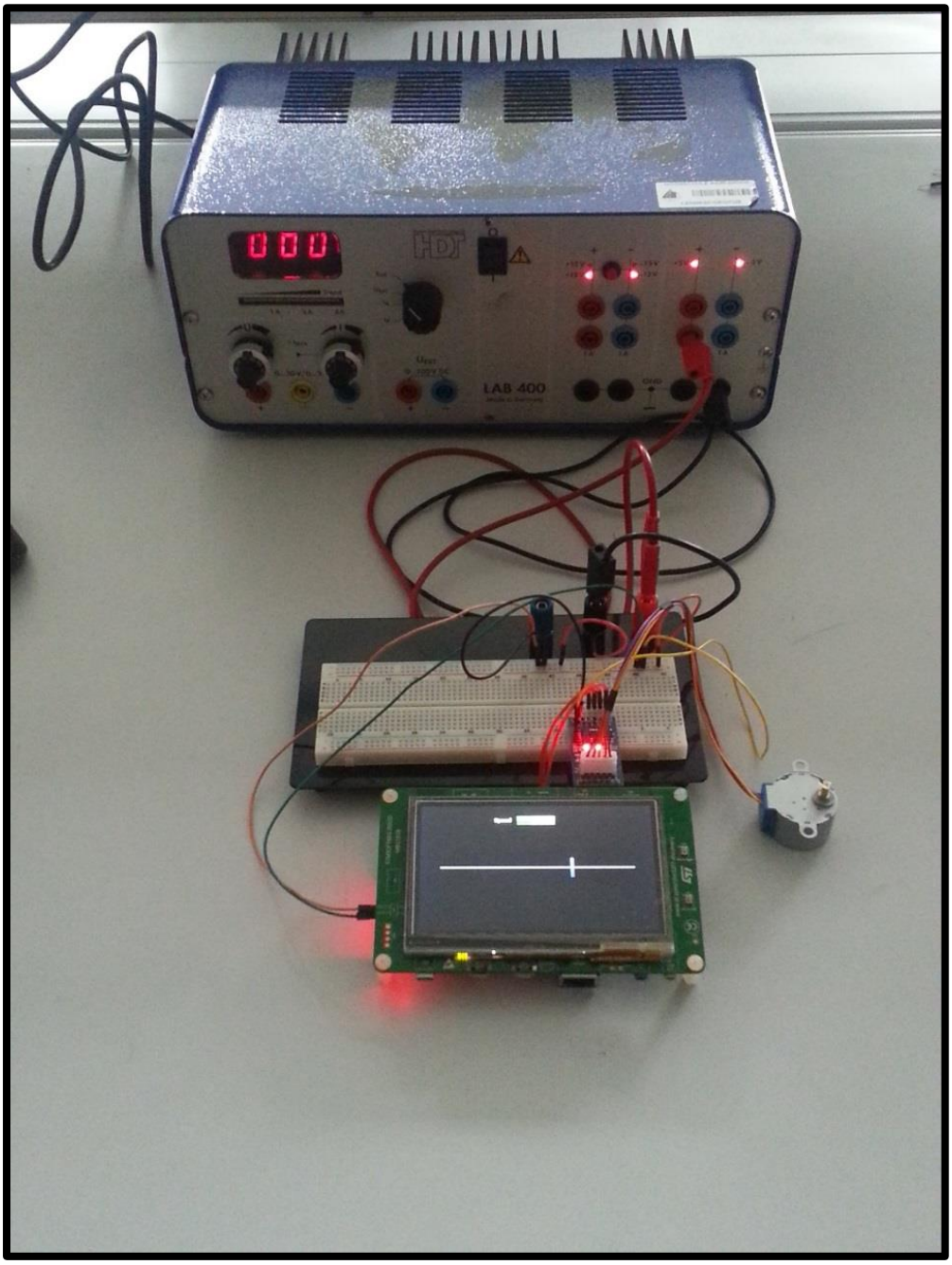
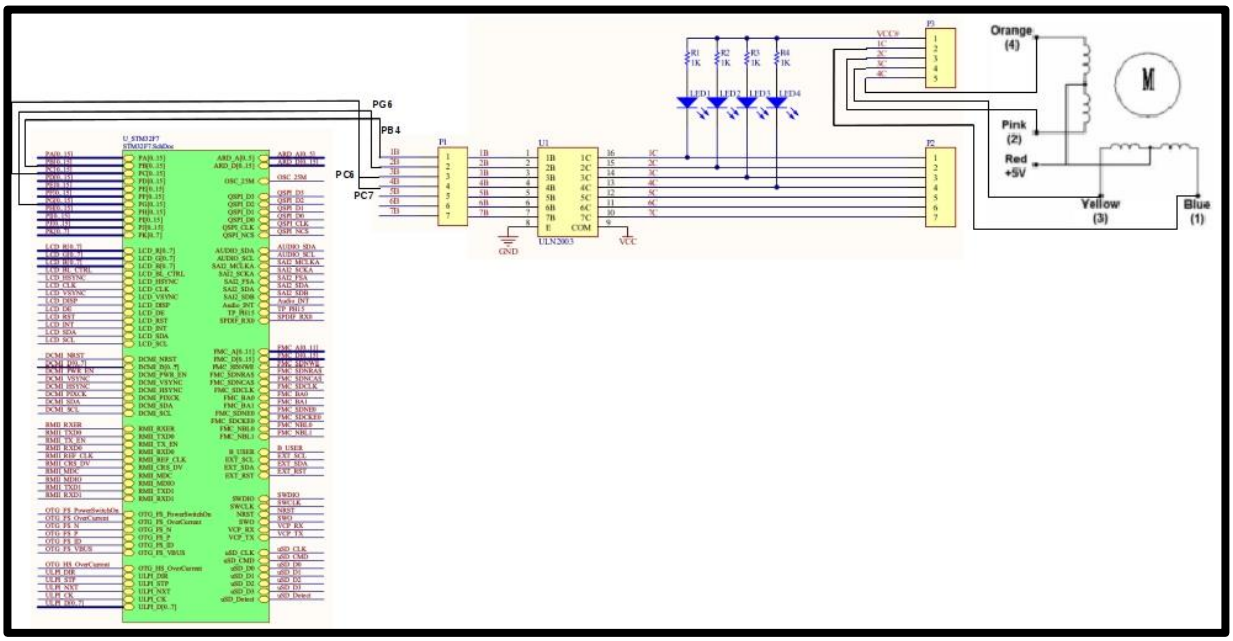
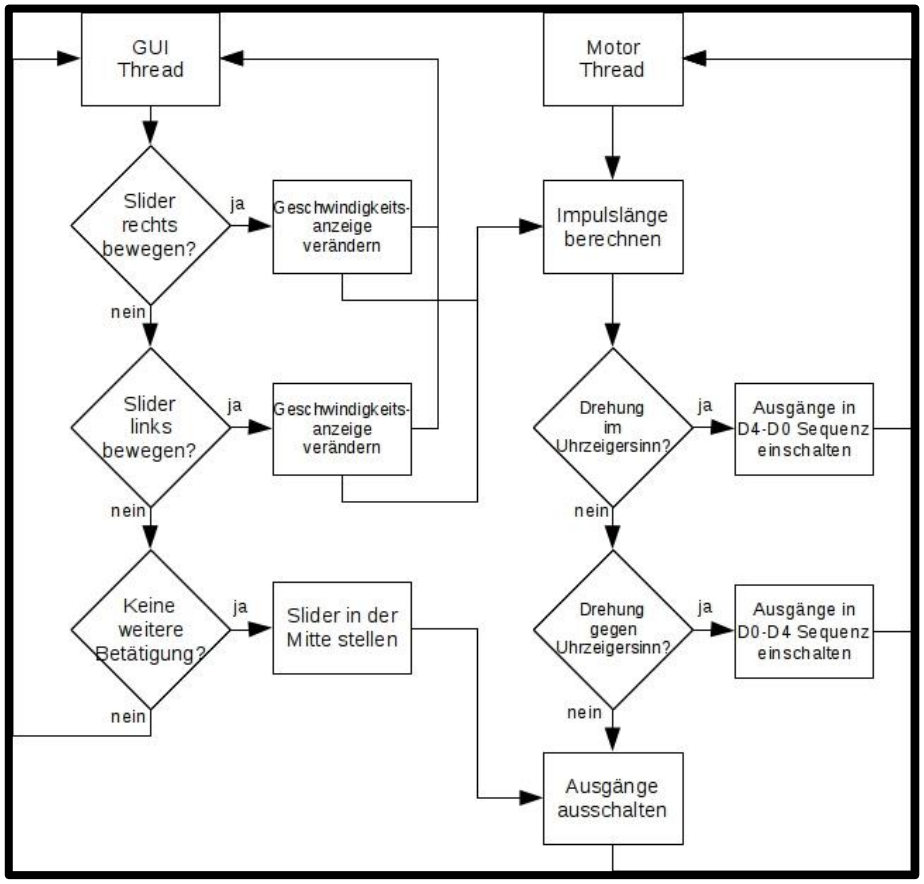


Abbildung 4: Versuchsaufbau

Schaltplan:



Flussdiagramm:



Quellcode:

1. **Gui_Single Thread**:

```
#include "cmsis_os.h"                   // CMSIS RTOS header file
#include "GUI.h"
#include "DIALOG.h"

#ifdef _RTE_
#include "RTE_Components.h"             // Component selection
#endif

extern WM_HWIN CreateWindow(void);

/*----------------------------------------------------------------------------
 *      GUIThread: GUI Thread for Single-Task Execution Model
 *---------------------------------------------------------------------------*/

void GUIThread(void const *argument);              // thread function
osThreadId tid_GUIThread;                          // thread id
osThreadDef(GUIThread, osPriorityNormal, 1, 2048);   // thread object

int Init_GUIThread (void) {

  tid_GUIThread = osThreadCreate (osThread(GUIThread), NULL);
  if (!tid_GUIThread) return(-1);

  return(0);
}

void GUIThread (void const *argument) {

  GUI_Init();                     /* Initialize the Graphics Component */

  /* Add GUI setup code here */
       CreateWindow();

  while (1) {

    /* All GUI related activities might only be called from here */

#ifdef RTE_Graphics_Touchscreen   /* Graphics Input Device Touchscreen enabled */
    GUI_TOUCH_Exec();             /* Execute Touchscreen support */
#endif
    GUI_Exec();                   /* Execute all GUI jobs ... Return 0 if nothing was
done. */
    GUI_X_ExecIdle();             /* Nothing left to do for the moment ... Idle
processing */
  }
}
```

2. Main_c

```
/**
  ******************************************************************************
  * @file    Templates/Src/main.c
  * @author  MCD Application Team
  * @version V1.0.2
  * @date    18-November-2015
  * @brief   STM32F7xx HAL API Template project (with modifications from ARM)
  ******************************************************************************
  * @attention
  *
  * <h2><center>&copy; COPYRIGHT 2015 STMicroelectronics</center></h2>
  *
  * Redistribution and use in source and binary forms, with or without modification,
  * are permitted provided that the following conditions are met:
  *   1. Redistributions of source code must retain the above copyright notice,
  *      this list of conditions and the following disclaimer.
  *   2. Redistributions in binary form must reproduce the above copyright notice,
  *      this list of conditions and the following disclaimer in the documentation
  *      and/or other materials provided with the distribution.
  *   3. Neither the name of STMicroelectronics nor the names of its contributors
  *      may be used to endorse or promote products derived from this software
  *      without specific prior written permission.
  *
  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
  * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
  * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
  * DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
  * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
  * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
  * CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
  * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
  * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
  *
  ******************************************************************************
  */

/*
 * This file contains modifications by ARM to provide it as User Code Template
 * within the STM32 Device Family Pack.
 */

/* Includes ------------------------------------------------------------------*/
#include "main.h"
#include "stm32746g_discovery_sdram.h"  // Keil.STM32F746G-Discovery::Board
Support:Drivers:SDRAM

#ifdef _RTE_
#include "RTE_Components.h"             // Component selection
#endif
#ifdef RTE_CMSIS_RTOS                   // when RTE component CMSIS RTOS is used
#include "cmsis_os.h"                   // CMSIS RTOS header file
```

```
#endif

#ifdef RTE_CMSIS_RTOS_RTX
extern uint32_t os_time;

uint32_t HAL_GetTick(void) {
  return os_time;
}
#endif

/** @addtogroup STM32F7xx_HAL_Examples
  * @{
  */

/** @addtogroup Templates
  * @{
  */

/* Private typedef -----------------------------------------------------------*/
/* Private define ------------------------------------------------------------*/
/* Private macro -------------------------------------------------------------*/
/* Private variables ---------------------------------------------------------*/
/* Private function prototypes -----------------------------------------------*/
static void SystemClock_Config(void);
static void Error_Handler(void);
static void MPU_Config(void);
static void CPU_CACHE_Enable(void);
static void GPIO_Init(void);
extern int Init_GUIThread(void);
extern int Init_MotorThread(void);

/* Private functions ---------------------------------------------------------*/

/**
  * @brief  Main program
  * @param  None
  * @retval None
  */
int main(void)
{
  /* This project template calls firstly two functions in order to configure MPU
feature
     and to enable the CPU Cache, respectively MPU_Config() and CPU_CACHE_Enable().
     These functions are provided as template implementation that User may integrate
     in his application, to enhance the performance in case of use of AXI interface
     with several masters. */

  /* Configure the MPU attributes as Write Through */
  MPU_Config();

  /* Enable the CPU Cache */
  CPU_CACHE_Enable();

#ifdef RTE_CMSIS_RTOS                   // when using CMSIS RTOS
  osKernelInitialize();                 // initialize CMSIS-RTOS
#endif

  /* STM32F7xx HAL library initialization:
       - Configure the Flash ART accelerator on ITCM interface
```

```
       - Configure the Systick to generate an interrupt each 1 msec
       - Set NVIC Group Priority to 4
       - Low Level Initialization
     */
  HAL_Init();

  /* Configure the System clock to have a frequency of 200 MHz */
  SystemClock_Config();


  /* Add your application code here
     */
       // Initialize user GPIO's
       GPIO_Init();

       // Board SDRAM initialize
       BSP_SDRAM_Init();

#ifdef RTE_CMSIS_RTOS                   // when using CMSIS RTOS
  // create 'thread' functions that start executing,
  // example: tid_name = osThreadCreate (osThread(name), NULL);
       Init_GUIThread();
       Init_MotorThread();
  osKernelStart();                      // start thread execution
#endif

  /* Infinite loop */
  while (1)
  {
              osDelay(100);
  }
}

/**
  * @brief  System Clock Configuration
  *         The system Clock is configured as follow :
  *            System Clock source            = PLL (HSE)
  *            SYSCLK(Hz)                     = 200000000
  *            HCLK(Hz)                       = 200000000
  *            AHB Prescaler                  = 1
  *            APB1 Prescaler                 = 4
  *            APB2 Prescaler                 = 2
  *            HSE Frequency(Hz)              = 25000000
  *            PLL_M                          = 25
  *            PLL_N                          = 400
  *            PLL_P                          = 2
  *            PLLSAI_N                       = 384
  *            PLLSAI_P                       = 8
  *            VDD(V)                         = 3.3
  *            Main regulator output voltage  = Scale1 mode
  *            Flash Latency(WS)              = 6
  * @param  None
  * @retval None
  */
static void SystemClock_Config(void)
{
  RCC_ClkInitTypeDef RCC_ClkInitStruct;
  RCC_OscInitTypeDef RCC_OscInitStruct;
```

```
  /* Enable HSE Oscillator and activate PLL with HSE as source */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
  RCC_OscInitStruct.HSEState = RCC_HSE_ON;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
  RCC_OscInitStruct.PLL.PLLM = 25;
  RCC_OscInitStruct.PLL.PLLN = 400;
  RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
  if(HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    Error_Handler();
  }

  /* activate the OverDrive to reach the 216 Mhz Frequency */
  if(HAL_PWREx_EnableOverDrive() != HAL_OK)
  {
    Error_Handler();
  }

  /* Select PLL as system clock source and configure the HCLK, PCLK1 and PCLK2
     clocks dividers */
  RCC_ClkInitStruct.ClockType = (RCC_CLOCKTYPE_SYSCLK | RCC_CLOCKTYPE_HCLK |
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_DIV2;
  if(HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_6) != HAL_OK)
  {
    Error_Handler();
  }
}

/**
  * @brief  This function is executed in case of error occurrence.
  * @param  None
  * @retval None
  */
static void Error_Handler(void)
{
  /* User may add here some code to deal with this error */
  while(1)
  {
  }
}

/**
  * @brief  Configure the MPU attributes as Write Through for SRAM1/2.
  * @note   The Base Address is 0x20010000 since this memory interface is the AXI.
  *         The Region Size is 256KB, it is related to SRAM1 and SRAM2  memory size.
  * @param  None
  * @retval None
  */
static void MPU_Config(void)
{
  MPU_Region_InitTypeDef MPU_InitStruct;

  /* Disable the MPU */
  HAL_MPU_Disable();
```

```
  /* Configure the MPU attributes as WT for SRAM */
  MPU_InitStruct.Enable = MPU_REGION_ENABLE;
  MPU_InitStruct.BaseAddress = 0x20010000;
  MPU_InitStruct.Size = MPU_REGION_SIZE_256KB;
  MPU_InitStruct.AccessPermission = MPU_REGION_FULL_ACCESS;
  MPU_InitStruct.IsBufferable = MPU_ACCESS_NOT_BUFFERABLE;
  MPU_InitStruct.IsCacheable = MPU_ACCESS_CACHEABLE;
  MPU_InitStruct.IsShareable = MPU_ACCESS_NOT_SHAREABLE;
  MPU_InitStruct.Number = MPU_REGION_NUMBER0;
  MPU_InitStruct.TypeExtField = MPU_TEX_LEVEL0;
  MPU_InitStruct.SubRegionDisable = 0x00;
  MPU_InitStruct.DisableExec = MPU_INSTRUCTION_ACCESS_ENABLE;

  HAL_MPU_ConfigRegion(&MPU_InitStruct);

  /* Enable the MPU */
  HAL_MPU_Enable(MPU_PRIVILEGED_DEFAULT);
}

/**
  * @brief  CPU L1-Cache enable.
  * @param  None
  * @retval None
  */
static void CPU_CACHE_Enable(void)
{
  /* Enable I-Cache */
  SCB_EnableICache();

  /* Enable D-Cache */
  SCB_EnableDCache();
}

#ifdef  USE_FULL_ASSERT

/**
  * @brief  Reports the name of the source file and the source line number
  *         where the assert_param error has occurred.
  * @param  file: pointer to the source file name
  * @param  line: assert_param error line source number
  * @retval None
  */
void assert_failed(uint8_t* file, uint32_t line)
{
  /* User can add his own implementation to report the file name and line number,
     ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */

  /* Infinite loop */
  while (1)
  {
  }
}
#endif

/**
       * @brief Configure pins as: Analog, Input, Output, EVENT_OUT, EXTI
       * @param  None
  * @retval None
```

```
  */
void GPIO_Init(void)
{

  GPIO_InitTypeDef GPIO_InitStruct;

  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOB_CLK_ENABLE();
  __HAL_RCC_GPIOA_CLK_ENABLE();
  __HAL_RCC_GPIOG_CLK_ENABLE();
  __HAL_RCC_GPIOI_CLK_ENABLE();
  __HAL_RCC_GPIOK_CLK_ENABLE();
  __HAL_RCC_GPIOD_CLK_ENABLE();
  __HAL_RCC_GPIOH_CLK_ENABLE();
  __HAL_RCC_GPIOC_CLK_ENABLE();

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOG, GPIO_PIN_6, GPIO_PIN_SET); // G6: pink

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOC, GPIO_PIN_7|GPIO_PIN_6, GPIO_PIN_SET); // C7: orange, C6:
gelb

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_4, GPIO_PIN_SET); // B4: blau

  /*Configure GPIO pin : PB4 */
  GPIO_InitStruct.Pin = GPIO_PIN_4;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
  HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

  /*Configure GPIO pins : PC7 PC6 */
  GPIO_InitStruct.Pin = GPIO_PIN_7|GPIO_PIN_6;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
  HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);

  /*Configure GPIO pin : PG6 */
  GPIO_InitStruct.Pin = GPIO_PIN_6;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
  HAL_GPIO_Init(GPIOG, &GPIO_InitStruct);
}

/**
  * @}
  */

/************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
```

3. Main_h

```
/**
  ******************************************************************************
```

```
  * @file    Templates/main.h
  * @author  MCD Application Team
  * @version V1.0.2
  * @date    18-November-2015
  * @brief   Header for main.c module
  ******************************************************************************
  * @attention
  *
  * <h2><center>&copy; COPYRIGHT 2015 STMicroelectronics</center></h2>
  *
  * Redistribution and use in source and binary forms, with or without modification,
  * are permitted provided that the following conditions are met:
  *   1. Redistributions of source code must retain the above copyright notice,
  *      this list of conditions and the following disclaimer.
  *   2. Redistributions in binary form must reproduce the above copyright notice,
  *      this list of conditions and the following disclaimer in the documentation
  *      and/or other materials provided with the distribution.
  *   3. Neither the name of STMicroelectronics nor the names of its contributors
  *      may be used to endorse or promote products derived from this software
  *      without specific prior written permission.
  *
  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
  * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
  * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
  * DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
  * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
  * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
  * CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
  * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
  * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
  *
  ******************************************************************************
  */

/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef __MAIN_H
#define __MAIN_H

/* Includes ------------------------------------------------------------------*/
#include "stm32f7xx_hal.h"

/* Exported types ------------------------------------------------------------*/
/* Exported constants --------------------------------------------------------*/
/* Exported macro ------------------------------------------------------------*/
/* Exported functions ------------------------------------------------------- */

#endif /* __MAIN_H */

/************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
```

4. **Motor Thread**

```
/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef __MAIN_H
#define __MAIN_H
```

```
/* Includes ------------------------------------------------------------------*/
#include "stm32f7xx_hal.h"

#endif /* __MAIN_H */

/* Includes ------------------------------------------------------------------*/
#include "cmsis_os.h"                   // CMSIS RTOS header file

#ifdef _RTE_
#include "RTE_Components.h"             // Component selection
#endif

#define LEFT 0
#define RIGHT 1

// Global motor variables
extern int Motor_Steps;
extern int Progress_Value;
extern int Motor_Direction;

/*----------------------------------------------------------------------------
 *      MotorThread: Motor Thread for Single-Task Execution Model
 *----------------------------------------------------------------------------*/

 void MotorThread(void const *argument);              // thread function
 osThreadId tid_MotorThread;                           // thread id
 osThreadDef(MotorThread, osPriorityNormal, 1, 512);   // thread object

       int Init_MotorThread(void)
       {
              tid_MotorThread = osThreadCreate(osThread(MotorThread), NULL);
              if (!tid_MotorThread)
              {
                     return (-1);
              }

              return (0);
       }

       void MotorThread(void const *argument)
       {
              // Variables
              int Delay_Time;

              while (1)
              {
                     /*     Delay time between every step, currently set for a
                                   maximum motor frequency of 50Hz. */
                     Delay_Time =  15 * 10 / Progress_Value;

                     switch (Motor_Direction)
                     {
                            case RIGHT:
                                   switch (Motor_Steps)
                                   {
                                          case 1:
                                                 HAL_GPIO_WritePin(GPIOC, GPIO_PIN_7,
GPIO_PIN_RESET);
```

```
                                                HAL_GPIO_WritePin(GPIOB, GPIO_PIN_4,
GPIO_PIN_SET);
                                                osDelay(Delay_Time);
                                        case 2:
                                                HAL_GPIO_WritePin(GPIOG, GPIO_PIN_6,
GPIO_PIN_SET);
                                                osDelay(Delay_Time);
                                        case 3:
                                                HAL_GPIO_WritePin(GPIOB, GPIO_PIN_4,
GPIO_PIN_RESET);
                                                osDelay(Delay_Time);
                                        case 4:
                                                HAL_GPIO_WritePin(GPIOC, GPIO_PIN_6,
GPIO_PIN_SET);
                                                osDelay(Delay_Time);
                                        case 5:
                                                HAL_GPIO_WritePin(GPIOG, GPIO_PIN_6,
GPIO_PIN_RESET);
                                                osDelay(Delay_Time);
                                        case 6:
                                                HAL_GPIO_WritePin(GPIOC, GPIO_PIN_7,
GPIO_PIN_SET);
                                                osDelay(Delay_Time);
                                        case 7:
                                                HAL_GPIO_WritePin(GPIOC, GPIO_PIN_6,
GPIO_PIN_RESET);
                                                osDelay(Delay_Time);
                                        case 8:
                                                HAL_GPIO_WritePin(GPIOB, GPIO_PIN_4,
GPIO_PIN_SET);
                                                osDelay(Delay_Time);
                                                break;
                                        default:
                                                HAL_GPIO_WritePin(GPIOB, GPIO_PIN_4,
GPIO_PIN_RESET);
                                                HAL_GPIO_WritePin(GPIOC,
GPIO_PIN_7|GPIO_PIN_6, GPIO_PIN_RESET);
                                                HAL_GPIO_WritePin(GPIOG, GPIO_PIN_6,
GPIO_PIN_RESET);
                                }
                                break;

                                case LEFT:
                                switch (Motor_Steps)
                                {
                                        case 1:
                                                HAL_GPIO_WritePin(GPIOB, GPIO_PIN_4,
GPIO_PIN_RESET);
                                                HAL_GPIO_WritePin(GPIOC, GPIO_PIN_7,
GPIO_PIN_SET);
                                                osDelay(Delay_Time);
                                        case 2:
                                                HAL_GPIO_WritePin(GPIOC, GPIO_PIN_6,
GPIO_PIN_SET);
                                                osDelay(Delay_Time);
                                        case 3:
                                                HAL_GPIO_WritePin(GPIOC, GPIO_PIN_7,
GPIO_PIN_RESET);
                                                osDelay(Delay_Time);
```

```
                                        case 4:
                                              HAL_GPIO_WritePin(GPIOG, GPIO_PIN_6,
GPIO_PIN_SET);
                                              osDelay(Delay_Time);
                                        case 5:
                                              HAL_GPIO_WritePin(GPIOC, GPIO_PIN_6,
GPIO_PIN_RESET);
                                              osDelay(Delay_Time);
                                        case 6:
                                              HAL_GPIO_WritePin(GPIOB, GPIO_PIN_4,
GPIO_PIN_SET);
                                              osDelay(Delay_Time);
                                        case 7:
                                              HAL_GPIO_WritePin(GPIOG, GPIO_PIN_6,
GPIO_PIN_RESET);
                                              osDelay(Delay_Time);
                                        case 8:
                                              HAL_GPIO_WritePin(GPIOC, GPIO_PIN_7,
GPIO_PIN_SET);
                                              osDelay(Delay_Time);
                                              break;
                                        default:
                                              HAL_GPIO_WritePin(GPIOG, GPIO_PIN_6,
GPIO_PIN_RESET);
                                              HAL_GPIO_WritePin(GPIOC,
GPIO_PIN_7|GPIO_PIN_6, GPIO_PIN_RESET);
                                              HAL_GPIO_WritePin(GPIOB, GPIO_PIN_4,
GPIO_PIN_RESET);
                                  }
                                  break;
                        } // End of first switch
                  } // End of while loop
            } // End of function

            /************************** End of file ****************************/
```

5. **Windows_DGL:**

```
/*********************************************************************
*                                                                    *
*                SEGGER Microcontroller GmbH & Co. KG                *
*        Solutions for real time microcontroller applications        *
*                                                                    *
**********************************************************************
*                                                                    *
* C-file generated by:                                               *
*                                                                    *
*        GUI_Builder for emWin version 5.30                          *
*        Compiled Jul  1 2015, 10:50:32                              *
*        (c) 2015 Segger Microcontroller GmbH & Co. KG               *
*                                                                    *
**********************************************************************
*                                                                    *
*        Internet: www.segger.com  Support: support@segger.com       *
*                                                                    *
**********************************************************************
*/

// USER START (Optionally insert additional includes)
```

```
// USER END

#include "DIALOG.h"

/*********************************************************************
*
*       Defines
*
**********************************************************************
*/
#define ID_WINDOW_0         (GUI_ID_USER + 0x00)
#define ID_SLIDER_0         (GUI_ID_USER + 0x01)
#define ID_PROGBAR_0         (GUI_ID_USER + 0x02)
#define ID_TEXT_0         (GUI_ID_USER + 0x03)


// USER START (Optionally insert additional defines)
#define LEFT 0
#define RIGHT 1
#define STOP 0
#define START 1
// USER END

/*********************************************************************
*
*       Static data
*
**********************************************************************
*/

// USER START (Optionally insert additional static data)
int Progress_Value;
int Motor_Direction;
int Motor_Steps;
// USER END

/*********************************************************************
*
*       _aDialogCreate
*/
static const GUI_WIDGET_CREATE_INFO _aDialogCreate[] = {
  { WINDOW_CreateIndirect, "Window", ID_WINDOW_0, 0, 0, 480, 272, 0, 0x0, 0 },
  { SLIDER_CreateIndirect, "Slider", ID_SLIDER_0, 40, 111, 400, 50, 0, 0x0, 0 },
  { PROGBAR_CreateIndirect, "Progbar", ID_PROGBAR_0, 200, 23, 80, 20, 0, 0x0, 0 },
  { TEXT_CreateIndirect, "Speed", ID_TEXT_0, 152, 22, 41, 20, 0, 0x0, 0 },
  // USER START (Optionally insert additional widgets)
  // USER END
};

/*********************************************************************
*
*       Static code
*
**********************************************************************
*/

// USER START (Optionally insert additional static code)
// USER END
```

```
/*********************************************************************
*
*       _cbDialog
*/
static void _cbDialog(WM_MESSAGE * pMsg) {
  WM_HWIN hItem;
  int     NCode;
  int     Id;
  // USER START (Optionally insert additional variables)
       int Slider_Position;
  // USER END

  switch (pMsg->MsgId) {
  case WM_INIT_DIALOG:
    //
    // Initialization of 'Window'
    //
    hItem = pMsg->hWin;
    WINDOW_SetBkColor(hItem, GUI_MAKE_COLOR(0x00000000));
    //
    // Initialization of 'Speed'
    //
    hItem = WM_GetDialogItem(pMsg->hWin, ID_TEXT_0);
    TEXT_SetTextColor(hItem, GUI_MAKE_COLOR(0x00FFFFFF));
    TEXT_SetFont(hItem, GUI_FONT_13B_1);
    TEXT_SetTextAlign(hItem, GUI_TA_HCENTER | GUI_TA_VCENTER);
    // USER START (Optionally insert additional code for further widget
initialization)
              //
    // Initialization of 'Slider'
    //
              hItem = WM_GetDialogItem(pMsg->hWin, ID_SLIDER_0);
    SLIDER_SetRange(hItem, 0, 200);
              SLIDER_SetValue(hItem, 100);
              //
    // Initialization of 'Progress Bar'
    //
              hItem = WM_GetDialogItem(pMsg->hWin, ID_PROGBAR_0);
              PROGBAR_SetMinMax(hItem, 0, 100);
              PROGBAR_SetValue(hItem, 0);
    // USER END
    break;
  case WM_NOTIFY_PARENT:
    Id    = WM_GetId(pMsg->hWinSrc);
    NCode = pMsg->Data.v;
    switch(Id) {
    case ID_SLIDER_0: // Notifications sent by 'Slider'
      switch(NCode) {
      case WM_NOTIFICATION_CLICKED:
        // USER START (Optionally insert code for reacting on notification message)
        // USER END
        break;
      case WM_NOTIFICATION_RELEASED:
        // USER START (Optionally insert code for reacting on notification message)
                          hItem = WM_GetDialogItem(pMsg->hWin, ID_SLIDER_0);
                          SLIDER_SetValue(hItem, 100);
                          hItem = WM_GetDialogItem(pMsg->hWin, ID_PROGBAR_0);
                          PROGBAR_SetValue(hItem, 0);
                          Motor_Steps = STOP;
```

```
        // USER END
        break;
      case WM_NOTIFICATION_VALUE_CHANGED:
        // USER START (Optionally insert code for reacting on notification message)
                        hItem = WM_GetDialogItem(pMsg->hWin, ID_SLIDER_0);
                        Slider_Position = SLIDER_GetValue(hItem);
                        switch ((int) (Slider_Position > 100))
                        {
                               case 0:
                                      Motor_Direction = LEFT;
                                      Motor_Steps = START;
                                      Progress_Value = (Slider_Position - 100) * -1;
                                      hItem = WM_GetDialogItem(pMsg->hWin,
ID_PROGBAR_0);
                                      PROGBAR_SetValue(hItem, Progress_Value);
                                      break;
                               case 1:
                                      Motor_Direction = RIGHT;
                                      Motor_Steps = START;
                                      Progress_Value = Slider_Position - 100;
                                      hItem = WM_GetDialogItem(pMsg->hWin,
ID_PROGBAR_0);
                                      PROGBAR_SetValue(hItem, Progress_Value);
                                      break;
                        }
        // USER END
        break;
      // USER START (Optionally insert additional code for further notification
handling)
      // USER END
      }
      break;
    // USER START (Optionally insert additional code for further Ids)
    // USER END
    }
    break;
  // USER START (Optionally insert additional message handling)
  // USER END
  default:
    WM_DefaultProc(pMsg);
    break;
  }
}

/*********************************************************************
*
*       Public code
*
**********************************************************************
*/
/*********************************************************************
*
*       CreateWindow
*/
WM_HWIN CreateWindow(void);
WM_HWIN CreateWindow(void) {
  WM_HWIN hWin;
```

```
  hWin = GUI_CreateDialogBox(_aDialogCreate, GUI_COUNTOF(_aDialogCreate), _cbDialog,
WM_HBKWIN, 0, 0);
  return hWin;
}

// USER START (Optionally insert additional public code)
// USER END

/************************* End of file *************************
```

Quellen:

Abbildung 1
http://www.bhphotovideo.com/c/product/1051846-REG/cinetics_cinetics_3_axis360_pro.html

Abbildung 2
http://www.bhphotovideo.com/c/product/1051846-REG/cinetics_cinetics_3_axis360_pro.html

Abbildung3
Eigene Darstellung

Abbildung 4
Eigene Darstellung