FUNCTIONS GLOSSARY OF K-JUNIOR

K-Team, December 2010

1 Functions of control and access to the robot (file "KJunior.h")

1.1. Configuration functions of the robot

```
// Initialisation of K-Junior
void K-Junior_init( void );

// Configuration of the method of reading for the IR sensors.
void KJunior_config_auto_refresh_sensors( int1 Bit );

// Configuration of the refresh rate of the TV remote receiver.
void KJunior_config_auto_refresh_tv_remote( int1 Bit );

// Enabling or disabling the "Serial Remote Control" mode.
void KJunior_config_rs232_control( int1 Bit );

// Enabling or disabling "TV Remote Control" mode.
void KJunior_config_tv_remote_control( int1 Bit );
```

1.2. Functions for reading the "Flags"

```
// Check if the IR sensors value is refreshed
int1 KJunior_flag_sensors_refreshed( void );

// Set to zero the refresh rate flag of the IR sensors
void KJunior_flag_sensors_reset( void );

// Check if the "Serial Remote Control" mode is activated
int1 KJunior_flag_rs232_filtering( void );

// Check if the TV remote control value is refreshed
int1 KJunior_flag_tv_data_refreshed( void );

// Check if the TV data are still emitting
int1 KJunior_flag_tv_data_emitting( void );

// Set to zero the refresh flag of the TV remote control
void KJunior_flag_tv_data_reset( void );
```

1.3. Robot Input and Output functions

```
// Get the proximity measure of an IR sensor
// Sensors : FRONT, FRONTLEFT, FRONTRIGHT, LEFT, RIGHT, REAR, GROUNDLEFT or GROUNDRIGHT.
signed int16 KJunior_get_proximity( char Sensor );

// Get the luminosity measure of an IR sensor
// Sensors : FRONT, FRONTLEFT, FRONTRIGHT, LEFT, RIGHT, REAR, GROUNDLEFT ou GROUNDRIGHT.
signed int16 KJunior_get_brightness( char Sensor );

// Read the state of the three switches
unsigned char KJunior_get_switch_state();

// Get the data from the TV receiver
unsigned char KJunior_get_tv_data( void );

// Get the address of the emitter from the TV receiver
unsigned char KJunior_get_tv_addr( void );

// Send a data and an address on the TV IR emitter
void KJunior_send_tv_value( unsigned char addr, unsigned char data );

// Set the speed of the motors: left (first value) and right (second value)
void KJunior_set_speed( signed int8 LeftSpeed, signed int8 RightSpeed );

// Produce a continuous sound with the Buzzer at the selected frequency
void KJunior_beep( unsigned char Freq );

// Switch on or off the left led
void KJunior_led_left( int1 State );

// Switch on or off the front-left led
void KJunior_led_frontleft( int1 State );

// Switch on or off the front-right led
void KJunior_led_frontright( int1 State );

// Switch on or off the right led
void KJunior_led_right( int1 State );

// Switch on or off the ON/OFF led
void KJunior_led_onoff( int1 State );

// Manually refresh the IR sensors
void KJunior_manual_refresh_sensors();
```

1.4. Delay and time Functions

```
// Insert a delay in seconds ( 0 to 65535 )
void KJunior_delay_s( unsigned int Delay );

// Insert a delay in milliseconds ( 0 to 65535)
void KJunior_delay_ms( unsigned int Delay );

// Insert a delay in microseconds ( 0 to 65535 )
void KJunior_delay_us( unsigned int Delay );

// Get the number of milliseconds elapsed from the robot start-up
unsigned int32 KJunior_get_time( void );

// Initialise or force the number of milliseconds elapsed
void KJunior_set_time( unsigned int32 Time );
```

1.5. Access functions to the I/O of the external connectors

```
// Reading of the B6 pin
int1 KJunior_ext_read_PINB6( void );

// Reading of the B7 pin
int1 KJunior_ext_read_PINB7( void );

// Writing to the B6 pin
void KJunior_ext_write_PINB6( int1 Bit );

// Writing to the B7 pin
void KJunior_ext_write_PINB7( int1 Bit );
```

2 Control and access functions to the "I/O Interface board" module ("hemgenio.h" file)

```
// Initialise the module; it's the very first function to call at the beginning of the code
void HemGenIO_Init( void );

// Read "Firmware" version
void HemGenIO_Read_Version( void );

// Read the state of a digital input
int1 HemGenIO_Read_Digital( char Input );

// Read the state of a digital output
void HemGenIO_Read_Digital( char Output, int1 State );

// Read the value of an analog input
unsigned char HemGenIO_Read_Analog( char Input );
```

3 Control and access functions to the "Ultrasonic detector" module ("hemultrasonicsensor.h" file)

```
// Initialise the module; it's the very first function to call at the beginning of the code
void HemUltraSon_Init( void );

// Read "Firmware" version
void HemUltraSon_Read_Version( void );

// Read the value of the luminosity sensor
unsigned char HemUltraSon_Read_Brightness( void );

// Start a distance acquisition in millimeters
void HemUltraSon_Start_Mesure( void );

// Read the value of an echo
unsigned int16 HemUltraSon_Read_Value( char EchoNumber );

// Initialise the range
void HemUltraSon_Init_Range_Register( unsigned char Value );
```

4 Control and access functions to the "Linear camera" module ("hemlincam.h" file)

// The pixel values are saved into the following arrays of variables:

unsigned char HemLinCam_Pixels_Zone1[34];

unsigned char HemLinCam_Pixels_Zone2[34];

unsigned char HemLinCam_Pixels_Zone3[34];

// Each of these arrays has a size of 34 pixels, respectively corresponding to the left, middle and right parts of the image.

// Initialise the module; it's the very first function to call at the beginning of the code
void HemLinCam_Init(void);

// Read "Firmware" version
void HemLinCam_Read_Version(void);

// Define the threshold value for the threshold pixels
void HemLinCam_Set_Threshold(unsigned char Value);

// Read the defined threshold value
unsigned char HemLinCam_Read_Threshold(void);

// Define the exposure time in milliseconds
void HemLinCam_Set_Exposition_Time(unsigned char Value);

// Read the defined exposure time in milliseconds
unsigned char HemLinCam_Read_Exposition_Time(void);

// Read the values of all the pixels; these values are stored in the arrays of variables "HemlinCam_Pixels_ZoneX"
void HemLinCam_Read_Pixels(void);

// Read the values of all the threshold pixels; these values are stored in the arrays of variables "HemlinCam_Pixels_ZoneX"
void HemLinCam_Read_Pixels_Tresholded(void);

// Switch on or off the module led
void HemLinCam_Set_Led_State(int1 State);

5 Control and access functions to the "Vocal synthesizer" module ("hemtexttospeech.h file)

```
// Initialise the module; it's the very first function to call at the beginning of the code
void HemTextToSpeech_Init( void );

// Read "Firmware" version
void HemTextToSpeech_Version( void );

// Define the speed of the speech during the pronunciation of the sentence
void HemTextToSpeech_Speed( char Value );

// Define the pitch value
void HemTextToSpeech_Pitch( char Value );

// Pronounce a dynamically loaded sentence
void HemTextToSpeech_Speak( char Phrase[], char Taille );

// Pronounce a sentence pre-loaded in the module
void HemTextToSpeech_Speak_PreDef( char Num );
```

6 Control and access functions to the "LCD display" module ("hemlcd.h" file)

```
// The text defined for each of the LCD display is stored in 2 arrays of 12 bytes.
char Line1[12], Line2[12];

// The state of the 3 push buttons is stored in 3 global variables
int1 SW1, SW2, SW3;

// Initialise the module; it's the very first function to call at the beginning of the code
void HemLCD_Init( void );

// Read "Firmware" version
void HemLCD_Read_Version( void );

// Define the intensity level of the display backlight
void HemLCD_Set_Backlight( unsigned char Value );

// Define the contrast level of the display
void HemLCD_Set_Contrast( unsigned char Value );

// Clear the screen
void HemLCD_Clear_Screen( void );

// Clear the content of the line 1
void HemLCD_Clear_Line1( void );

// Clear the content of the line 2
void HemLCD_Clear_Line2( void );

//  Read the values of the 3 push buttons (stored in SW1, SW2 et SW3)
void HemLCD_Read_Interruptors( void );

// Display the line 1, left-justified (WARNING: use Line1[] or Line2[])
void HemLCD_Line1_Left( char* Data );

// Display the line 1, middle-justified (WARNING: use Line1[] or Line2[])
void HemLCD_Line1_Centered( char* Data );

// Display the line 1, right-justified (WARNING: use Line1[] or Line2[])
void HemLCD_Line1_Right( char* Data );

// Display the line 2, left-justified (WARNING: use Line1[] or Line2[])
void HemLCD_Line2_Left( char* Data );

// Display the line 2, middle-justified (WARNING: use Line1[] or Line2[])
```

```
void HemLCD_Line2_Centered( char* Data );

// Display the line 2, right-justified (WARNING: use Line1[] or Line2[])
void HemLCD_Line2_Right( char* Data );
```

7 Control and access functions to the "Gripper" module ("hemgripper.h" file)

```
// Initialize the Gripper turret
void HemGripper_Init( void )

// Read the firmware version (OS) of the Gripper turret connected on the I2C Bus
char HemGripper_Read_Version( void )

// Read the position of the Arm (0-7000)
Unsigned int16 HemGripper_Read_Arm_Position( void )

// Read the position of the Gripper (0-5100)
Unsigned int16 HemGripper_Read_Gripper_Position( void )

// Read the position consign of the Arm (0-7000)
Unsigned int16 HemGripper_Read_Arm_Consign( void )

// Read the position consign of the Gripper (0-5100)
Unsigned int16 HemGripper_Read_Gripper_Consign( void )

// Read the speed of the Arm used for the consign movements (0-255)
Unsigned char HemGripper_Read_Arm_Speed( void )

// Read the speed of the Gripper used for the consign movements (0-255)
Unsigned char HemGripper_Read_Gripper_Speed( void )

// Read the flag which indicate if the Arm must be turn off or not when a stable position is reached
Unsigned char HemGripper_Read_Arm_Disable( void )

// Read the flag which indicate if the Gripper must be turn off or not when the open position is reached
Unsigned char HemGripper_Read_Gripper_Disable( void )


// Read the Voltage level of the battery
Unsigned int16 HemGripper_Read_Battery_Voltage( void )
```

```
// Read the value of the available remaining capacity of the Battery
Unsigned char HemGripper_Read_Battery_Capacity( void )

// Set the Arm position to reach without speed control
void HemGripper_Set_Arm_Position( unsigned int16 value )

// Set the Gripper position to reach without speed control
void HemGripper_Set_Gripper_Position( unsigned int16 value )

// Set the Arm position to reach using the speed defined with the
Hemisson_Set_Arm_Speed() function.
void HemGripper_Set_Arm_Consign( unsigned int16 value )

// Set the Gripper position to reach using the speed defined with the
Hemisson_Set_Gripper_Speed() function.
void HemGripper_Set_Gripper_Consign( unsigned int16 value )

// Configure the speed to use for a Arm movement with speed control
void HemGripper_Set_Arm_Speed( unsigned char value )

// Configure the speed to use for a Gripper movement with speed control
void HemGripper_Set_Gripper_Speed( unsigned char value )

// Configure the flag to define if the Arm must turn off its motor when a stable Position
is reached
void HemGripper_Arm_Disable( unsigned char value )

// Configure the flag to define if the Gripper must turn off its motor when the open
position is reached
void HemGripper_Gripper_Disable( unsigned char value )
```

END OF FILE