ST600μNET Library Interface Guide
Version: 11.65

by

Albert Tello

www.rmv.com

Contents

Chapter 1

Introduction

1.1 Brief Introduction

The ST600µNET has been developed in C++, with a fundamental goal to be ported all Operating Systems and IoT ready. This document will describe all functions developed to control the ST600µNET Series, in the way an application can be written in the mayor popular languages compiled or scripted like: C Sharp, C++, .NET, Python, Delphi, Perl, and JAVA. Further versions of the library will support Web Service for the mayor O.S., which will allow to create an application written in JavaScript.

1.2 HID Connection

The ST600µNET Series has on board a HID USB-to-SMBus bridge interface, which ST600uNET library provides a simple API to configure and operate the boards. ST600uNET library provides interface abstraction so that users can develop their application without writing any USB HID Code. C libraries implementing the ST600uNET library Interface Specification are provided for Windows 7,8.1,10 and Mac OS X 10.5 and later. Similarly, various include files are provided to import library functions into C-Sharp .NET, and Visual Basic .NET.

Chapter 2

HID Connection Functions

2.1 ST600uNET_Connect

int ST600uNET_Connect (unsigned char * ST600uNET_Found ,char *ProductString, char * SerialNumber);
***Parameters**:*
char ProductString[255],
char SerialNumber[255];
unsigned char ST600uNET_Found, by default is assigned a value of 0xFF, which means "Not connected".

***Return**:*
"0": NO ERROR
Parameters Returned:
SerialNumber ="RMVxxxxxxxxxx", (encrypted serial number)
ProductString="ST600uNET HID Controller".
ST600uNET_Found =1.
"1": ERROR
Parameters Returned:
SerialNumber ="xxxx-xxx", (encrypted serial number)
ProductString="".
ST600uNET_Found =0xFF.

2.2 ST600uNET_Close

int ST600uNET_Close (char * SerialNumber);
***Return**:*
"0": NO ERROR
Parameters Returned:
SerialNumber ="",null string.
"1": ERROR
Parameters Returned:
SerialNumber ="RMVxxxxxxx", same string when HID port was open.

Chapter 3

Motor Parameters Functions

3.1 ST600uNet_SetMotorParameters

ST600uNet_SetMotorParameters (unsigned char `motor_address_mask`, unsigned int `winding_current`, float `inductance_in_mH`, unsigned char `motor_power`, unsigned char `bipolar_unipolar`, float `winding_resistance_in_Ohms`,unsigned char `step_angle`, unsigned short int * `Error_number`);

This function will provide to RMV859 controller all motion parameters in order to perform different motions like : profiling movement,PI, Close Loop, Current Control, etc. This function needs to be called every times any motor parameters are changed.
`Note1: It is very important to send the correct motor parameters otherwise will cause wrong current-voltage calculation.`

Parameters:

unsigned char *motor_address_mask*: RMV859 Address:($ADDR0 \Rightarrow 1$, $ADDR1 \Rightarrow 2, ADDR2 \Rightarrow 4$, $ADDR3 \Rightarrow 8, ALL \Rightarrow 0$)

unsigned int *winding_current*: Valid value (in mA) from 400 mA, to 4500 mA.

float *inductance_in_mH*: Value in mH, this parameter must be precise in order to setup motor pulse with.

unsigned char *motor_power*: Valid value from:18 Volts to 50 Volts.

unsigned char *bipolar_unipolar*:"0"=Bipolar, "1"=Unipolar.

float *winding_resistance_in_Ohms*: Valid values from $0\,Ohm < RESISTANCE \leq 63.00\,Ohms$.

unsigned char *step_angle*: "1" $\Rightarrow$ 1.8°,"2" $\Rightarrow$ 0.9°,"3" $\Rightarrow$ 1.2°, "4" $\Rightarrow$ 0.72°,"5" $\Rightarrow$ 0.36°

Return:

"0": NO ERROR

Parameters Returned:

unsigned short int * `Error_number` = 0.

"1": ERROR

Parameters Returned:

unsigned short int * `Error_number` ="Error Number", please see Errors Section, what was the cause.

3.2 ST600uNet_ReadMotorParameters

ST600uNet_ReadMotorParameters (unsigned char motor_address_mask,
unsigned int * winding_current_in_mAmp, float * inductance_in_mH,unsigned char * motor_power,
unsigned int * motor_config_1, float * winding_resistance_in_Ohms, unsigned int * speed_open_loop,
unsigned char* step_angle, unsigned short int * Error_number);

This function will read all motor parameters from RMV959 controller, stored in Flash memory, if a check sum error is detected, and the flag ERROR_READ_FLASH_MOTOR_PARAMETERS has been set in "motor_current_flags".

Parameters:

unsigned char *motor_address_mask*: RMV859 Address:($ADDR0 \Rightarrow 1$, $ADDR1 \Rightarrow 2, ADDR2 \Rightarrow 4$, $ADDR3 \Rightarrow 8, ALL \Rightarrow 0$)

unsigned int * *winding_current_in_mAmp*: Winding current in mAmp.

float * *inductance_in_mH*: Motor inductance in mH.

unsigned char * *motor_power*:DC Motor power supply, maximum 50VDC.

unsigned int * *motor_config_1*: Motor Configure "1". Please refer to Motion Configuration for a FULL Flags Definitions.

unsigned int * *winding_resistance_in_Ohms*: Motor resistance, the maximum allowed is MOTOR_RESISTANCE_MAX = 62 Ohms.

unsigned int * *speed_open_loop*: Not Used in the firmware version.

unsigned char * *step_angler*: Motor steps per revolution.

unsigned short int * *Error_number* $\Rightarrow$

Return:

"0": NO ERROR

Parameters Returned:

unsigned short int * Error_number = 0.

"1": ERROR

Parameters Returned:

unsigned short int * Error_number ="Error Number", please see Errors Section, what was the cause.

3.3 ST600uNet_SAVE_ParametersToInternal_FLASH

ST600uNet_SAVE_ParametersToInternal_FLASH (unsigned char `motor_addr_mask`, unsigned short int `what_to_save`, unsigned short int* `Error_number`);

This function save the motor manufacturer parameters into RMV859 controller internal FLASH. Note2: also this function can be used to save others pre-programmed parameters to Flash memory .
When motor parameters must be saved, the flag `SAVE_MOTOR_PARAMETERS_TO_FLASH` must be set. If an error in writing parameters occurred, please CALL `ST600uNet_Get_Internal_Report` and verify if `ERROR_WRITE_FLASH_MOTOR_PARAMETERS` is set.

Parameters:

unsigned char *motor_address_mask*: RMV859 Address:($ADDR0 \Rightarrow 1$, $ADDR1 \Rightarrow 2$, $ADDR2 \Rightarrow 4$, $ADDR3 \Rightarrow 8$, $ALL \Rightarrow 0$)

unsigned short int *what_to_save*: `SAVE_MOTOR_PARAMETERS_TO_FLASH`

Return:

"0": NO ERROR

Parameters Returned:

unsigned short int * `Error_number` = 0.

"1": ERROR

Parameters Returned:

unsigned short int * `Error_number` ="Error Number", please see Errors Section, what was the cause.

Chapter 4

Motion Configuration Functions

4.1 ST600uNet_SetMotionConfig

ST600uNet_SetMotionConfig (unsigned char `motor_address_mask`,
unsigned char `bipolar_unipolar`, unsigned char `operating_mode`,
unsigned char `step_resolution`, unsigned char `control_mode`, unsigned char `sampling_rate`,
unsigned char `enable_fifo`, unsigned char `sequential_motion_flag`,
unsigned short int* `Error_number`);

Parameters:

unsigned char *motor_address_mask*: RMV859 Address:($ADDR0 \Rightarrow 1$,
$ADDR1 \Rightarrow 2, ADDR2 \Rightarrow 4$,
$ADDR3 \Rightarrow 8, ALL \Rightarrow 0$)

unsigned char *bipolar_unipolar*: $Bipolar \Rightarrow$ "0", $Unipolar \Rightarrow$ "1".

unsigned char *operating_mode*: Three motor control modes :

`FIXED_VOLTAGE` $\Rightarrow$ 0, The PWM motor duty cycle is calculated from motor parameters.

`FIXED_CURRENT` $\Rightarrow$ 1, The maximum amplitude of current in both motor windings is sampled during one complete sine wave. If the maximum current amplitude is lower than the desired value, the drive voltage is increased gradually by adjusting the PWM duty cycle until the desired amplitude is reached. If the current is too high the duty cycle is decreased, but not less than the initial value corresponding to the rated motor voltage.

`CLOSED_LOOP` $\Rightarrow$ 3 PI Controller:

$$V * u = R * i + L\frac{di}{dt}$$

L = Motor resistance
R = Motor inductance
V = DC voltage
i = Instantaneous motor current
u = PWM duty cycle percentage

`ENCODER_CLOSE_POS` $\Rightarrow$ 4 Not enabled in this firmware version.

`ENCODER_CLOSE_SPEED` $\Rightarrow$ 5 Not enabled in this firmware version.

unsigned char *step_resolution*: Number of micro-steps per full step.

`ST_FULLSTEP` $\Rightarrow$ 0 Full Steps
`ST_HALFSTEP` $\Rightarrow$ 1 Half Steps
`ST_1_4STEP` $\Rightarrow$ 2 $1/4\ of\ Steps$
`ST_1_8STEP` $\Rightarrow$ 3 $1/8\ of\ Steps$

ST_1_16STEP ⇒ 4 1/16 *of Steps*
ST_1_32STEP ⇒ 5 1/32 *of Steps*
ST_1_64STEP ⇒ 6 1/64 *of Steps*
ST_1_128STEP ⇒ 7 1/128 *of Steps*
ST_1_256STEP ⇒ 8 1/256 *of Steps*

unsigned char *control_mode*: Different kind of motions:

SPEED_CONTROL ⇒ 1: The motor will change Speed "up-down", in every tick of(2,1, or 5 msec), according to the acceleration-deceleration values programmed. Please refer to Section 4.5, for acceleration-speed-deceleration parameters.

POSITION_CONTROL ⇒ 2: When the desired position is reached, the motor should stop immediately to avoid position oscillations. Since a fixed deceleration rate is imposed to the motor, position controller must take it into account and begin the deceleration at right time, before it is too late.Please refer to Section 4.5, for acceleration-speed-deceleration parameters.

PROFILE_SPEED_CONTROL ⇒ 3 This method is used when FIFO memory is enabled.Speed, Acceleration, Deceleration are calculated in Section 4.6. The parameters for this movement are: i)Speed *(unsigned short int*), ii) Acceleration (*unsigned short int*), and finally iii) Steps Number (*long*). The motor will accelerate until speed is reached, it will drive at speed constant until de steps number are reached, at that time RMV859 will execute another motion from internal FIFO, only if "TRIGGER" allows.

S_CURVE_CONTROL ⇒ 4 NO ENABLED in this firmware version.

TRAPEZOIDAL_CONTROL ⇒ 5 In this method a traditional Trapezoidal motion is performed. Speed, Acceleration, Deceleration are calculated in Section 4.6. The parameters for this movement are: i)Speed*(unsigned short int*), ii) Acceleration (*unsigned short int*), iii) Steps Number (*long*), iv) and finally Deceleration (*unsigned short int*).

PROFILE_TRAPEZOIDAL_CONTROL ⇒ 6 In this method a traditional Trapezoidal motion is performed, when FIFO is ENABLED. Speed, Acceleration, Deceleration are calculated in Section 4.6. The parameters for this movement are: i)Speed*(unsigned short int*), ii) Acceleration (*unsigned short int*), iii) Steps Number (*long*), iv) and finally Deceleration (*unsigned short int*). After step number has been reached, RMV859 will extract another set of motion from FIFO and automatically will start only "TRIGGER" allows.

unsigned char *sampling_rate*:Three fixed sampling rate can be chosen: when SPEED_CONTROL, or POSITION_CONTROL has been selected in *control_mode*. In every "Tick" Speed will be accelerated-decelerated or slew rate constant according to the parameters calculated in Section 4.5.

SAMPLING_RATE_TWO_MSEC ⇒ 2 mSec: Two milliseconds.

SAMPLING_RATE_ONE_MSEC ⇒ 1 mSec: One millisecond.

SAMPLING_RATE_FIVE_MSEC ⇒ 5 mSec: Five milliseconds.

unsigned char *enable_fifo*: ENABLE ⇒ 1
DISABLE ⇒ 0

unsigned char *sequential_motion_flag*: ENABLE ⇒ 1
DISABLE ⇒ 0

Return:

"0": NO ERROR

Parameters Returned:

unsigned short int * `Error_number` = 0.

"1": ERROR

Parameters Returned:

unsigned short int * `Error_number` ="Error Number", please see Errors Section, what was the cause.

4.2 ST600uNet_Read_Full_MotionConfig

ST600uNet_Read_Full_MotionConfig (unsigned char `motor_address_mask`, unsigned short int * `motor_config_1`, unsigned short int* `motor_config_2`, unsigned short int * `sequential_motion_setting`, unsigned short int* `Error_number`)

Parameters:

unsigned char *motor_address_mask*: RMV859 Address:($ADDR0 \Rightarrow 1$, $ADDR1 \Rightarrow 2, ADDR2 \Rightarrow 4$, $ADDR3 \Rightarrow 8, ALL \Rightarrow 0$)

unsigned short int ** *motor_config_1: The following flags are returned:

$Bit[0]$ ⇒ `Bipolar_Unipolar`;
`BIPOLAR` ⇒ $[x, x, x, x, x, x, x, 0]$,
`UNIPOLAR` ⇒ $[x, x, x, x, x, x, x, 1]$,

$Bit[3, 2, 1]$ ⇒ `Operation_Mode`:
`FIXED_VOLTAGE` ⇒ $[x, x, x, x, 0, 0, 0, x]$,
`FIXED_CURRENT` ⇒ $[x, x, x, x, 0, 1, 0, x]$,
`CLOSED_LOOP` ⇒ $[x, x, x, x, 0, 1, 1, x]$

$Bit[7, 6, 5, 4]$ ⇒ `Step_Resolution`:
`ST_FULLSTEP` ⇒ $[0, 0, 0, 0, x, x, x, x]$,
`ST_HALFSTEP` ⇒ $[0, 0, 0, 1, x, x, x, x]$,
`ST_1_4STEP` ⇒ $[0, 0, 1, 0, x, x, x, x]$,
`ST_1_8STEP` ⇒ $[0, 0, 1, 1, x, x, x, x]$,
`ST_1_16STEP` ⇒ $[0, 1, 0, 0, x, x, x, x]$,
`ST_1_32STEP` ⇒ $[0, 1, 0, 1, x, x, x, x]$,
`ST_1_64STEP` ⇒ $[0, 1, 1, 0, x, x, x, x]$,
`ST_1_128STEP` ⇒ $[0, 1, 1, 1, x, x, x, x]$,
`ST_1_256STEP` ⇒ $[1, 0, 0, 0, x, x, x, x]$,

unsigned short int ** *motor_config_2: The following flags are returned.

$Bit[3, 2, 1, 0]$ ⇒ `Control_Mode`:
`SPEED_CONTROL` ⇒ $[x, x, x, x, 0, 0, 0, 1]$,
`POSITION_CONTROL` ⇒ $[x, x, x, x, 0, 0, 1, 0]$,
`PROFILE_SPEED_CONTROL` ⇒ $[x, x, x, x, 0, 0, 1, 1]$,
`TRAPEZOIDAL_CONTROL` ⇒ $[x, x, x, x, 0, 1, 0, 1]$,
`PROFILE_TRAPEZOIDAL_CONTROL` ⇒ $[x, x, x, x, 0, 1, 1, 0]$,

$Bit[5, 4]$ ⇒ `Speed_Sampling_Rate`;
`SAMPLING_RATE_TWO_MSEC` ⇒ $[x, x, 0, 0, x, x, x, x]$,
`SAMPLING_RATE_ONE_MSEC` ⇒ $[x, x, 0, 1, x, x, x, x]$,

SAMPLING_RATE_FIVE_MSEC $\Rightarrow [x, x, 1, 0, x, x, x, x]$,

$Bit[7, 6]$ $\Rightarrow$ FIFO_Enable_Flag;

FIFO_ENABLE $\Rightarrow [0, 1, x, x, x, x, x, x]$,

FIFO_DISABLE $\Rightarrow [0, 0, x, x, x, x, x, x]$,

***unsigned short int** * sequential_motion_setting*: The following flags are returned:

$Bit[0]$ $\Rightarrow$ Sequential_Motion_Flag;

ENABLE $\Rightarrow [x, x, x, x, x, x, x, x][x, x, x, x, x, x, x, 1]$,

DISABLE $\Rightarrow [x, x, x, x, x, x, x, x][x, x, x, x, x, x, x, 0]$,

$Bit[7]$ $\Rightarrow$ Pivot_Controller; Dimension:16 Bits

PIVOT_CONTROLLER_ENABLE $\Rightarrow [x, x, x, x, x, x, x, x][1, x, x, x, x, x, x, x]$,

PIVOT_CONTROLLER_DISABLE $\Rightarrow [x, x, x, x, x, x, x, x][0, x, x, x, x, x, x, x]$,

$Bit[11, 10, 9, 8]$ $\Rightarrow$ SEQ_Motion_Controllers_Enabled:

ALL_Possible_Enabled $\Rightarrow [x, x, x, x, 1, 1, 1, 1][x, x, x, x, x, x, x, x]$,

ALL_Controllers_Disabled $\Rightarrow [x, x, x, x, 0, 0, 0, 0][x, x, x, x, x, x, x, x]$,

$Bit[15, 14, 13, 12]$ $\Rightarrow$ SEQ_Motion_Controllers_MASK:

ALL_Motion_Campleted_MASK $\Rightarrow [1, 1, 1, 1, x, x, x, x][x, x, x, x, x, x, x, x]$,

ALL_Controllers_Running $\Rightarrow [0, 0, 0, 0, x, x, x, x][x, x, x, x, x, x, x, x]$,

***Return**:*

"0": NO ERROR

Parameters Returned:

unsigned short int * Error_number = 0.

"1": ERROR

Parameters Returned:

unsigned short int * Error_number ="Error Number", please see Errors Section, what was the cause.

4.3 ST600uNet_SetMotionOperationMode

ST600uNet_SetMotionOperationMode (unsigned char motor_address_mask, unsigned char operating_mode, unsigned short int* Error_number);

***Parameters**:*

***unsigned char** motor_address_mask*: RMV859 Address:($ADDR0 \Rightarrow 1$, $ADDR1 \Rightarrow 2, ADDR2 \Rightarrow 4$, $ADDR3 \Rightarrow 8, ALL \Rightarrow 0$)

***unsigned char** operating_mode*: Three motor control modes :

FIXED_VOLTAGE $\Rightarrow$ 0, The PWM motor duty cycle is calculated from motor parameters. This method run as voltage open loop, micro-stepping is allowed. When motor running more at high speed the torque will decrease. For motor power is suggested between 24Volts to 36 Volts.

FIXED_CURRENT $\Rightarrow$ 1, The maximum amplitude of current in both motor windings is sampled during one complete sine wave. If the maximum current amplitude is lower than the desired value, the drive voltage is increased gradually by adjusting the PWM duty cycle until the desired amplitude is reached. If the current is too high the duty cycle is decreased, but not less than the initial value corresponding to the rated motor voltage.

CLOSED_LOOP $\Rightarrow$ 3 PI Controller:

$V * u = R * i + L\frac{di}{dt}$

L = Motor resistance

R = Motor inductance

V = DC voltage
i = Instantaneous motor current
u = PWM duty cycle percentage

`ENCODER_CLOSE_POS` ⇒ 4 Not enabled in this firmware version.

`ENCODER_CLOSE_SPEED` ⇒ 5 Not enabled in this firmware version.

***Return**:*

"0": NO ERROR

Parameters Returned:

unsigned short int * `Error_number` = 0.

"1": ERROR

Parameters Returned:

unsigned short int * `Error_number` ="Error Number", please see Errors Section, what was the cause.

4.4 ST600uNet_SetStepResolution

ST600uNet_SetStepResolution (unsigned char `motor_address_mask`, unsigned char `step_resolution`, unsigned short int* `Error_number`);

***Parameters**:*

***unsigned char** `motor_address_mask`*: RMV859 Address:($ADDR0 \Rightarrow 1$, $ADDR1 \Rightarrow 2, ADDR2 \Rightarrow 4$, $ADDR3 \Rightarrow 8, ALL \Rightarrow 0$)

***unsigned char** `step_resolution`*: This function can change step resolution , in any kind of motion. The total micro-steps per full step are as follows:

`ST_FULLSTEP` ⇒ 0 Full Steps
`ST_HALFSTEP` ⇒ 1 Half Steps
`ST_1_4STEP` ⇒ 2 1/4 *of Steps*
`ST_1_8STEP` ⇒ 3 1/8 *of Steps*
`ST_1_16STEP` ⇒ 4 1/16 *of Steps*
`ST_1_32STEP` ⇒ 5 1/32 *of Steps*
`ST_1_64STEP` ⇒ 6 1/64 *of Steps*
`ST_1_128STEP` ⇒ 7 1/128 *of Steps*
`ST_1_256STEP` ⇒ 8 1/256 *of Steps*

***Return**:*

"0": NO ERROR

Parameters Returned:

unsigned short int * `Error_number` = 0.

"1": ERROR

Parameters Returned:

unsigned short int * `Error_number` ="Error Number", please see Errors Section, what was the cause.

4.5 Acceleration_Deceleration_Equations_in_step/sec^2

When RMV859 controller perform SPEED_CONTROL or POSITION_CONTROL, the units of measurement for motion parameters are linear as follows:

Acceleration, Deceleration = $steps/sec^2$;
Speed = $steps/sec$;
Number of Steps = 32 bits step counter;

Acceleration_Deceleration Equation:

$$\mathbf{ACCEL} = \frac{(\mathbf{5.12} * \mathbf{ACCELERATION})}{\mathbf{1024}}; \text{ Eq.1}$$

Where $10000 =< ACCELERATION <= 10000$

$$\mathbf{DECEL} = \frac{(\mathbf{5.12} * \mathbf{DECELERATION})}{\mathbf{1024}}; \text{ Eq.2}$$

Where $10000 =< DECELERATION <= 10000$

Example:

Acceleration=10000; Deceleration=5000; Speed=500; Sampling Rate = 0.5 sec

If $(Speed_{final} - Speed_{initial}) > 0 \Rightarrow ACCEL \Rightarrow$ Eq.1
or $(Speed_{final} - Speed_{initial}) < 0 \Rightarrow DECEL \Rightarrow$Eq.2
TICK = 2 mSec:
I) Increment: $Speed_k = (Speed_{K-1} + ACCEL)$ Eq.1
I) Increment: $Speed_k = (Speed_{K-1} + ACCEL)$ Eq.2

4.6 Acceleration_Deceleration_Equations_in_rad/sec^2

When RMV859 controller perform PROFILE_SPEED_CONTROL , TRAPEZOIDAL_CONTROL, and PROFILE_TRAPEZOIDAL_CONTROL the units of measurement for motion parameters are radians as follows:
Acceleration: Data given in 0.01 rad/sec^2 , (100 = 1 rad/sec^2).
Deceleration: Data given in 0.01 rad/sec^2 , (100 = 1 rad/sec^2).
Speed Rate: Data given in 0.01 rad/sec^1, (100 = 1 rad/sec^1).
Step Number Counter Range: -(2^{31}) = -2,147,483,648 to (2^{31} -1) = +2,147,483, 647

Chapter 5

Limit Switch, HOME, TRIGGER Functions

5.1 ST600uNet_Motor_SEEK_HOME

ST600uNet_Motor_SEEK_HOME (unsigned char `motor_address_mask`,
short int Speed, unsigned short int* `Error_number`);

This function SEEK HOME switch, if Speed $>$ 0 the motor moves CW, otherwise CCW. When limit switch is reached the motor stop. The Operation Mode when this function is called is `SPEED_CONTROL`.

Parameters:
unsigned char *motor_address_mask*: RMV859 Address:($ADDR0 \Rightarrow 1$, $ADDR1 \Rightarrow 2, ADDR2 \Rightarrow 4$, $ADDR3 \Rightarrow 8, ALL \Rightarrow 0$)
short int *Speed*: Where Speed is:
Maximum Seek Home Speed = `MAX_SEEK_HOME_SPEED` = 1000 step/sec;
Minimum Seek Home Speed = `MIN_SEEK_HOME_SPEED` =-1000 step/sec;

`MIN_SEEK_HOME_SPEED` =< Speed =< `MIN_SEEK_HOME_SPEED` ;

Return:
"0": NO ERROR
Parameters Returned:
unsigned short int * `Error_number` = 0.
"1": ERROR
Parameters Returned:
unsigned short int * `Error_number` ="Error Number", please see Errors Section, what was the cause.

5.2 ST600uNet_Motor_Move_From_Limit_SW

ST600uNet_Motor_Move_From_Limit_SW (unsigned char `motor_address_mask`,
short int `number_of_steps`, unsigned short int* `Error_number`);

This function will move out the axis from a limit switch (left or right).
Motion direction must be inverse from the previous that reached the limit switch.

The direction of the motion depend upon the sign of step numbers.

Parameters:

unsigned char *motor_address_mask*: RMV859 Address:($ADDR0 \Rightarrow 1$, $ADDR1 \Rightarrow 2, ADDR2 \Rightarrow 4$, $ADDR3 \Rightarrow 8, ALL \Rightarrow 0$)

short int *number_of_steps*: Where `number_of_steps` is:

Maximum Number of Steps= `LSW_MAX_STEP` = 1000 steps;
Minimum Number of Step = `LSW_MIN_STEPS` = 1 step/sec;

`LSW_MAX_STEPS` >= `number_of_steps` >= `LSW_MIN_STEPS`;

Return:

"0": NO ERROR
Parameters Returned:
unsigned short int * `Error_number` = 0.

"1": ERROR
Parameters Returned:
unsigned short int * `Error_number` ="Error Number", please see Errors Section, what was the cause.

5.3 ST600uNet_MotorTrigger

ST600uNet_MotorTrigger (unsigned char `motor_address_mask`,
unsigned char `trigger_flag`, unsigned short int* `Error_number`);
This function Trigger any motion programmed into RMV859 controller. Belows shows different methods to trigger. This function supports broadcasting addressing (`motor_address_mask`=0), all address will be triggered.

Parameters:

unsigned char *motor_address_mask*: RMV859 Address:($ADDR0 \Rightarrow 1$, $ADDR1 \Rightarrow 2, ADDR2 \Rightarrow 4$, $ADDR3 \Rightarrow 8, ALL \Rightarrow 0$)

short int *trigger_flag*: Different Trigger flags:

`DIRECT_TRIGGER_ON` $\Rightarrow$ 1 : Will trigger intermediately the motion stored. If this flag still un-change every motion in memory will be triggered. In case of Sequential Motion, *"Pivot Controllers"* must be triggered with method.

`DIRECT_TRIGGER_OFF` $\Rightarrow$ 0 : Clear Trigger flag. Any movement stored will be hold until the flag is set again.

`SEQ_TRIGGER_ON` $\Rightarrow$ 2 : HARDWARE Sequential Motion Flag, please refer to "Sequential Motion Functions chapter", to review in details this method.

`SEQ_TRIGGER_OFF` $\Rightarrow$ 3 : HARDWARE Sequential Motion is OFF,

`DIRECT_TRIGGER_ALWAYS` $\Rightarrow$ 4 :In this method RMV859 always is triggered, when the controller is configured as *"Pivot" or as alone "Direct Trigger"*.

Return:

"0": NO ERROR
Parameters Returned:

unsigned short int * `Error_number` = 0.

"1": ERROR

Parameters Returned:

unsigned short int * `Error_number` ="Error Number", please see Errors Section, what was the cause.

Chapter 6

Functions: JOG, Speed Control, Motor Steps Move, STOP, Delay, RESET

6.1 ST600uNet_MotorJOG

ST600uNet_MotorJOG (unsigned char `motor_address_mask`,
short int `speed`,unsigned short int* `bytes_available` ,unsigned short int* `Error_number`);

Parameters:

unsigned char *motor_address_mask*: RMV859 Address:($ADDR0 \Rightarrow 1$, $ADDR1 \Rightarrow 2, ADDR2 \Rightarrow 4$, $ADDR3 \Rightarrow 8, ALL \Rightarrow 0$)

short int *speed*: JOG Speed :
`MAXIMUM_SPEED` >= `speed` >= `MINIMUM_SPEED`;

unsigned short int * *bytes_available*: Return how many words are available.
The maximum buffer to store different speeds is:
`MAX_BUFFER_SIZE_SPPOS_CNTRL` = 10;

Return:

"0": NO ERROR
Parameters Returned:
unsigned short int * `Error_number` = 0.

"1": ERROR
Parameters Returned:
unsigned short int * `Error_number` ="Error Number", please see Errors Section, what was the cause.

6.2 ST600uNet_MotorSpeedControl

ST600uNet_MotorSpeedControl (unsigned char `motor_address_mask`, short int `speed_kind`,
short int `speed`,unsigned short int `acceleration`,unsigned short int `deceleration`,
unsigned short int* `bytes_available`, unsigned short int* `Error_number`);
This function is used when Motion Mode has been selected: `SPEED_CONTROL`.

Parameters:

unsigned char *motor_address_mask*: RMV859 Address:($ADDR0 \Rightarrow 1$, $ADDR1 \Rightarrow 2, ADDR2 \Rightarrow 4$,

$ADDR3 \Rightarrow 8, ALL \Rightarrow 0$)

short int *speed_kind*: What kind of slew rate is sent.
SPEED_RATE $\Rightarrow$ $'SR'$;
SPEED_INITIAL $\Rightarrow$ $'SI'$;
SPEED_FINAL $\Rightarrow$ $'SF'$;

short int *speed*: New slew rate to be programmed.
MAXIMUM_SPEED >= speed >= MINIMUM_SPEED;

short int *acceleration*: Acceleration calculated in Section 4.5.
MAXIMUM_ACCEL) >= acceleration >= MINIMUM_ACCEL_DECEL;

short int *deceleration*: Deceleration calculated in Section 4.5.
MAXIMUM_DECEL >= deceleration >= MINIMUM_ACCEL_DECEL;

unsigned short int * *bytes_available*: Return how many words are available.
The maximum buffer to store different speeds is:
MAX_BUFFER_SIZE_SPPOS_CNTRL = 10;

Return:
"0": NO ERROR
Parameters Returned:
unsigned short int * Error_number = 0.
"1": ERROR
Parameters Returned:
unsigned short int * Error_number ="Error Number", please see Errors Section, what was the cause.

6.3 ST600uNet_MotorStepsMove

ST600uNet_MotorStepsMove (unsigned char motor_address_mask, long step_number, short int speed,unsigned short int acceleration,unsigned short int deceleration, unsigned short int* Error_number);
This function provide a trapezoidal motion, which direction is determined by the sign of step_number. Acceleration-Deceleration and Speed the units measurement and parameter calculations are described in Section 4.6.

Parameters:
unsigned char *motor_address_mask*: RMV859 Address:($ADDR0 \Rightarrow 1$, $ADDR1 \Rightarrow 2, ADDR2 \Rightarrow 4$, $ADDR3 \Rightarrow 8, ALL \Rightarrow 0$)

long *step_number*: The axis move CW if step numbers are positive, and CCW if are negative.
Step Number Counter Range: -(2^{31}) = -2,147,483,648 to (2^{31} -1) = +2,147,483, 647

short int *speed*: Slew Rate after acceleration has been applied.
The unit of measurement is = 0.01 rad/sec, a speed of 100 = 1 rad/sec.
MAXIMUM_SPEED >= speed >= MINIMUM_SPEED;

short int *acceleration*: Acceleration calculated in Section 4.6.
MAXIMUM_ACCEL) >= acceleration >= MINIMUM_ACCEL_DECEL;
The unit of measurement is = 0.01 ad/sec^2, a acceleration of 100 = 1 rad/sec^2.

short int *deceleration*: Slew Rate will Decelerate until speed is zero, Deceleration is calculated in Section 4.6.

MAXIMUM_DECEL >= deceleration >= MINIMUM_ACCEL_DECEL;

The unit of measurement is = 0.01 ad/sec^2, a deceleration of 100 = 1 rad/sec^2.

Return:

"0": NO ERROR

Parameters Returned:

unsigned short int * Error_number = 0.

"1": ERROR

Parameters Returned:

unsigned short int * Error_number ="Error Number", please see Errors Section, what was the cause.

6.4 ST600uNet_MotorSoftwareStop

ST600uNet_MotorStepsMove (unsigned char motor_address_mask, unsigned char stop_parameters, unsigned short int* Error_number);

Parameters:

unsigned char *motor_address_mask*: RMV859 Address:($ADDR0 \Rightarrow 1$, $ADDR1 \Rightarrow 2, ADDR2 \Rightarrow 4$, $ADDR3 \Rightarrow 8, ALL \Rightarrow 0$)

unsigned char *stop_parameters*:Three different kind of STOP:

STOP_MOTOR_ONLY $\Rightarrow$ 1: Normal Motion Stop

STOP_MOTOR_RESET_CMDS $\Rightarrow$ 2: Motor Stop, and RMV859 controller will initialize all motion parameters to the same after "power on reset".

STOP_MOTOR_RESET_FIFO $\Rightarrow$ 3: Motor Stop, and RMV859 controller will initialize internal FIFO, and disabled it.

Return:

"0": NO ERROR

Parameters Returned:

unsigned short int * Error_number = 0.

"1": ERROR

Parameters Returned:

unsigned short int * Error_number ="Error Number", please see Errors Section, what was the cause.

6.5 ST600uNet_Set_Motion_Delay

ST600uNet_Set_Motion_Delay (unsigned char motor_address_mask, unsigned char select_msec_or_usec, unsigned short int delay, unsigned short int* Error_number);

Parameters:

unsigned char *motor_address_mask*: RMV859 Address:($ADDR0 \Rightarrow 1$, $ADDR1 \Rightarrow 2, ADDR2 \Rightarrow 4$, $ADDR3 \Rightarrow 8, ALL \Rightarrow 0$)

unsigned char *select_msec_or_usec*: Delay can be in millisecond or micro-second:

DELAY_IN_MSEC $\Rightarrow$ 1: Delay unit is in Millisecond;

DELAY_IN_USEC $\Rightarrow$ 2: Delay unit is in Microsecond;

unsigned short int *delay*: Delay value.

Return:

"0": NO ERROR

Parameters Returned:

unsigned short int * `Error_number` = 0.

"1": ERROR

Parameters Returned:

unsigned short int * `Error_number` ="Error Number", please see Errors Section, what was the cause.

6.6 ST600uNet_Set_StepNumbersinPositionControl

ST600uNet_Set_StepNumbersinPositionControl (unsigned char `motor_addr_mask`, unsigned short int `which_mode`, long `step_numbers`, short int `speed`, unsigned short int `acceleration`, unsigned short int `deceleration`, unsigned short int * `Error_number`);

This function is used in `STEP_NUM_POSITION_CNTRL` only, further firmware versions will allow different positions control. The parameters units are calculated in Section 4.5:

Speed = step/sec; Acceleration = Deceleration = step/sec^2

Parameters:

unsigned char *motor_address_mask*: RMV859 Address:($ADDR0 \Rightarrow 1$, $ADDR1 \Rightarrow 2, ADDR2 \Rightarrow 4$, $ADDR3 \Rightarrow 8, ALL \Rightarrow 0$)

unsigned char *which_mode*: Only one mode is allowed:
`STEP_NUM_POSITION_CNTR` $\Rightarrow$ $'NP'$: Position Control;

long *step_numbers*: The axis move CW if step numbers are positive, and CCW if are negative.

Step Number Counter Range: -(2^{31}) = -2,147,483,648 to (2^{31} -1) = +2,147,483, 647

short int *speed*: New slew rate to be programmed.
`MAXIMUM_SPEED` >= `speed` >= `MINIMUM_SPEED`;

unsigned short int *acceleration*: Acceleration calculated in Section 4.5.
`MAXIMUM_ACCEL)` >= `acceleration` >= `MINIMUM_ACCEL_DECEL`;

unsigned short int *deceleration*: Deceleration calculated in Section 4.5.
`MAXIMUM_DECEL` >= `deceleration` >= `MINIMUM_ACCEL_DECEL`;

Return:

"0": NO ERROR

Parameters Returned:

unsigned short int * `Error_number` = 0.

"1": ERROR

Parameters Returned:

unsigned short int * `Error_number` ="Error Number", please see Errors Section, what was the cause.

6.7 ST600uNet_Reset

ST600uNet_Reset (unsigned char `motor_addr_mask`, unsigned short int `what_to_reset`, unsigned short int* `Error_number`);

This function provide two options on RESET: i) `RESET_MOTION_REGISTERS`, ii) `RESET_RMV859_CONTROLLER`, both methods are described belows:

Parameters:

unsigned char `motor_address_mask`: RMV859 Address:($ADDR0 \Rightarrow 1$, $ADDR1 \Rightarrow 2, ADDR2 \Rightarrow 4$, $ADDR3 \Rightarrow 8, ALL \Rightarrow 0$)

unsigned short int `what_to_reset`: Two different resets:

`RESET_MOTION_REGISTERS` $\Rightarrow$ $'QR'$: Reset only internal motion parameters: FIFO, Position buffer, Acceleration-Deceleration buffer, and Speed buffer.

`RESET_RMV859_CONTROLLER` $\Rightarrow$ $'QF'$: Similar than "Power ON RESET".

Return:

"0": NO ERROR

Parameters Returned:

unsigned short int * `Error_number` = 0.

"1": ERROR

Parameters Returned:

unsigned short int * `Error_number` ="Error Number", please see Errors Section, what was the cause.

Chapter 7

Adjust Current, System GAIN, Holding Torque

7.1 ST600uNet_Adjust_Current

ST600uNet_Adjust_Current (unsigned char *motor_addr_mask*, unsigned short int *flag_zero_or_maximum_current*, unsigned short int *winding_max_current*, unsigned short int * *Error_number*); This function will adjust chopper driver current for the motor parameters' programmed. Same function is used to adjust current by : Zero, Half, and Maximum. It is very important to call this function in "precise order" :

I) : $\Rightarrow$ Adjustment by Zero Winding Current
II) : $\Rightarrow$ Adjustment by Half Winding Current
III) : $\Rightarrow$ Adjustment by Maximum Winding Current

Parameters:

unsigned char *motor_address_mask*: RMV859 Address:($ADDR0 \Rightarrow 1$, $ADDR1 \Rightarrow 2$, $ADDR2 \Rightarrow 4$, $ADDR3 \Rightarrow 8$, $ALL \Rightarrow 0$)

unsigned short int flag_zero_or_maximum_current: This variable adjusts winding current according motor's parameters.

`ADJUST_CURRENT_BY_ZERO` $\Rightarrow$ 1;
`ADJUST_CURRENT_BY_MIDDLE` $\Rightarrow$ 3;
`ADJUST_CURRENT_BY_MAXIMUM` $\Rightarrow$ 2;

unsigned short int winding_max_current:The maximum motor winding current.

Return:

"0": NO ERROR
Parameters Returned:
unsigned short int * `Error_number` = 0.

"1": ERROR
Parameters Returned:
unsigned short int * `Error_number` ="Error Number", please see Errors Section, what was the cause.

7.2 ST600uNet_Change_SystemGAIN

ST600uNet_Change_SystemGAIN (unsigned char `motor_addr_mask`, unsigned char `set_or_read_system_gain`, unsigned short int `new_system_gain`, unsigned short int * `actual_gain`, unsigned short int* `Error_number`); This function change the Gain Proportional K for PI

controller. The default value is 7200 calculated on Motor power of 24 Volts.

Parameters:

unsigned char *motor_address_mask*: RMV859 Address:($ADDR0 \Rightarrow 1$, $ADDR1 \Rightarrow 2$, $ADDR2 \Rightarrow 4$, $ADDR3 \Rightarrow 8$, $ALL \Rightarrow 0$)

unsigned char set_or_read_system_gain: Write or Read System GAIN.

`READ_SYSTEM_GAIN` $\Rightarrow$ 1;

`READ_SYSTEM_GAIN` $\Rightarrow$ 2;

unsigned short int new_system_gain: New Value.

Return:

"0": NO ERROR

Parameters Returned:

unsigned short int * `Error_number` = 0.

"1": ERROR

Parameters Returned:

unsigned short int * `Error_number` ="Error Number", please see Errors Section, what was the cause.

7.3 ST600uNet_Set_HoldingTorque

ST600uNet_Set_HoldingTorque (unsigned char `motor_addr_mask`, unsigned short int `which_holding_current`, unsigned short int * `Error_number`); This function setup a programmable percentage of a full winding current as holding torque.

`HOLDING_TORQUE1` $\Rightarrow$ $'H1'$ Full Winding Current;

`HOLDING_TORQUE2` $\Rightarrow$ $'H2'$ 1/4 of Full Winding Current;

`HOLDING_TORQUE3` $\Rightarrow$ $'H3'$ 1/2 of Full Winding Current;

`HOLDING_TORQUE4` $\Rightarrow$ $'H4'$ 3/4 of Full Winding Current;

Return:

"0": NO ERROR

Parameters Returned:

unsigned short int * `Error_number` = 0.

"1": ERROR

Parameters Returned:

unsigned short int * `Error_number` ="Error Number", please see Errors Section, what was the cause.

Chapter 8

FIFO Functions: Write, Space Available, Delete

8.1 ST600uNet_FIFOWriteMotion

ST600uNet_FIFOWriteMotion (unsigned char `motor_address_mask`, unsigned short int `buffer`, unsigned short int `buffer_size`, unsigned short int `fifo_command`, unsigned short int * `fifo_data_available`, unsigned short int* `Error_number`);
This function writes commands to internal FIFO, the maximum capacity are 700 words. Only two different motions are allowed in this version firmware: `FIFO_PROFILE_SPEED_CONTROL` and `FIFO_PROFILE_TRAPEZOIDAL_CONTROL`.

Parameters:
unsigned char *motor_address_mask*: RMV859 Address:($ADDR0 \Rightarrow 1$, $ADDR1 \Rightarrow 2, ADDR2 \Rightarrow 4$, $ADDR3 \Rightarrow 8, ALL \Rightarrow 0$)
unsigned short int *buffer: Word Buffer which commands are sent.
unsigned short int buffer_size: Length of the buffer array,
the maximum length in one call is `FIFO_MAX_SIZE_ONE_TIME` 28 words;
unsigned short int fifo_command: Two FIFO commands are allowed `FIFO_PROFILE_SPEED_CONTROL` and `FIFO_PROFILE_TRAPEZOIDAL_CONTROL`.
unsigned short int * fifo_data_available: Return words available in FIFO.
Return:
"0": NO ERROR
Parameters Returned:
unsigned short int * `Error_number` = 0.
"1": ERROR
Parameters Returned:
unsigned short int * `Error_number` ="Error Number", please see Errors Section, what was the cause.

8.2 ST600uNet_FIFOGetFreeSpace

ST600uNet_FIFOGetFreeSpace (unsigned char `motor_address_mask`,
unsigned short int * `fifo_data_available`, unsigned short int* `Error_number`);
Parameters:
unsigned char *motor_address_mask*: RMV859 Address:($ADDR0 \Rightarrow 1$, $ADDR1 \Rightarrow 2, ADDR2 \Rightarrow 4$,

$ADDR3 \Rightarrow 8, ALL \Rightarrow 0$)

unsigned short int *fifo_data_available: Return words available in FIFO.

Return:

"0": NO ERROR

Parameters Returned:

unsigned short int * `Error_number` = 0.

"1": ERROR

Parameters Returned:

unsigned short int * `Error_number` ="Error Number", please see Errors Section, what was the cause.

8.3 ST600uNet_FIFO_DELETE

ST600uNet_FIFO_DELETE (unsigned char `motor_address_mask`, unsigned short int * `fifo_data_available`' unsigned short int* `Error_number`); ***Parameters***:

unsigned char *motor_address_mask*: RMV859 Address:($ADDR0 \Rightarrow 1$, $ADDR1 \Rightarrow 2, ADDR2 \Rightarrow 4$, $ADDR3 \Rightarrow 8, ALL \Rightarrow 0$)

unsigned short int *fifo_data_available: Return words available in FIFO, after been deleted must return 700 words.

Return:

"0": NO ERROR

Parameters Returned:

unsigned short int * `Error_number` = 0.

"1": ERROR

Parameters Returned:

unsigned short int * `Error_number` ="Error Number", please see Errors Section, what was the cause.

Chapter 9

Report Functions: Motion Flags, Counters, Status.

9.1 ST600uNet_Get_Internal_Report

ST600uNet_Get_Internal_Report (unsigned char `motor_addr_mask`, unsigned short int * `parameters_array`, unsigned char `array_length`, unsigned short int `what_kind_of_report`, unsigned short int* `Error_number`);
This function return different motor report according to the variable `what_kind_of_report`, It is responsibility of the application software to create an *unsigned short int array[Indx]*, where Indx=20 for example. Only two reports are supported in this firmware version: `CHOPPER_DRIVER_CURRENT_REPORT` and `MOTOR_PARAMETERS_REPORT`.

Parameters:

unsigned char *motor_address_mask*: RMV859 Address:($ADDR0 \Rightarrow 1$, $ADDR1 \Rightarrow 2, ADDR2 \Rightarrow 4$, $ADDR3 \Rightarrow 8, ALL \Rightarrow 0$)

unsigned short int *parameters_array: Buffer array Report

unsigned short int array_length: An array size of 20 is acceptable.

unsigned short int what_kind_of_report: The following description shows each report parameters returned and its flags.

I) `CHOPPER_DRIVER_CURRENT_REPORT`: $Length \Rightarrow 1\,WORD$;
In Table:1 shows details the flags involved in this report.

II) `MOTOR_PARAMETERS_REPORT`: $Length \Rightarrow 6\,WORD$
In Table:2 shows details the flags involved in this report.

CHOPPER_DRIVER_CURRENT_REPORT		
WORD I		
Flag Name:	Bit Position	Description
CURRENT_PARAMETERS_SAVED	0x0001	Current Parameters Saved.
DRIVER_ZERO_CURRENT_ADJUSTED	0x0002	Zero current calibration has been done.
DRIVER_MAX_CURRENT_ADJUSTED	0x0004	Maximum current calibration has been done.
MOTOR_PARAM_SAVED	0x0008	Manufacturer Motor Parameters saved.
CHOPPER_DRIVER_CAN_NOT_SET_MAX_MIN	0x0010	Error in adjusting chopper current.
CHOPPER_OVER_TEMPERATURE	0x0020	Set when chopper driver temperature exceed max allowed.
CHOPPER_OVER_CURRENT_UNDERVOL	0x0040	Set when chopper driver voltage is lower than $< 18\, volts$.
DRIVER_HALF_CURRENT_ADJUSTED	0x0080	Middle point current calibration has been done.
ERROR_READ_FLASH_MOTOR_PARAMETERS	0x0100	Set when checksum in Flash Memory is wrong, in motor parameters section.
ERROR_WRITE_FLASH_CURRENT_PARAMETERS	0x0200	Set when an error in writing motor parameters into flash memory.
ERROR_WRITE_FLASH_CURRENT_PARAMETERS	0x0800	Set when an error in writing current chopper driver parameters into slash memory.
RESERVED	0x1000	Bit no used
RESERVED	0x2000	Bit no used
RESERVED	0x4000	Bit no used
RESERVED	0x8000	Bit no used

TABLE:1 Flags CHOPPER_DRIVER_CURRENT_REPORT

II) MOTOR_PARAMETERS_REPORT: $Length \Rightarrow 6\, WORD$

MOTOR_PARAMETERS_REPORT		
I) WORD $\Rightarrow$ motor_internals_flags_1		
Flag Name:	Bit Position	Description
DIGITAL_INPUT_REPORT	0x0001	RMV859 Isolated Digital Input.
MOTOR_RUNNING	0x0002	Motor running '1', '0' is not running.
MOTION_COMPLETED	0x0004	Set '1' when motion is completed. '0' Motor running.
MOTOR_STOPPED	0x0008	Set when motor is has been STOPPED.
LSW_RIGHT	0x0010	Limit Switch Right Input has been reached.
LSW_LEFT	0x0020	Limit Switch Left Input has been reached.
HARDWARE_STOP	0x0040	External Hardware Switch STOP .
AT_HOME	0x0080	The axis is AT HOME switch.
EXT_TRIGGER	0x0100	Set when Extern Trigger push button.
RMV_TASKER_TRIGGER	0x0200	rmv_Tasker Module TRIGGER pulse.
RESERVED	0x0400	Bit no used
RESERVED	0x0800	Bit no used
RESERVED	0x1000	Bit no used
RESERVED	0x2000	Bit no used
RESERVED	0x4000	Bit no used
RESERVED	0x8000	Bit no used

MOTOR_PARAMETERS_REPORT		
II) WORD ⇒ motor_current_flags		
Flag Name:	Bit Position	Description
CURRENT_PARAMETERS_SAVED	0x0001	Current Parameters Saved.
DRIVER_ZERO_CURRENT_ADJUSTED	0x0002	Zero current calibration has been done.
DRIVER_MAX_CURRENT_ADJUSTED	0x0004	Maximum current calibration has been done.
MOTOR_PARAM_SAVED	0x0008	Manufacturer Motor Parameters saved.
CHOPPER_DRIVER_CAN_NOT_SET_MAX_MIN	0x0010	Error in adjusting chopper current.
CHOPPER_OVER_TEMPERATURE	0x0020	Set when chopper driver temperature exceed max allowed.
CHOPPER_OVER_CURRENT_UNDERVOL	0x0040	Set when chopper driver voltage is lower than $< 18\ volts$.
DRIVER_HALF_CURRENT_ADJUSTED	0x0080	Middle point current calibration has been done.
ERROR_READ_FLASH_MOTOR_PARAMETERS	0x0100	Set when checksum in Flash Memory is wrong, in motor parameters section.
ERROR_WRITE_FLASH_CURRENT_PARAMETERS	0x0200	Set when an error in writing motor parameters into flash memory.
ERROR_WRITE_FLASH_CURRENT_PARAMETERS	0x0800	Set when an error in writing current chopper driver parameters into slash memory.
RESERVED	0x1000	Bit no used
RESERVED	0x2000	Bit no used
RESERVED	0x4000	Bit no used
RESERVED	0x8000	Bit no used

MOTOR_PARAMETERS_REPORT		
III) WORD ⇒ motor_config_1		
Flag Name:	Bits Position	Macro Description
BIPOLAR_UNIPOLAR	0x0001	MOTOR_KIND(X)= '0' Bipolar, '1' Unipolar.
OPERATION_MODE:	0x000E	OPERATION_MODE(X) = Step Mode .
FIXED_VOLTAGE		OPERATION_MODE(X) ="0".
FIXED_CURRENT		OPERATION_MODE(X) ="2".
CLOSE_LOOP		OPERATION_MODE(X) ="3".
ENCODER_CLOSE_POS	Not Enabled	OPERATION_MODE(X) ="4".
ENCODER_CLOSE_SPEED	Not Enabled	OPERATION_MODE(X) ="5".
STEP_RESOLUTION:	0x00E0	STEP_RESOLUTION(X) = Step Resolution .
FULL_STEP		STEP_RESOLUTION(X) ="0".
HALH_STEP		STEP_RESOLUTION(X) ="1".
1_4_STEP		STEP_RESOLUTION(X) ="2".
1_8_STEP		STEP_RESOLUTION(X) ="3".
1_16_STEP		STEP_RESOLUTION(X) ="4".
1_32_STEP		STEP_RESOLUTION(X) ="5".
1_64_STEP		STEP_RESOLUTION(X) ="6"
1_128_STEP		STEP_RESOLUTION(X) ="7".
1_256_STEP		STEP_RESOLUTION(X) ="8".
RESERVED	0xFF00	Bits no used

MOTOR_PARAMETERS_REPORT		
IV) WORD ⇒ motor_config_2		
Flag Name:	Bits Position	Macro Description
CONTROL_MODE: SPEED_CONTROL POSITION_CONTROL PROFILE_SPEED_CONTROL S_CURVE_CONTROL TRAPEZOIDAL_CONTROL PROFILE_TRAPEZOIDAL_CONTROL	0x000F Not Enabled	CONTROL_MODE(X) = Control Mode . CONTROL_MODE(X) =”1”. CONTROL_MODE(X) =”2”. CONTROL_MODE(X) =”3”. CONTROL_MODE(X) =”4”. CONTROL_MODE(X) =”5”. CONTROL_MODE(X) =”6”.
SPEED_SAMPLING_RATE: SAMPLING_RATE_TWO_MSEC SAMPLING_RATE_ONE_MSEC SAMPLING_RATE_FIVE_MSEC	0x0030	SPEED_CONTROL_MODE(X)=Sampling Rate. SPEED_CONTROL_MODE(X)=”0”. SPEED_CONTROL_MODE(X)=”1”. SPEED_CONTROL_MODE(X)=”2”.
FIFO_ENABLED: ENABLED_DISABLE	0x00C0	FIFO_FLAGS(X)=FIFO Flags. FIFO_FLAGS(X) ⇒ ”1”$ENABLE$;”0”$DISABLE$.
RESERVED	0xFF00	Bits no used

MOTOR_PARAMETERS_REPORT		
V) WORD ⇒ sequential_motion_config		
Flag Name:	Bits Position	Macro Description
SEQUENTIAL_MOTION_ENABLE :	0x0001	SEQ_MOTION_FLAG(X) ⇒ ”1”$ENABLE$;”0”$DISABLE$.
PIVOT_CONTROLLER	0x0080	PIVOT_CONTROLLER(X) ⇒ ”1”$ENABLE$;”0”$DISABLE$.
SEQ_MOTION_CONTROLLERS_ENABLED	0x0F00	SEQ_MOTION_CONTROLLERS_ENABLED(X) = RMV859 Controller Enabled. ”1” Enable, ”0” disable
SEQ_MOTION_MASK_VALUE	0xF000	SEQ_MOTION_MASK_VALUE(X) ⇒ Motor Running:”0”, Motion Completed:”1”

MOTOR_PARAMETERS_REPORT		
VI) WORD ⇒ PI_GAIN		
Flag Name:	Bits Position	Macro Description
PI_GAIN: System GAIN	0xFFFF	PI_GAIN = ”unsigned short int”, System GAIN in PI controller

TABLE:2 Flags MOTOR_PARAMETERS_REPORT

Return:

"0": NO ERROR

Parameters Returned:

unsigned short int * `Error_number` = 0.

"1": ERROR

Parameters Returned:

unsigned short int * `Error_number` ="Error Number", please see Errors Section, what was the cause.

9.2 ST600uNet_Get_StepsCounter_MotorF_Report

ST600uNet_Get_StepsCounter_MotorF_Report (unsigned char `motor_addr_mask`, unsigned short int * `motor_internal_flags_1`, unsigned short int * `motor_internal_flags_2`, long * `step_counter`, unsigned short int* `Error_number`);

This function return the following motion registers and flags: 1) `motor_internals_flags_1`, 2) `motor_current_flags`, and a 32 bits Step Counter Register.

Parameters:

unsigned char *motor_address_mask*: RMV859 Address:($ADDR0 \Rightarrow 1$, $ADDR1 \Rightarrow 2$, $ADDR2 \Rightarrow 4$, $ADDR3 \Rightarrow 8$, $ALL \Rightarrow 0$)

unsigned short int motor_internal_flags_1: This function return the flags that are referred to real time motion.

MOTOR_INTERNAL_FLAGS_1		
1 WORD ⇒ motor_internals_flags_1		
Flag Name:	Bit Position	Description
DIGITAL_INPUT_REPORT	0x0001	RMV859 Isolated Digital Input.
MOTOR_RUNNING	0x0002	Motor running '1', '0' is not running.
MOTION_COMPLETED	0x0004	Set '1' when motion is completed. '0' Motor running.
MOTOR_STOPPED	0x0008	Set when motor is has been STOPPED.
LSW_RIGHT	0x0010	Limit Switch Right Input has been reached.
LSW_LEFT	0x0020	Limit Switch Left Input has been reached.
HARDWARE_STOP	0x0040	External Hardware Switch STOP .
AT_HOME	0x0080	The axis is AT HOME switch.
EXT_TRIGGER	0x0100	Set when Extern Trigger push button.
RMV_TASKER_TRIGGER	0x0200	`rmv_Tasker` Module TRIGGER pulse.
RESERVED	0x0400	Bit no used
RESERVED	0x0800	Bit no used
RESERVED	0x1000	Bit no used
RESERVED	0x2000	Bit no used
RESERVED	0x4000	Bit no used
RESERVED	0x8000	Bit no used

unsigned short int motor_internal_flags_2: This function return the flags are referred to motor parameter setup, over current, over temperature, and read-write Flash memory status.

MOTOR_INTERNAL_FLAGS_2		
1 WORD ⇒ motor_internal_flags_2		
Flag Name:	Bit Position	Description
CURRENT_PARAMETERS_SAVED	0x0001	Current Parameters Saved.
DRIVER_ZERO_CURRENT_ADJUSTED	0x0002	Zero current calibration has been done.
DRIVER_MAX_CURRENT_ADJUSTED	0x0004	Maximum current calibration has been done.
MOTOR_PARAM_SAVED	0x0008	Manufacturer Motor Parameters saved.
CHOPPER_DRIVER_CAN_NOT_SET_MAX_MIN	0x0010	Error in adjusting chopper current.
CHOPPER_OVER_TEMPERATURE	0x0020	Set when chopper driver temperature exceed max allowed.
CHOPPER_OVER_CURRENT_UNDERVOL	0x0040	Set when chopper driver voltage is lower than $< 18\ volts$.
DRIVER_HALF_CURRENT_ADJUSTED	0x0080	Middle point current calibration has been done.
ERROR_READ_FLASH_MOTOR_PARAMETERS	(*) 0x0100	Set when checksum in Flash Memory is wrong, in motor parameters section.
ERROR_WRITE_FLASH_CURRENT_PARAMETERS	(*) 0x0200	Set when an error in writing motor parameters into flash memory.
ERROR_WRITE_FLASH_CURRENT_PARAMETERS	(*) 0x0800	Set when an error in writing current chopper driver parameters into slash memory.
RESERVED	0x1000	Bit no used
RESERVED	0x2000	Bit no used
RESERVED	0x4000	Bit no used
RESERVED	0x8000	Bit no used

Note3: (*) When Error occur RMV959 will not RESET its self.

long step_counter: 32 bits Step Counter Register.

Return:

"0": NO ERROR

Parameters Returned:

unsigned short int * `Error_number` = 0.

"1": ERROR

Parameters Returned:

unsigned short int * `Error_number` ="Error Number", please see Errors Section, what was the cause.

Chapter 10

Sequential Trigger Functions

10.1 ST600uNet_Set_Sequential_Motion

ST600uNet_Set_Sequential_Motion (unsigned char `motor_addr_mask`,
unsigned char `sequential_controllers_enable_addr`, unsigned char `pivot_controller`,
unsigned short int* `Error_number`);

This function will initialize "A Sequential Motion Movement" to the controller address which the command is sent. This function must be sent at the bigging of the sequential motion session.
If a new set of controllers need to be enabled-disabled, this function must be called again. The Controllers Enabled are set in: `sequential_controllers_enable_addr`, ON $\Rightarrow$ 1, OFF $\Rightarrow$ 0; Pivot controller: will be triggered by himself (trough a normal Trigger Command or External Trigger push button) and MASK Interrupt register identical to motion completed masks programmed in `ST600uNet_Sequential_Motion_Interrupt_MASK`.

Parameters:

unsigned char *motor_address_mask*: RMV859 Address:($ADDR0 \Rightarrow 1$, $ADDR1 \Rightarrow 2, ADDR2 \Rightarrow 4$, $ADDR3 \Rightarrow 8, ALL \Rightarrow 0$)

unsigned char sequential_controllers_enable_addr: This parameter enables RMM859 controller for Sequential Trigger, ON $\Rightarrow$ 1, OFF $\Rightarrow$ 0;

Sequential Controllers Enable Address								
1 BYTE $\Rightarrow$ sequential_controllers_enable_addr								
Enable Addr.	X	X	X	X	ADDR3	ADDR2	ADDR1	ADDR0
0,1	X	X	X	X	0	0	1	1
0,1,2	X	X	X	X	0	1	1	1
0,1,2,3	X	X	X	X	1	1	1	1

unsigned char pivot_controller:Select which RMV859 controller will work as "Pivot Controller". Only one Pivot controller can be sent in one command.

Pivot Controller								
1 BYTE pivot_controller								
Enable Addr.	X	X	X	X	ADDR3	ADDR2	ADDR1	ADDR0
0	X	X	X	X	0	0	0	1
1	X	X	X	X	0	0	1	0
2	X	X	X	X	0	1	0	0
3	X	X	X	X	1	0	0	0

***Return**:*

"0": NO ERROR

Parameters Returned:

unsigned short int * `Error_number` = 0.

"1": ERROR

Parameters Returned:

unsigned short int * `Error_number` ="Error Number", please see Errors Section, what was the cause.

10.2 ST600uNet_Sequential_Motion_Interrupt_MASK

ST600uNet_Sequential_Motion_Interrupt_MASK (unsigned char `motor_addr_mask`, unsigned char `matching_kte`, unsigned short int* `Error_number`);
This function allows hardware trigger for multiple RMV859 controllers according to a pre-programmed sequence in `matching_kte`. The motor is triggered when "the value in `matching_kte`" is identical with the inputs from Motion Completed from the another RMV859 controllers. This function can be called when motor is running.

`Motor is Running ⇒ 0, Motion Completed ⇒ 1`

***Parameters**:*

***unsigned char** `motor_address_mask`*: RMV859 Address:($ADDR0 \Rightarrow 1$, $ADDR1 \Rightarrow 2, ADDR2 \Rightarrow 4$, $ADDR3 \Rightarrow 8, ALL \Rightarrow 0$)

unsigned char matching_kte: Sequential Motion Mask Bits:3,2,1,0,

MASK KTE							
1 BYTE matching_kte							
7	6	5	4	ADDR3	ADDR2	ADDR1	ADDR0
X	X	X	X	Mot.Cmpltd.⇒ 1 Mot.Running⇒ 0	Mot.Cmpltd.⇒ 1 Mot.Running⇒ 0	Mot.Cmpltd.⇒ 1 Mot.Running⇒ 0	Mot.Cmpltd.⇒ 1 Mot.Running⇒ 0

***Return**:*

"0": NO ERROR

Parameters Returned:

unsigned short int * `Error_number` = 0.

"1": ERROR

Parameters Returned:

unsigned short int * `Error_number` ="Error Number", please see Errors Section, what was the cause.

Example:

RMV859 Address Selected= ADDR0 ,"ADDR1", ADDR3. Pivot controller=ADDR0;
The three controllers will be triggered when:

i) Pivot Controller=ADDR0 ⇒ Motor Running ⇒ "0"
ii) ADDR1 ⇒ Motion Completed ⇒ "1",
ii) ADDR3 ⇒ Motion Completed ⇒ "1",
matching_kte = [0,0,0,0,1,x,1,0] ;

Before trigger is asserted:

$ADDR0 \Rightarrow 1 = MotionCompleted\ (Pivot\ Controller)$

$ADDR1 \Rightarrow 1 = MotionCompleted$

$ADDR3 \Rightarrow 1 = MotionCompleted$

When: $ADRR0\ goes\ 1 \Downarrow 0\ \ ADDR0, ADDR1, ADDR3 \Rightarrow$ `MOTOR_RUNNING`

Note4: Any `kind of trigger synchronization` for complex curves can be set with this method. Triggering one controller the rest will be trigger with No Jitter, which allows very high resolution motions.

10.3 ST600uNet_Disable_Sequential_Motion

ST600uNet_Disable_Sequential_Motion (unsigned char `motor_addr_mask`, unsigned short int* `Error_number`); This function disable Sequential Trigger Motion, after this command is executed RMV859 controller will be triggered by: `DIRECT_TRIGGER_ON`, or `DIRECT_TRIGGER_ALWAYS`, or `EXT_TRIGGER` push button.

Parameters:

unsigned char *motor_address_mask*: RMV859 Address:($ADDR0 \Rightarrow 1$, $ADDR1 \Rightarrow 2, ADDR2 \Rightarrow 4$, $ADDR3 \Rightarrow 8, ALL \Rightarrow 0$)

Return:

"0": NO ERROR

Parameters Returned:

unsigned short int * `Error_number` = 0.

"1": ERROR

Parameters Returned:

unsigned short int * `Error_number` ="Error Number", please see Errors Section, what was the cause.

Chapter 11

Analog Digital IO's Functions

11.1 ST600uNet_Digital_Input_Output

ST600uNet_Digital_Input_Output (unsigned char `motor_addr_mask`, unsigned char `output_value`, unsigned char * `input_value`, unsigned short int* `Error_number`);
This function write to a isolated digital outputs, and read a isolated digital input. Some address have two Outputs others ones. In regards to digital Input the total number is "four" each address has one isolated Input. The total of digital Outputs are "six", please see the details bellows.

Parameters:

unsigned char *motor_address_mask*: RMV859 Address:($ADDR0 \Rightarrow 1$, $ADDR1 \Rightarrow 2, ADDR2 \Rightarrow 4$, $ADDR3 \Rightarrow 8, ALL \Rightarrow 0$)

unsigned char output_value: The digital outputs description is as follows:

OUTPUT			
1 BYTE output_value			
ADDR3	ADDR2	ADDR1	ADDR0
OUTPUT5⇒ 0*or*1	OUTPUT4⇒ 0*or*1	OUTPUT2⇒ 0*or*1 OUTPUT3⇒ 0*or*1	OUTPUT0⇒ 0*or*1 OUTPUT1⇒ 0*or*1

Example: Address ADDR0, write OUTPUT0 and OUTPUT1 to HIGH
`output_value` ⇒ 0x03

unsigned char * input_value: Each RMV859 controller has "one" isolated input. The value returned in *input_value* is "0" or "1".

Return:

"0": NO ERROR
Parameters Returned:
unsigned short int * `Error_number` = 0.

"1": ERROR
Parameters Returned:
unsigned short int * `Error_number` ="Error Number", please see Errors Section, what was the cause.

11.2 ST600uNet_Analog_Input

ST600uNet_Analog_Input (unsigned char `motor_addr_mask`, float * `adc_channel`, unsigned short int* `Error_number`);
Three Analog Channels are ready in the ST600μNET to remeasure the following: 0 to +5V, 4-20mA, and heat-sink temperature. Monitoring the chopper driver temperature in real time is very important feature when ST600μNET operates in very harsh environment.

The following description show which channel is in its address:

$ADDR0 \Rightarrow$ NO Analog Channel is connected.
$ADDR1 \Rightarrow$ Heat-Sink Temperature Channel.
$ADDR2 \Rightarrow$ 4-20mA Channel.
$ADDR3 \Rightarrow$ 0 to +5V Channel.

Parameters:

unsigned char *motor_address_mask*: RMV859 Address:($ADDR0 \Rightarrow 1$, $ADDR1 \Rightarrow 2, ADDR2 \Rightarrow 4$, $ADDR3 \Rightarrow 8, ALL \Rightarrow 0$)

float * adc_channel: Analog returned from RMV859 controller.

Return:

"0": NO ERROR
Parameters Returned:
unsigned short int * `Error_number` = 0.

"1": ERROR
Parameters Returned:
unsigned short int * `Error_number` ="Error Number", please see Errors Section, what was the cause.

Chapter 12

Shaft Encoder Functions

12.1 This page is kipped in blank

Chapter 13

Error Functions

13.1 ST600uNet_GetError

ST600uNet_GetError (char * Error,unsigned char, `buffer_length`));
When an Error occurred, calling this function the application will know what caused this error, and will return the error string which maximum length = 255 bytes. Internally ST600μNET Library, will check the buffer length if it is not enough will return "1".

Parameters:
char * Error: Return ASCII String with a error description, maximum 255 characters.Please refer to Section 13.5 to see the error table.
unsigned char buffer_length: *Maximum length* $\Rightarrow$ 255
Return:
"0": NO ERROR
Parameters Returned:

"1": ERROR
Parameters Returned:

13.2 ST600uNet_Get_RMV859_Kernel_LastError

ST600uNet_Get_RMV859_Kernel_LastError (unsigned char `motor_addr_mask`, unsigned short int *`kernel_error`, unsigned short int * `last_command`, unsigned short int *`Error_number`);
This function return as parameters which a) RMV859 Kernel Error number, b) last command sent.

Parameters:
unsigned char motor_address_mask: RMV859 Address:($ADDR0 \Rightarrow 1$, $ADDR1 \Rightarrow 2$, $ADDR2 \Rightarrow 4$, $ADDR3 \Rightarrow 8$, $ALL \Rightarrow 0$)
unsigned char kernel_error: Please refer to Section 13.6 for Kernel Error Description.
unsigned char last_command: Value for Last Command sent.

Return:
"0": NO ERROR
Parameters Returned:

unsigned short int * `Error_number` = 0.

"1": ERROR

Parameters Returned:

unsigned short int * `Error_number` ="Error Number", please see Errors Section, what was the cause.

13.3 ST600uNet_Get_RMV859_Kernel_Chopper_Error

ST600uNet_Get_RMV859_Kernel_Chopper_Error (unsigned char `motor_address_mask`,unsigned short int * `kernel_error`, unsigned short int * `state_status`, unsigned short int *`over_current_counter`, unsigned short int * `over_temp_counter`, unsigned short int * `Error_number`);

This return the status of RMV859 Kernel errors, in the parameter `state_status` will return if the kernel is the state of ERROR, and chopper driver request ATTENTION. In addition, over current and over temperature counters are provided.

Parameters:

unsigned char *motor_address_mask*: RMV859 Address:($ADDR0 \Rightarrow 1$, $ADDR1 STATE_E RROR2, ADDR2 \Rightarrow 4$, $ADDR3 \Rightarrow 8, ALL \Rightarrow 0$)

unsigned short int * kernel_error: Kind of Error. When an error is caused by chopper fault, ATTENTION = 0xA700 flag is received. The reason can be over current or over temperature.

unsigned short int * state_status: This parameter return the Kernel State at the moment this function is called. When a motion "stops" due to an error this parameter will indicates RMV859 Kernel will be at : `STATE_ERROR`. All STATES of RMV859 Kernel's that are concerned to the application software are defined as follows:

`STATE_RUN` $\Rightarrow$ 2
`STATE_STOP` $\Rightarrow$ 3
`STATE_ERROR` $\Rightarrow$ 4
`STATE_MOTION_CMPLT` $\Rightarrow$ 7
`STATE_LSWT_REACHED` $\Rightarrow$ 8

unsigned short int* over_current_counter: The chopper driver has independent fast-reacting current detectors, with pre-programmable trip threshold on all high-side and low-side power-stage FETs. The over protection current is done cycle-by-cycle, where the current limit is calculated from motor parameters.
When the current exceed its region:
(`Current_Limit` − `Safe_Porcentage`) $\Rightarrow$ `Current_Counter` + 1,
RMV859 will be noticed via an interrupt which will increase the counter until `MAX_FAULT_REPETITION` = 1000, which the motion is stopped.
Please refer to `ST600uNet_Reset_RMV859_Internal_Errors` to clear all errors.
Before a new command is sent , please closely review motor contentions.

unsigned short int* over_temperature_counter:
The chopper driver has a two-level temperature-protection system that asserts an interrupt warning signal to RMV859 (which increment the counter) when the device junction temperature exceeds 125 °C (nominal) and, if the device junction temperature exceeds 150°C (nominal), the chopper is put into thermal shutdown,

resulting in all half-bridge outputs being set in the high impedance.
RMV859 will stop the motion when:
i) Temperature Counter at 125°C is >= `MAX_OTW_REPETITION` = 100.
ii) Temperature is higher or equal than 150°C.

Return:
"0": NO ERROR
Parameters Returned:
unsigned short int * `Error_number` = 0.
"1": ERROR
Parameters Returned:
unsigned short int * `Error_number` ="Error Number", please see Errors Section, what was the cause.

13.4 ST600uNet_Reset_RMV859_Internal_Error

ST600uNet_Reset_RMV859_Internal_Error (unsigned char `motor_address_mask`, unsigned short int * `Error_number`);
This function clear all kernel internal errors, and over current-temperature counters.

Parameters:
unsigned char *motor_address_mask*: RMV859 Address:($ADDR0 \Rightarrow 1$, $ADDR1STATE_ERROR2, ADDR2 \Rightarrow 4$, $ADDR3 \Rightarrow 8, ALL \Rightarrow 0$)

Return:
"0": NO ERROR
Parameters Returned:
unsigned short int * `Error_number` = 0.
"1": ERROR
Parameters Returned:
unsigned short int * `Error_number` ="Error Number", please see Errors Section, what was the cause.

13.5 Error Table for ST600µNET Library and HID SMB Bus

The following Error Table correspond ST600µNET Library:

ERRORS LIST ST600µNET Library and HID		
Error Defined:	HEX	Comments
HID_SMBUS_SUCCESS	0x01	HID_SMBUS_SUCCESS, NO ERROR.
HID_SMBUS_DEVICE_NOT_FOUND	0x01	HID_SMBUS_DEVICE_NOT_FOUND.
HID_SMBUS_INVALID_HANDLE	0x02	HID_SMBUS_INVALID_HANDLE.
HID_SMBUS_INVALID_DEVICE_OBJECT	0x02	HID_SMBUS_INVALID_DEVICE_OBJECT.
HID_SMBUS_INVALID_PARAMETER	0x04	HID_SMBUS_INVALID_PARAMETER.
HID_SMBUS_INVALID_REQUEST_LENGTH	0x05	HID_SMBUS_INVALID_REQUEST_LENGTH.
HID_SMBUS_READ_ERROR	0x10	HID_SMBUS_READ_ERROR, Read a wrong Address.
HID_SMBUS_WRITE_ERROR	0x11	HID_SMBUS_WRITE_ERROR, Write a wrong Open Object or Address.
HID_SMBUS_READ_TIMED_OUT	0x12	HID_SMBUS_READ_TIMED_OUT, No response from RMV959.
HID_SMBUS_WRITE_TIMED_OUT	0x13	HID_SMBUS_WRITE_TIMED_OUT, No write acknowledge .
HID_SMBUS_DEVICE_IO_FAILED	0x14	HID_SMBUS_DEVICE_IO_FAILED, NOT USED.
HID_SMBUS_DEVICE_ACCESS_ERROR	0x15	HID_SMBUS_DEVICE_ACCESS_ERROR.
HID_SMBUS_DEVICE_NOT_SUPPORTED	0x16	HID_SMBUS_DEVICE_NOT_SUPPORTED.
ST_ADDRESS_NOT_SUPPORTED	0x17	ST_ADDRESS_NOT_SUPPORTED. Wrong RMV859 address.
ST_SMBUS_NOT_OPENED	0x18	ST_SMBUS_NOT_OPENED. HID bus is not Opened.
ST600_MOTOR_POWER	0x30	ST600UNET_WRONG_MOTOR_PWR. Motor power not supported.
ST600_MOTOR_WINCURR	0x31	ST600UNET_WRONG_MAX_WINDING_CURRENT.
ST600_MOTOR_STEP_ANGLE	0x32	ST600UNET_WRONG_STEP_ANGLE.
ST600_MOTOR_SPEED	0x33	ST600UNET_WRONG_MAX_MOTOR_SPEED.
ST600_MOTOR_FIFO_CMD	0x34	ST600UNET_WRONG_FIFO_COMMAND.
ST600_MOTOR_FIFO_BUF	0x35	ST600UNET_WRONG_FIFO_BUFFER_LENGTH.
ST600_FIFO_WRONG_BUFFER_DATA	0x36	ST600UNET_WRONG_FIFO_BUFFER_DATA.
KERNEL_SECURE_PASSWORD	0xB0	ST600_SECURE_FUNCTION_WRONG_PASSWORD. For only reserved functions.
WRONG_POINTER_AS_PARAMETER	0x37	WRONG_POINTER_AS_PARAMETER. Pointer passed as parameter is corrupt.
HEX_FILE_LOADING_FAILED	0x38	HEX_FILE_COULD_NOT_BEEN_LOADED.
BOOTLOADER_NEEDS_TO_BE_RUN	0x39	HEX_FILE_COULD_NOT_BEEN_LOADED.
NO_COMMAND_ACKNOWLEDGE_RECEIVED	0x3A	NO_COMMAND_ACKNOWLEDGE_HAS_BEEN_RECEIVED.
FLASH_MEMORY_WRONG_VERIFICATION	0x3B	FLASH_MEMORY_WRONG_VERIFICATION_CRC_DATA. Wrong CRC in updating Flash Memory.
WRONG_FUNCTION_PARAMETERS	0x3C	WRONG_FUNCTION_PARAMETERS_SENT.
ERROR_SAVING_PARAMETERS_TO_FLASH	0x3D	ERROR_SAVING_PARAMETERS_TO_FLASH_NO_ACKN.
NO_STEPS_TO_MOVE	0x3E	ERROR_NO_STEPS_TO_MOVE.
WRONG_SEQUENTIAL_MOTION_PARAMETERS	0x3F	WRONG_SEQUENTIAL_MOTION_PARAMETERS.
WRONG_STEPS_TO_MOVE	0x40	WRONG_STEPS_NUMBER_SENT.
ERROR_IN_ADJUSTING_CHOPPER_CURRENT	0x41	ERROR_IN_ADJUSTING_CHOPPER_CURRENT.
WRONG_MOTOR_RESISTANCE_VALUE	0x42	WRONG_MOTOR_RESISTANCE_VALUE.
WRONG_SEEK_HOME_SPEED	0x43	WRONG_VALUE_IN_SEEK_HOME_MOTOR_SPEED.
COMMAND_NOT_ACKNOWLEDGED	0x44	NO ACKNOWLEDGE was received from last Command.....
HID_SMBUS_UNKNOWN_ERROR	0xFF	HID_SMBUS_UNKNOWN_ERROR. The Error was not found in the parser.

13.6 RMV859 Kernel Errors Table Description

The following Table describes RMV859 kernel errors. Calling the function ST600uNet_Get_RMV859_Kernel_Last
will return as parameter which kernel errors occurred.

RMV859 KERNEL ERRORS		
Error Defined:	HEX	Comments
CMD_NOTFOUND	0x80	The command sent, is not found.
WRONG_SPEED	0x81	Wrong Speed Value.
WRONG_PARAMETER	0x82	The parameter sent, is wrong.
CMD_NOT_ALLOWED_WHEN_RUN	0x83	This command is not allowed, when motor is running.
WRONG_STEP_RESOLUTION	0x84	The step desired is not supported.
NOT_CONFIGURED_AS_SPEED_CONTROL	0x85	Calling to SPEED_CONTROL motion, before flag is enabled.
WRONG_CHECK_SUM	0x86	Wrong Check Sum, in Flash Memory operations.
FIFO_FULL_WRITE_NOT_COMPLT	0x87	FIFO FULL can not write all data sent.
FIFO_WRONG_CHECK_SUM	0x88	FIFO wrong check sum.
FIFO_DATA_CORRUPTED	0x89	Set when Read FIFO does not return any value.
WRONG_FIFO_LENGTH	0x8A	Set when a Read FIFO length is not odd.
WRONG_CHECKSUM_FLASH	0x8B	FIFO read or write wrong check sum.
SPEED_BUFFER_IS_FULL	0x8C	Buffer for SPEED_CONTROL is full.
POSITION_BUFFER_IS_FULL	0x8D	Buffer for POSITION_CONTROL is full.
WRONG_HOLDING_TORQUE_PARAM	0x8E	Wrong Stop Holding torque parameters.
TRAPEZOIDAL_MOTION_MUST_BE_ENABLED	0x8F	The flag for Trapezoidal motion must be enabled first.
DISABLE_FIFO_FLAG_FIRST	0x90	FIFO flag needs to be disabled first.
NO_STEPS_TO_MOVE	0x91	Zero step has been sent to move.
WRONG_TRAPEZOIDAL_PARAMETERS	0x92	The parameters are not in boundary limits
RESERVED	0x93	RESERVED
RESERVED	0x94	RESERVED
RESERVED	0x95	RESERVED
RESERVED	0x96	RESERVED
WRONG_SEQUENTIAL_PARAMETERS	0x97	Sequential mask controller con not be Zero.
WRONG_SEQUENTIAL_PARAMETERS	0x97	Sequential mask controller con not be Zero.
SPEED_CONTROL_FLAG_NEEDS_TO_BE_ENABLED	0x98	SPEED Control Motion must be enabled first.
AXIS_IS_NOT_AT_LIMIT_SWITCH	0x99	The axis is not at the Limit Switch.
WRONG_NUMBER_STEPS_IN_SW_FUNCTION	0x9A	Step numbers are not in boundary Limit.
WRONG_DIRECTION_TO_MOVE_FROM_LIMIT_SWT	0x9B	The Direction must be oppositive from the previous motion.
SEQ_MOTION_NEEDS_BE_ENABLED_FIRST	0x9C	The Sequential Motion flag must be enabled first in ST600uNet_Set_Sequential_Motion.
CMD_SENT_TO_ADDRESS_MUST_BE_ENABLED	0x9D	A Sequential Motion Setup command was sent, which address has not been enabled.
WRONG_ADDRESS_TO_READ_AN_ADC_CHANNEL	0x9E	The address sent for an Analog Input read, is not supported.
WRONG_ADDRESS_TO_READ_AN_ADC_CHANNEL	0x9E	The address sent for an Analog Input read, is not supported.
RESERVED	0x9F	RESERVED

Note5: These errors are produced by RMV859 controller and be retrieved by command.

Chapter 14

RMV859 Firmware and ST600μNET.dll Versions

14.1 ST600uNet_GET_Library_Version

ST600uNet_GET_Library_Version (unsigned char * `mayor_version`, unsigned char * `minor_version`, unsigned short int * `compiled_year`, unsigned char * `compiled_month`, unsigned short int * `Error_number`);

Parameters:
unsigned short char* mayor_version: Mayor Version.
unsigned short char* minor_version: Minor Version.
unsigned short int* compiled_year: Year compiled.
unsigned short char* compiled_month: Month of compilation.

Return:
"0": NO ERROR
Parameters Returned:
unsigned short int * `Error_number` = 0.
"1": ERROR
Parameters Returned:
unsigned short int * `Error_number` ="Error Number", please see Errors Section, what was the cause.

14.2 ST600uNet_GET_RMV859_Firware_Version

ST600uNet_GET_RMV859_Firware_Version (unsigned char `motor_address_mask`, unsigned char *`rmv859_mayor_version`,unsigned char * `rmv859_middle_version`, unsigned char * `rmv859_minor_version`,unsigned short int * `rmv859_compiled_year`, unsigned char * `rmv859_compiled_month`, unsigned short int * `Error_number`);

Parameters:
unsigned char *motor_address_mask*: RMV859 Address:($ADDR0 \Rightarrow 1$, $ADDR1 STATE_E RROR2, ADDR2 \Rightarrow 4$, $ADDR3 \Rightarrow 8, ALL \Rightarrow 0$)

unsigned short int * rmv859_mayor_version: Mayor Version.
unsigned short int * rmv859_middle_version: Latest Features .
unsigned short int * rmv859_minor_version: Secondary upgrades.
unsigned short int * rmv859_compiled_year: Year of compilation.
unsigned short int * rmv859_compiled_month: Month of compilation.

Return:
"0": NO ERROR
Parameters Returned:
unsigned short int * `Error_number` = 0.
"1": ERROR
Parameters Returned:
unsigned short int * `Error_number` ="Error Number", please see Errors Section, what was the cause.