RugbyPro SDK V1.03 Release User Guide

1 REVISION HISTORY

Table 1 Revision History

Rev #	Date	Action	By
0.3	09/08/2021	First draft	JackPan
0.4	09/10/2021	Fixed PWM setting for LED PN measurement	JackPan
0.41	09/10/2021	Fixed Internal temperature sensor gain	JackPan
1.00	12/09/2021	Add frameId log function, add ls_read_status_ext function,fixed auxPWM dutycycle configuration issues	JackPan
1.01	12/31/2021	Fixed temperature sensor measurement inaccurate issues, Fixed Aux PWM init issues	JackPan
1.02	01/18/2022	Update LIN stack to V2.04, fix multiPDUs with NAD=0x7F, 0x3D response valid NAD issues.	JackPan
1.03	04/14/2022	Fixed RAM access issue when LED_NUM is not 8, Fixed disable buck issue which cause MCU halt	JackPan
1.04	12/26/2022	Fixed ColorMixing possiblely no output when target color is close to R(x,y) G(x,y) or B(x,y) of target RGB LED coordinates.	JackPan

2 TABLE OF CONTENTS

3 CHANGES V1.04

3.1 BUG FIXED

Fixed ColorMixing possiblely no output when target color is close to R(x,y) G(x,y) or B(x,y) of target RGB LED coordinates.

Update ColorMixing Library to 2.1.2

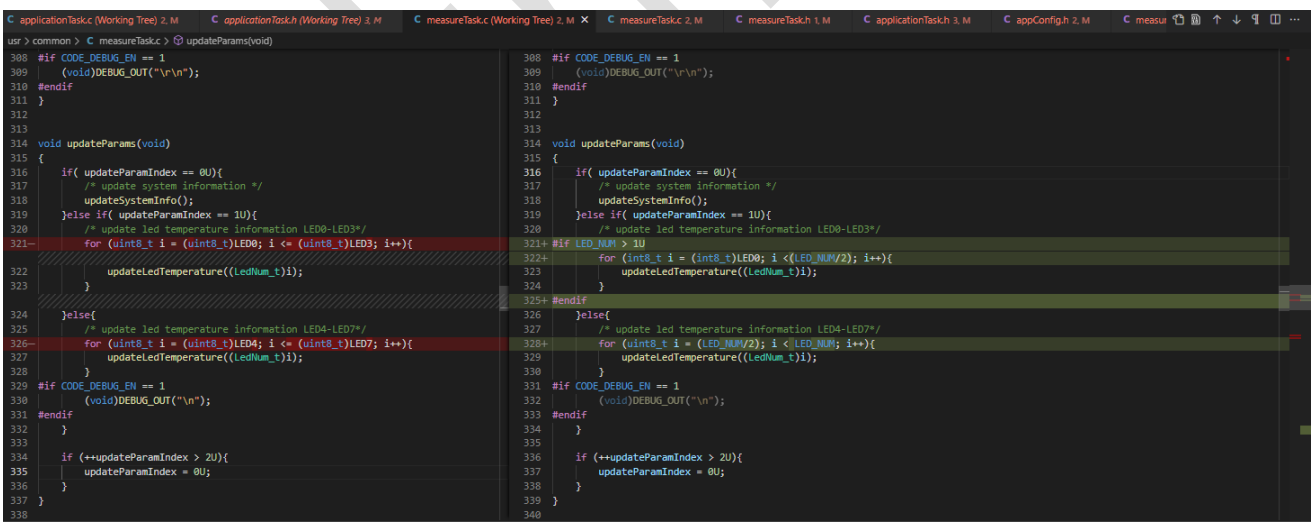
3.2 NEW FEATURES

None

4 CHANGES V1.03

4.1 BUG FIXED

1. libdev_rugbyPro_app.01.03_lin_2.0.3_colorlib_2.0.1_ :



Fixed the RAM access issue when LED_NUM is not equal to 8.

2. Remove waiting buck off which may cause MCU halt when Battery voltage is very low.

```c
void BUCK_Handler(void)
{
    uint8_t status = BUCKCTRL_REG_BUCKIRQ.STATUS.BYTE;
    if (buckIsrCallback != NULL){
        buckIsrCallback(status);
    }
    BUCKCTRL_REG_BUCKIRQ.CLEAR.BYTE = 0xFFU;
}


void BUCK_DisableBuck(void)
{
    BUCKCTRL_REG_CTRL.ENAREQ     = 0U;                  /* disable buck */
    while(BUCKCTRL_REG_STATUS.BUCKOFF == 0U){}           /* wait until disable finished */
}

void BUCK_EnableBuck(void)
{
    BUCKCTRL_REG_CTRL.ENAREQ     = 1U;                  /* enable buck */
}

void BUCK_RestartBuck(void)
{
    BUCKCTRL_REG_CTRL.ENAREQ     = 0U;                  /* disable buck */
    while(BUCKCTRL_REG_STATUS.BUCKOFF == 0U){}          /* wait until disable finished */
    BUCKCTRL_REG_CTRL.ENAREQ     = 1U;                  /* enable buck */
}
```

```c
void BUCK_Handler(void)
{
    uint8_t status = BUCKCTRL_REG_BUCKIRQ.STATUS.BYTE;
    if (buckIsrCallback != NULL){
        buckIsrCallback(status);
    }
    BUCKCTRL_REG_BUCKIRQ.CLEAR.BYTE = 0xFFU;
}


void BUCK_DisableBuck(void)
{
    BUCKCTRL_REG_CTRL.ENAREQ     = 0U;                  /* disable buck */

}

void BUCK_EnableBuck(void)
{
    BUCKCTRL_REG_CTRL.ENAREQ     = 1U;                  /* enable buck */
}
```

3. Change LF clock to 256KHz:

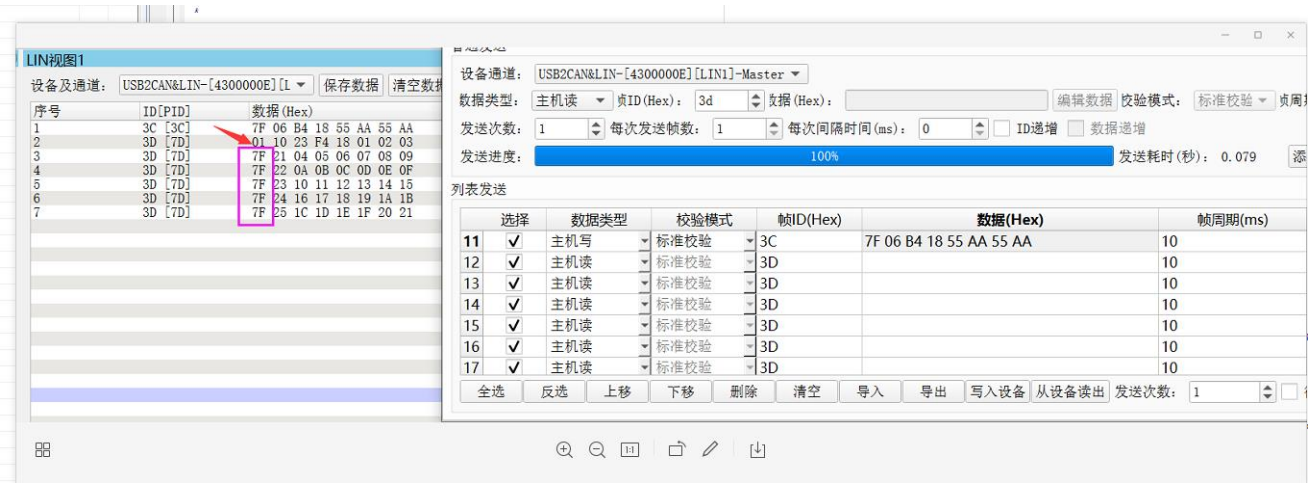
```c
*/

#include <clock_device.h>

#define SYSTEM_CLOCK_HIGH_FREQ  (16000U)/* KHz */
#define SYSTEM_CLOCK_LOW_FREQ   (16U)   /* KHz */
static uint32_t systemClock = SYSTEM_CLOCK_HIGH_FREQ;

void Clock_SystemMainClockInit(uint8_t divider)
{
    CRGA_SFRS->SYSCLKCTRL.HFRCENA = 1U;
    while(CRGA_SFRS->SYSCLKCTRL.HFRCSTS == 0U){}
    /* Enable trim access write enable */
    SYSCTRLA_SFRS->TRIM_ACCESS_KEY.KEY = 0x0EU;
    SYSCTRLA_SFRS->HF_OSC_TRIM.TRIM_HF_RC = HWCFG_GetRCHFClockCalibValue();
    SYSCTRLA_SFRS->LF_OSC_TRIM = HWCFG_GetRCLFClockCalibValue();

    SYSCTRLA_SFRS->PMU_TRIM.RESISTOR_TRIM =  HWCFG_GetV2ITrimValue();
    /* Disable trim access write enable */
    SYSCTRLA_SFRS->TRIM_ACCESS_KEY.KEY = 0U;
    CRGA_SFRS->SYSCLKCTRL.SYSCLKSEL = (uint8_t)CLOCK_RC_16MHz;
    CRGA_SFRS->SYSCLKCTRL.DIVSYSCLK = (uint8_t)divider;
    /*  frequency spread    */
    SYSCTRLA_SFRS->HF_OSC_TRIM.SSCDEEP  = 3U;/*  3  */
    SYSCTRLA_SFRS->HF_OSC_TRIM.SSCDIV   = 15U;
    SYSCTRLA_SFRS->HF_OSC_TRIM.SSCENA   = 1U;

    systemClock = SYSTEM_CLOCK_HIGH_FREQ;
    systemClock = systemClock >> (uint8_t)divider;

    SYSCTRLA_SFRS->PMU_TRIM.TRIM = HWCFG_GetMajorVBGCode();
}
```

```c
*/

#include <clock_device.h>

#define SYSTEM_CLOCK_HIGH_FREQ  (16000U)/* KHz */
#define SYSTEM_CLOCK_LOW_FREQ   (16U)   /* KHz */
static uint32_t systemClock = SYSTEM_CLOCK_HIGH_FREQ;

void Clock_SystemMainClockInit(uint8_t divider)
{
    CRGA_SFRS->SYSCLKCTRL.HFRCENA = 1U;
    while(CRGA_SFRS->SYSCLKCTRL.HFRCSTS == 0U){}
    /* Enable trim access write enable */
    SYSCTRLA_SFRS->TRIM_ACCESS_KEY.KEY = 0x0EU;
    SYSCTRLA_SFRS->HF_OSC_TRIM.TRIM_HF_RC = HWCFG_GetRCHFClockCalibValue();
    SYSCTRLA_SFRS->LF_OSC_TRIM = HWCFG_GetRCLFClockCalibValue();
    CRGA_SFRS->LFCLKCTRL.CLKLFSEL = 1;
    SYSCTRLA_SFRS->PMU_TRIM.RESISTOR_TRIM =  HWCFG_GetV2ITrimValue();
    /* Disable trim access write enable */
    SYSCTRLA_SFRS->TRIM_ACCESS_KEY.KEY = 0U;
    CRGA_SFRS->SYSCLKCTRL.SYSCLKSEL = (uint8_t)CLOCK_RC_16MHz;
    CRGA_SFRS->SYSCLKCTRL.DIVSYSCLK = (uint8_t)divider;
    /*  frequency spread    */
    SYSCTRLA_SFRS->HF_OSC_TRIM.SSCDEEP  = 3U;/*  3  */
    SYSCTRLA_SFRS->HF_OSC_TRIM.SSCDIV   = 15U;
    SYSCTRLA_SFRS->HF_OSC_TRIM.SSCENA   = 1U;

    systemClock = SYSTEM_CLOCK_HIGH_FREQ;
    systemClock = systemClock >> (uint8_t)divider;

    SYSCTRLA_SFRS->PMU_TRIM.TRIM = HWCFG_GetMajorVBGCode();
}
```

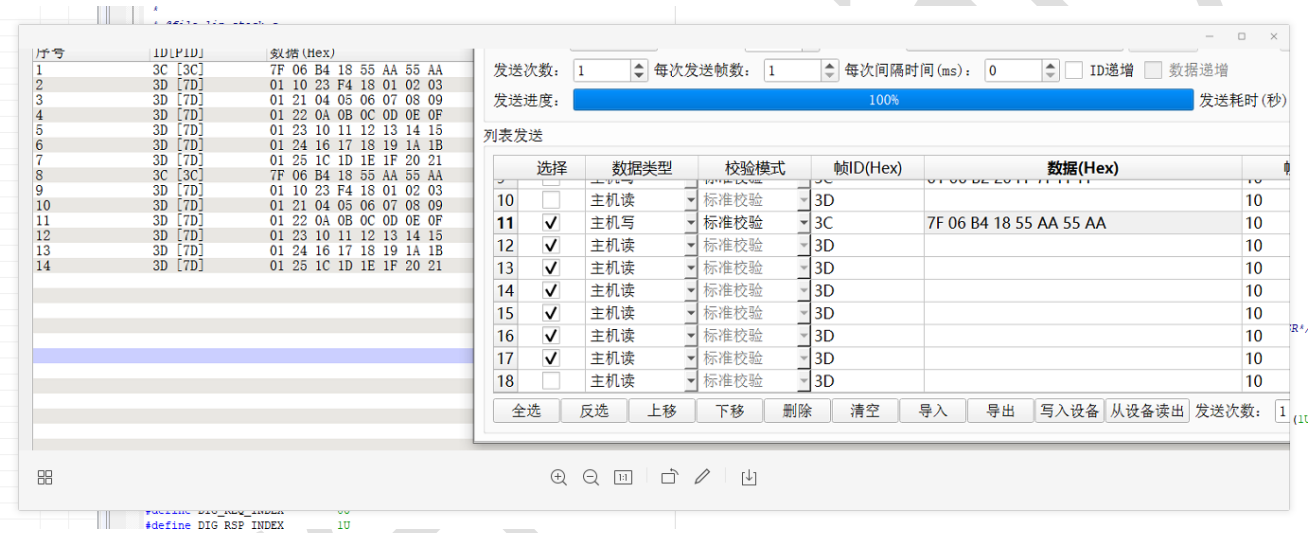
5 CHANGES V1.02

Update LIN stack to V2.04, fix multiPDUs with NAD=0x7F, 0x3D response valid NAD issues.

Before change:



After change:



6 CHANGES V1.00

6.1 BUG FIXED

1) Fixed temperature sensor measurement inaccurate issues: increase sampling cycle when select Vtemp channel:

libdev_rugbyPro_app.01.01_lin_2.0.3_colorlib_2.0.1_:

```
diff --git a/drivers/hal/src/adc_device.c b/drivers/hal/src/adc_device.c
index 291d392..2a14c3a 100644
--- a/drivers/hal/src/adc_device.c
+++ b/drivers/hal/src/adc_device.c
@@ -48,6 +48,14 @@ void ADC_Init(AdcMeasureItem_t item, uint8_t channel, LedType_t type, AdcSyncMod
 {
     adcMeasParamm.item      = item;
     adcMeasParamm.channel   = channel;
+
+    if(item == ADC_MEASURE_ITEM_VTEMP){
+        ADC_SFRS->CONF.SAMPCYC      = 44U;
+    }else{
+        ADC_SFRS->CONF.SAMPCYC      = 10U;
+    }
+    ADC_SFRS->CONF.ATTEN = 0U;
+
     switch(item){
     case ADC_MEASURE_ITEM_VTEMP:/* 1x */
       ADC_SFRS->CNTRL.CH1SEL = (uint8_t)ADC_CH1_CH3_SEL_TEMP_SENSOR;
diff --git a/drivers/hal/src/pwmAux_device.c b/drivers/hal/src/pwmAux_device.c
index d79d3f4..3016388 100644
--- a/drivers/hal/src/pwmAux_device.c
+++ b/drivers/hal/src/pwmAux_device.c
@@ -25,13 +25,15 @@ void PWMAUX_Init(PWMAUX_Port_t port,PWMAUX_ClockSource_t source,PWMAUX_Prescaler
```

libdev_rugbyPro_app.01.01_lin_2.0.3_colorlib_2.0.1_16leds:

```
diff --git a/drivers/hal/src/pwm_device.c b/drivers/hal/src/pwm_device.c
index 6386278..7ff3674 100644
--- a/drivers/hal/src/pwm_device.c
+++ b/drivers/hal/src/pwm_device.c
@@ -164,6 +164,7 @@ void ADC_Handler(void)
     }else{
         vTempCode = ADC_SFRS->DATA1;
         vBattCode = ADC_SFRS->DATA0345;
+        ADC_SFRS->CONF.SAMPCYC = 10U;
     }
     if (++ledNo >= LED_NUM){
         ledNo = 0U;
@@ -187,6 +188,7 @@ void ADC_Handler(void)
         ledConvertCount = 0U;
         ADC_REG_CNTRL_CH1_CH2 = (uint8_t)ADC_CH1_CH3_SEL_TEMP_SENSOR + ( ledPhyChannel << 2);
         ADC_SFRS->CNTRL.CH5EQ  = (uint8_t)ADC_CH3_THEN_CH1;
+        ADC_SFRS->CONF.SAMPCYC = 40U;
         ledConvertType = LED_CONVERT_TYPE_VTEMP_VBAT;
     }
     ADC_REG_CNTRL_CONVERT = 0xC1U;
diff --git a/usr/common/appConfig.h b/usr/common/appConfig.h
index 1927d2e..1cfd329 100644
```

2) Fixed Aux PWM init issues:

```
--- a/drivers/hal/src/pwmAux_device.c
+++ b/drivers/hal/src/pwmAux_device.c
@@ -25,13 +25,14 @@ void PWMAUX_Init(PWMAUX_Port_t port,PWMAUX_ClockSource_t source,PWMAUX_Prescaler
     IOCTRLA_SFRS->GPIO[(uint8_t)port].PULLMODE = (uint8_t)GPIO_PULL_NONE;
     GPIO_SFRS->GPIO_CFG[(uint8_t)port].DIR     = (uint8_t)GPIO_DIR_OUTPUT;

-    PWM_AUX_SFRS->BASE_SEL = (uint8_t)source;
     PWM_AUX_SFRS->TIMER[(uint8_t)source].PRESCALESEL = (uint8_t)devider;
     PWM_AUX_SFRS->TIMER[(uint8_t)source].PERIOD = peroid;
     if (port <= PWMAUX_PORT_5){
         PWM_AUX_SFRS->ENAREQ.ENAREQ |= 1U << (uint8_t)port;                    /*  CHN:0-4   */
+        PWM_AUX_SFRS->BASE_SEL |= (uint8_t)source << (uint8_t)port;
     }else{
         PWM_AUX_SFRS->ENAREQ.ENAREQ |= 1U << ((uint8_t)port - (uint8_t)PWMAUX_PORT_5);    /*  CHN:1-4   */
+        PWM_AUX_SFRS->BASE_SEL |= (uint8_t)source << ((uint8_t)port - (uint8_t)PWMAUX_PORT_5);
     }
 }
```

```
diff --git a/drivers/hal/src/pwmAux_device.c b/drivers/hal/src/pwmAux_device.c
index d79d3f4..3016388 100644
--- a/drivers/hal/src/pwmAux_device.c
+++ b/drivers/hal/src/pwmAux_device.c
@@ -25,13 +25,15 @@ void PWMAUX_Init(PWMAUX_Port_t port,PWMAUX_ClockSource_t source,PWMAUX_Prescaler
     IOCTRLA_SFRS->GPIO[(uint8_t)port].PULLMODE = (uint8_t)GPIO_PULL_NONE;
     GPIO_SFRS->GPIO_CFG[(uint8_t)port].DIR     = (uint8_t)GPIO_DIR_OUTPUT;

-    PWM_AUX_SFRS->BASE_SEL = (uint8_t)source;
+
     PWM_AUX_SFRS->TIMER[(uint8_t)source].PRESCALESEL = (uint8_t)devider;
     PWM_AUX_SFRS->TIMER[(uint8_t)source].PERIOD = peroid;
     if (port <= PWMAUX_PORT_5){
         PWM_AUX_SFRS->ENAREQ.ENAREQ |= 1U << (uint8_t)port;                    /*  CHN:0-4   */
+        PWM_AUX_SFRS->BASE_SEL |= (uint8_t)source << (uint8_t)port;
     }else{
         PWM_AUX_SFRS->ENAREQ.ENAREQ |= 1U << ((uint8_t)port - (uint8_t)PWMAUX_PORT_5);    /*  CHN:1-4   */
+        PWM_AUX_SFRS->BASE_SEL |= (uint8_t)source << ((uint8_t)port - (uint8_t)PWMAUX_PORT_5);
     }
 }

diff --git a/usr/common/appConfig.h b/usr/common/appConfig.h
index 7f549ae..8809ef6 100644
--- a/usr/common/appConfig.h
+++ b/usr/common/appConfig.h
```

6.2 New Features

7 Changes V1.00

7.1 BUG Fixed

Fixed Aux PWM duty cycle issue

```
void PWMAUX_SetMatchValue(PwmAuxChannel_t channel, uint16_t matchRisingValue, uint16_t matchFaillValue)
{
    PWM_AUX_REG_PULSE((uint8_t)channel) = (uint32_t)matchFaillValue + ((uint32_t)matchRisingValue << 16);
    PWM_AUX_SFRS->UPDATE = 0x03U;
}
```

7.2 New Features

Add Frame ID log function which can be used for time sync when doing the dynamic light sequency.

```
Task.c (Working Tree) M   applicationTask.c (Working Tree) M   linStackTask.h (Working Tree) 1, M   cstartup_M.c (Working Tree)   hwcfg.h (Working Tree) M   uart_device.h (Working Tree)   clock_device.c (Working Tree)   pwm_device.c (Workir
usr > library > linStackTask.h > ls_FrameIdLogCallback_t
343  void ls_registerDiagReqLogPrint(ls_DiagReqLogCallback_t callback);
344
345  /*
```

```
344  void ls_registerDiagReqLogPrint(ls_DiagReqLogCallback_t callback);
345
346+ /*
347+ *  Register got an valid frameId from LIN bus ignore whether it is registered in the table, it's an ISR function!!!!
348+ *  @param [in] callback: get the frameId .
349+ *  @return none
350+ */
351+ void ls_registerFrameIdLogPrint_ISR(ls_FrameIdLogCallback_t callback);
352+
353  /*
```

Add an extra error report function which would be easy to be understanded through source code, it is the same as function ls_read_sys_status().

```
/*
 *  Read current error flag and system information, simulate with ls_read_sys_status
 *  @return:
    l_u8 error_in_resp                  :1;
    l_u8 success_in_transfer            :1;
    l_u8 overrun                        :1;
    l_u8 goto_sleep                     :1;
    l_u8 bus_activity                   :1;
    l_u8 event_trig_frame_collision     :1;
    l_u8 save_config                    :1;
    l_u8 parity_error                   :1;
    l_u8 pid;
*/
lin_status_t ls_read_sys_status_ext(void);
```

8 CHANGES V0.42

8.1 BUG FIXED

```
appConfig.h (Working Tree)   applicationTask.c (Working Tree)   systemInit.c (Working Tree) 2, M   measureTask.c (Working Tree) 2, M   clock_device.c (Working Tree)   linslaveTask_J2602.c (Working Tree)   uart_device.c (Working Tree)
usr > common > measureTask.c > MES_MnftGetLedInfo(LedNum_t, uint16_t *, uint16_t *, uint16_t *)
    uint8_t lastCHSEQ;
    uint8_t lastCH2;
    if (led_no < LED_NUM){
        ADC_UnregisterIRQ();
        lastCHSEQ   = ADC_SFRS->CNTRL.CHSEQ;
        lastCH2     = ADC_REG_CNTRL.CH2SEL;

        if ((led_no - ((led_no >> 1)*2U)) == 0U){
#if LED_BOARD_TYPE == BOARD_TYPE_DP
        GPIO_SetGPIOs((1U << GPIO_PORT_2) | (1U << GPIO_PORT_3), (1U << GPIO_PORT_2));
#elif LED_BOARD_TYPE == BOARD_TYPE_SY
        GPIO_SetGPIOs((1U << GPIO_PORT_2) | (1U << GPIO_PORT_3), (1U << GPIO_PORT_3));  /* 0,2,4,6,8,10,12,14,16 */

#endif
        }else{
#if LED_BOARD_TYPE == BOARD_TYPE_DP
        GPIO_SetGPIOs((1U << GPIO_PORT_2) | (1U << GPIO_PORT_3), (1U << GPIO_PORT_3));
#elif LED_BOARD_TYPE == BOARD_TYPE_SY
        GPIO_SetGPIOs((1U << GPIO_PORT_2) | (1U << GPIO_PORT_3), (1U << GPIO_PORT_2));  /* 0,2,4,6,8,10,12,14,16 */

#endif
        }
```

```
    uint8_t lastCHSEQ;
    uint8_t lastCH2;
    if (led_no < LED_NUM){
        ADC_UnregisterIRQ();
        lastCHSEQ   = ADC_SFRS->CNTRL.CHSEQ;
        lastCH2     = ADC_REG_CNTRL.CH2SEL;

        if ((led_no - ((led_no >> 1)*2U)) == 0U){
#if LED_BOARD_TYPE == BOARD_TYPE_DP
        GPIO_SetGPIOs((1U << GPIO_PORT_2) | (1U << GPIO_PORT_3), (1U << GPIO_PORT_2));
#elif LED_BOARD_TYPE == BOARD_TYPE_SY
        GPIO_SetGPIOs((1U << GPIO_PORT_2) | (1U << GPIO_PORT_3), (1U << GPIO_PORT_3));  /* 0,2,4,6,8,10,12,14,16 */
#elif LED_BOARD_TYPE == BOARD_TYPE_S2
        GPIO_SetGPIOs((1U << GPIO_PORT_2) | (1U << GPIO_PORT_3), (1U << GPIO_PORT_2));
#endif
        }else{
#if LED_BOARD_TYPE == BOARD_TYPE_DP
        GPIO_SetGPIOs((1U << GPIO_PORT_2) | (1U << GPIO_PORT_3), (1U << GPIO_PORT_3));
#elif LED_BOARD_TYPE == BOARD_TYPE_SY
        GPIO_SetGPIOs((1U << GPIO_PORT_2) | (1U << GPIO_PORT_3), (1U << GPIO_PORT_2));  /* 0,2,4,6,8,10,12,14,16 */
#elif LED_BOARD_TYPE == BOARD_TYPE_S2
        GPIO_SetGPIOs((1U << GPIO_PORT_2) | (1U << GPIO_PORT_3), (1U << GPIO_PORT_3));
#endif
        }
```

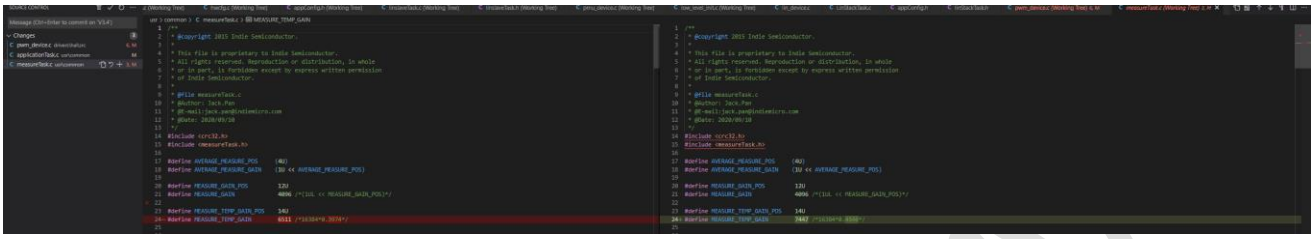
8.2 NEW FEATURES

None

9 CHANGES V0.41

9.1 BUG FIXED

Fixed internal temperature sensor gain:



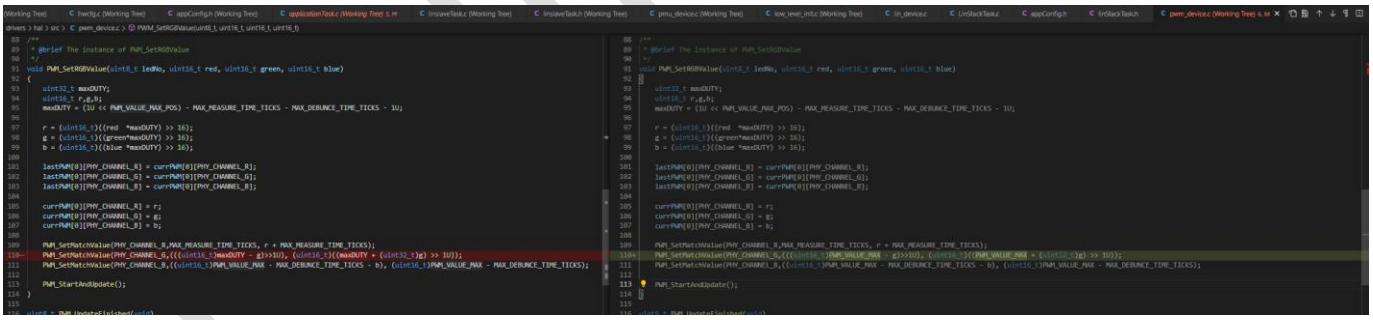
9.2 NEW FEATURES

None

10 CHANGES V0.4

10.1 BUG FIXED

Fixed PWM load average issue which would affect LED PN measurement



10.2 NEW FEATURES

None

11 CHANGES V0.3

11.1 BUG FIXED

None

11.2 NEW FEATURES

1. Add Macro AUTO_ADDRESSING_EN for SNPD Option specific LIN slew rate configuration:

linslaveTask.c appConfig.h

```
#include <clock_device.h>
#include <pwm_device.h>
#include <string.h>

#define PRAGMA(x) _Pragma(#x)
#define BEGIN_PACK PRAGMA(pack(push, 1))
#define END_PACK   PRAGMA(pack(pop))

#define LIN_STACK_TYPE_LIN2_2A          (0U)
#define LIN_STACK_TYPE_SAEJ2602         (1U)
#define LIN_STACK_TYPE_PRIVATE          (2U)

#define LIN_STACK_TYPE                  LIN_STACK_TYPE_LIN2_2A

#define AUTO_ADDRESSING_EN              (0U)
```

```
linslaveTask.c | appConfig.h | lin_device.c ×

    SYSCTRLA_REG_TRIM_ACCESS_KEY.KEY   = 0x0E;
    LINS_REG_CTRL.SLEEP = 0U;                    /*  wake up lins when init if sleep  */
    /* Config GPIO to LIN mode, enable transmission */
    IOCTRLA_SFRS->LIN.LINS_HWMODE       = 1U;     /* Hardware Mode Enabled. LIN slave peripheral writes/read the LIN I/O pin.*/
    IOCTRLA_SFRS->LIN.LINS_PU30K_ENA    = 1U;     /* LIN 30K pullup enable.*/
    IOCTRLA_SFRS->LIN.LINS_TXENA        = 1U;     /* LIN transmit enable.*/
    IOCTRLA_SFRS->LIN.LINS_RXENA        = 1U;     /* LIN receive enable. */
    /* LINS Pullup Disable in dominant TimeOut condition. Set to disable LINS 30K pullup in case that lin bus is
       shorted to ground(Bus idle dominant timeout is detected) for saving power. LINS Pullup will be recovered
       automatically if bus idle dominant timeout is released by any bus activity. Only reset by power-on sequence.
    */
    IOCTRLA_SFRS->LIN.LINS_PUOFF_TIMEOUT = 1U;
    IOCTRLA_SFRS->LINTXDMONITOR.LINSTXDMONITORENA = 1U;     /* LINS TxD Monitor enable   */

#if LIN_STACK_TYPE == LIN_STACK_TYPE_SAEJ2602
    LINS_SFRS->BUSTIME.BUSINACTIVE      = E_LIN_TIME_INACTIVE_SEC_4;
#else
    LINS_SFRS->BUSTIME.BUSINACTIVE      = E_LIN_TIME_INACTIVE_SEC_6;
#endif
    LINS_SFRS->BUSTIME.WUPREPEAT        = E_LIN_TIME_WAKEUP_REPEAT_MS_200;
    LINS_SFRS->DL.DISAUTOSLEEP          = 1U;
    LINS_SFRS->CTRL.RST_INT_ERR         = 0x03U;            /* reset error,reset interrupt */

#if AUTO_ADDRESSING_EN == 0U
    IOCTRLA_SFRS->LIN.SWON              = 0U; /* 1: LIN slave switch On between LIN_IN and LIN_OUT pin*/
    SYSCTRLA_SFRS->LIN.TXLINSRISESLOPE = 0U;
    SYSCTRLA_SFRS->LIN.TXLINS_DR_SLOPE = LIN_TX_SLEW_RATE_2_1V_PER_US;
    SYSCTRLA_SFRS->LIN.TXLINMRISESLOPE = 0U;
    SYSCTRLA_SFRS->LIN.TXLINM_DR_SLOPE = LIN_TX_SLEW_RATE_2_1V_PER_US;
#else
    IOCTRLA_SFRS->LIN.SWON              = 1U; /* 1: LIN slave switch On between LIN_IN and LIN_OUT pin*/
    IOCTRLA_SFRS->LIN.LINM_PU1K_ENA     = 1U; /* LIN 1K pullup enable.*/
    SYSCTRLA_SFRS->LIN.TXLINSRISESLOPE = 1U;
    SYSCTRLA_SFRS->LIN.TXLINS_DR_SLOPE = LIN_TX_SLEW_RATE_5_1V_PER_US;
    SYSCTRLA_SFRS->LIN.TXLINMRISESLOPE = 1U;
    SYSCTRLA_SFRS->LIN.TXLINM_DR_SLOPE = LIN_TX_SLEW_RATE_5_1V_PER_US;
#endif
```

2. Add Diagnostic Request command(0x3C) monitor callback for adapting user's requirements.

```
linslaveTask.c * | appConfig.h | lin_device.c

void DiagReqLogPrint(const LIN_Device_Frame_t * const frame)
{

}

void LINS_TaskHandler(void)
{
    switch(linsTaskState){
    case TASK_STATE_ACTIVE:
      break;
    case TASK_STATE_INIT:
      ls_register_services(LIN_PROTOCOL_LIN2_2A,UnconditionalCmdsTable, (l_u8)(sizeof(UnconditionalCmdsTable)/sizeof(LIN_Device_Frame_t)), DIAG_DATA_BUFF_SIZE,&linsFramesCallback);
      (void)ls_set_tp_timeout(N_AS, N_CR);
      ls_registerDiagReqLogPrint(DiagReqLogPrint);
      /* Note that please don't change the following glitch configuration!!!!!*/
      (void)ls_set_lins_rx_glitch_filter_1st(0x00U,0x08U);
      (void)ls_set_lins_rx_glitch_filter_2nd(0x0AU,0x10U);
      (void)ls_set_lins_rx_glitch_filter_3rd(0x30U,0x30U);
      (void)l_sys_init();
      linsTaskState = TASK_STATE_ACTIVE;
      break;
    default:
      break;
    }
}
```

3. Add Milky Way UART driver and application:

```
LinStackTask.c | appConfig.h * | lin_device.c | linslaveTask.c

/* ************************System config ******************************/
#if AUTO_ADDRESSING_EN == 1U
    #define LIN_MASTER_EN                       (0U)
#else
    #define LIN_MASTER_EN                       (0U)
#endif

#define WATCH_DOG_EN                            (0U)
/* ************************debug config ******************************/
#define ENABLE_FUNCTION_VALIDATION              (0U)

#define UART_MILKY_WAY_EN                       (0U)
#define CODE_DEBUG_EN                           (0U)

#if CODE_DEBUG_EN == 1U
  #include <stdio.h>
  #include <stdlib.h>
  #include <string.h>
  #define DEBUG_OUT(...) printf(__VA_ARGS__)
#else
  #define DEBUG_OUT(...)
#endif
```

LinStackTask.c | appConfig.h | lin_device.c | linslaveTask.c | applicationTask.c *

```c
        response = TRUE;
        break;
      default:
        break;
    }
    return response;
}


#if UART_MILKY_WAY_EN == 1U

void APPL_UART_ISR(uint8_t* buff, uint8_t length)
{
    rxBuffLength = length;
    memcpy(rxBuff,buff,length);
    TM_PostTask(TASK_ID_APPL);

}

void handleColorFromUart(uint8_t* buff, uint8_t length)
{
    APPXYY_t *color = (APPXYY_t *)buff;
    for (uint8_t i = 0; i < LED_NUM; i++){
        (void)CLM_SetXYY((LedNum_t)i,MES_GetLedTemperature((LedNum_t)i), color->xyY[i].x, color->xyY[i].y, (uint16_t)color->xyY[i].Y*10U, 0);
    }
}
#endif
```

LinStackTask.c | appConfig.h | lin_device.c | linslaveTask.c | applicationTask.c * | systemInit.c

```c
{
    /* Init set BOR voltage level for cpu low voltage safety*/
    PMU_BORInit(BOR_1V5_THRS_1385mV, BOR_3V3_THRS_3034mV);
    /* Disable wake up timer */
    PMU_WakeTimerInit(WAKEUP_TIMEER_DISABLE, WAKEUP_TIMEER_INTERVAL_32768ms);
}


void SYS_Init(void)
{
    /* Enable trim revise access enable*/
    HWCFG_TrimAccessUnlock();
    CRGA_SFRS->MODULERSTREQ.BYTE = 0xFFU;
    /* Init system clock */
    Clock_SystemMainClockInit(SYS_MAIN_CLOCK_DIV);
    pmu_init();
    /* Init global timer engine for driving soft timer */
    SysTick_Init(SOFT_TIMER_INTERVAL *1000U * MAIN_CPU_CLOCK, SoftTimer_ExpireCallback);
#if WATCH_DOG_EN == 1U
      WDTA_Enable(WDTA_INTERVAL_8S); /* 8s */
#endif
    /* Init gpios settings */
    gpios_init();
    /* Init LED current and PWM settings */
    leds_driver_Init();

#if (CODE_DEBUG_EN == 1U) || (UART_MILKY_WAY_EN == 1U)
    (void)UART_Init(BAUDRATE_DIV_16MHz_1000000, BITSIZE_8BITS, PARITY_NONE,STOPS_1BITS);
    UART_RegisterIRQ(0,APPL_UART_ISR);
#endif

    /* Init buck settings */
    BUCK_Init(BUCK_OUTPUT_3800mV);

    /* Disable trim revise access until reset*/
```

LinStackTask.c | appConfig.h | lin_device.c | linslaveTask.c | applicationTask.c | systeminit.c | uart_device.c

```c
{
    uartRxDataCallback = rxDowncallback;
    UART_SFRS->MSGCTRL.ENA_ADDR_MATCH   = 0U;
    UART_SFRS->MSGCTRL.ADDR_MATCH       = address;
    UART_SFRS->MSGCTRL.MAX_BYTES_RXD    = 3U;
    uartAddress = address;
    UART_SFRS->FIFOLEVELCTL.RXMULTIPLEXFERDONECNT = 9;

    UART_SFRS->MSGCTRL.UFIFOSOFTRESET = 1U;// FIFO reset

    UART_SFRS->INT.CLEAR.RXMULTDONE = 1U;
    UART_SFRS->INT2.CLEAR.RXTOUT    = 1U;

    UART_SFRS->INT.ENABLE.RXMULTDONE = 1U;
    UART_SFRS->INT2.ENABLE.RXTOUT    = 1U;

    NVIC_EnableIRQ(UART_IRQn);
}

void UART_UnRegisterIRQ(void)
{
    uartRxDataCallback = NULL;
    UART_SFRS->INT.ENABLE.RXMULTDONE = 0U;
    UART_SFRS->INT.ENABLE.TXDONE     = 0U;
    UART_SFRS->INT.ENABLE.BREAKKERR  = 0U;
    NVIC_DisableIRQ(UART_IRQn);
}


int8_t UART_SendBuff(uint8_t *buff, uint16_t length)
{
    int8_t result = 0;

    for (uint8_t i = 0U; i < length; i++){
        UART_SFRS->DATA.BYTE = buff[i];
        while(UART_SFRS->INT.STATUS.TXDONE == 0U){}
        UART_SFRS->INT.CLEAR.TXDONE = 1U;
    }

    return result;
}


void UART_Handler(void)
{
    uint16_t mateData;
    uint8_t fifoCount = UART_SFRS->FIFOSTATUS.RXCOUNT;
    for (uint8_t i = 0U; i < fifoCount; i++){
        mateData = UART_REG_BREAK_DATA;
        if ((mateData & 0x0800U) != 0U){// it's a break signal
            rxBuffCount = 0U;
        }else{// it's data
            if (rxBuffCount < VALID_RX_PACKAGE_SIZE){
                rxBuff[rxBuffCount++] = (uint8_t)mateData;
                if (rxBuff[0] == uartAddress){
                    if (rxBuffCount == VALID_RX_PACKAGE_SIZE){
                        uartRxDataCallback(rxBuff,rxBuffCount);
                    }
                }
            }
        }
    }
    UART_SFRS->INT.CLEAR.BYTE = 0xFFU;
    UART_SFRS->INT2.CLEAR.RXTOUT = 0xFFU;
}
```