RealplumPro SDK 1.20 Update 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
0.42	09/10/2021	Fixed Vbatt ADC channel selection	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	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	05/10/2022	Add LIN stack APIs for LIN status management: 1. l_u8 ls_clr_save_configuration_flag(void); 2. lin_status_t ls_ifc_read_status(void); 3. l_u8 ls_ifc_clear_error_status(void); 4. l_u8 ls_read_error_code_log(void); 5. l_u8 ls_clr_error_code_log(void); 6. l_u8 ls_read_go_to_sleep_flag(void); 7. l_u8 ls_clr_go_to_sleep_flag(void); 8. l_u8 ls_clr_overrun_flag(void); Add task pending API for task state query: 1. uint8_t TM_TaskIsPending(TM_TaskId_t taskId); Add LIN wakeup callback function	JackPan
1.1	9/19/2022	Add physical Intensity APIs: 1. CLM_SetXYAbsY 2. CLM_SetAbsLUV 3. CLM_SetSRGBAbsL	Jack.Pan
1.2	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 V1.10

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 V1.10

4.1 BUG FIXED

None

4.2 NEW FEATURES

1. Add physical intensity APIs for Color consistency display in different modules and environment.

```
/*
 *  Set xy absolute physical intensity output  for example: P(x,y) = (0.33333,0.33333) x = 0.33333*65536 = 21845, y = 0.33333*65536 = 21845  intensity: lumen or mcd
 *  @param [in] temperature: -40-100 in 1C
 *  @param [in] x: 0-65535
 *  @param [in] y: 0-65535
 *  @param [in] intensity: 0-65535, lumen or mcd, if intensity value is higher than maximum intensity of the target color, it would display maximum intensity of the target color.
 *  @param [in] transitionTime: 0-65535ms
 *  @return 0
 */
uint8_t CLM_SetXYAbsY(LedNum_t ledIndex,ColorTemperature_t temperature, uint16_t x, uint16_t y, uint16_t intensity, uint16_t transitionTime);
```

```
/*
 *  Set uv absolute phy intensity output,intensity: lumen or mcd
 *  @param [in] temperature: -40-100 in 1C
 *  @param [in] u: 0-1023
 *  @param [in] v: 0-1023
 *  @param [in] intensity: 0-65535, lumen or mcd, if intensity value is higher than maximum intensity of the target color, it would display maximum intensity of the target color.
 *  @param [in] transitionTime: 0-65535ms
 *  @return 0
 */
uint8_t CLM_SetAbsLUV(LedNum_t ledIndex,ColorTemperature_t temperature, uint16_t u, uint16_t v, uint16_t intensity, uint16_t transitionTime);
```

```
/*
 *  Set sRGB absolute phy intensity output lumen or mcd,
 *  @param [in] temperature: -40-100 in 1C
 *  @param [in] red: 0-255
 *  @param [in] green: 0-255
 *  @param [in] blue: 0-255
 *  @param [in] intensity: 0-65535, lumen or mcd, if intensity value is higher than maximum intensity of the target color, it would display maximum intensity of the target color.
 *  @param [in] transitionTime: 0-65535ms
 *  @return 0
 */
uint8_t CLM_SetSRGBAbsL(LedNum_t ledIndex,ColorTemperature_t temperature, uint8_t red, uint8_t green, uint8_t blue,uint16_t intensity, uint16_t transitionTime);
```

2. Add function for bypassing intensity degradation:

```
/*
 *  ColorMixing Intensity Degradation enable, there is about 3% Intensity Degradation by default
 *  @param [in] enable: 0U: disable degradation, others: enable degradation
 *  @return 0
 */
uint8_t CLM_SetIntensityDegradation(uint8_t enable);

/*
 *  Get ColorMixing Intensity Degradation enable state.
 *  @param [in] none
 *  @return 0U: disable degradation, 1U: enable degradation
 */
uint8_t CLM_GetIntensityDegradation(void);
```

5 CHANGES V1.03

5.1 BUG FIXED

None

5.2 NEW FEATURES

```
/*
 *  Save current LIN Stack parameters such as NAD, PIDs etc. refer to pdsTask for more inform
 *  @param [in] none.
 *  @return 0.
 */
l_u8  ls_save_configuration(void);

/*
```

```
/*
 *  Save current LIN Stack parameters such as NAD, PIDs etc. refer to pdsTask for more information.
 *  @param [in] none.
 *  @return 0.
 */
l_u8  ls_save_configuration(void);

/*
 *  clear save_configuration flag.
 *  @param [in] none.
 *  @return 0.
 */
l_u8 ls_clr_save_configuration_flag(void);
```

```
/*
```

```
/*
 *  Read current error flag and system information, simulate with ls_read_sys_status not don't clear the 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_ifc_read_status(void);

/*
 *  reset lin error status.
 *  @param [in] errCode.
 *  @return: 0U;
 */
l_u8 ls_ifc_clear_error_status(void);
```

```
l_u8  ls_read_error_code(void);
```

```
l_u8  ls_read_error_code(void);

/*
 *  Read current lin stack communication error reasons, this is only for internal use and don't clear the error log
 *  @param [in] none.
 *  @return: ((1U << ERROR_BITERR_POS) | (1U << ERROR_CHK_POS) | (1 << ERROR_PARITY_POS) | (1U << ERROR_BITMON_POS) | (1U << ERROR_FRA
 */
l_u8 ls_read_error_code_log(void);

/*
 *  clear the erroe code log.
 *  @param [in] none.
 *  @return: 0;
 */
l_u8 ls_clr_error_code_log(void);
```

```
void ls_set_J2602_hardware_reset(uint8_t hardwareReset);
```

```
void ls_set_J2602_hardware_reset(uint8_t hardwareReset);

/*
 *  read go_to_sleep flag
 *  @param [in] none.
 *  @return: state: 1: got go_to_sleep command, 0: not got go_to_sleep command
 */
l_u8 ls_read_go_to_sleep_flag(void);
/*
 *  Clear go_to_sleep flag
 *  @param [in] none.
 *  @return: 0U
 */
l_u8 ls_clr_go_to_sleep_flag(void);

/*
 *  Clear overrun flag
 *  @param [in] none.
 *  @return: 0U
 */
l_u8 ls_clr_overrun_flag(void);
```

```
/* *****************internal function declarations*************************/
void DiagnosticSubscribedCmdsHandle(const DiagReqInfo_t * const frameInfo);
void UnconditionalSubscribedCmdsHandle(LIN_Device_Frame_t const *frame);
void UnconditionalPublishedCmdsISR(LIN_Device_Frame_t *const frame);

void DiagnosticSleepRequestHandle(SleepRequestType_t type);
void BusWakeUpRequestHandle(BusWakeUpRequestResult_t result);
void DiagReqLogPrint(const LIN_Device_Frame_t * const frame);
```

```
/* *****************internal function declarations*************************/
void DiagnosticSubscribedCmdsHandle(const DiagReqInfo_t * const frameInfo);
void UnconditionalSubscribedCmdsHandle(LIN_Device_Frame_t const *frame);
void UnconditionalPublishedCmdsISR(LIN_Device_Frame_t *const frame);
void UnconditionalPublishedCmdsTxFinishedISR(uint8_t fid, uint8_t resvd);
void DiagnosticSleepRequestHandle(SleepRequestType_t type);
void BusWakeUpRequestHandle(BusWakeUpRequestResult_t result);
void DiagReqLogPrint(const LIN_Device_Frame_t * const frame);
```

```
/*  LIN command handle callbacks declarations  */
static ls_LinsFramesCallback_t linsFramesCallback ={
  UnconditionalSubscribedCmdsHandle,        /* received data from master */
  UnconditionalPublishedCmdsISR,            /* send data to master, it's an interrupt fun
  NULL,                                     /* send data to master finished, it's an inte
  DiagnosticSubscribedCmdsHandle,           /* Diagnostic sigle PDU and MultiPDU received
  NULL,                                     /* special functional NAD (0x7E) handler*/
  DiagnosticSleepRequestHandle,             /* Diagnostic sleep request from master */
  BusWakeUpRequestHandle,                   /* Get the bus wake up result this would take
  NULL,                                     /* SAE J2602 -1 Configuration Messages */
  NULL,                                     /*  5.7.2.5 DNN Based Broadcast Messages */
};
```

```
/*  LIN command handle callbacks declarations  */
static ls_LinsFramesCallback_t linsFramesCallback ={
  UnconditionalSubscribedCmdsHandle,        /* received data from master */
  UnconditionalPublishedCmdsISR,            /* send data to master, it's an interrupt function, please fill the data as fast as
  UnconditionalPublishedCmdsTxFinishedISR,  /* send data to master finished, it's an interrupt function, please fill the data as
  DiagnosticSubscribedCmdsHandle,           /* Diagnostic sigle PDU and MultiPDU received data from master */
  NULL,                                     /* special functional NAD (0x7E) handler*/
  DiagnosticSleepRequestHandle,             /* Diagnostic sleep request from master */
  BusWakeUpRequestHandle,                   /* Get the bus wake up result this would take effects after call API: ls_send_wake_u
  NULL,                                     /* SAE J2602 -1 Configuration Messages */
  NULL,                                     /*  5.7.2.5 DNN Based Broadcast Messages */
};
```

```
/*
  Received sleep command from lin master or bus idle timeout occurs
*/
void DiagnosticSleepRequestHandle(SleepRequestType_t type)
{

    /* Set lin to sleep mode */
    /*LINS_SetSlaveModuleToSleep();*/
    /* Set system to hibranate mode*/
    PMU_EnterDeepSleepMode();
}
```

```
/*
  Received sleep command from lin master or bus idle timeout occurs
*/
void DiagnosticSleepRequestHandle(SleepRequestType_t type)
{
    ls_clr_go_to_sleep_flag();
    /* Set lin to sleep mode */
    /*LINS_SetSlaveModuleToSleep();*/
    /* Set system to hibranate mode*/
    PMU_EnterDeepSleepMode();
}
```

```
/*
 User defined published frame callback function which from LINS ISR
 please fill frame data[0-7] only, please don't do any changes on the left info except data[
*/
void UnconditionalPublishedCmdsISR(LIN_Device_Frame_t *const frame)
{
    /*   feed buffer to frame   */
    if (frame->frame_id == UnconditionalCmdsTable[FID_STATUS_FRAME_INDEX].frame_id){
        /* status management frame */
        lin_status_t status = ls_read_sys_status_ext();
        frame->data[0] = (uint8_t)status.payload;
        frame->data[1] = LINS_GetPIDFromFID(status.pid);
        frame->data[2] = ls_read_error_code();
        frame->data[3] = MES_DataIntegrityError() | ((uint8_t)SAFM_GetBatteryState() << 2) |
        /* MUST BE 4 bytes Aligned  */
        uint32_t *ledPNState = frame->data32 + 1U;
        *ledPNState = 0U;
        for (uint8_t i = 0U; i < LED_NUM; i++){
            (*ledPNState) |= (uint32_t)SAFM_GetLedPNStateState((LedNum_t)i) << (i*2U);
        }
    }else if (frame->frame_id == UnconditionalCmdsTable[USER_DATA_REPORT_INDEX].frame_id){
```

```
void UnconditionalPublishedCmdsTxFinishedISR(uint8_t fid, uint8_t resvd)
{
    if ( fid == UnconditionalCmdsTable[FID_STATUS_FRAME_INDEX].frame_id){
        ls_ifc_clear_error_status();
        ls_clr_error_code_log();
    }
}

/*
 User defined published frame callback function which from LINS ISR
 please fill frame data[0-7] only, please don't do any changes on the left info except data[x]!!!!
*/
void UnconditionalPublishedCmdsISR(LIN_Device_Frame_t *const frame)
{
    /*   feed buffer to frame   */
    if (frame->frame_id == UnconditionalCmdsTable[FID_STATUS_FRAME_INDEX].frame_id){
        /* status management frame */
        lin_status_t status = ls_ifc_read_status();
        frame->data[0] = (uint8_t)status.payload;
        frame->data[1] = LINS_GetPIDFromFID(status.pid);
        frame->data[2] = ls_read_error_code_log();
        frame->data[3] = MES_DataIntegrityError() | ((uint8_t)SAFM_GetBatteryState() << 2) | ((uint8_t)SAFM_GetChipTemperatureState()
        /* MUST BE 4 bytes Aligned  */
        uint32_t *ledPNState = frame->data32 + 1U;
        *ledPNState = 0U;
        for (uint8_t i = 0U; i < LED_NUM; i++){
            (*ledPNState) |= (uint32_t)SAFM_GetLedPNStateState((LedNum_t)i) << (i*2U);
        }
    }else if (frame->frame_id == UnconditionalCmdsTable[USER_DATA_REPORT_INDEX].frame_id){
```

```
void TM_DisableTask(TM_TaskId_t taskId);
```

```
void TM_DisableTask(TM_TaskId_t taskId);

/*
 *  get the state of specified task.
 *  @param [in] taskId - task to be disabled.
 *  @return 0: task is not pending, other: task is pending to be handle
 */
uint8_t TM_TaskIsPending(TM_TaskId_t taskId);
```

```
void TM_PostTask(TM_TaskId_t taskId)
{
    __atomic_enter()
    taskFlags |= (uint16_t)(1UL << taskId);
    __atomic_exit()
}
```

```
void TM_PostTask(TM_TaskId_t taskId)
{
    __atomic_enter()
    taskFlags |= (uint16_t)(1UL << taskId);
    __atomic_exit()
}

uint8_t TM_TaskIsPending(TM_TaskId_t taskId)
{
    uint16_t tasks =  taskFlags;
    return ((tasks && (uint16_t)(1UL << taskId))!= 0U);
}
```

```
typedef void (*lin_cb_func_t)(LIN_Device_Frame_t const *frame);
typedef void (*lin_stack_cb_func_t)(void);

void LINS_Init(void);
void LINS_AutoAddressingSwitch(AutoaddressingSwitchState_t state);
void LINS_SetSlaveModuleToSleep(void);
void LINS_RegisterLINStack_IRQ(lin_stack_cb_func_t callback);

void LINM_Init(void);
void LINM_RegisterLINM_IRQ(lin_cb_func_t cb);
void LINM_UnRegisterLINM_IRQ(void);
int8_t LINM_SendFrame(LIN_Device_Frame_t *frame);
```

```
typedef void (*lin_cb_func_t)(LIN_Device_Frame_t const *frame);
typedef void (*lin_stack_cb_func_t)(void);
typedef void (*lin_wakeup_cb_func_t)(void);

void LINS_Init(void);
void LINS_AutoAddressingSwitch(AutoaddressingSwitchState_t state);
void LINS_SetSlaveModuleToSleep(void);
void LINS_RegisterLINStack_IRQ(lin_stack_cb_func_t callback,lin_wakeup_cb_func_t wakeupCallback);

void LINM_Init(void);
void LINM_RegisterLINM_IRQ(lin_cb_func_t cb);
void LINM_UnRegisterLINM_IRQ(void);
int8_t LINM_SendFrame(LIN_Device_Frame_t *frame);
```

```
#if LIN_STACK_TYPE == LIN_STACK_TYPE_PRIVATE
  static void LINS_SLAVE_ISR(void);
#else
  static lin_stack_cb_func_t lin_stack_callback = NULL;

#endif

static lin_cb_func_t linm_callback = NULL;
```

```
#if LIN_STACK_TYPE == LIN_STACK_TYPE_PRIVATE
  static void LINS_SLAVE_ISR(void);
#else
  static lin_stack_cb_func_t lin_stack_callback = NULL;
  static lin_wakeup_cb_func_t lin_wakeup_callback = NULL;
#endif

static lin_cb_func_t linm_callback = NULL;
```

```
void LINS_RegisterLINStack_IRQ(lin_stack_cb_func_t callback)
{
#if LIN_STACK_TYPE != LIN_STACK_TYPE_PRIVATE
    lin_stack_callback = callback;

    NVIC_EnableIRQ(LINS_IRQn);
#endif
}
```

```
void LINS_RegisterLINStack_IRQ(lin_stack_cb_func_t callback,lin_wakeup_cb_func_t wakeupCallback)
{
#if LIN_STACK_TYPE != LIN_STACK_TYPE_PRIVATE
    lin_stack_callback = callback;
    lin_wakeup_callback = wakeupCallback;
    NVIC_EnableIRQ(LINS_IRQn);
#endif
}
```

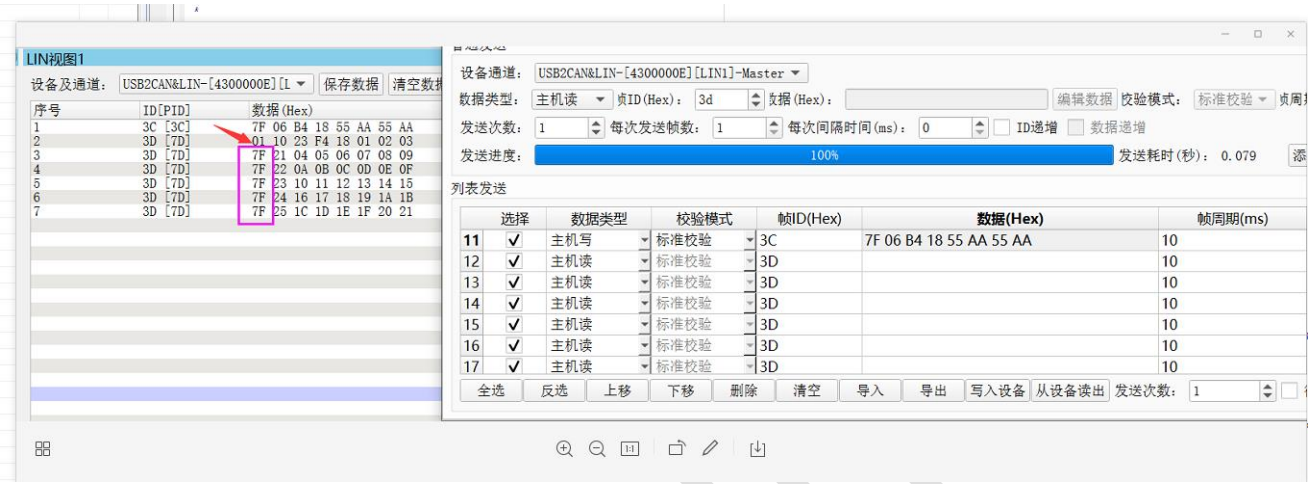
```
void WULINS_Handler(void)
{
    if (WICA_REG_STATUS.LINS != 0U){
        LINS_REG_CTRL.SLEEP = 0U;                    /*  wake up lins occur

        WICA_REG_CTRL.LINSIRQCLR = 1U;
    }
}
```

```
void WULINS_Handler(void)
{
    if (WICA_REG_STATUS.LINS != 0U){
        LINS_REG_CTRL.SLEEP = 0U;                    /*  wake up lins occurs when wake up signal detected  */
        if (lin_wakeup_callback != NULL){
            lin_wakeup_callback();
        }
        WICA_REG_CTRL.LINSIRQCLR = 1U;
    }
}
```

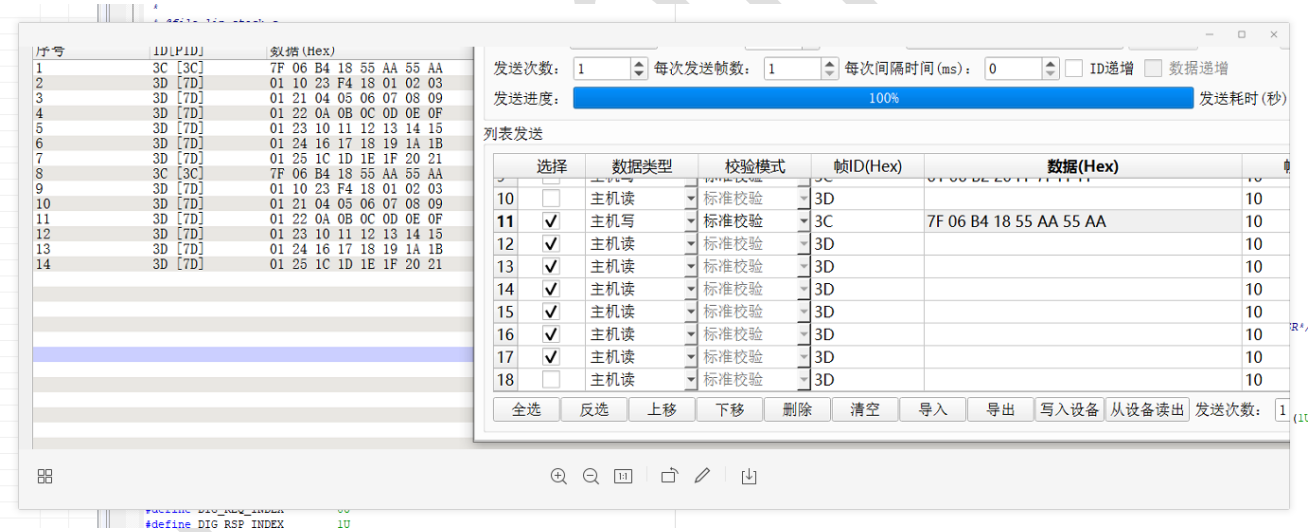
6 CHANGES V1.02

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

Before change:



After change:



7 CHANGES V1.01

7.1 BUG FIXED

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

libdev_realplumPro_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 f2ee729..a67863a 100644
--- a/drivers/hal/src/adc_device.c
+++ b/drivers/hal/src/adc_device.c
@@ -47,9 +47,8 @@ void ADC_Init(AdcMeasureItem_t item, uint8_t channel, LedType_t type, AdcSyncMod
 {
     adcMeasParamm.item       = item;
     adcMeasParamm.channel    = channel;
-
-    if(item == ADC_MEASURE_ITEM_LED_VBAT_VTEMP){
-        ADC_SFRS->CONF.SAMPCYC      = 40U;
+    if(item == ADC_MEASURE_ITEM_VBAT_VTEMP){
+        ADC_SFRS->CONF.SAMPCYC      = 44U;
     }else{
         ADC_SFRS->CONF.SAMPCYC      = 10U;
     }
diff --git a/usr/common/measureTask.c b/usr/common/measureTask.c
index c41ab14..fc405a7 100644
--- a/usr/common/measureTask.c
+++ b/usr/common/measureTask.c
:...skipping...
commit 9cb069f5a541790a8350232d99560bf48bdb3280 (HEAD -> V3.6)
Author: Jack.Pan <jack.pan@indiemicro.com>
Date:   Fri Jan 7 15:07:05 2022 +0800

    0

diff --git a/drivers/hal/src/adc_device.c b/drivers/hal/src/adc_device.c
index f2ee729..a67863a 100644
--- a/drivers/hal/src/adc_device.c
+++ b/drivers/hal/src/adc_device.c
@@ -47,9 +47,8 @@ void ADC_Init(AdcMeasureItem_t item, uint8_t channel, LedType_t type, AdcSyncMod
 {
     adcMeasParamm.item       = item;
     adcMeasParamm.channel    = channel;
-
-    if(item == ADC_MEASURE_ITEM_LED_VBAT_VTEMP){
-        ADC_SFRS->CONF.SAMPCYC      = 40U;
+    if(item == ADC_MEASURE_ITEM_VBAT_VTEMP){
+        ADC_SFRS->CONF.SAMPCYC      = 44U;
     }else{
         ADC_SFRS->CONF.SAMPCYC      = 10U;
     }
diff --git a/usr/common/measureTask.c b/usr/common/measureTask.c
index c41ab14..fc405a7 100644
--- a/usr/common/measureTask.c
+++ b/usr/common/measureTask.c
@@ -148,7 +148,7 @@ void measureParamStart(void)
       }
       break;
     case ADC_MEASURE_ITEM_LED_VBAT_VTEMP:
-      add_data_to_buff(&adcResult.vChipTemp.currCode, measGeneralAdcCode[0], &adcResult.vChipTemp.count,adcResult.vChipTemp.buff);
+//      add_data_to_buff(&adcResult.vChipTemp.currCode, measGeneralAdcCode[0], &adcResult.vChipTemp.count,adcResult.vChipTemp.buff);
       add_data_to_buff(&adcResult.vBatt.currCode,     measGeneralAdcCode[1], &adcResult.vBatt.count,adcResult.vBatt.buff);
       if (PWM_LedIsOn(0U) != 0U){
           add_data_to_buff(&adcResult.vLedPN[0][measLedColorNo].currCode, measGeneralAdcCode[2], &adcResult.vLedPN[0][measLedColorNo].count,adcResult.vLedPN[0][measLedColorNo].buff);
```

libdev_realplumPro_app.01.01_lin_2.0.3_colorlib_2.0.1_4leds:

```
diff --git a/drivers/hal/src/pwm_device.c b/drivers/hal/src/pwm_device.c
index ddfd696..d544174 100644
--- a/drivers/hal/src/pwm_device.c
+++ b/drivers/hal/src/pwm_device.c
@@ -75,6 +75,7 @@ void ADC_Handler(void)
     }else{
         tempCode = ADC_SFRS->DATA0345;
         battCode  = ADC_SFRS->DATA1;
+        ADC_SFRS->CONF.SAMPCYC = 10U;
     }
 #if LED_NUM == 1U
     LedPowerSwitch = 0U;
@@ -174,6 +175,7 @@ void PWM_Handler(void)
     }else{
         ledConvertCount = 0U;
         ADC_SFRS->CNTRL.CHSEQ  = (uint8_t)ADC_CH3_THEN_CH1;
+        ADC_SFRS->CONF.SAMPCYC      = 40U;
         ledConvert = FALSE;
     }
     ADC_SFRS->CNTRL.CONVERT    = 1U;
```

7.2 NEW FEATURES

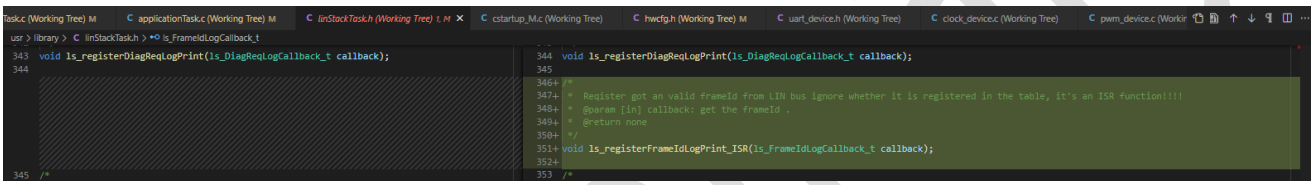
8 CHANGES V1.00

8.1 BUG FIXED

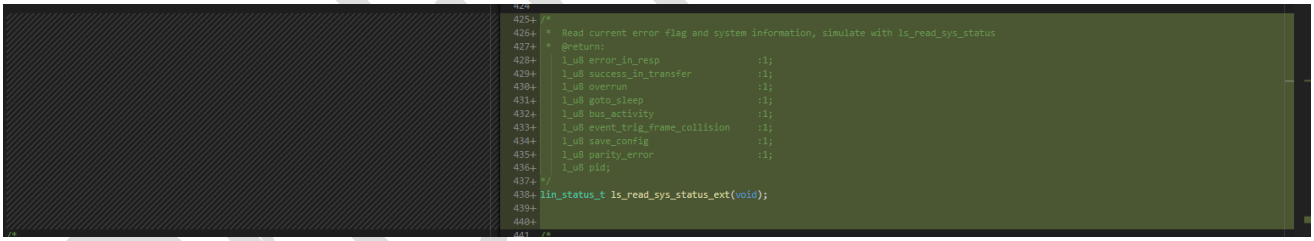
None

8.2 NEW FEATURES

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



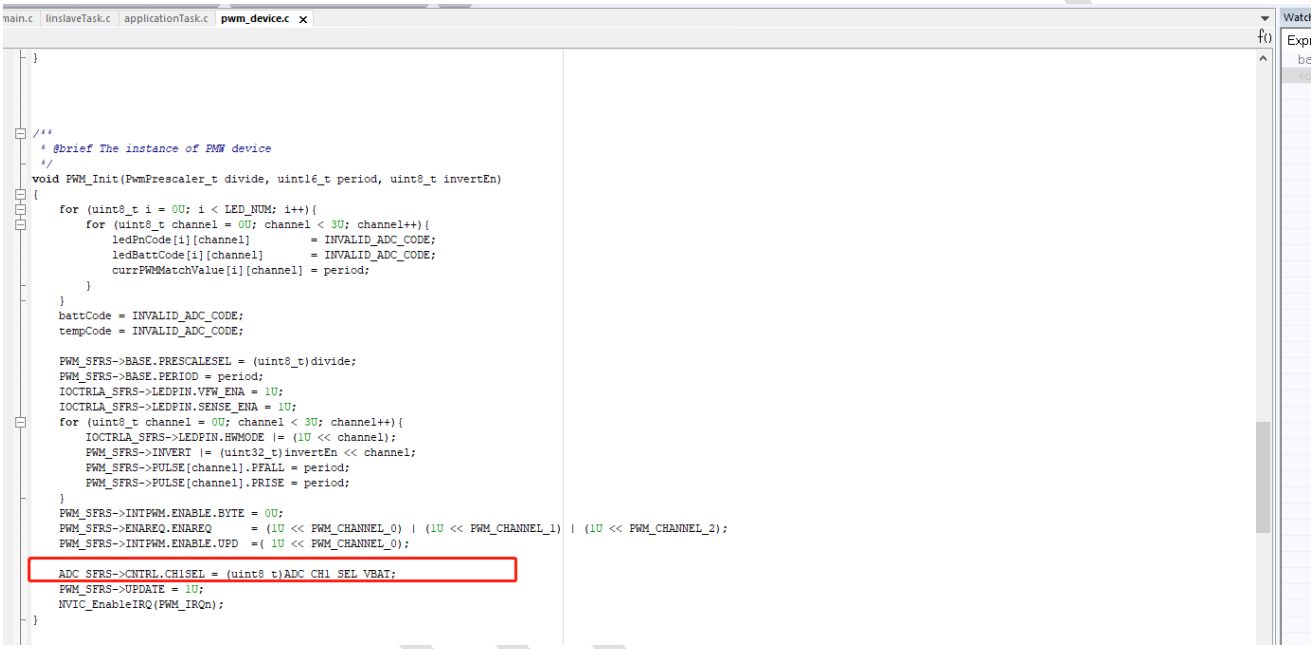
Add an extra error report function which would be easy to be understanded through source code, it is the same as function ls_read_sys_status().



9 CHANGES V0.41

9.1 BUG FIXED

Fixed VBATT ADC channel selection:



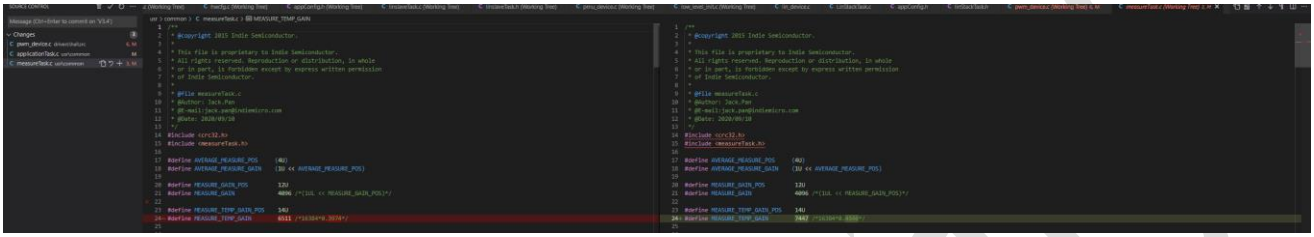
9.2 NEW FEATURES

None

10 CHANGES V0.41

10.1 BUG FIXED

Fixed internal temperature sensor gain:



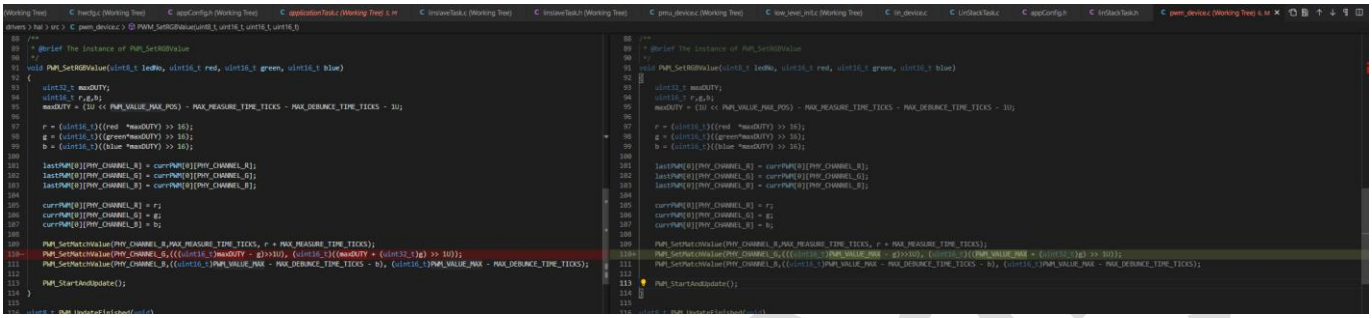
10.2 NEW FEATURES

None

11 CHANGES V0.4

11.1 BUG FIXED

Fixed PWM load average issue which would affect LED PN measurement



11.2 NEW FEATURES

None

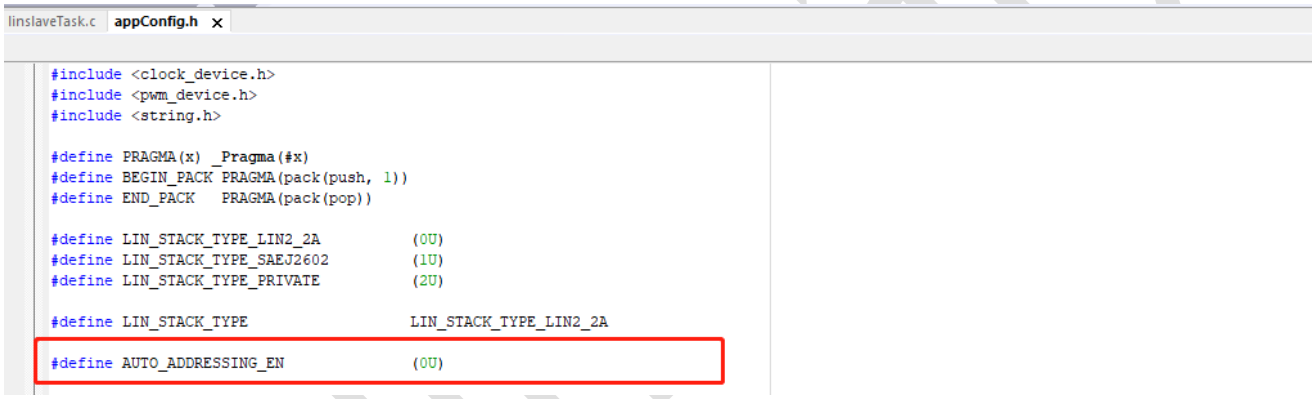
12 CHANGES V0.3

12.1 BUG FIXED

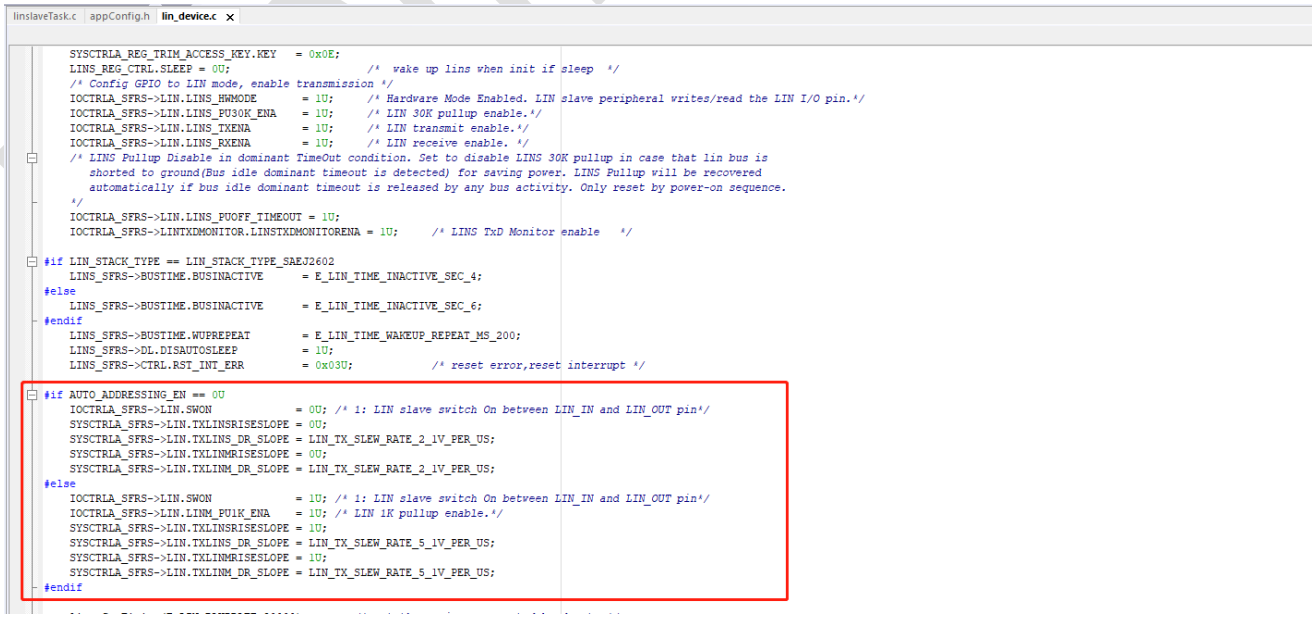
None

12.2 NEW FEATURES

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

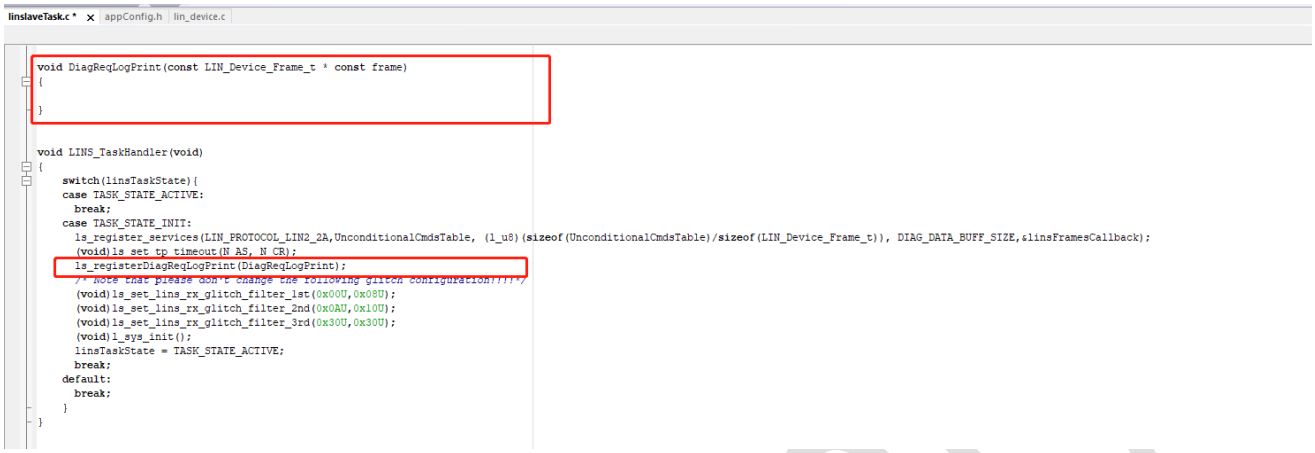


```
linslaveTask.c | appConfig.h

#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;
    }
}
```