

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 frameld 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 possibly 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

1	REVISION HISTORY	2
2	TABLE OF CONTENTS	3
3	CHANGES V1.03	4
3.1	BUG Fixed	4
4	CHANGES V1.02	5
5	CHANGES V1.00	6
5.1	BUG Fixed	6
5.2	New Features	8
6	CHANGES V1.00	8
6.1	BUG Fixed	8
6.2	New Features	8
7	CHANGES V0.42	9
7.1	BUG Fixed	9
7.2	New Features	9
8	CHANGES V0.41	9
8.1	BUG Fixed	10
8.2	New Features	10
9	CHANGES V0.4	10
9.1	BUG Fixed	10
9.2	New Features	10
10	CHANGES V0.3	11
10.1	BUG Fixed	11
10.2	New Features	11

3 CHANGES V1.04

3.1 BUG FIXED

Fixed ColorMixing possibly 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

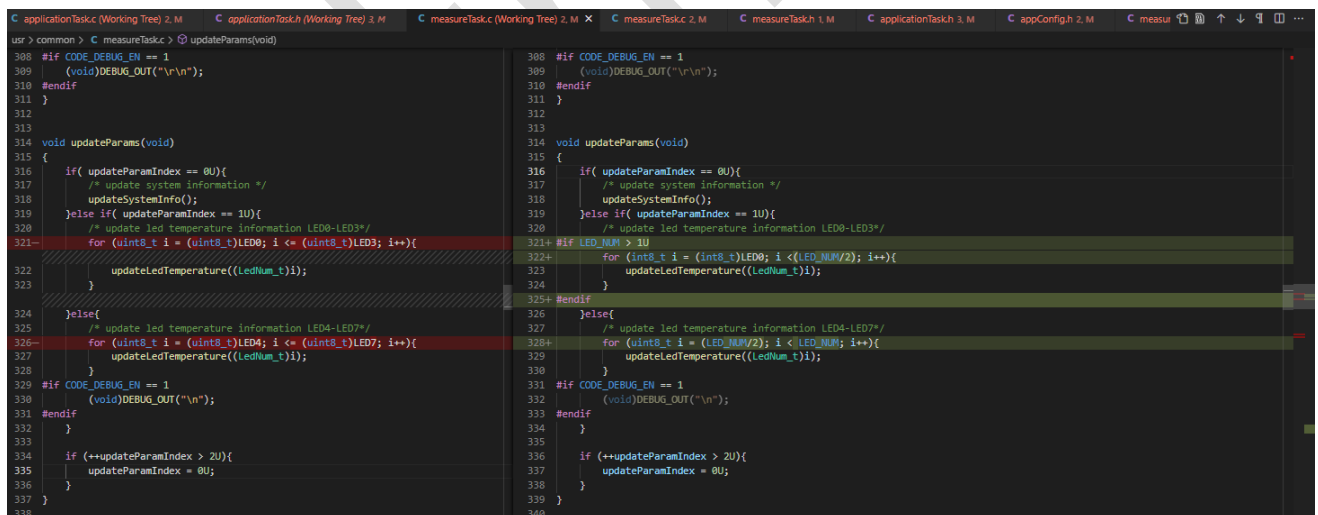
3.2 NEW FEATURES

None

4 CHANGES V1.03

4.1 BUG FIXED

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



```

308 #if CODE_DEBUG_EN == 1
309     (void)DEBUG_OUT("\r\n");
310 #endif
311 }
312
313
314 void updateParams(void)
315 {
316     if( updateParamIndex == 0U){
317         /* update system information */
318         updateSystemInfo();
319     }else if( updateParamIndex == 1U){
320         /* update led temperature information LED0-LED3*/
321         for (uint8_t i = (uint8_t)LED0; i <= (uint8_t)LED3; i++){
322             updateLedTemperature((LedNum_t)i);
323         }
324     }else{
325         /* update led temperature information LED4-LED7*/
326         for (uint8_t i = (uint8_t)LED4; i <= (uint8_t)LED7; i++){
327             updateLedTemperature((LedNum_t)i);
328         }
329     }
330 #if CODE_DEBUG_EN == 1
331     (void)DEBUG_OUT("\r\n");
332 #endif
333 }
334
335 if (++updateParamIndex > 2U){
336     updateParamIndex = 0U;
337 }
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359 #if CODE_DEBUG_EN == 1
360     (void)DEBUG_OUT("\r\n");
361 #endif
362 }
363
364
365 void updateParams(void)
366 {
367     if( updateParamIndex == 0U){
368         /* update system information */
369         updateSystemInfo();
370     }else if( updateParamIndex == 1U){
371         /* update led temperature information LED0-LED3*/
372         #if LED_NUM > 3U
373         for (uint8_t i = (uint8_t)LED0; i < (LED_NUM/2); i++){
374             updateLedTemperature((LedNum_t)i);
375         }
376         #endif
377     }else{
378         /* update led temperature information LED4-LED7*/
379         for (uint8_t i = (LED_NUM/2); i < (LED_NUM); i++){
380             updateLedTemperature((LedNum_t)i);
381         }
382     }
383 #if CODE_DEBUG_EN == 1
384     (void)DEBUG_OUT("\r\n");
385 #endif
386 }
387
388 if (++updateParamIndex > 2U){
389     updateParamIndex = 0U;
390 }
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

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:\appConfig.h (Working Tree) M  C:\applicationTask.c (Working Tree) 2, M  C:\systemInit.c (Working Tree)  C:\clock_device.c (Working Tree) 2, M  C:\buck_device.c (Working Tree) 2, M  C:\inslaveTask_J2602.c (Working Tree)  C:\uart_de...
drivers > hal > src > C:\buck_device.c > (buckSettingTable
131
132 void BUCK_Handler(void)
133 {
134     uint8_t status = BUCKCTRL_REG_BUCKIRQ_STATUS.BYTE;
135     if (buckIsrCallback != NULL){
136         buckIsrCallback(status);
137     }
138     BUCKCTRL_REG_BUCKIRQ_CLEAR.BYTE = 0xFFU;
139 }
140
141
142 void BUCK_DisableBuck(void)
143 {
144     BUCKCTRL_REG_CTRL.ENAREQ = 0U; /* disable buck */
145     while(BUCKCTRL_REG_STATUS.BUCKOFF == 0U){} /* wait until disable finished */
146 }
147
148 void BUCK_EnableBuck(void)
149 {
150     BUCKCTRL_REG_CTRL.ENAREQ = 1U; /* enable buck */
151 }
152
153 void BUCK_RestartBuck(void)
154 {
155     BUCKCTRL_REG_CTRL.ENAREQ = 0U; /* disable buck */
156     while(BUCKCTRL_REG_STATUS.BUCKOFF == 0U){} /* wait until disable finished */
157     BUCKCTRL_REG_CTRL.ENAREQ = 1U; /* enable buck */
158 }
159
131
132 void BUCK_Handler(void)
133 {
134     uint8_t status = BUCKCTRL_REG_BUCKIRQ_STATUS.BYTE;
135     if (buckIsrCallback != NULL){
136         buckIsrCallback(status);
137     }
138     BUCKCTRL_REG_BUCKIRQ_CLEAR.BYTE = 0xFFU;
139 }
140
141
142 void BUCK_DisableBuck(void)
143 {
144     BUCKCTRL_REG_CTRL.ENAREQ = 0U; /* disable buck */
145 }
146
147 void BUCK_EnableBuck(void)
148 {
149     BUCKCTRL_REG_CTRL.ENAREQ = 1U; /* enable buck */
150 }
151

```

3. Change LF clock to 256KHz:

```

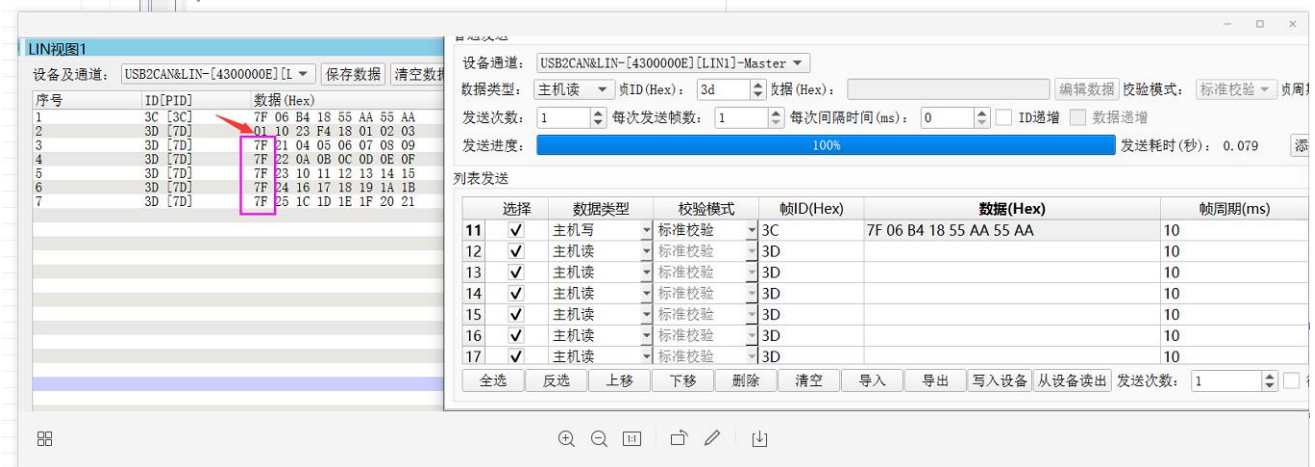
C:\appConfig.h 2, M  C:\measureTask.h (Working Tree) 1, M  C:\appConfig.h (Working Tree) 2, M  C:\startup_M.c (Working Tree)  C:\hwclp.h (Working Tree)  C:\uart_device.h (Working Tree)  C:\clock_device.c (Working Tree) 2, M  C:\...
drivers > hal > src > C:\clock_device.c > Clock_SystemMainClockInit(uint8_t)
13
14
15 #include <clock_device.h>
16
17 #define SYSTEM_CLOCK_HIGH_FREQ (16000U) /* KHz */
18 #define SYSTEM_CLOCK_LOW_FREQ (16U) /* KHz */
19 static uint32_t systemClock = SYSTEM_CLOCK_HIGH_FREQ;
20
21 void Clock_SystemMainClockInit(uint8_t divider)
22 {
23     CRGA_SFRS->SYSCLKCTRL.HFRCENA = 1U;
24     while(CRGA_SFRS->SYSCLKCTRL.HFRCSTS == 0U){}
25     /* Enable trim access write enable */
26     SYSCTRLA_SFRS->TRIM_ACCESS_KEY.KEY = 0x0EU;
27     SYSCTRLA_SFRS->HF_OSC_TRIM.TRIM_HF_RC = HwCfG_GetHfClockCalibValue();
28     SYSCTRLA_SFRS->LF_OSC_TRIM = HwCfG_GetLfcClockCalibValue();
29
30     SYSCTRLA_SFRS->PMU_TRIM.RESISTOR_TRIM = HwCfG_GetV2TrimValue();
31     /* Disable trim access write enable */
32     SYSCTRLA_SFRS->TRIM_ACCESS_KEY.KEY = 0U;
33     CRGA_SFRS->SYSCLKCTRL.SYSCLKSEL = (uint8_t)CLOCK_RC_16MHz;
34     CRGA_SFRS->SYSCLKCTRL.DIVSYSCLK = (uint8_t)divider;
35     /* frequency spread */
36     SYSCTRLA_SFRS->HF_OSC_TRIM.SSCDEEP = 3U; /* 3 */
37     SYSCTRLA_SFRS->HF_OSC_TRIM.SSCDIV = 15U;
38     SYSCTRLA_SFRS->HF_OSC_TRIM.SSCENA = 1U;
39
40     systemClock = SYSTEM_CLOCK_HIGH_FREQ;
41     systemClock = systemClock >> (uint8_t)divider;
42
43     SYSCTRLA_SFRS->PMU_TRIM.TRIM = HwCfG_GetMajorVBGCode();
44 }
45
13
14
15 #include <clock_device.h>
16
17 #define SYSTEM_CLOCK_HIGH_FREQ (16000U) /* KHz */
18 #define SYSTEM_CLOCK_LOW_FREQ (16U) /* KHz */
19 static uint32_t systemClock = SYSTEM_CLOCK_HIGH_FREQ;
20
21 void Clock_SystemMainClockInit(uint8_t divider)
22 {
23     CRGA_SFRS->SYSCLKCTRL.HFRCENA = 1U;
24     while(CRGA_SFRS->SYSCLKCTRL.HFRCSTS == 0U){}
25     /* Enable trim access write enable */
26     SYSCTRLA_SFRS->TRIM_ACCESS_KEY.KEY = 0x0EU;
27     SYSCTRLA_SFRS->HF_OSC_TRIM.TRIM_HF_RC = HwCfG_GetHfClockCalibValue();
28     SYSCTRLA_SFRS->LF_OSC_TRIM = HwCfG_GetLfcClockCalibValue();
29
30     CRGA_SFRS->LFCLKCTRL.CLKPSSEL = 1;
31     SYSCTRLA_SFRS->PMU_TRIM.RESISTOR_TRIM = HwCfG_GetV2TrimValue();
32     /* Disable trim access write enable */
33     SYSCTRLA_SFRS->TRIM_ACCESS_KEY.KEY = 0U;
34     CRGA_SFRS->SYSCLKCTRL.SYSCLKSEL = (uint8_t)CLOCK_RC_16MHz;
35     CRGA_SFRS->SYSCLKCTRL.DIVSYSCLK = (uint8_t)divider;
36     /* frequency spread */
37     SYSCTRLA_SFRS->HF_OSC_TRIM.SSCDEEP = 3U; /* 3 */
38     SYSCTRLA_SFRS->HF_OSC_TRIM.SSCDIV = 15U;
39     SYSCTRLA_SFRS->HF_OSC_TRIM.SSCENA = 1U;
40
41     systemClock = SYSTEM_CLOCK_HIGH_FREQ;
42     systemClock = systemClock >> (uint8_t)divider;
43
44     SYSCTRLA_SFRS->PMU_TRIM.TRIM = HwCfG_GetMajorVBGCode();
45 }
46

```

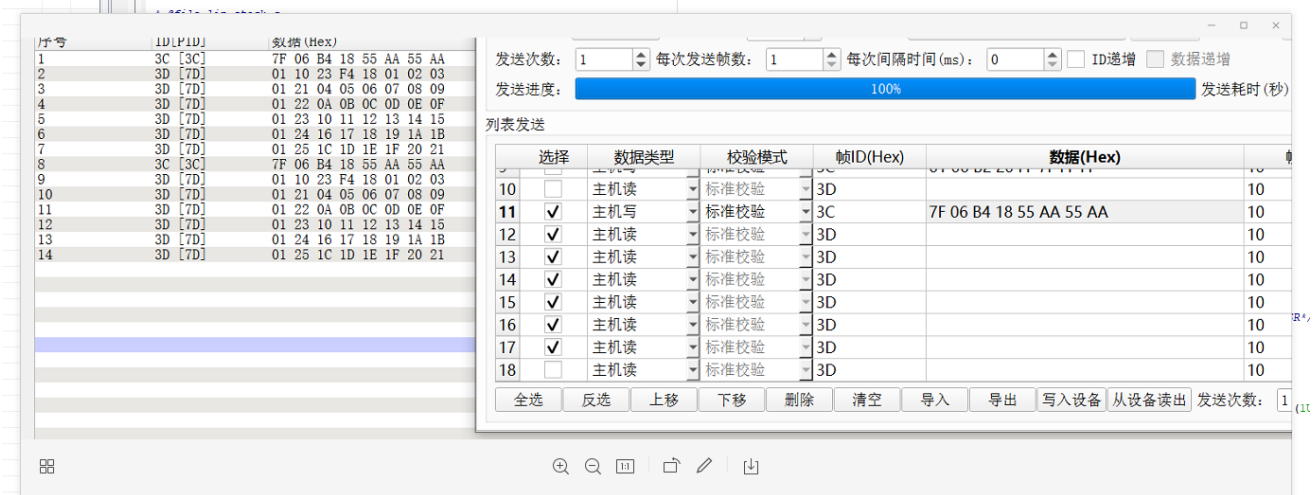
5 CHANGES V1.02

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 2916592..2a1dca3 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
{
    adcMeasParam.item = item;
    adcMeasParam.channel = channel;
+    if(item == ADC_MEASURE_ITEM_VTEMP){
+        ADC_SFRR->CONF_SAMPVCYC = 440;
+    }
+    else{
+        ADC_SFRR->CONF_SAMPVCYC = 100;
+    }
+    ADC_SFRR->CONF_ATTEN = 0;
+    switch(item){
+        case ADC_MEASURE_ITEM_VTEMP: /* 1x */
+            ADC_SFRR->CTRL_CHSEL = (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 PWM_Aux_Init(PWM_Aux_Port_t port, PWM_Aux_ClockSource_t source, PWM_Aux_Prescaler
@@ -25,13 +25,15 @@ void PWM_Aux_Init(PWM_Aux_Port_t port, PWM_Aux_ClockSource_t source, PWM_Aux_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_SFRR->DATA1;
    vBattCode = ADC_SFRR->DATA0345;
+    ADC_SFRR->CONF_SAMPVCYC = 100;
}
if (++ledNo >= LED_NUM){
    ledNo = 0;
@@ -187,6 +188,7 @@ void ADC_Handler(void)
    ledConvertCount = 0;
    ADC_REG_CTRL_CH1_CH2 = (uint8_t)ADC_CH1_CH3_SEL_TEMP_SENSOR + ( ledPhyChannel << 2);
    ADC_SFRR->CTRL_CHSEL = (uint8_t)ADC_CH1_CH3_SEL_TEMP_SENSOR;
+    ADC_SFRR->CONF_SAMPVCYC = 440;
    ledConvertType = LED_CONVERT_TYPE_VTEMP_VBAT;
}
ADC_REG_CTRL_CONVERT = 0x0;
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 PWM_Aux_Init(PWM_Aux_Port_t port, PWM_Aux_ClockSource_t source, PWM_Aux_Prescaler
IOCTRLA_SFRR->GPIO[(uint8_t)port].PULLMODE = (uint8_t)GPIO_PULL_NONE;
GPIO_SFRR->GPIO_CFG[(uint8_t)port].DIR = (uint8_t)GPIO_DIR_OUTPUT;

- PWM_Aux_SFRR->BASE_SEL = (uint8_t)source;
PWM_Aux_SFRR->TIMER[(uint8_t)source].PRESCALESEL = (uint8_t)devider;
PWM_Aux_SFRR->TIMER[(uint8_t)source].PERIOD = peroid;
if (port <= PWM_Aux_PORT_5){
    PWM_Aux_SFRR->ENAREQ_ENAREQ |= 1U << (uint8_t)port; /* CHN:0-4 */
+    PWM_Aux_SFRR->BASE_SEL |= (uint8_t)source << (uint8_t)port;
}
else{
    PWM_Aux_SFRR->ENAREQ_ENAREQ |= 1U << ((uint8_t)port - (uint8_t)PWM_Aux_PORT_5); /* CHN:1-4 */
+    PWM_Aux_SFRR->BASE_SEL |= (uint8_t)source << ((uint8_t)port - (uint8_t)PWM_Aux_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 PWM_Aux_Init(PWM_Aux_Port_t port, PWM_Aux_ClockSource_t source, PWM_Aux_Prescaler
IOCTRLA_SFRR->GPIO[(uint8_t)port].PULLMODE = (uint8_t)GPIO_PULL_NONE;
GPIO_SFRR->GPIO_CFG[(uint8_t)port].DIR = (uint8_t)GPIO_DIR_OUTPUT;

- PWM_Aux_SFRR->BASE_SEL = (uint8_t)source;
+ PWM_Aux_SFRR->TIMER[(uint8_t)source].PRESCALESEL = (uint8_t)devider;
PWM_Aux_SFRR->TIMER[(uint8_t)source].PERIOD = peroid;
if (port <= PWM_Aux_PORT_5){
    PWM_Aux_SFRR->ENAREQ_ENAREQ |= 1U << (uint8_t)port; /* CHN:0-4 */
+    PWM_Aux_SFRR->BASE_SEL |= (uint8_t)source << (uint8_t)port;
}
else{
    PWM_Aux_SFRR->ENAREQ_ENAREQ |= 1U << ((uint8_t)port - (uint8_t)PWM_Aux_PORT_5); /* CHN:1-4 */
+    PWM_Aux_SFRR->BASE_SEL |= (uint8_t)source << ((uint8_t)port - (uint8_t)PWM_Aux_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

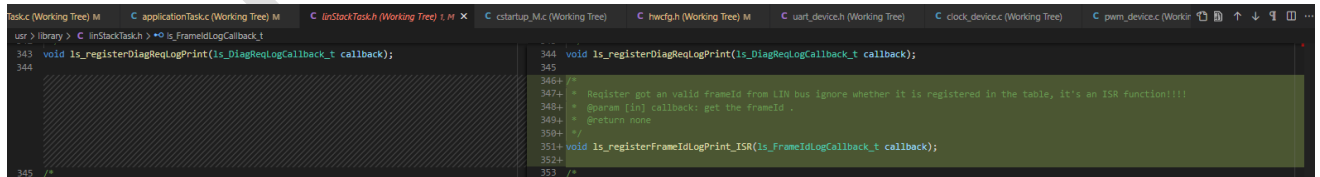
7.1 BUG FIXED

Fixed Aux PWM duty cycle issue

```
void PwmAux_SetMatchValue(PwmAuxChannel_t channel, uint16_t matchRisingValue, uint16_t matchFallValue)
{
    Pwm_AUX_REG_PULSE(((uint8_t)channel) - (uint32_t)matchFallValue + ((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 sequence.



```
Task.c (Working Tree) M applicationTask.c (Working Tree) M arStackTask.h (Working Tree) t M x c startup_M.c (Working Tree) hwdfgh (Working Tree) M uart_device.h (Working Tree) clock_device.c (Working Tree) pwm_device.c (Working Tree)
usr > library > C arStackTask.h > #include FrameIdLogCallback_t
343 void ls_registerDiagReqLogPrint(ls_DiagReqLogCallback_t callback);
344
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 understood through source code, it is the same as function `ls_read_sys_status()`.

```

424+
425+ /*
426+  * Read current error flag and system information, simulate with ls_read_sys_status
427+  * @return:
428+  * 1:usb error_in_resp
429+  * 1:usb success_in_transfer
430+  * 1:usb overrun
431+  * 1:usb goto_sleep
432+  * 1:usb bus_activity
433+  * 1:usb event_type_frame_collision
434+  * 1:usb save_config
435+  * 1:usb parity_error
436+  * 1:usb pid;
437+ */
438+ lin_status_t ls_read_sys_status_ext(void);
439+
440+
441+

```

8 CHANGES V0.42

8.1 BUG FIXED

```

C: appConfig.h (Working Tree) C: applicationTask.c (Working Tree) C: systemInit.c (Working Tree) 2. M X C: measureTask.c (Working Tree) 2. M X C: clock_device.c (Working Tree) C: inslaveTask_J2602.c (Working Tree) C: uart_device.c (Working Tree) C: lish.c
usr > common > C: measureTask.c > MES_MinIGetInfoLedNum_t uint16_t*, uint16_t*, uint16_t*
446 uint8_t lastCHSEQ;
447 uint8_t lastCH2;
448 if (led_no < LED_NUM){
449     ADC_UnregisterIRQ();
450     lastCHSEQ = ADC_SFRR->CNTRL.CHSEQ;
451     lastCH2 = ADC_REG_CNTRL.CH2SEL;
452
453     if ((led_no - ((led_no >> 1)*2U)) == 0U){
454 #if LED_BOARD_TYPE == BOARD_TYPE_DP
455     GPIO_SetGPIOs((1U << GPIO_PORT_2) | (1U << GPIO_PORT_3), (1U << GPIO_PORT_2));
456 #elif LED_BOARD_TYPE == BOARD_TYPE_SY
457     GPIO_SetGPIOs((1U << GPIO_PORT_2) | (1U << GPIO_PORT_3), (1U << GPIO_PORT_3)); /* 0,2,4,6,8,10,12,14,16 */
458 #endif
459     }else{
460 #if LED_BOARD_TYPE == BOARD_TYPE_DP
461     GPIO_SetGPIOs((1U << GPIO_PORT_2) | (1U << GPIO_PORT_3), (1U << GPIO_PORT_3));
462 #elif LED_BOARD_TYPE == BOARD_TYPE_SY
463     GPIO_SetGPIOs((1U << GPIO_PORT_2) | (1U << GPIO_PORT_3), (1U << GPIO_PORT_2)); /* 0,2,4,6,8,10,12,14,16 */
464 #endif
465     }
466 }

```

8.2 NEW FEATURES

None

9 CHANGES V0.41

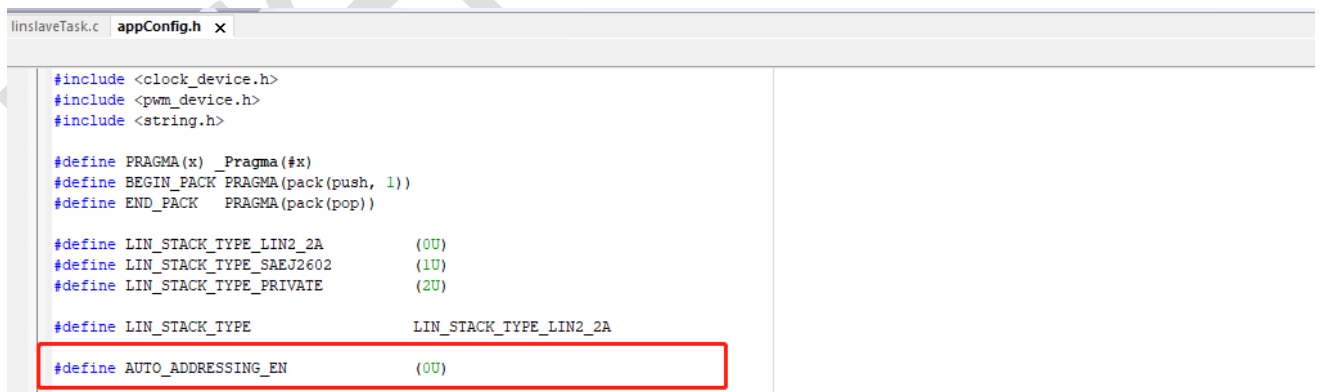
11 CHANGES V0.3

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 x
#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 x

SYSTRLA_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 Tx 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.DISAUTOKEEP = 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*/
SYSTRLA_SFRS->LIN.TXLINSRISSESLOPE = 0U;
SYSTRLA_SFRS->LIN.TXLINS_DR_SLOPE = LIN_TX_SLEW_RATE_2_1V_PER_US;
SYSTRLA_SFRS->LIN.TXLINNRISSESLOPE = 0U;
SYSTRLA_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_FU1K_ENA = 1U; /* LIN 1K pullup enable.*/
SYSTRLA_SFRS->LIN.TXLINSRISSESLOPE = 1U;
SYSTRLA_SFRS->LIN.TXLINS_DR_SLOPE = LIN_TX_SLEW_RATE_5_1V_PER_US;
SYSTRLA_SFRS->LIN.TXLINNRISSESLOPE = 1U;
SYSTRLA_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 x | 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, (1_u8)(sizeof(UnconditionalCmdsTable)/sizeof(LIN_Device_Frame_t)), DIAG_DATA_BUFF_SIZE,&linsFramesCallback);
(void)ls_set_timeout(N AS, N CR);
ls_registerDiagReqLogPrint(DiagReqLogPrint);
/* =====> User please add some code for adding custom configuration!!!! */
(void)ls_set_lins_rx_glitch_filter_1st(0x00U,0x00U);
(void)ls_set_lins_rx_glitch_filter_2nd(0x00U,0x10U);
(void)ls_set_lins_rx_glitch_filter_3rd(0x30U,0x30U);
(void)ls_sys_init();
linsTaskState = TASK_STATE_ACTIVE;
break;
default:
break;
}
}
```

3. Add Milky Way UART driver and application:

```
linStackTask.c | appConfig.h x | lin_device.c | linSlaveTask.c

/* =====> System config ===== */
#define LIN_MASTER_EN (0U)
#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 | x
    response = TRUE;
    break;
default:
    break;
}
return response;
}

#endif

void APP_UART_ISR(uint8_t* buff, uint8_t length)
{
    rxBufflength = length;
    memcpy(rxBuff, buff, length);
    TM_PostTask(TASK_ID_APP1);
}

void handleColorFromDart(uint8_t* buff, uint8_t length)
{
    APPXYT_t *color = (APPXYT_t *)buff;
    for (uint8_t i = 0; i < LED_NUM; i++) {
        (void)CWM_SetXY((LedNum_t)i, WES_GetLedTemperature((LedNum_t)i), color->xyY[i].x, color->xyY[i].y, (uint16_t)color->xyY[i].Y*100, 0);
    }
}

#endif

```

```

linStackTask.c | appConfig.h | lin_device.c | linSlaveTask.c | applicationTask.c | systemInit.c | x
/* Init set BOR voltage level for cpu low voltage safety*/
PMU_BORInit(BOR_VS_THRS_1385mV, BOR_VVS_THRS_3034mV);
// Disable wake up timer */
PMU_WakeTimeInit(WAKEUP_TIMER_DISABLE, WAKEUP_TIMER_INTERVAL_32768ms);

void SYS_Init(void)
{
    /* Enable trim revise access enable*/
    BMTF_TrimAccessEnClock();
    CROA_SFPS->MODULERSTREQ_BYTE = 0xFF;
    /* Init system clock */
    Clock_SystemMainClockInit(SYS_MAIN_CLOCK_DIV);
    psm_init();
    /* Init global timer engine for driving soft timer */
    SysTick_Init(SOFT_TIMER_INTERVAL *10000 * MAIN_CPU_CLOCK, SoftTimer_ExpireCallback);
}

/*if WATCH_DOG_EN == 10
WDTA_Enable(WDTA_INTERVAL_8S); /* 8s */
#endif

/* Init gpio settings */
gpio_init();
/* Init LED current and PWM settings */
leds_driver_Init();

/*if (CODE_DEBUG_EN == 10) || (UART_MILKY_WAY_EN == 10)
(void)UART_Init(SAURATE_DIV_16MHz_1000000, BITSIZE_8BITS, PARITY_NONE, STOPS_1BITS);
UART_RegisterIRQ((UART_ID_APP1));
#endif

/* Init Buck settings */
Buck_Init(BUCK_OUTPUV_3800mV);

/* Disable trim revise access until reset*/

```

```

linStackTask.c | appConfig.h | lin_device.c | linSlaveTask.c | applicationTask.c | systemInit.c | uart_device.c x
{
    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->PIPOLEVELCTL.RXMULTIPLEXERDONECNT = 9;

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

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

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

    NVIC_EnableIRQ(UART_IRQn);
}

void UART_UnRegisterIRQ(void)
{
    uartRxDataCallback = NULL;
    UART_SFRS->INT.ENABLE.RXMULTIDONE = 0U;
    UART_SFRS->INT.ENABLE.TXDONE = 0U;
    UART_SFRS->INT.ENABLE.BREAKERR = 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;
}

```