ST uviedol v upgrade knižnice Cube pre (mimochodom, pomerne čerstvú) pod-rodinu STM32L4xx nové rozhranie s názvom **Low Layer**.
Toto rozhranie postupne v nasledujúcich mesiacoch]pribudne do všetkých knižníc Cube. ST LL vyvíja ako náhradu za staršie SPL(Standard Peripheral Library), ktorý pre novšie rodiny (vrátane STM32L4xx) nie je dodávaný.

Rozhranie je zdokumentované spôsobom obvyklým pre Cube, v doxygenovskom autogenerovanom STM32Cube_FW_L4_V1.1.0\Drivers\STM32L4xx_HAL_Driver\STM32L486xx_User_Manual.chm ako aj v STM32Cube_FW_L4_V1.1.0\Documentation\STM32CubeL4GettingStarted.pdf

Implementácia je vo forme makier a static inline funkcií, v súboroch STM32Cube_FW_L4_V1.1.0\Drivers\STM32L4xx_HAL_Driver\Inc\stm32l4xx_ll_XXX.h (dva súbory sa nachádzajú aj v adresári ..\Src).

Niekoľko príkladov použitia rozhrania LL pre rôzne periférie sa dajú nájsť v STM32Cube_FW_L4_V1.1.0\Projects\STM32L476RG-Nucleo\Examples_LL\. Skrátená ukážka z príkladu "blikača":

```
1.
2.
3.  int main(void)
4.  {
5.    /* Configure the system clock to 80 MHz */
6.    SystemClock_Config();
7.
8.    /* -2- Configure IO in output push-pull mode to drive external LED */
9.    Configure_GPIO();
10.
11.   /* Toggle IO in an infinite loop */
12.   while (1)
13.   {
14.     LL_GPIO_TogglePin(LED2_GPIO_PORT, LED2_PIN);
15.
```

```
16.     /* Insert delay 250 ms */
17.     LL_mDelay(250);
18.   }
19. }
20.
21.
22. __STATIC_INLINE void Configure_GPIO(void)
23. {
24.   /* Enable the LED2 Clock */
25.   LED2_GPIO_CLK_ENABLE();
26.
27.   /* Configure IO in output push-pull mode to drive external LED2 */
28.   LL_GPIO_SetPinMode(LED2_GPIO_PORT, LED2_PIN,             LL_GPIO_MODE_OUTPUT);
29.   LL_GPIO_SetPinOutputType(LED2_GPIO_PORT, LED2_PIN,             LL_GPIO_OUTPUT_PUSHPULL);
30.   LL_GPIO_SetPinSpeed(LED2_GPIO_PORT, LED2_PIN,             LL_GPIO_SPEED_LOW);
31.   LL_GPIO_SetPinPull(LED2_GPIO_PORT, LED2_PIN,             LL_GPIO_PULL_NO);
32. }
33.
34. __STATIC_INLINE void SystemClock_Config(void)
35. {
```

```
36.
    /* MSI configuration and activation */
37.
    LL_FLASH_SetLatency(LL_FLASH_LATENCY_4);
38.
    LL_RCC_MSI_Enable();
39.
    while(LL_RCC_MSI_IsReady()              != 1)
40.
    {
41.
    };
42.


43.
    /* Main PLL configuration and activation */
44.
    LL_RCC_PLL_ConfigDomain_SYS(LL_RCC_PLLSOURCE_MSI, LL_RCC_PLLM_DIV_1, 40, LL_RCC_PLLR_DIV_2);
45.
    LL_RCC_PLL_Enable();
46.
    LL_RCC_PLL_EnableDomain_SYS();
47.
    while(LL_RCC_PLL_IsReady()              != 1)
48.
    {
49.
    };
50.


51.
    /* Sysclk activation on the main PLL */
52.
    LL_RCC_SetAHBPrescaler(LL_RCC_SYSCLK_DIV_1);
53.
    LL_RCC_SetSysClkSource(LL_RCC_SYS_CLKSOURCE_PLL);
54.
    while(LL_RCC_GetSysClkSource()              != LL_RCC_SYS_CLKSOURCE_STATUS_PLL)
55.
    {
```

```
};


/* Set APB1 & APB2 prescaler*/
LL_RCC_SetAPB1Prescaler(LL_RCC_APB1_DIV_1);
LL_RCC_SetAPB2Prescaler(LL_RCC_APB2_DIV_1);


/* Set systick to 1ms in using frequency set to 80MHz */
/* This frequency can be calculated through LL RCC macro */
/* ex: __LL_RCC_CALC_PLLCLK_FREQ(__LL_RCC_CALC_MSI_FREQ(LL_RCC_MSIRANGESEL_RUN, LL_RCC_MSIRANGE_6),


                                 LL_RCC_PLLM_DIV_1, 40, LL_RCC_PLLR_DIV_2)*/
LL_Init1msTick(80000000);


/* Update CMSIS variable (which can be updated also through SystemCoreClockUpdate function) */
LL_SetSystemCoreClock(80000000);
}
```

Hodnotenie tohoto počinu nechávam na čitateľoch uzone.