# WT32L064 EVB
## User Guide

**Rev. 1.0**

**April 2022**



**Disclaimers**

**Right to make change** –

This document provides technical information for user. Weltrend Semiconductor, Inc. reserves the right to make change without further notice to any products herein.

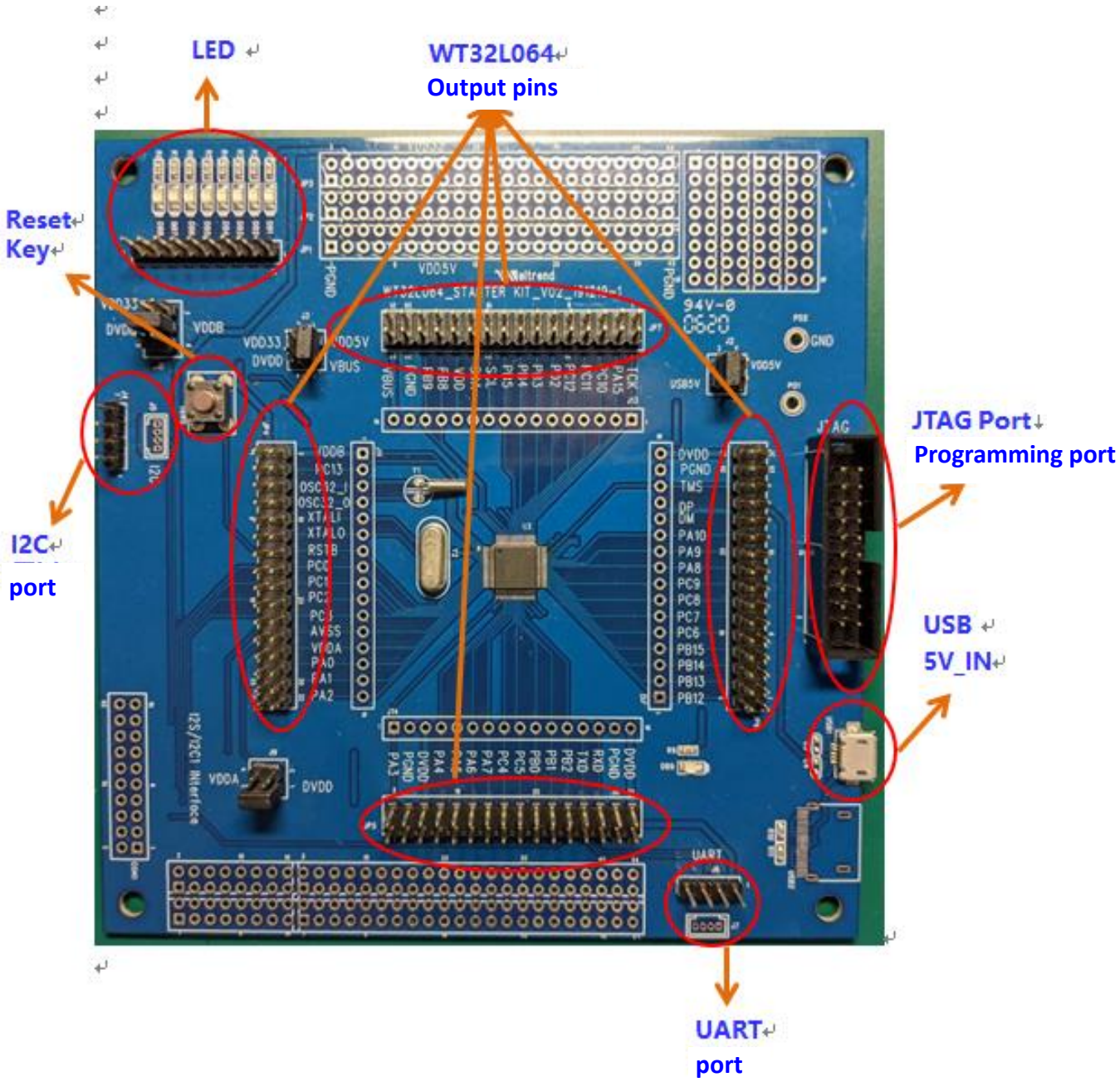
# Table of Contents

## 1. EVB Introduction

The WT32L064 is an ultra-low power ARM Cortex M0-based 32-bit microcontroller with 64KB embedded flash memory and 8KB SRAM.

This Starter Kit Board uses a 64-pin LQFP package and demonstrates its functions.

### 1.1 Overview:



### 1.2 EVB Port:

#### 1.2.1 Micro USB Port (USB1)

EVB DC voltage input port (support voltage: 5V)

| Pad Number | Description |
|---|---|
| 1 | VBUS |
| 2 | D- |
| 3 | D+ |
| 4 | GND |
| 5 | GND |

#### 1.2.2 I2C Interface Port (J4/J5)

SLAVE I2C interface

| Pad Number | Description |
|---|---|
| 1 | DVDD (3.3V) |
| 2 | SCL |
| 3 | SDA |
| 4 | GND |

#### 1.2.3 UART Interface Port (J6/J7)

UART interface

| Pad Number | Description |
|---|---|
| 1 | DVDD (3.3V) |
| 2 | RXD |
| 3 | TXD |
| 4 | GND |

### 1.2.4 JTAG Interface Port (J6/J7)

This is JTAG interface, used as a programming program and debugging.

| Pad Number | Description | Pad Number | Description |
|---|---|---|---|
| 1 | VDD33 | 2 | VDD33 |
| 3 | NC | 4 | GND |
| 5 | NC | 6 | GND |
| 7 | TMS | 8 | GND |
| 9 | TCK | 10 | GND |
| 11 | NC | 12 | GND |
| 13 | NC | 14 | GND |
| 15 | RSTB | 16 | GND |
| 17 | NC | 18 | GND |
| 19 | NC | 20 | GND |

### 1.2.5 WT32L064 Pin Output pin (JP4/JP5/FP6/JP7)

This is WT32L064 pin output interface, used for external testing.

| JP4 | | | |
|---|---|---|---|
| **Pad Number** | **Description** | **Pad Number** | **Description** |
| 1-2 | VDDB | 3-4 | GPIOC13 |
| 5-6 | GPIOC14/LXTALI | 7-8 | GPIOC15/LXTALO |
| 9-10 | GPIOD0/HXTALI | 11-12 | GPIOD1/HXTALO |
| 13-14 | NRST | 15-16 | GPIOC0 |
| 17-18 | GPIOC1 | 19-20 | GPIOC2 |
| 21-22 | GPIOC3 | 23-24 | AVSS |
| 25-26 | VDDA | 27-28 | GPIOA0 |
| 29-30 | GPIOA1 | 31-32 | GPIOA2 |

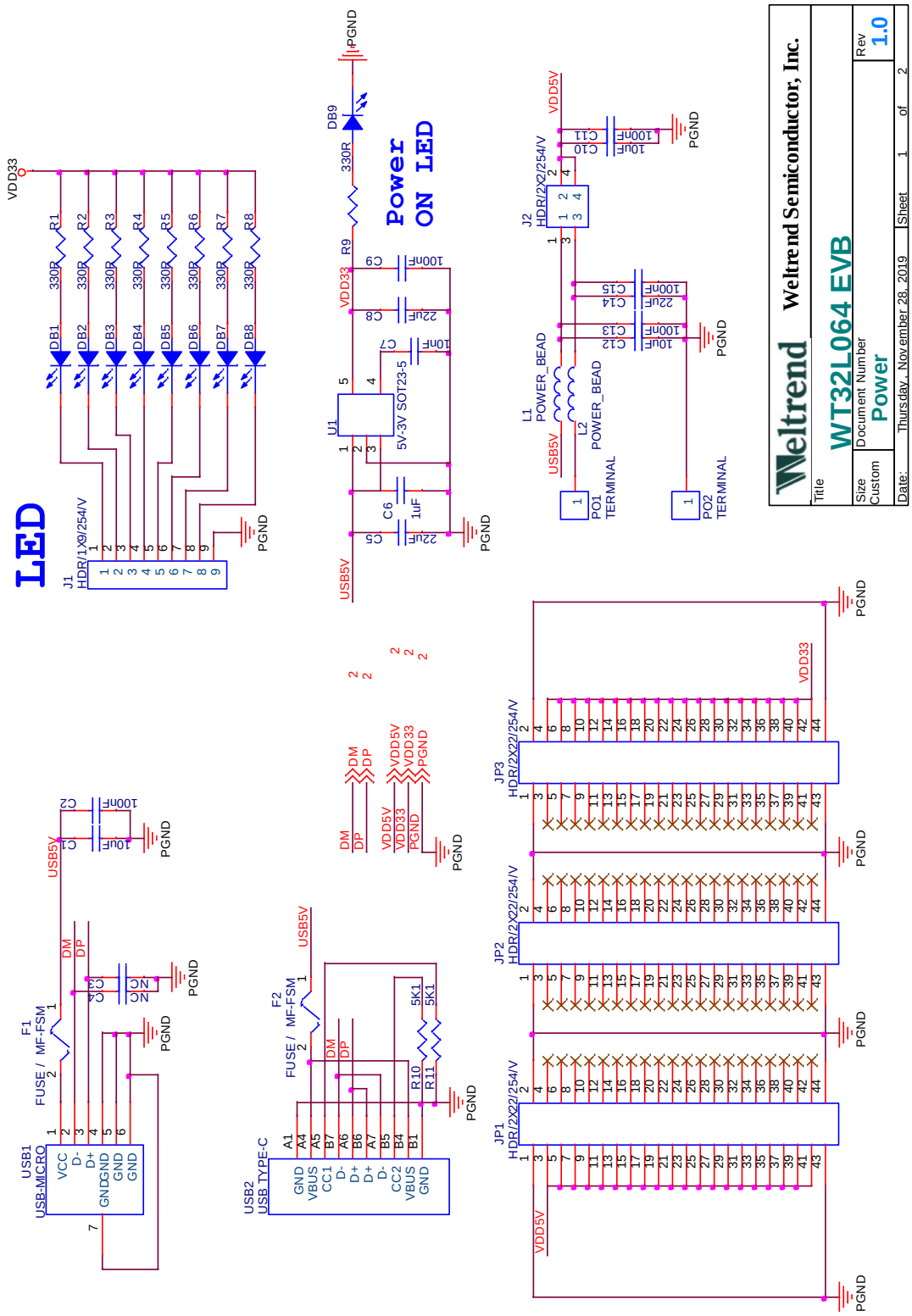
| JP5 | | | |
|---|---|---|---|
| **Pad Number** | **Description** | **Pad Number** | **Description** |
| 1-2 | GPIOA2 | 3-4 | VSS |
| 5-6 | DVDD | 7-8 | GPIOA4 |
| 9-10 | GPIOA5 | 11-12 | GPIOA6 |
| 13-14 | GPIOA7 | 15-16 | GPIOC4 |
| 17-18 | GPIOC5 | 19-20 | GPIOB0 |
| 21-22 | GPIOB1 | 23-24 | GPIOB2 |
| 25-26 | GPIOB10/TXD | 27-28 | GPIOB11/RXD |
| 29-30 | VSS | 31-32 | DVDD |

| JP6 | | | |
|---|---|---|---|
| **Pad Number** | **Description** | **Pad Number** | **Description** |
| 1-2 | GPIOB12 | 3-4 | GPIOB13 |
| 5-6 | GPIOB14 | 7-8 | GPIOB15 |
| 9-10 | GPIOC6 | 11-12 | GPIOC7 |
| 13-14 | GPIOC8 | 15-16 | GPIOC9 |
| 17-18 | GPIOA8 | 19-20 | GPIOA9 |
| 21-22 | GPIOA10 | 23-24 | GPIOA11/DM |
| 25-26 | GPIOA12/DP | 27-28 | GPIOA13/TMS |
| 29-30 | VSS | 31-32 | DVDD |

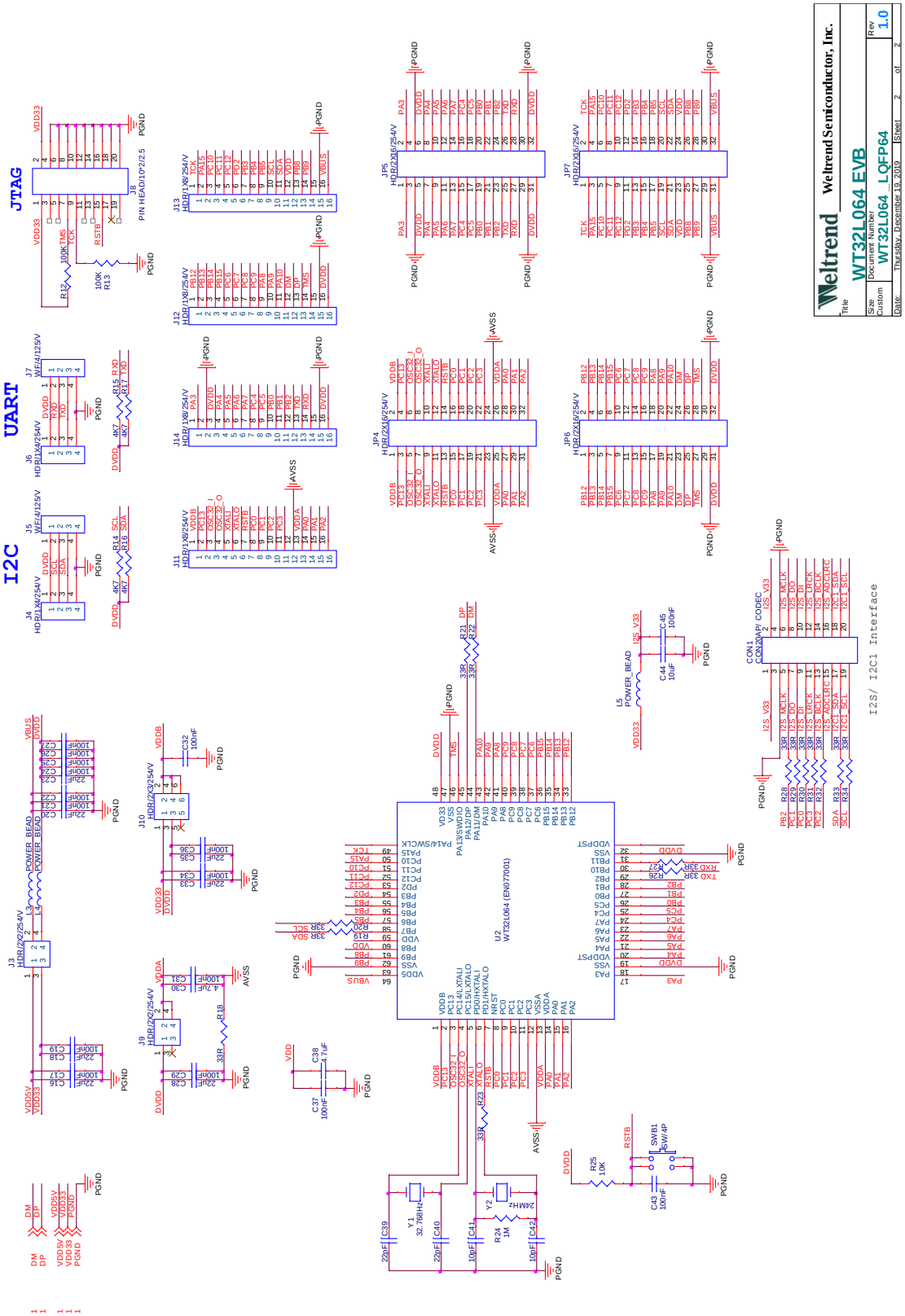
| JP7 | | | |
|---|---|---|---|
| **Pad Number** | **Description** | **Pad Number** | **Description** |
| 1-2 | GPIOA14/SWCLK | 3-4 | GPIOA15 |
| 5-6 | GPIOC10 | 7-8 | GPIOC11 |
| 9-10 | GPIOC12 | 11-12 | GPIOD2 |
| 13-14 | GPIOB3 | 15-16 | GPIOB4 |
| 17-18 | GPIOB5 | 19-20 | GPIOB6 |
| 21-22 | GPIOB7 | 23-24 | VDD |
| 25-26 | GPIOB8 | 27-28 | GPIOB9 |
| 29-30 | VSS | 31-32 | VDD5 |

## 1.3 EVB Schematic Diagram:

### 1.3.1 Power

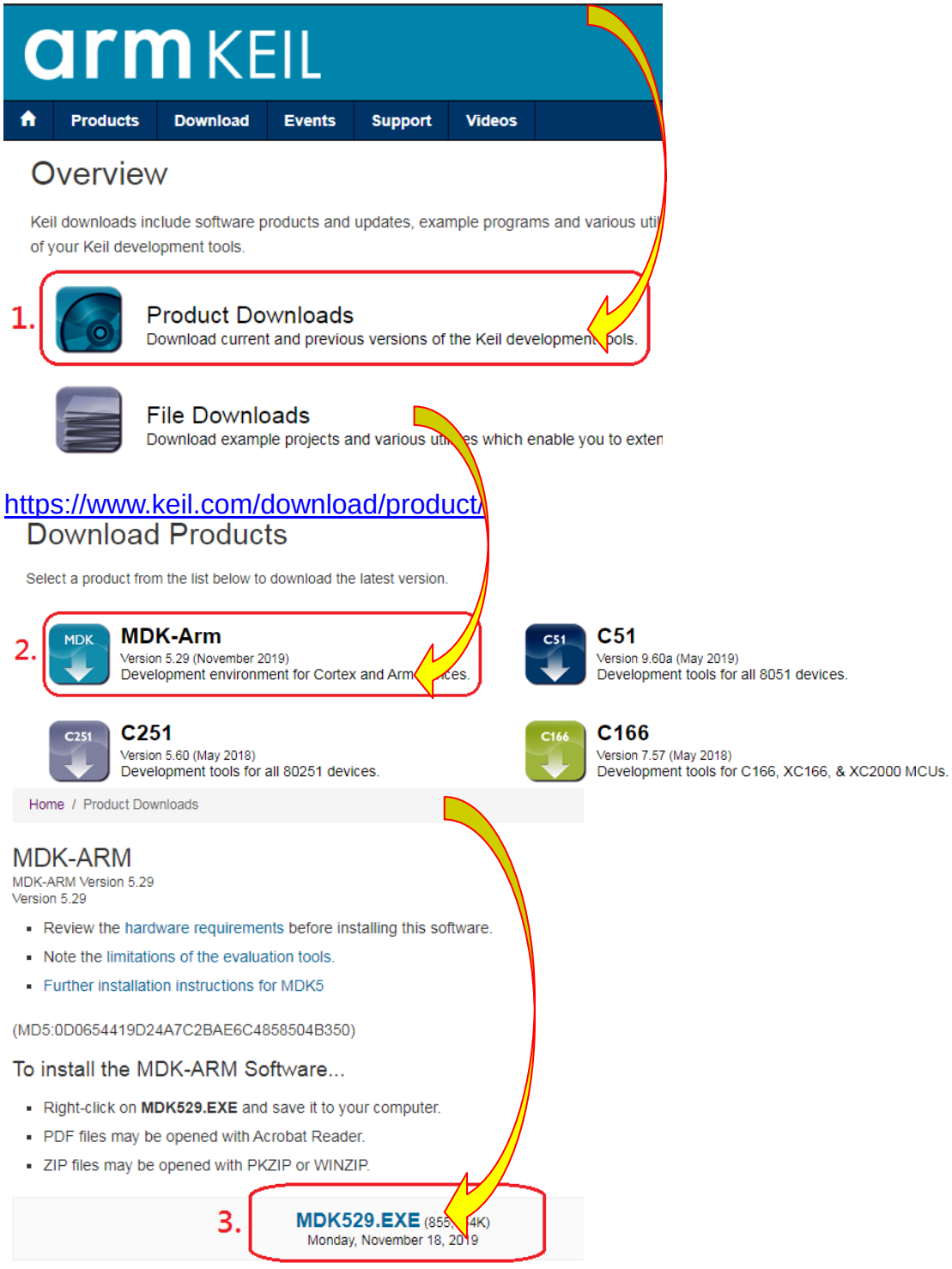


### 1.3.2 WT32L064

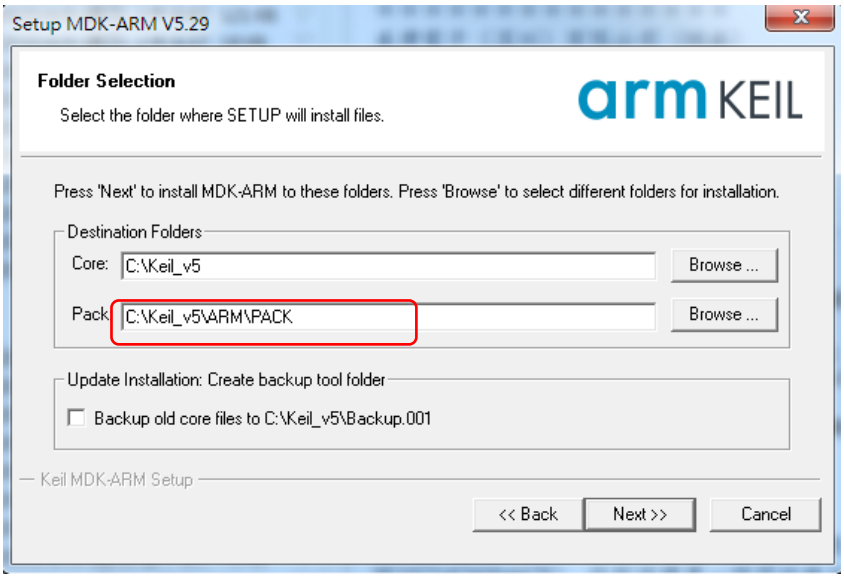


## 2. ARM-MDK Installation & Environment setting

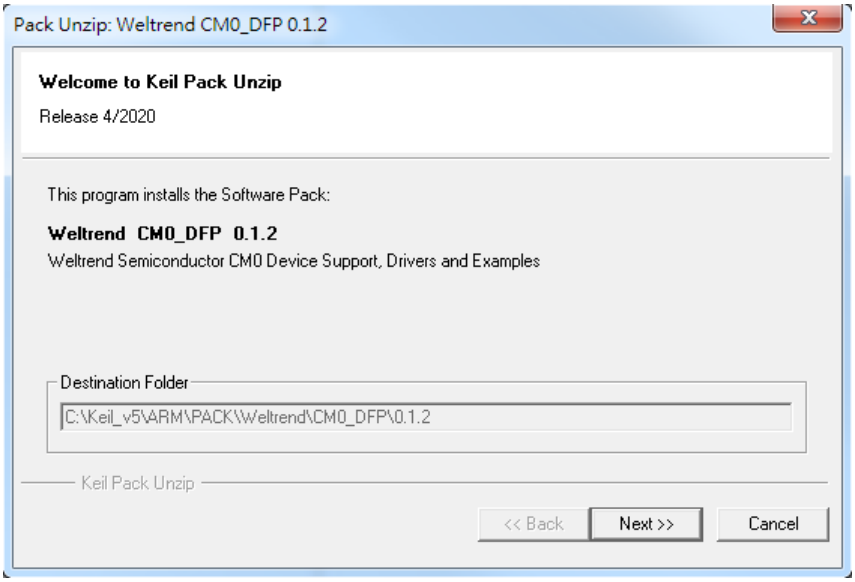
(Step 1) **Please download ARM-MDK** https://www.keil.com/download/



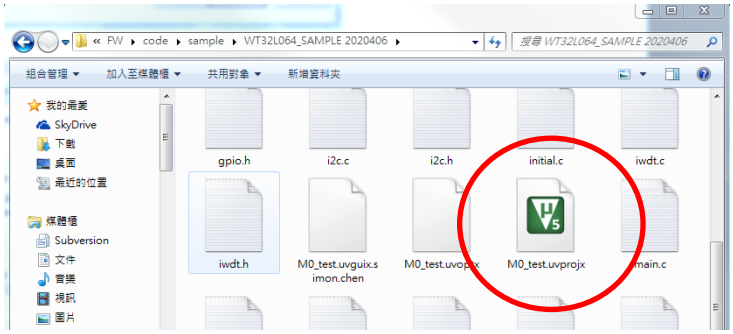
The default PACK path will be asked during installation, please specify **C:\Keil_v5\ARM\PACK** as follows to avoid subsequent PACK installation problems.




**(Step 2)** After downloading and installing MDK, please install Weltrend PACK file (Weltrend.CM0_DFP.0.1.x.pack) in your PC.



**(Step 3)** After installing the ARM-MDK, the basic 32KB is free for use, or you can purchase the software by yourself. After installation, please open the relevant WT32L064 project on your computer for compilation.



## 3. CMSIS Middleware Driver

### 3.1 Definition:

ARM® Cortex™ Microcontroller Software Interface Standard (CMSIS) is a set of firmware libraries that can drive ARM processors. The firmware interface provides a standard function directly for the peripheral with the same name and is easy to use, which can be used repeatedly by software, reducing developing time for microcontroller developers. Based on this framework, the manufacturer provides a set of basic peripheral applications of sample programs or peripheral libraries, which can directly focus on the application side to speed up program operation and editing.

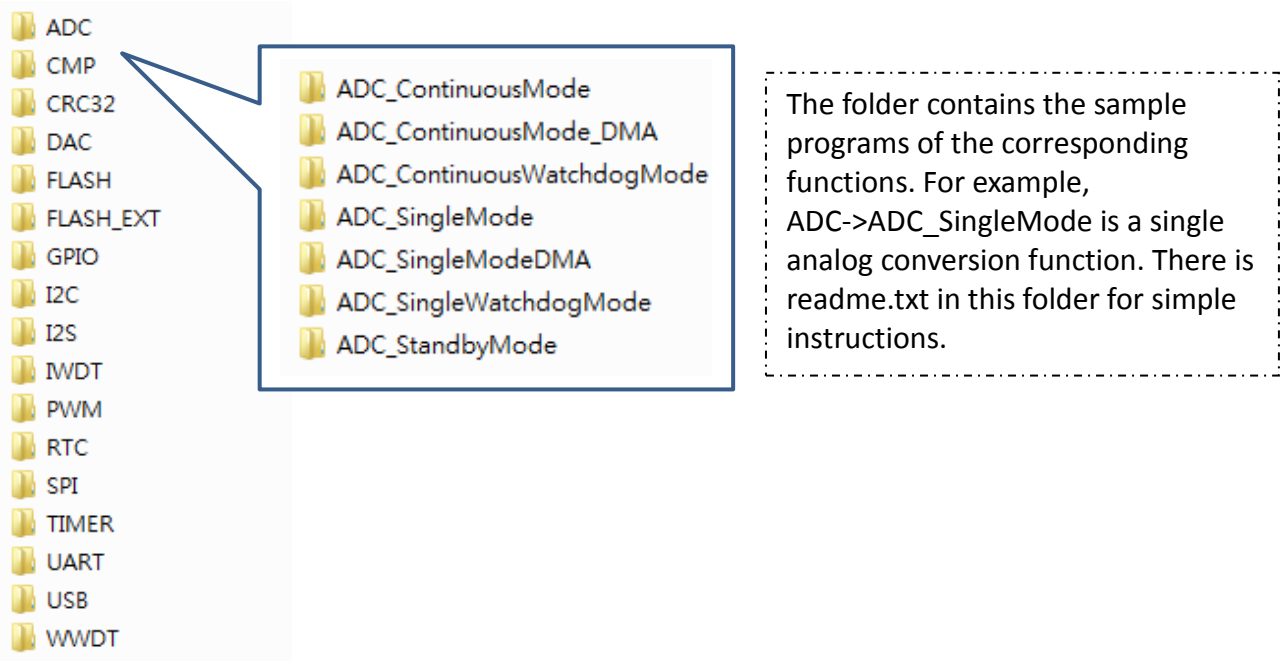
### 3.2 CMSIS Content:

After installing the WT32L064 PACK, the default CMSIS path is ***C:\Keil_v5\ARM\Packs\Weltrend\CM0_DFP\0.1.x \WT32L064\StdPeriph_Driver***. The header file is placed in the **Include** folder, the original file is placed in the **Source** folder, and which contents include the basic settings for all peripherals of WT32L064. The files list is as follows.

| File Name | Description |
|---|---|
| wt32l064_adc | Analog detect ADC related function |
| wt32l064_crc32 | CRC32 function |
| wt32l064_crs | Calibrated IC Internal Frequency related function |
| wt32l064_dac | Analog output DAC related function |
| wt32l064_dma | DMA related function |
| wt32l064_flash | EEPROM programming FLASH related function |
| wt32l064_gpio | GPIO related function |
| wt32l064_i2c | I2C related function |
| wt32l064_i2s | I2S related function |
| wt32l064_iwdt | IWDT Independent Watchdog related function |
| wt32l064_pmu | PMU Power Control Unit related function |
| wt32l064_pwm | PWM related function |
| wt32l064_rcc | RCC frequency control unit related function |
| wt32l064_rtc | RTC Timer related function |
| wt32l064_spi | SPI related function |
| wt32l064_timer | TIMER related function |
| wt32l064_usart | UART related function |
| wt32l064_usbd | USB related function |
| wt32l064_wwdt | WWDT Window Watchdog related function |

## 4. Structure of PACK Example Program

There are basic example programs for various application units. After PACK is installed, refer to the following path ***C:\...\Arm\Packs\Weltrend\ CM0_DFP\0.1.x \WT32L064\Examples***. There are sub-units in the folder that contain original files and Project. The following is an ADC sample program, which can be classified into single conversion and continuous conversion according to its function, as shown in the figure below.




### 4.1 Examples folder

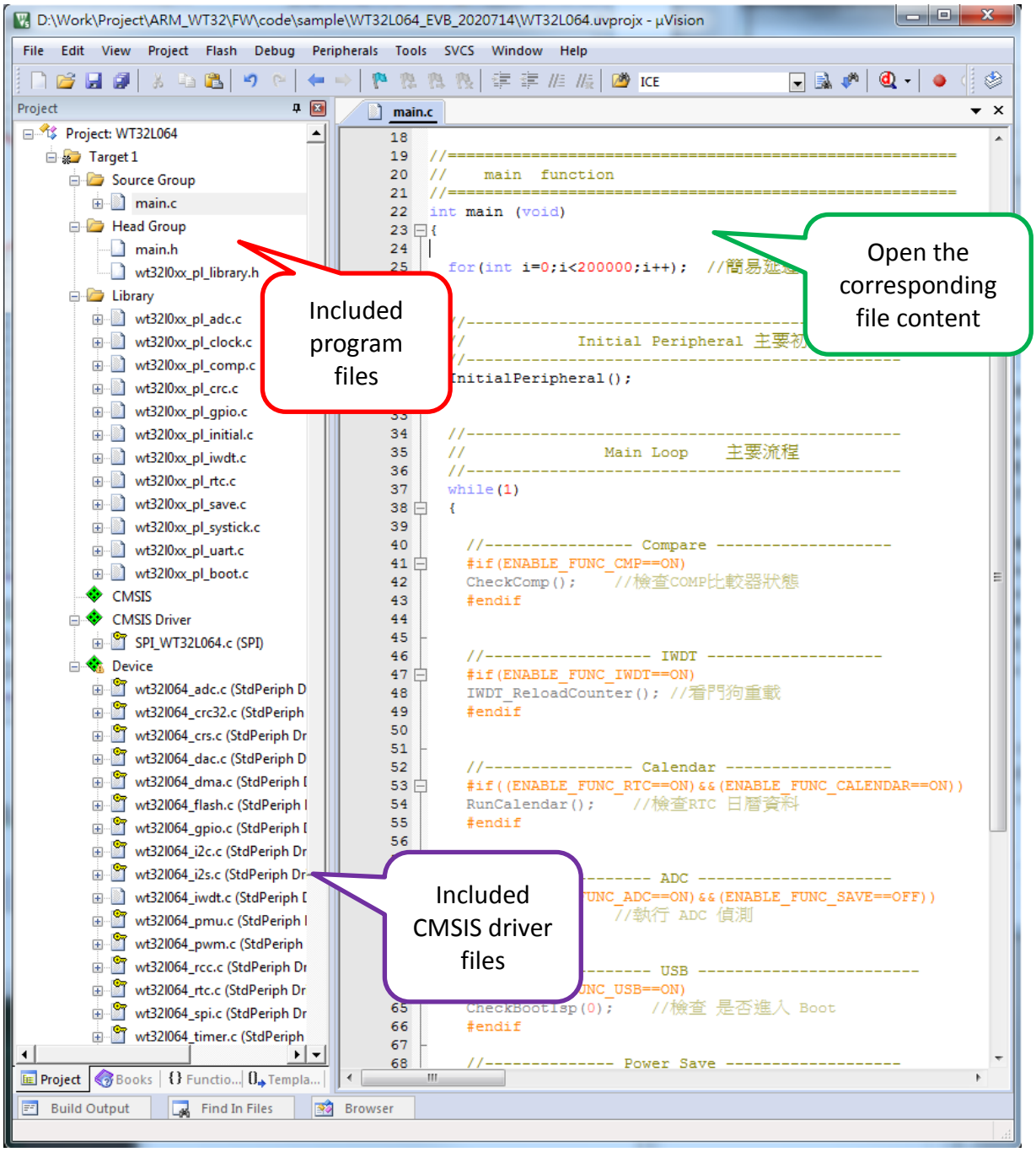
| Root Folder | Folder Name | Description |
|---|---|---|
| ADC | ADC_ContinuousMode | Continuous ADC detect |
| | ADC_ContinuousMode_DMA | Use DAM as continuous ADC detect |
| | ADC_ContinuousWatchdogMode | Use consecutive ADC as WDT detect |
| | ADC_SingleMode | Single ADC detect |
| | ADC_SingleModeDMA | Using DAM as a single ADC detect |
| | ADC_SingleWatchdogMode | Use single ADC as WDT detect |
| | ADC_StandbyMode | Using Low Power ADC Mode |
| CMP | CMP | Comparator Example |
| CR32 | CRC32 | CRC32 calculation example |
| DAC | DAC | DAC output example |
| | DAC_HighCurrent | DAC high thrust output example |
| FLASH | FLASH_PROGRAM | Example of programming area (EEPROM) |
| | FLASH_PROGRAM_INT | Program area (EEPROM) interrupt example |

| Root Folder | Folder Name | Description |
|---|---|---|
| FLASH_EXT | FLASH_OB_EEPROM | Example of programming data area (EEPROM) |
| | FLASH_OB_LEVEL | Example of programming data area (EEPROM) |
| | FLASH_OB_READ_PROTECTION | Anti-read encryption in the data area (OB) |
| | FLASH_OB_WRITE_PROTECTION | Write-proof encryption in the data area (OB) |
| GPIO | GPIO | GPIO Basic Example |
| | GPIO_Bit_Set_Reset | Example of setting GPIO bits |
| | GPIO_Input | Example of setting GPIO input |
| | GPIO_Interrupt | Example of setting GPIO interrupt |
| | GPIO_Output | Example of setting GPIO output |
| | GPIO_Toggle | Example of setting GPIO output inversion |
| I2C | I2C_Master_Slave_DMA_FLAG | I2C Slave mode and DMA move |
| | I2C_Master_Slave_DMA_INT | I2C Slave mode and DMA interrupt |
| | I2C_Master_Slave_FLAG | I2C Slave Mode |
| | I2C_Master_Slave_FLAG_EEPROM | I2C Slave mode and EEPROM programming |
| | I2C_Master_Slave_INT | Each group of I2C master and slave mode transfers to each other |
| I2S | I2S_DMA | I2S Slave mode and DMA transfer |
| | I2S_INT | I2S Slave mode and DMA interrupt |
| | I2S_POLLING | I2S Slave mode |
| IWDT | IWDT | Watchdog setting example |
| PWM | PWM | PWM pulse modulation example |
| RTC | RTC_1sec | RTC Timer setting 1 second example |
| | RTC_Alarm | RTC Timer setting Alarm example |
| SPI | MSPI_DMA_FLAG | SPI using DMA transfer example |
| | MSPI_DMA_INT | SPI using DMA transfer and interrupt example |
| | MSPI_FLAG | SPI transfer example |
| | MSPI_FLAG_FLASH_MX25L4006 | SPI with MX25L4006 transmission example |
| | MSPI_INT | SPI transfer and interrupt example |
| TIMER | TMR_Capture_Mode | Timer capture mode example |
| | TMR_Compare_Mode | Timer compare mode example |
| | TMR_Counter_Mode | Timer count mode example |
| | TMR_DMA_Mode | Example of using Timer with DMA |
| | TMR_PWM_MODE | Timer output PWM usage example |
| | TMR_Timer_Mode | Timer common timing example |
| UART | UART_DMA | Serial transfer with DMA usage |

| Root Folder | Folder Name | Description |
|---|---|---|
| | | example |
| | UART_HalfDuplexMode | Serial transfers use the half-duplex example |
| | UART_InterruptAndFlagManage | Serial transfer using interrupt example |
| | UART_IrDA_Mode | Serial transfer using IRDA example |
| | UART_TxRx | Serial Transmission Simultaneous Transmit and Receive Example |
| USB | USB_HID | HID KEYBOARD Simple Example |
| | USB_HID_AUDIO_WM8731 | HID with I2S to play WM8731 music |
| WWDT | WWDT | Windows Watchdog |

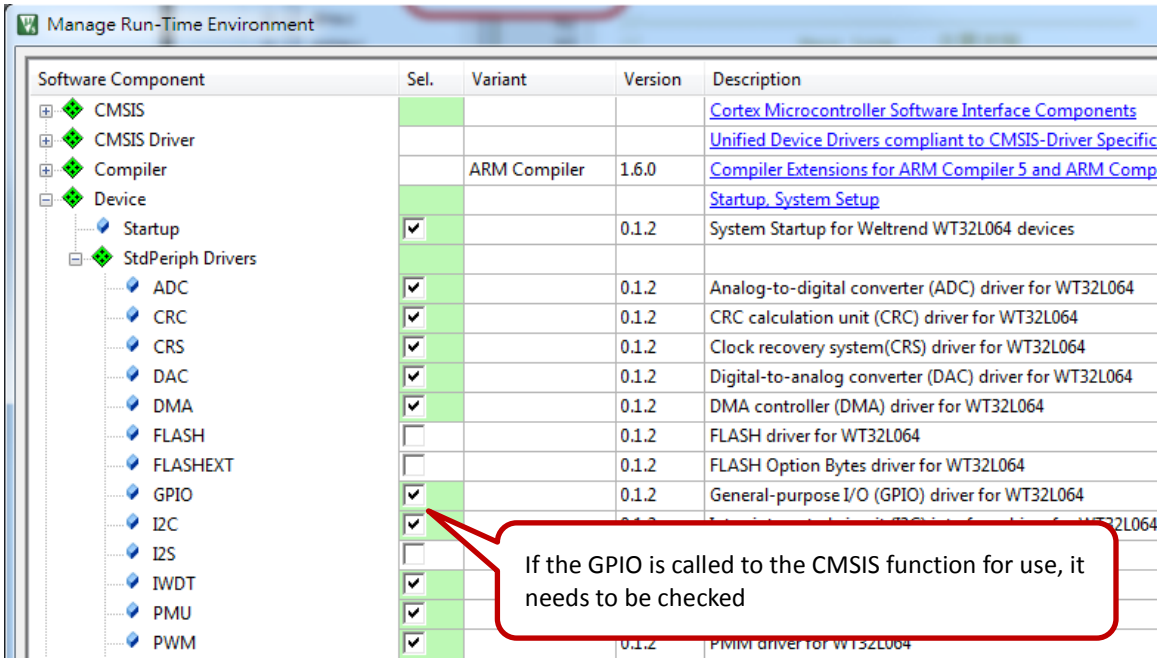
## 5. Examples of Codes operation

The following illustrates how to use reference example. Please refer to the example programs in the previous chapter, place the peripheral program library of the project into individual files according to the peripheral functions. After starting the project, the screen is as follows. It is divided into three parts: the project contains files, CMSIS drivers, and individual source file content.



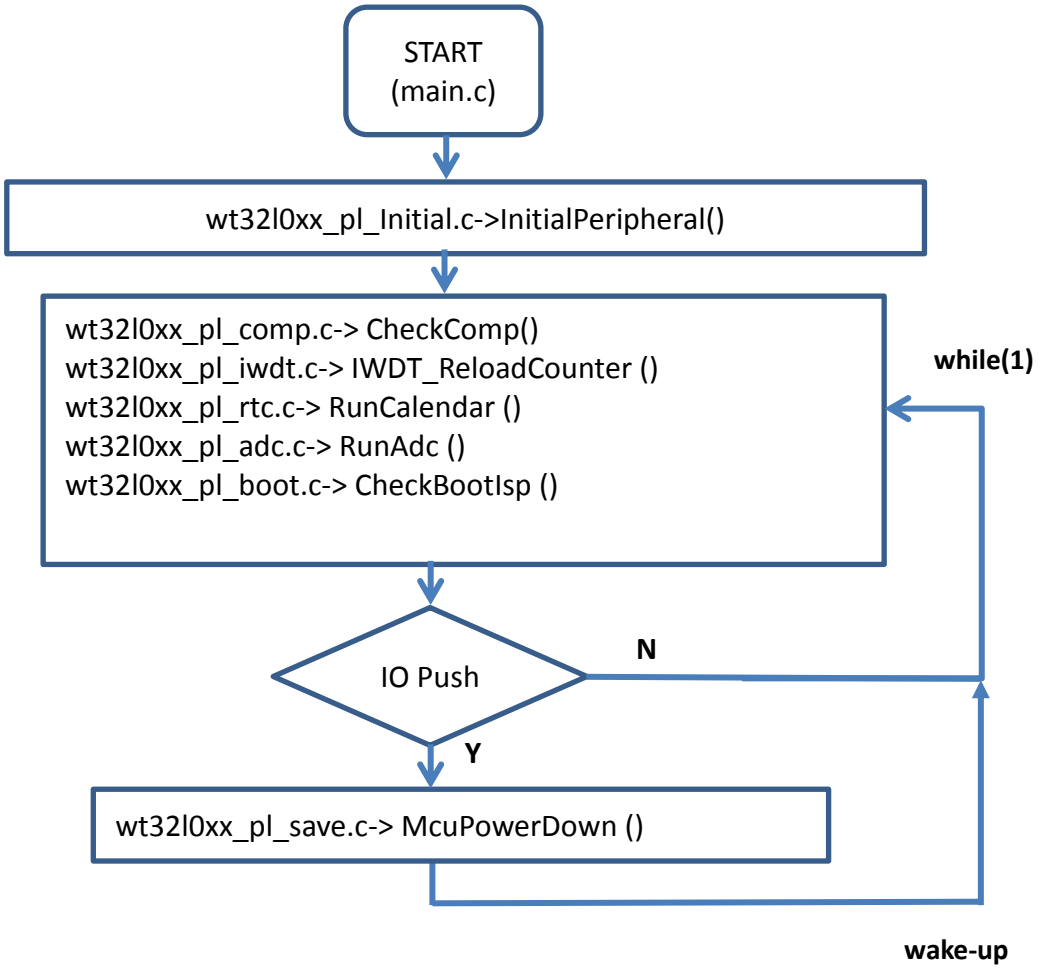
Add CMSIS driver layer functions for peripheral functions, click Manage Run-Time Environment on the top menu of ARM-MDK as shown below.


Click Device -> StdPeriph Drivers in sequence as shown in the figure below, you can add the required functions according to the application requirements, such as: ADC, DAC, FLASH, GPIO, I2C, etc. The general example program has been added to the reference CMSIS, if any missing parts or checked items can be re-selected.



### 5.1 Sample Flowchart

The following illustrates the flow chart of the sample program, and the main file contents and functions are as below.




According to the file name and function in the project, the description is as follows:

### 5.2 Main Program Flow

main.c: The function used is as follows:

1.) InitialPeripheral() ----------refer to wt32l0xx_pl_initial.c, Initialize the surrounding
2.) CheckComp () ---------- refer to wt32l0xx_pl_comp.c, Comparator output result
3.) IWDT_ReloadCounter() --- refer to wt32l0xx_pl_iwdt.c, reset Watchdog counter
4.) RunCalendar() ---------- refer to wt32l0xx_pl_rtc.c, Check calendar value
5.) RunAdc() ---------- refer to wt32l0xx_pl_adc.c, Perform ADC detection
6.) CheckBootIsp() ---------- refer to wt32l0xx_pl_boot.c, Check boot status
7.) McuPowerDown() ---------- refer to wt32l0xx_pl_save.c, Execute power saving function

The main loop content of the program is as follows:

```
int main(void)
{
   for (int i = 0; i < 200000; i++);        //Delay

   //-----------------------------------------------
   //                Initial Peripheral
   //-----------------------------------------------
   InitialPeripheral();

   //-----------------------------------------------
   //                 Main Loop
   //-----------------------------------------------
   while (1)    {
         //---------------- Compare -------------------
         #if(ENABLE_FUNC_CMP==ON)
         CheckComp();            // Check COMP comparator status
         #endif

         //------------------ IWDT -------------------
         #if(ENABLE_FUNC_IWDT==ON)
         IWDT_ReloadCounter();//Watchdog reload
         #endif

         //---------------- Calendar -------------------
         #if((ENABLE_FUNC_RTC==ON)&&(ENABLE_FUNC_CALENDAR==ON))
         RunCalendar();          // Check RTC calendar data
         #endif

         //------------------ ADC --------------------
         #if((ENABLE_FUNC_ADC==ON)&&(ENABLE_FUNC_SAVE==OFF))
         RunAdc();               // Perform ADC detection
         #endif

         //------------------ BOOT -------------------------
         #if(ENABLE_FUNC_BOOT==ON)
         CheckBootIsp();         // Check whether to enter Boot
         #endif

         //-------------- Power Save -------------------
         if (GPIO_ReadInputDataBit(GPIOA, GPIO_Pin_2) == 0) {        SysDelay(100);
               if (GPIO_ReadInputDataBit(GPIOA, GPIO_Pin_2) == 0) {        //debounce

               //-------- Sleep / Stop / Standby ----------
               #if(ENABLE_FUNC_SAVE==ON)
               McuPowerDown();             // enter power saving mode
               #endif
               }
         }
      };//while(1);
   }
```

- wt32l0xx_pl_library.h: Peripheral function switches, please turn on or off individual functions in sequence according to your needs. The program content is as follows.

```
//------------ Enable Function for Project --------------------------
// Please enable the following functions in sequence, use ON to enable, OFF to disable

//------------- Core -------------------------
#define       SELECT_CORE_1p2V                    OFF       // OFF:1.8V       // ON:
VCORE=1.2V
#define       ENABLE_FUNC_CLOCK                   ON        // Set IRC 16M~32KHz
#define       ENABLE_FUNC_LSI                     OFF  // Set whether to enable LSI 37KHz


//------------ IO LED -------------------------
#define       ENABLE_FUNC_GPIO                    ON // Set whether the GPIO function is enabled
#if(ENABLE_FUNC_GPIO==ON)
#define ENABLE_GPIO_INT                 OFF       // Set whether GPIO Interrupt is enabled
#define       ENABLE_LED_BLINK        ON        // Set whether the GPIO Port-C LED is enabled
#define       ENABLE_LED_RESET            OFF       // Set GPIO to test whether Reset is enabled
#endif
#define       ENABLE_FUNC_SYSTICK         ON        // Set whether the systick is activated

//------------ Digital ---------------
#define       ENABLE_FUNC_UART            ON        // Set whether the UART function is enabled
#if(ENABLE_FUNC_UART==ON)
#define ENABLE_FUNC_UART0               ON        // Set whether UART0 is enabled
#define ENABLE_FUNC_UART1               OFF       // Set whether UART1 is enabled
#define ENABLE_HW_IRDA              OFF  // Set whether IRDA is enabled or not. Use UART0+1
#endif

#define       ENABLE_FUNC_IWDT                OFF  // Set whether IWDT is enabled


//----------- Analog -----------------
#define       ENABLE_FUNC_CMP             ON        // Set whether COMPARE is enabled
#define       ENABLE_HW_CMP_SPEED_HI  OFF       //HI:4.5uA  LO:5.5uA

#define       ENABLE_FUNC_ADC             OFF       // Set whether the ADC is enabled
#if(ENABLE_FUNC_ADC==ON)
#define ENABLE_HW_ADC_AWD               OFF
#define ENABLE_HW_ADC_ALL               OFF
#endif


//-------- RTC -------
#define       ENABLE_FUNC_RTC                 OFF  // Set whether RTC is enabled
#if(ENABLE_FUNC_RTC==ON)
#define       ENABLE_FUNC_ALARM               OFF  //RTC Enable first (59 sec)
#define       ENABLE_FUNC_CALENDAR    OFF               //RTC Enable first (not for sleep)
#define       ENABLE_RESET_RTC                OFF  //ON: Test RTC keep RAM data
#endif
```

Power saving function switch

```
//------ Power Save ------
#define ENABLE_LPRUN_MODE              OFF      //GPIO cannot change without BLDO

#if(ENABLE_LPRUN_MODE==OFF)
#define     ENABLE_FUNC_SAVE           OFF
#if(ENABLE_FUNC_SAVE==ON)
#define ENABLE_STANDBY_MODE            OFF
#define ENABLE_SLEEP_MODE              OFF      //ENABLE_FUNC_SYSTICK must OFF
#define ENABLE_STOP_MODE               ON
#endif
```

Wake-up function

```
//----- wake up -----
#if(ENABLE_FUNC_SAVE==ON)
#define     ENABLE_WAKE_GPIO           ON       //STADBY must OFF
#define     ENABLE_WAKEUP_CMP          OFF
#define     ENABLE_WAKEUP_ADC          OFF
#define     ENABLE_WAKEUP_DAC          OFF      //Only Output
#define     ENABLE_WAKEUP_RTC          OFF
#define     ENABLE_WAKEUP_IWDT         OFF
#endif
#endif
```

### 5.3 Initialization of peripheral functions

Use wt32l0xx_pl_initial.c Initialize peripheral functions: Systemtick, GPIO, UART, WDT, ADC, RTC, and Comparator

**initialization order: InitialSysClock() -> InitialGpio() -> InitiSysTick() -> InitialUart0() -> InitialIwdt() -> InitialAdc() -> InitialRtc()->InitialComp()**

#### 5.3.1 Working frequency selection

wt32l0xx_pl_clock.h For the selection of operating frequency, four types of HIS, MSI, HSE and PLL can be selected. The program is as follows

```
//--------- Use PLL for HSI 32MHz -----------------------------
#define CLOCK_PLL_HSI_X2_32MHZ    ON    //ON: Peripheral IO type settings, including functions are as follows, please refer to Chapter 4

//--------- Use PLL for USB 48MHz -----------------------------
#define USB_PLL        0          // 0:HSI48M, 1:PLL(From external crystal)

//--------- Select Frequency for MSI --------------------------
#define MSI_65K                   PMU_MSIClock_Range0
#define MSI_131K                  PMU_MSIClock_Range1
#define MSI_262K                  PMU_MSIClock_Range2
#define MSI_524K                  PMU_MSIClock_Range3
#define MSI_1M                    PMU_MSIClock_Range4
#define MSI_2M                    PMU_MSIClock_Range5
```

```
#define MSI_4M              PMU_MSIClock_Range6          //4.2MHz

#define MSI_CLOCK           MSI_4M          // MSI is selected, System operating frequency

//--------- Select MCU Clock Type ------------------------------
#define     CLK_HSI         0       //Internal OSC 16MHz
#define     CLK_MSI         1       //Internal OSC 65K~4M
#define     CLK_PLL         2       //Use Multiple X with HSI or HSE
#define     CLK_HSE         3 //External OSC 1~25MHz

#define     SYS_CLOCK_SEL   CLK_MSI         // The type of frequency selected by the system
```

Choose speed

Choose a type

- wt32l0xx_pl_clock.c Working frequency setting function, including the following functions

  1.) InitialSysClock () ---------- Perform system frequency selection, excerpted as follows.

```
#if(SYS_CLOCK_SEL==CLK_HSI)          // Use HSI as system frequency
      PMU_PowerClockCmd(PMU_PowerClock_HSI, ENABLE);
      PMU_SYSCLKConfig(PMU_SystemClk_HSI16);

#elif(SYS_CLOCK_SEL==CLK_MSI)        // Use MSI as system frequency
      PMU_MSIConfig(MSI_CLOCK);                                  //Speed Setting
      PMU_PowerClockCmd(PMU_PowerClock_MSI, ENABLE);    //Power-On PLL
      PMU_SYSCLKConfig(PMU_SystemClk_MSI);       //Select System clock

#elif(SYS_CLOCK_SEL==CLK_PLL)        // Use PLL for system frequency
#elif(SYS_CLOCK_SEL==CLK_HSE)        // Use HSE as system frequency

      PMU_PowerClockCmd(PMU_PowerClock_HSE, ENABLE);
      PMU_SYSCLKConfig(PMU_SystemClk_HSE);
```

  2.) Delayms() ---------- Execute delay function

  3.) DelayCount() ---------- Execute delay function

### 5.3.2 Description of peripheral functions

- wt32l0xx_pl_gpio.c Peripheral IO type settings, including functions are as follows, please refer to Chapter 4

  1.) GPIO_Handler ()----------Interrupt service GPIO function

  2.) InitialGpio ()----------Initialize GPIO function

```
4 types of GPIO: GPIO_Mode_IN => basic input
                 GPIO_Mode_OUT => basic output
                 GPIO_Mode_AF => Composite use function，EX:UART、SPI、I2C …
                 GPIO_Mode_AN => Analog input function，EX:ADC、USB、COMP …
```

- wt32l0xx_pl_systick.c Built-in 24-bit timer settings, including the following functions.

1.) SysTick_Handler ()----------Interrupt service systick function
2.) InitiSysTick ()----------Initialize the systick function
3.) SysDelay ()----------Initialize the systick function using the systick delay function

- wt32l0xx_pl_uart.c Asynchronous transceiver transmission settings, including functions are as follows, please refer to Chapter 5

1.) UART0_Handler ()----------Interrupt Service UART0 Function
2.) UART1_Handler()---------- Interrupt Service UART1 Function
3.) InitialUart0 ()----------Initialize UART0 function
4.) InitialUart1()----------Initialize UART1 function
5.) fputc ()----------Use the emit serial data function with printf()
6.) fgetc()----------Use the receive serial data function with printf()
7.) DRV_IntToStr()----------Number to string
8.) Str2Num()----------String to Number
9.) uart_send_str()---------- Use UART0/1 to transmit serial data
10.) uart_clear_str()----------Clear list contents

- wt32l0xx_pl_adc.c analog detection settings, including the following functions

1.) ADC_Handler ()----------Interrupt Service ADC Function
2.) InitialAdc ()----------Initialize ADC function
3.) InitialAllAdc ()----------Initialize all channel functions of ADC
4.) RunAdc()----------Perform ADC target channel conversion function
5.) RunAllAdc ()----------Perform ADC conversion functions for all channels
6.) RunAdcConvert()---------- Execute ADC channel single conversion function
7.) API_AverADCData ()-------- ADC channel conversion function, calculate average
8.) ADC_StartOfConversion_1() - Start ADC module conversion
9.) ADC_StopOfConversion_1() - Stop ADC module conversion
10.) HEX2BCD()----------Hexadecimal to decimal

- wt32l0xx_pl_save.c----- Power saving function settings, including the following functions:

1.) McuPowerDown ()----------Execute the pre-operation of the power saving

function and call Save()

2.) Save()----------Execute power saving function according to the setting of SLEEP, STOP, STANDBY

- wt32l0xx_pl_rtc.c-- Real-time counter settings, including functions as follows:

1.) InitialRtc () ---------- Perform RTC initialization

2.) RTC_AlarmCmd ()----------Execute DAC interrupt function

3.) RTC_Handler()----------Execute the RTC interrupt function

4.) RunCalendar()----------Execute the RTC calendar function

5.) SetAlarm()----------Set the RTC alarm function

- wt32l0xx_pl_comp.c comparator settings, including the following functions

1.) CheckComp () ---------- Check CPM0 function

2.) CMP0_VOUT_Handler ()----------Execute the CPM0 interrupt function

3.) CMP1_VOUT_Handler()----------Execute the CMP1 interrupt function

4.) InitialComp()----------Initialize the COMP comparator

5.) RumComp()----------Execute the COMP comparator

- wt32l0xx_pl_iwdt.c-- Watchdog settings, including functions as follows

1.) InitialIwdt () ---------- Initialize the watchdog

- wt32l0xx_pl_boot.c ---The general serial bus enters the Boot setting, including the following functions. For more details, please refer to "WT32L064_032 Application File_Using USB to Update ISP Instructions_v1x.pdf"

1.) CheckBootIsp()----------Check if you need to enter the Boot Updater

2.) Enter_bootisp()----------Set the Boot function to be executed at startup and execute the MCU reset

## 6. Revision History

| Version | History | Date |
|---|---|---|
| 1.0 | Initial issue | 2022/04/22 |
| | | |