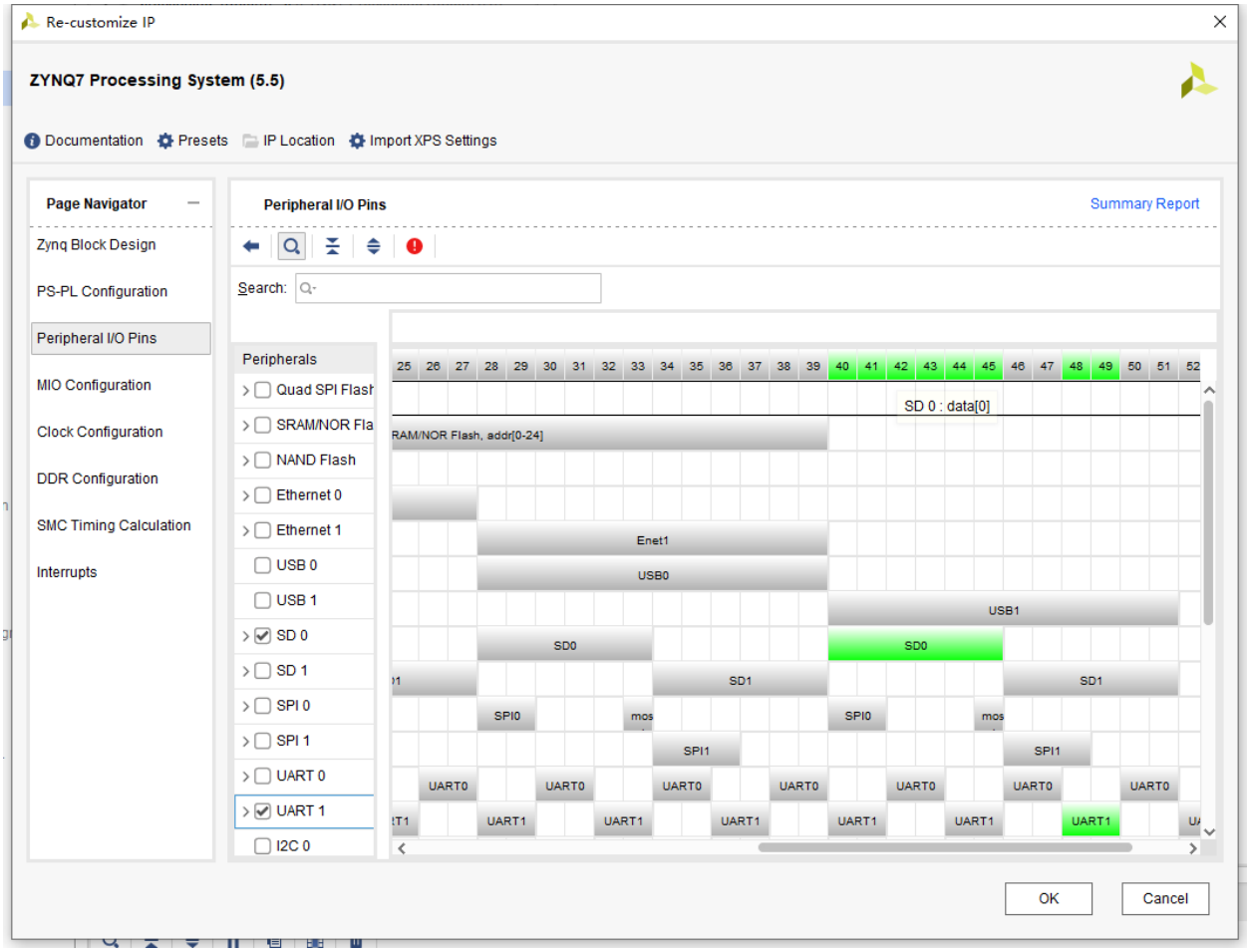
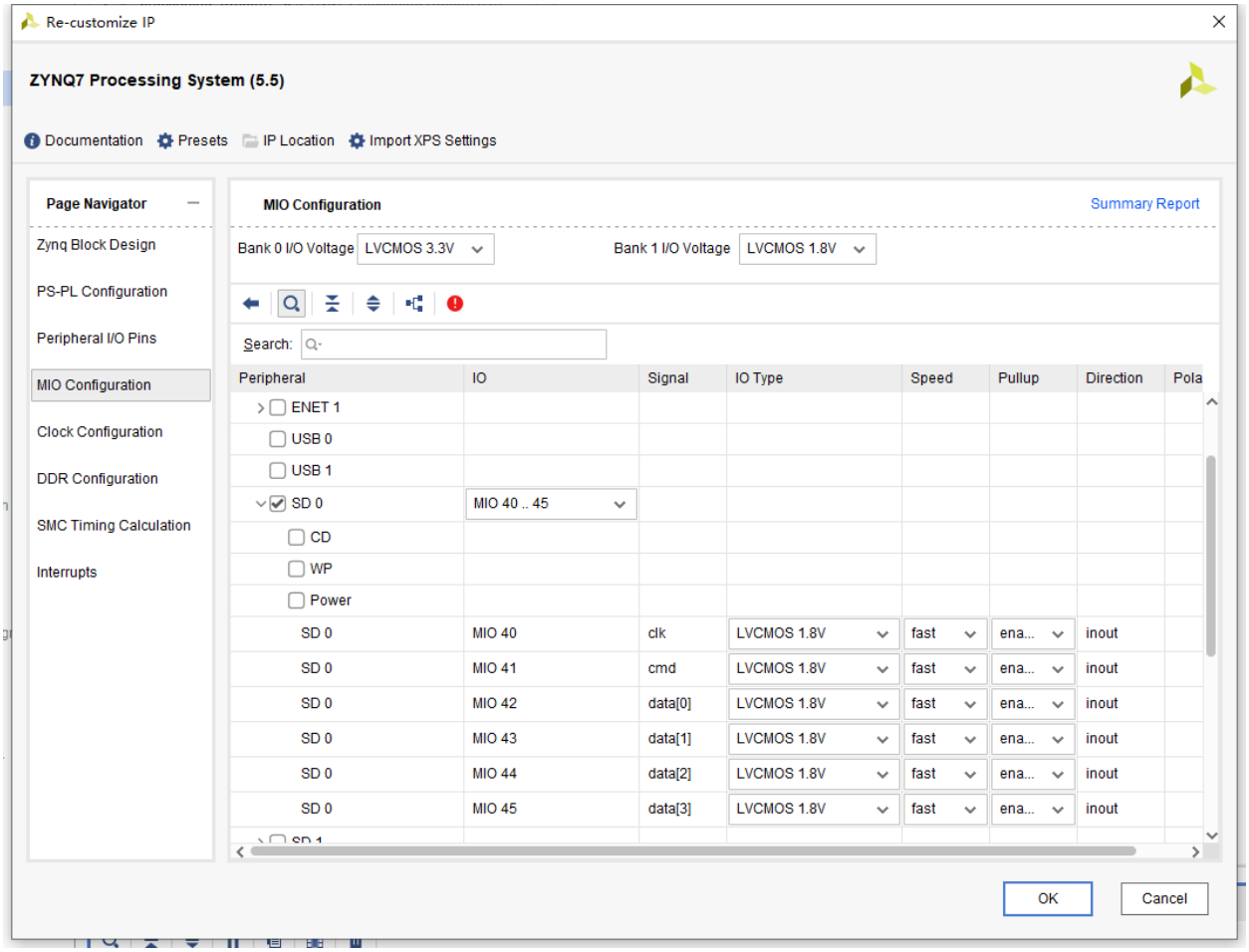
[L09]ARM 库调用(文件系统和RTOS)

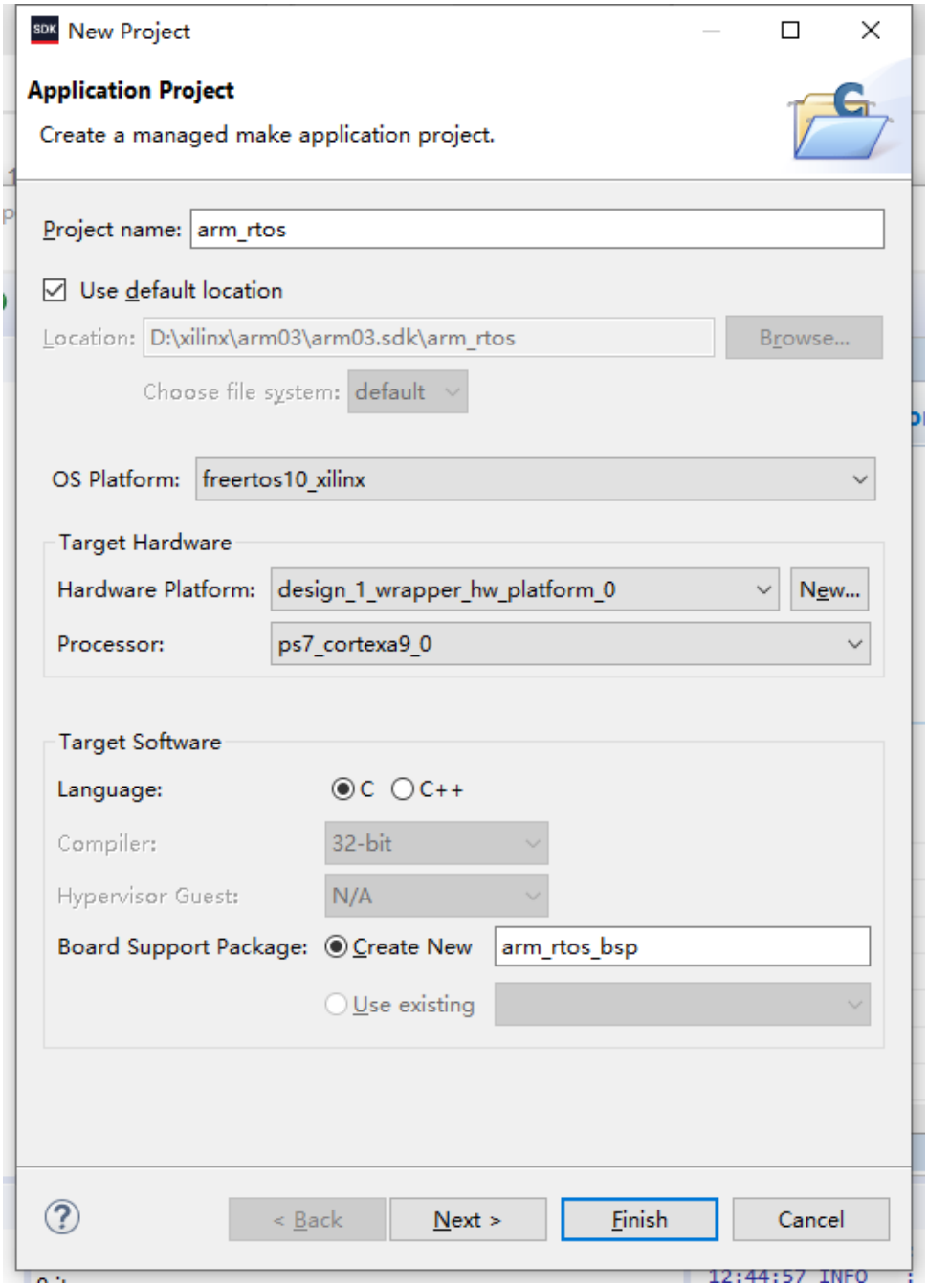
在ZYNQ Processing System中配置SD外设.



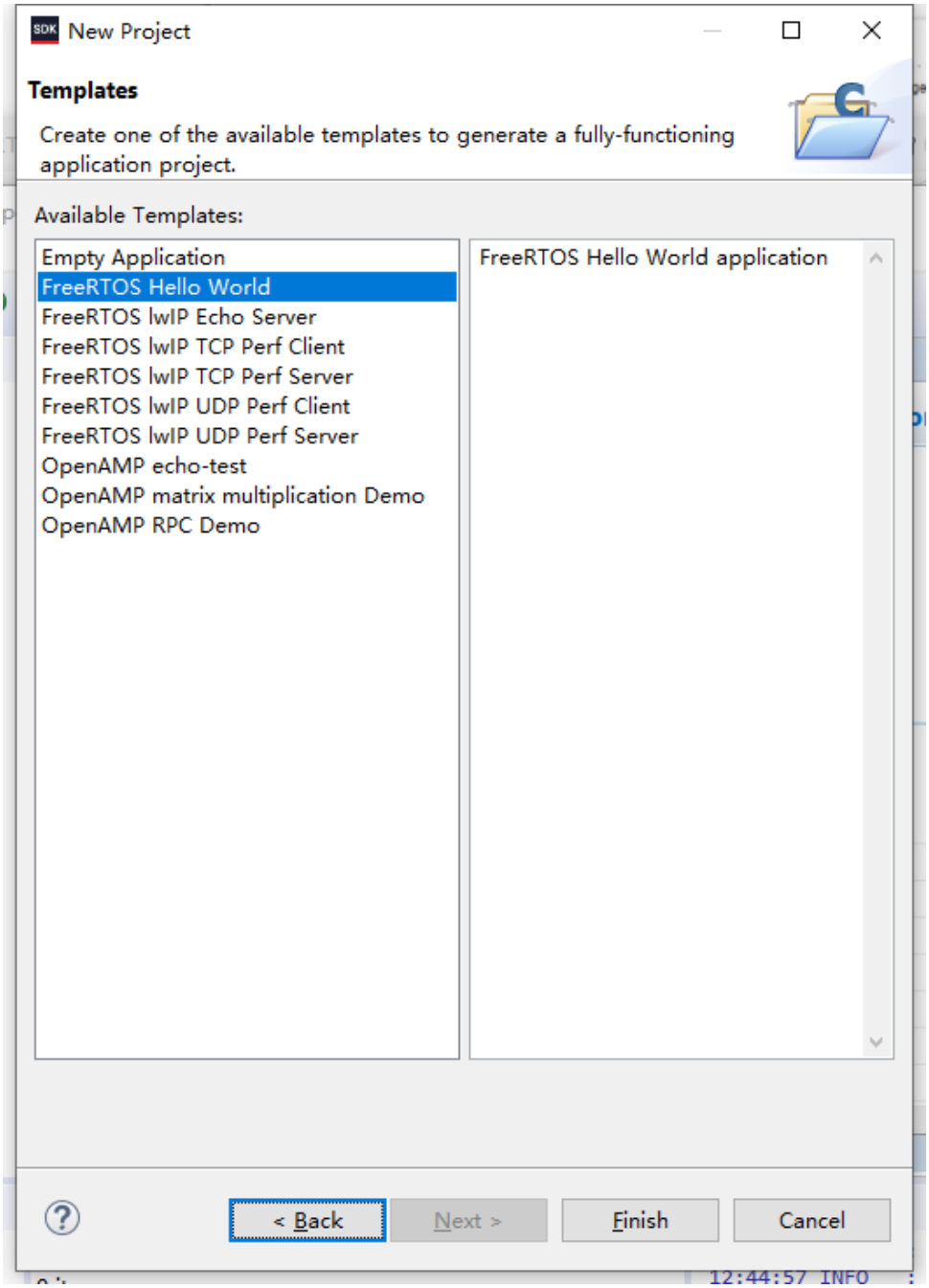
两个BANK电压不同,并且做好IO配置.



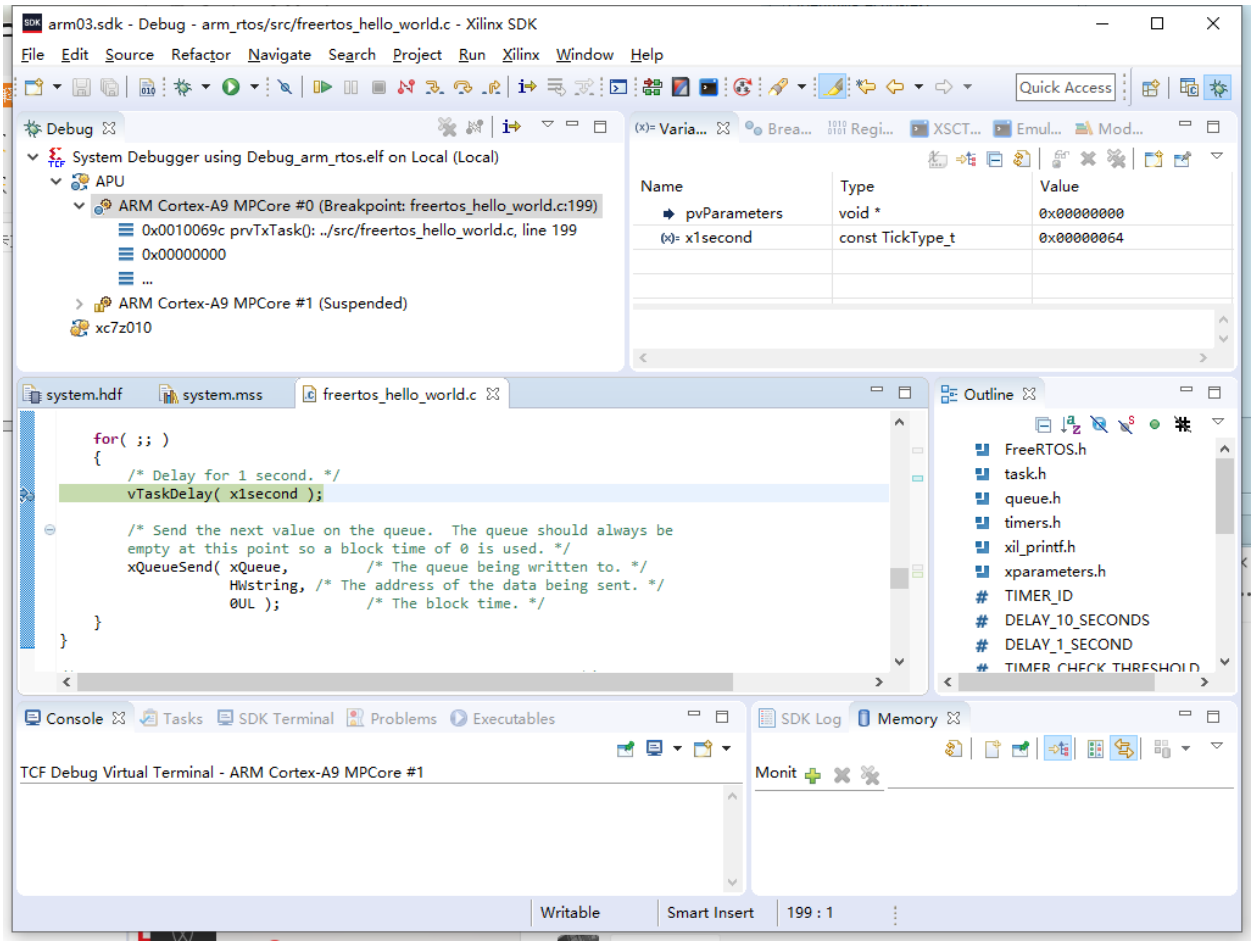
然后记得配置好内存,没有其他IP不需要拉PL相关配置,然后配置好各种,由于无PL,所以无需生成bitstream,最后导出SDK,创建一个RTOS工程.(注:生成导出bitstream也无所谓的~)



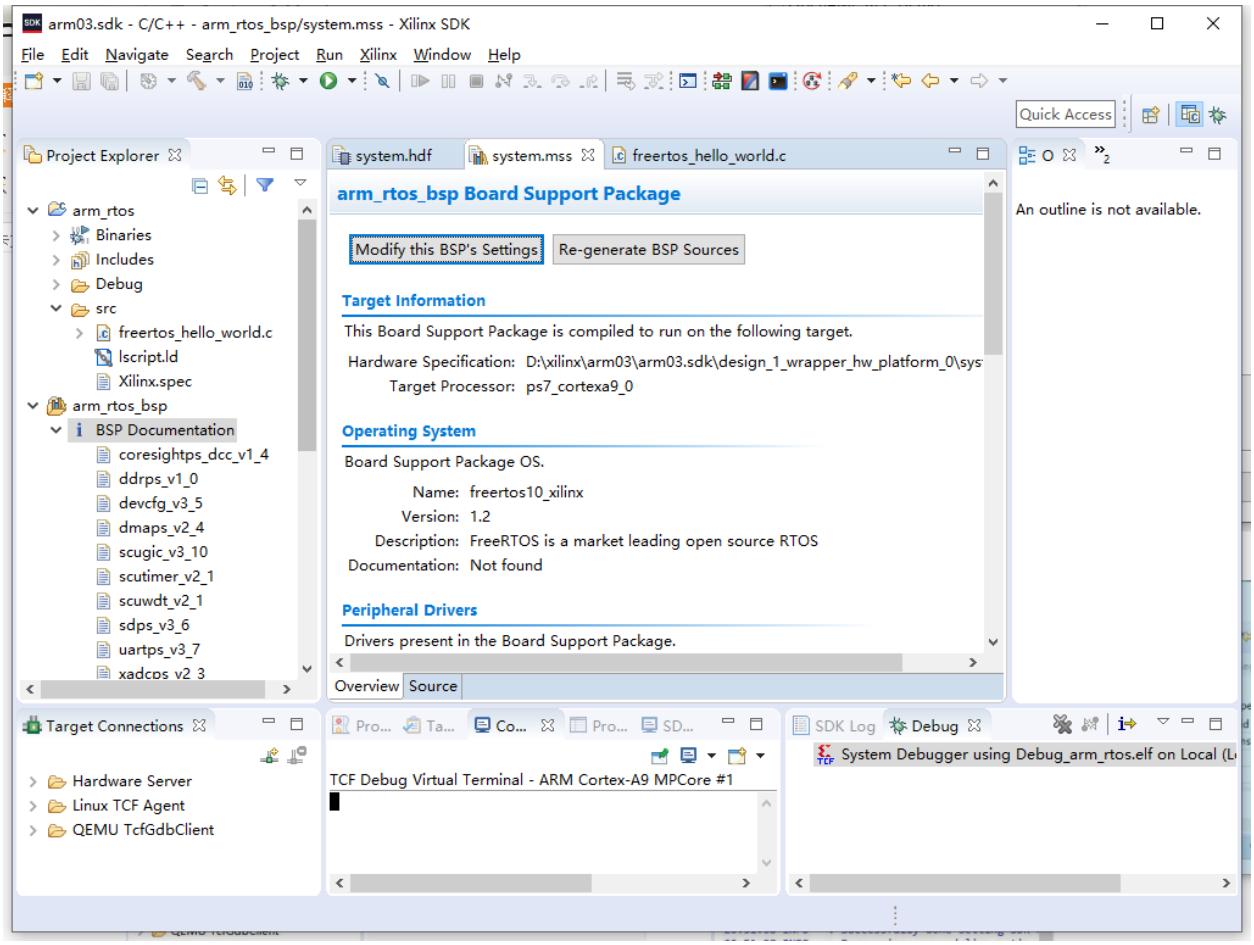
创建Hello World.


New Project
Templates
Create one of the available templates to generate a fully-functioning application project.
Available Templates:
Empty Application
FreeRTOS Hello World
FreeRTOS lwIP Echo Server
FreeRTOS lwIP TCP Perf Client
FreeRTOS lwIP TCP Perf Server
FreeRTOS lwIP UDP Perf Client
FreeRTOS lwIP UDP Perf Server
OpenAMP echo-test
OpenAMP matrix multiplication Demo
OpenAMP RPC Demo
FreeRTOS Hello World application
< Back
Next >
Finish
Cancel

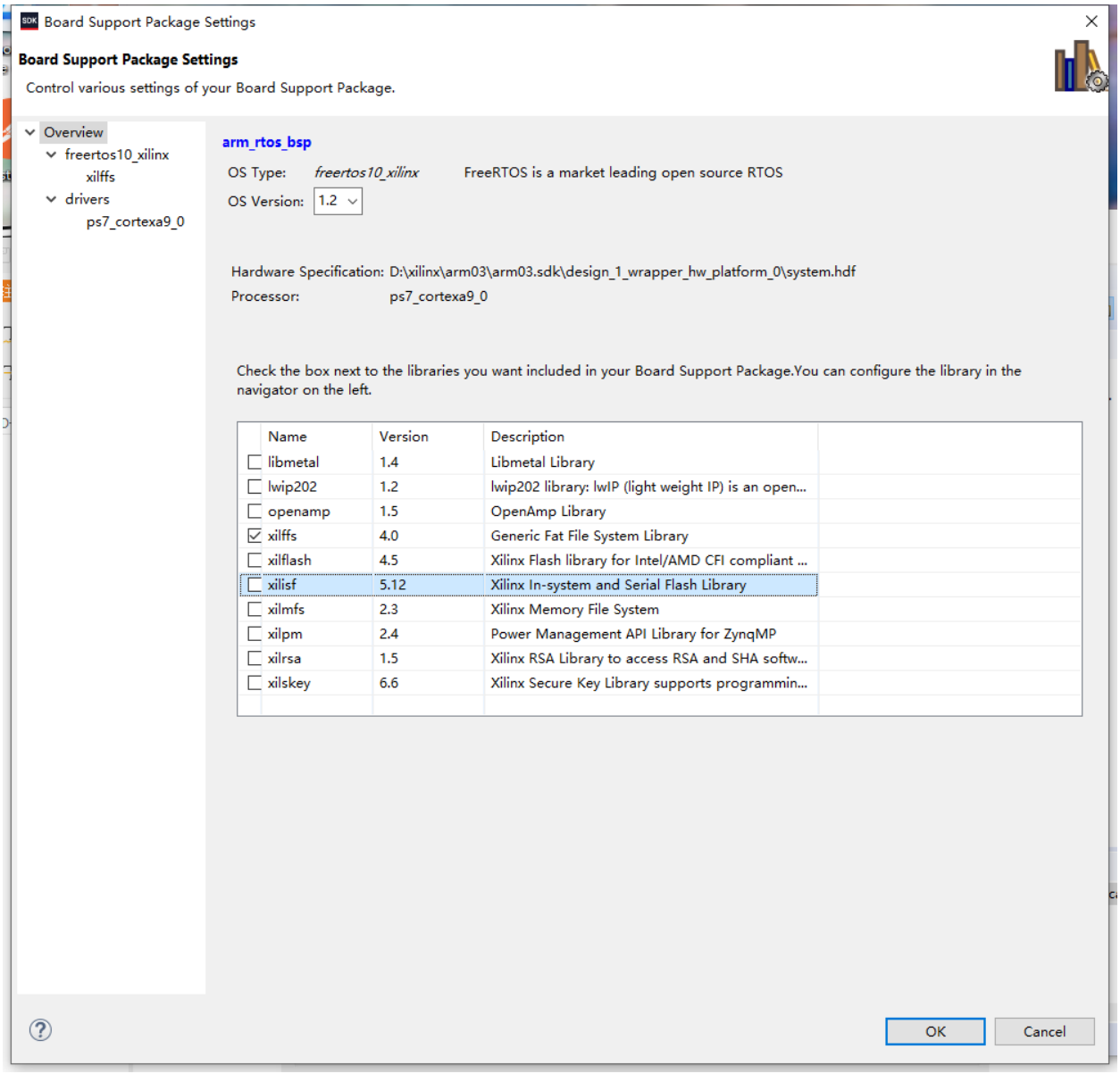
简单运行一下.



打开BSP配置,选择Modify this BSP’s Settings.



勾选xilffs,开启FAT32支持.



最后Re-generate BSP Sources,然后修改代码.

```
/*
    Copyright (C) 2017 Amazon.com, Inc. or its affiliates.  All Rights Reserved.
    Copyright (C) 2012 - 2018 Xilinx, Inc. All Rights Reserved.

    Permission is hereby granted, free of charge, to any person obtaining a copy of
    this software and associated documentation files (the "Software"), to deal in
    the Software without restriction, including without limitation the rights to
    use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of
    the Software, and to permit persons to whom the Software is furnished to do so,
    subject to the following conditions:

    The above copyright notice and this permission notice shall be included in all
    copies or substantial portions of the Software. If you wish to use our Amazon
```

```
    FreeRTOS name, please do so in a fair use way that does not cause confusion.

    THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
    IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS
    FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
    COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER
    IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
    CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

    http://www.FreeRTOS.org
    http://aws.amazon.com/freertos

    ***************************************************************************
     *                                                                         *
     *    Having a problem?  Start by reading the FAQ "My application does   *
     *    not run, what could be wrong?".  Have you defined configASSERT()?  *
     *                                                                         *
     *    http://www.FreeRTOS.org/FAQHelp.html                                 *
     *                                                                         *
    ***************************************************************************

    ***************************************************************************
     *                                                                         *
     *    FreeRTOS provides completely free yet professionally developed,    *
     *    robust, strictly quality controlled, supported, and cross          *
     *    platform software that is more than just the market leader, it     *
     *    is the industry's de facto standard.                               *
     *                                                                         *
     *    Help yourself get started quickly while simultaneously helping     *
     *    to support the FreeRTOS project by purchasing a FreeRTOS           *
     *    tutorial book, reference manual, or both:                          *
     *    http://www.FreeRTOS.org/Documentation                              *
     *                                                                         *
    ***************************************************************************

    ***************************************************************************
     *                                                                         *
     *   Investing in training allows your team to be as productive as       *
     *   possible as early as possible, lowering your overall development    *
     *   cost, and enabling you to bring a more robust product to market     *
     *   earlier than would otherwise be possible.  Richard Barry is both    *
     *   the architect and key author of FreeRTOS, and so also the world's   *
     *   leading authority on what is the world's most popular real time     *
     *   kernel for deeply embedded MCU designs.  Obtaining your training    *
     *   from Richard ensures your team will gain directly from his in-depth *
     *   product knowledge and years of usage experience.  Contact Real Time *
     *   Engineers Ltd to enquire about the FreeRTOS Masterclass, presented  *
     *   by Richard Barry:  http://www.FreeRTOS.org/contact
     *                                                                         *
    ***************************************************************************

    ***************************************************************************
     *                                                                         *
     *    You are receiving this top quality software for free.  Please play *
     *    fair and reciprocate by reporting any suspected issues and         *
     *    participating in the community forum:                              *
     *    http://www.FreeRTOS.org/support                                    *
     *                                                                         *
     *    Thank you!                                                         *
```

```
     *                                                                       *
    ***************************************************************************

    http://www.FreeRTOS.org - Documentation, books, training, latest versions,
    license and Real Time Engineers Ltd. contact details.

    http://www.FreeRTOS.org/plus - A selection of FreeRTOS ecosystem products,
    including FreeRTOS+Trace - an indispensable productivity tool, a DOS
    compatible FAT file system, and our tiny thread aware UDP/IP stack.

    http://www.FreeRTOS.org/labs - Where new FreeRTOS products go to incubate.
    Come and try FreeRTOS+TCP, our new open source TCP/IP stack for FreeRTOS.

    http://www.OpenRTOS.com - Real Time Engineers ltd license FreeRTOS to High
    Integrity Systems ltd. to sell under the OpenRTOS brand.  Low cost OpenRTOS
    licenses offer ticketed support, indemnification and commercial middleware.

    http://www.SafeRTOS.com - High Integrity Systems also provide a safety
    engineered and independently SIL3 certified version for use in safety and
    mission critical applications that require provable dependability.

    1 tab == 4 spaces!
*/

/* FreeRTOS includes. */
#include "FreeRTOS.h"
#include "task.h"
#include "queue.h"
#include "timers.h"
/* Xilinx includes. */
#include "xil_printf.h"
#include "xparameters.h"

#include "ff.h"
#include <stdio.h>

#define TIMER_ID  1
#define DELAY_10_SECONDS  10000UL
#define DELAY_1_SECOND    1000UL
#define TIMER_CHECK_THRESHOLD 9
/*-----------------------------------------------------------*/

/* The Tx and Rx tasks as described at the top of this file. */
static void prvSDTask( void *pvParameters );
static void prvTxTask( void *pvParameters );
static void prvRxTask( void *pvParameters );
static void vTimerCallback( TimerHandle_t pxTimer );
/*-----------------------------------------------------------*/

/* The queue used by the Tx and Rx tasks, as described at the top of this
file. */
static TaskHandle_t xTxTask;
static TaskHandle_t xRxTask;
static QueueHandle_t xQueue = NULL;
static TimerHandle_t xTimer = NULL;
char HWstring[15] = "Hello World";
long RxtaskCntr = 0;

int main( void )
```

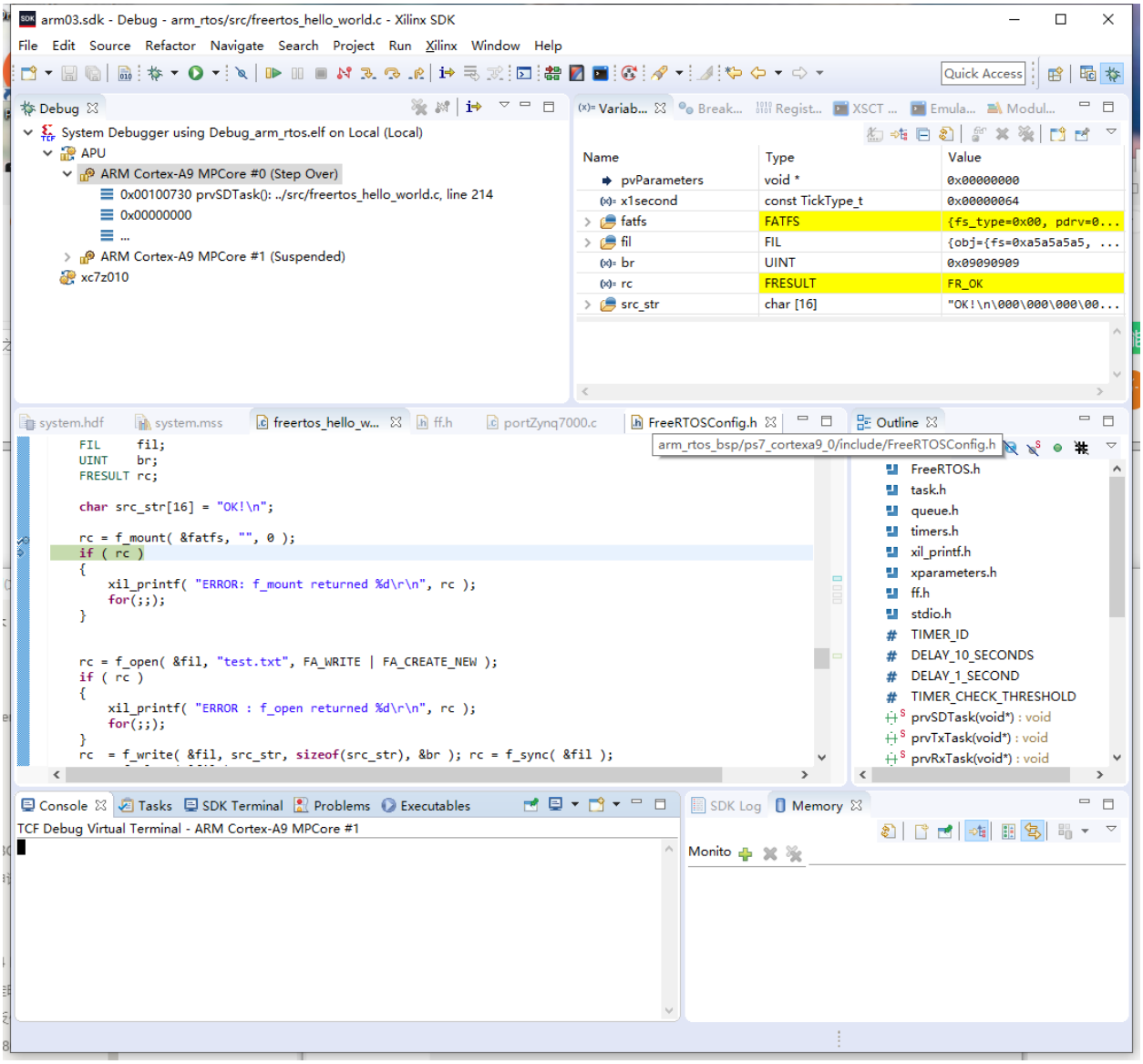
```
{
  const TickType_t x10seconds = pdMS_TO_TICKS( DELAY_10_SECONDS );

  xil_printf( "Hello from Freertos example main\r\n" );

  /* Create the two tasks.  The Tx task is given a lower priority than the
  Rx task, so the Rx task will leave the Blocked state and pre-empt the Tx
  task as soon as the Tx task places an item in the queue. */
  xTaskCreate(  prvTxTask,          /* The function that implements the task. */
          ( const char * ) "Tx",    /* Text name for the task, provided to assist debugging only. */
          configMINIMAL_STACK_SIZE,   /* The stack allocated to the task. */
          NULL,             /* The task parameter is not used, so set to NULL. */
          tskIDLE_PRIORITY,     /* The task runs at the idle priority. */
          &xTxTask );

  xTaskCreate( prvRxTask,
         ( const char * ) "GB",
         configMINIMAL_STACK_SIZE,
         NULL,
         tskIDLE_PRIORITY + 1,
         &xRxTask );

  xTaskCreate( prvSDTask,
         ( const char * ) "SD",
         configMINIMAL_STACK_SIZE,
         NULL,
         tskIDLE_PRIORITY,
         NULL );

  /* Create the queue used by the tasks.  The Rx task has a higher priority
  than the Tx task, so will preempt the Tx task and remove values from the
  queue as soon as the Tx task writes to the queue - therefore the queue can
  never have more than one item in it. */
  xQueue = xQueueCreate(  1,            /* There is only one space in the queue. */
              sizeof( HWstring ) ); /* Each space in the queue is large enough to hold a uint32_t. */

  /* Check the queue was created. */
  configASSERT( xQueue );

  /* Create a timer with a timer expiry of 10 seconds. The timer would expire
   after 10 seconds and the timer call back would get called. In the timer call back
   checks are done to ensure that the tasks have been running properly till then.
   The tasks are deleted in the timer call back and a message is printed to convey that
   the example has run successfully.
   The timer expiry is set to 10 seconds and the timer set to not auto reload. */
  xTimer = xTimerCreate( (const char *) "Timer",
              x10seconds,
              pdFALSE,
              (void *) TIMER_ID,
              vTimerCallback);
  /* Check the timer was created. */
  configASSERT( xTimer );

  /* start the timer with a block time of 0 ticks. This means as soon
     as the schedule starts the timer will start running and will expire after
     10 seconds */
  xTimerStart( xTimer, 0 );

  /* Start the tasks and timer running. */
```

```
  vTaskStartScheduler();

  /* If all is well, the scheduler will now be running, and the following line
  will never be reached.  If the following line does execute, then there was
  insufficient FreeRTOS heap memory available for the idle and/or timer tasks
  to be created.  See the memory management section on the FreeRTOS web site
  for more details. */
  for( ;; );
}

/*-----------------------------------------------------------*/
static void prvSDTask( void *pvParameters )
{
  const TickType_t x1second = pdMS_TO_TICKS( DELAY_1_SECOND );

  FATFS   fatfs;
  FIL   fil;
  UINT  br;
  FRESULT rc;

  char src_str[16] = "OK!\n";

  rc = f_mount( &fatfs, "", 0 );
  if ( rc )
  {
    xil_printf( "ERROR: f_mount returned %d\r\n", rc );
    for(;;);
  }


  rc = f_open( &fil, "test.txt", FA_WRITE | FA_CREATE_NEW );
  if ( rc )
  {
    xil_printf( "ERROR : f_open returned %d\r\n", rc );
    for(;;);
  }
  rc  = f_write( &fil, src_str, sizeof(src_str), &br ); rc = f_sync( &fil );
  rc  = f_close( &fil );

  rc = f_open( &fil, "test.txt", FA_READ );
  if ( rc )
  {
    xil_printf( "ERROR : f_open returned %d\r\n", rc );
    for(;;);
  }
  rc  = f_lseek( &fil, 0 );
  rc  = f_read( &fil, src_str, 16, &br );
  xil_printf( src_str );
  rc = f_close( &fil );

  for( ;; )
  {
    /* Delay for 1 second. */
    vTaskDelay( x1second );
  }
}

/*-----------------------------------------------------------*/
static void prvTxTask( void *pvParameters )
```

```
{
const TickType_t x1second = pdMS_TO_TICKS( DELAY_1_SECOND );

  for( ;; )
  {
    /* Delay for 1 second. */
    vTaskDelay( x1second );

    /* Send the next value on the queue.  The queue should always be
    empty at this point so a block time of 0 is used. */
    xQueueSend( xQueue,     /* The queue being written to. */
          HWstring, /* The address of the data being sent. */
          0UL );      /* The block time. */
  }
}

/*-----------------------------------------------------------*/
static void prvRxTask( void *pvParameters )
{
char Recdstring[15] = "";

  for( ;; )
  {
    /* Block to wait for data arriving on the queue. */
    xQueueReceive(  xQueue,       /* The queue being read. */
            Recdstring, /* Data is read into this address. */
            portMAX_DELAY );  /* Wait without a timeout for data. */

    /* Print the received data. */
    xil_printf( "Rx task received string from Tx task: %s\r\n", Recdstring );
    RxtaskCntr++;
  }
}

/*-----------------------------------------------------------*/
static void vTimerCallback( TimerHandle_t pxTimer )
{
  long lTimerId;
  configASSERT( pxTimer );

  lTimerId = ( long ) pvTimerGetTimerID( pxTimer );

  if (lTimerId != TIMER_ID) {
    xil_printf("FreeRTOS Hello World Example FAILED");
  }

  /* If the RxtaskCntr is updated every time the Rx task is called. The
   Rx task is called every time the Tx task sends a message. The Tx task
   sends a message every 1 second.
   The timer expires after 10 seconds. We expect the RxtaskCntr to at least
   have a value of 9 (TIMER_CHECK_THRESHOLD) when the timer expires. */
  if (RxtaskCntr >= TIMER_CHECK_THRESHOLD) {
    xil_printf("FreeRTOS Hello World Example PASSED");
  } else {
    xil_printf("FreeRTOS Hello World Example FAILED");
  }

  vTaskDelete( xRxTask );
```

```
  vTaskDelete( xTxTask );
}
```

测试结果:



注意:可能会因为Stack不足,需要自信调整处理,简单的单片机开发应该不难.