

# MD14\_PowerDown\_Retention\_ram

## 版本：

Config Tool Version: 2.7.6

MD1x SDK version: 1.3.1

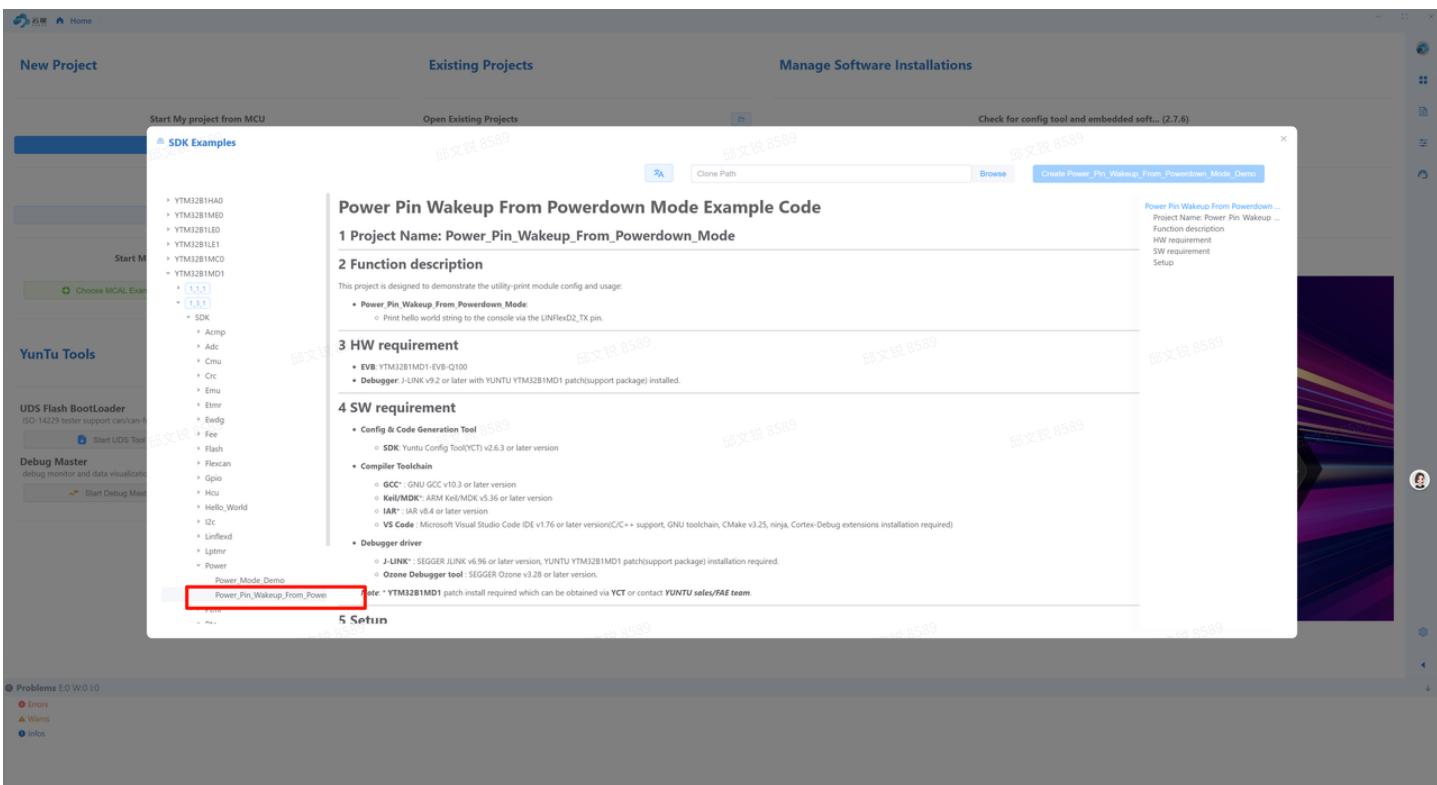
## 前言：

云途MD，ME系列有PowerDown下唤醒RAM保持的功能（retention\_ram）。MD和ME的retention\_ram都在整片RAM的中间。所以如果把retention\_ram这段分出来，就会把主RAM分割为两段，主RAM和RAM1，就需要把分割出来的RAM1像主RAM一样利用起来。

本文将简述云途YTM32B1MDx系列PowerDown下唤醒RAM保持，并且分割出来的RAM1像主RAM一样利用起来的实现过程及使用方法。ME系列可参考本文。

## 1. retention\_ram

### 1. 工程基于MD14的SDK Power\_Pin\_Wakeup\_From\_Powerdown\_Mode\_Demo修改



### 2. YT\_Link配置一段retention\_ram,将剩余的RAM空间利用起来。MD14的retention\_ram区域如下：

WPS AI

开始 插入 编辑 页面 批注 工具 保护 转换 全文翻译

书签

- 14.5.3 Watchdog Reset
- 14.6 Low Power Mode Entry Acknowledge
- 14.7 Software Reset
- 14.8 Core Lockup Reset
- 14.9 Debug Reset
- 15 Power Control Unit (PCU)
- 15.1 Introduction
- 15.1.1 Block Diagram
- 15.1.2 Features
- 15.2 Register Definition
- 15.2.1 PCU Memory Map
- 15.2.1.1 PCU SSTS Register
- 15.2.1.2 PCU STS Register
- 15.2.1.3 PCU INT Register
- 15.2.1.4 PCU CTRL Register
- 15.3 Functional Description
- 15.3.1 Reset
- 15.3.2 Interrupt
- 16 Power Mode
- 16.1 Introduction
- 16.1.1 Active Mode
- 16.1.2 Sleep Mode
- 16.1.3 Deepsleep Mode
- 16.1.4 Standby Mode
- 16.1.5 Powerdown Mode
- 16.2 Power Modes Table
- 16.3 Wake-up Sources
- 17 Wake-up Unit (WKU)
- VI Communication Interfaces
- VI Timer
- V Analog
- IX: Security, Integrity and Safety
- X Appendix

Memory and memory interface

	Flash	FF	FF	Optional FF (DPDEN) <sup>3</sup>	Optional FF (DPDEN)	OFF
SRAM	FF	FF	FF	FF	FF	2x16KB blocks retained <sup>4</sup>
REGFILE	FF	FF	FF	FF	FF	
Communication interfaces						
FlexCAN	FF	FF	Static	Static	OFF	
LINFlexD	FF	FF	Static	Static	OFF	
SPI	FF	FF	FF	FF	FF	
I2C	FF	FF	FF	FF	FF	
SENT	FF	FF	FF	FF	FF	
Timers						
TMR	FF	FF	Static	Static	OFF	
pTMR	FF	FF	Static	Static	OFF	

Yuntu Microelectronics YTM32B1MDx Reference Manual, REV. 1.3, 2024/02 143

CHAPTER 16. POWER MODE

Table 16.1 continued from previous page

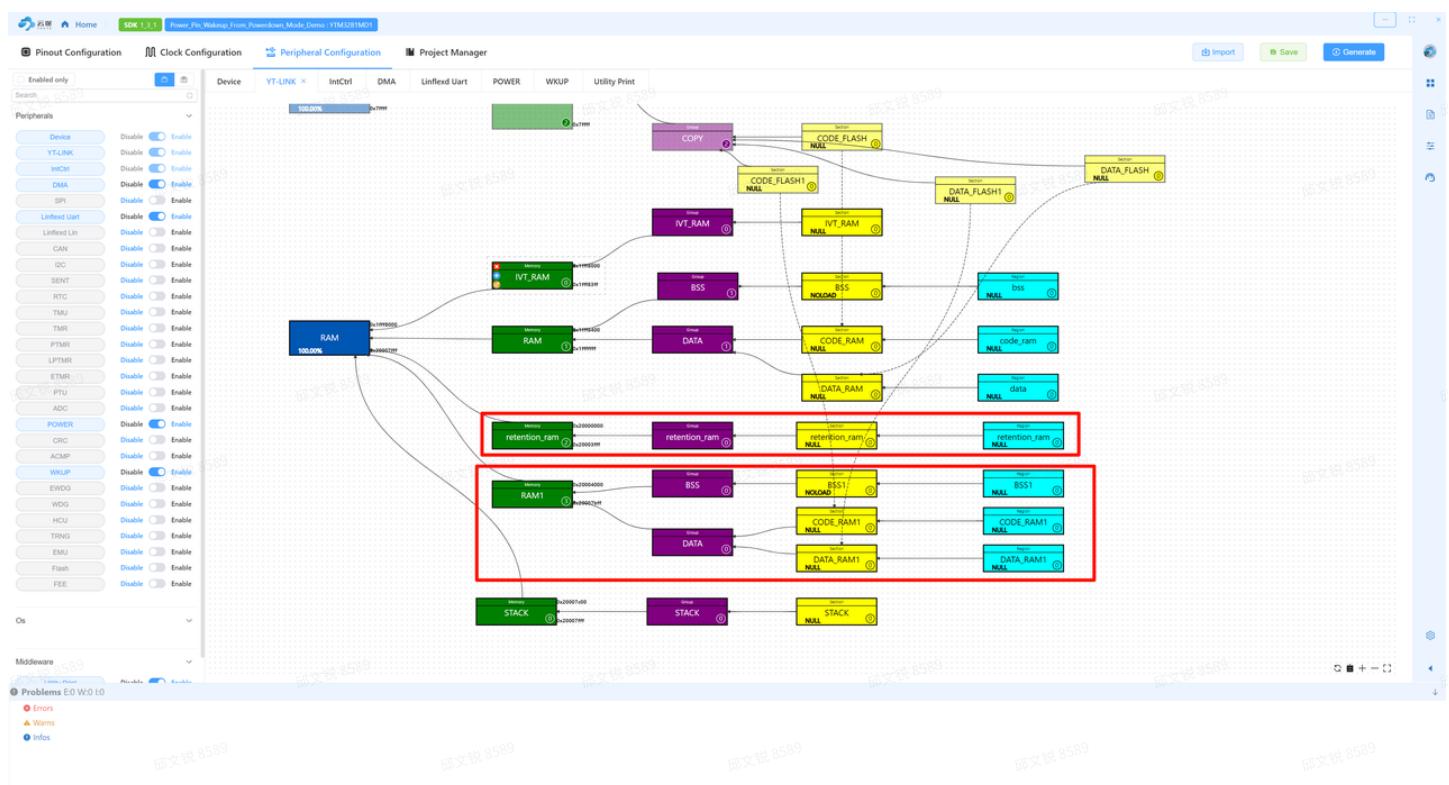
Module	Active	Sleep	Deepsleep	Standby	Powderdown
ETM	FF	FF	FF	Static	OFF
UgTHR	FF	FF	FF	FF	FF
RTC	FF	FF	FF	FF	FF
PTU	FF	FF	FF	Static	OFF
TMU	FF	FF	FF	FF	OFF
Analog					
ADC	FF	FF	FF	Static	OFF
ACMP	FF	FF	FF	FF	FF
Human-machine interfaces					
GPIO	FF	FF	FF	FF	OFF
WKU	Static	Static	Static	Static	FF

1. FF: Full Function  
2. RPM: Reduced Power Mode  
3. When DPDEN=1, Etach enters its deep Powderdown mode  
4. The content of SRAM with below address is retained:  
0x3FFC000-0xFFFFFFF retained  
0x2000000-0x20003FFF retained

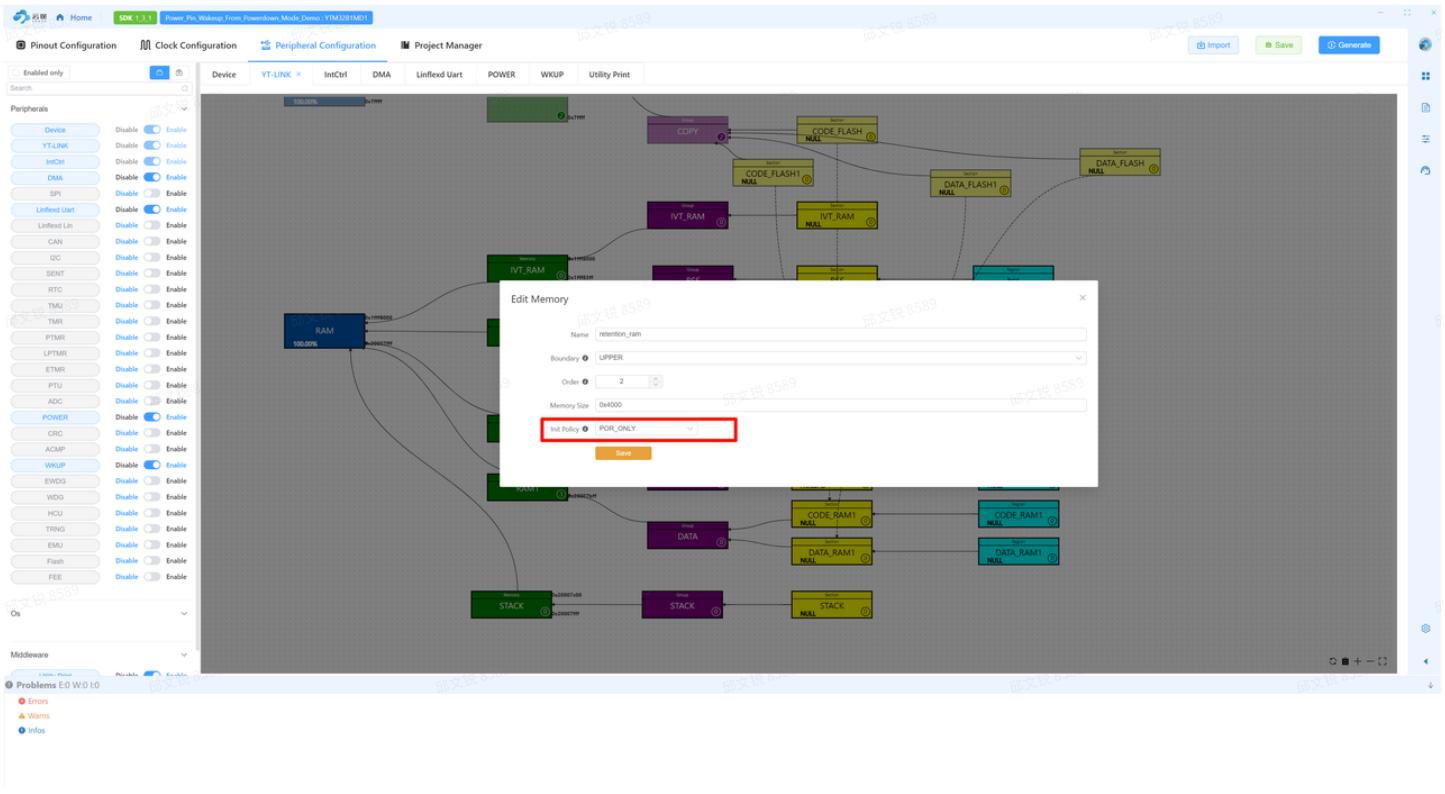
### 16.3 Wake-up Sources

Table 16.2: Wakeup Sources

	Deepsleep	Standby	Powderdown
--	-----------	---------	------------



同时配置这段RAM为POR\_ONLY，确保其Powerdown唤醒不被初始化，实现变量保持的功能。



### 3. 在程序中清除RCU上电复位标志位，保证PowerDown唤醒后retenram\_ram不会重复初始化：

```

int main(void)
{
    /* USER CODE BEGIN 1 */
    RCU->RSSR |= 0x1;
    status_t status = STATUS_SUCCESS;
    /* USER CODE END 1 */
    Board_Init();
    /* USER CODE BEGIN 2 */
    WKU_DRV_SetPinIsolation(false);
    PRINTF("Build %s %s\n", __DATE__, __TIME__);
    PRINTF("Power Pin Wakeup from Powerdown Mode!\n\n");
    OSIF_TimedDelay(5000);

    /* If POR or Reset pin */
    if ((RCU_RSSR_POR_LVD_MASK | RCU_RSSR_PIN_MASK) & RCU->RSSR)
    {
        PRINTF("POR or PIN reset.\n");
        RCU->RSSR = RCU_RSSR_POR_LVD_MASK | RCU_RSSR_PIN_MASK;
    }
    else
    {
        REGFILE->DR[0]++;
        PRINTF("Powerdown wakeup count: %d\n", REGFILE->DR[0]);
    }

    WKU_DRV_SetPinIsolation(true);
    PRINTF("Start to enter Powerdown mode.\n");
    // for(uint32_t i = 0; i < 0x400; i++)
    // {
    //     PRINTF("retention_value[%d] = %d\n", retention_value[i]);
    // }
    // Enter into Powerdown mode
    for(uint32_t i = 0; i < 0x400; i++)
    {
        retention_value[i] = 0;
    }
}

```

The screenshot shows the Visual Studio Code editor with the main.c file open. The code handles RCU initialization, powerdown entry, and retention value management. A red box highlights the line `RCU->RSSR |= 0x1;` in the `USER CODE BEGIN 1` section. The bottom status bar shows the build log for 'Power\_Pin\_Wakeup\_From\_Powerdown\_Mode\_Demo' project on 'YTM32B1MD1'.

### 4. 在进入PowerDown之前给测试数组赋值，按键唤醒后查看这段数组的内存：

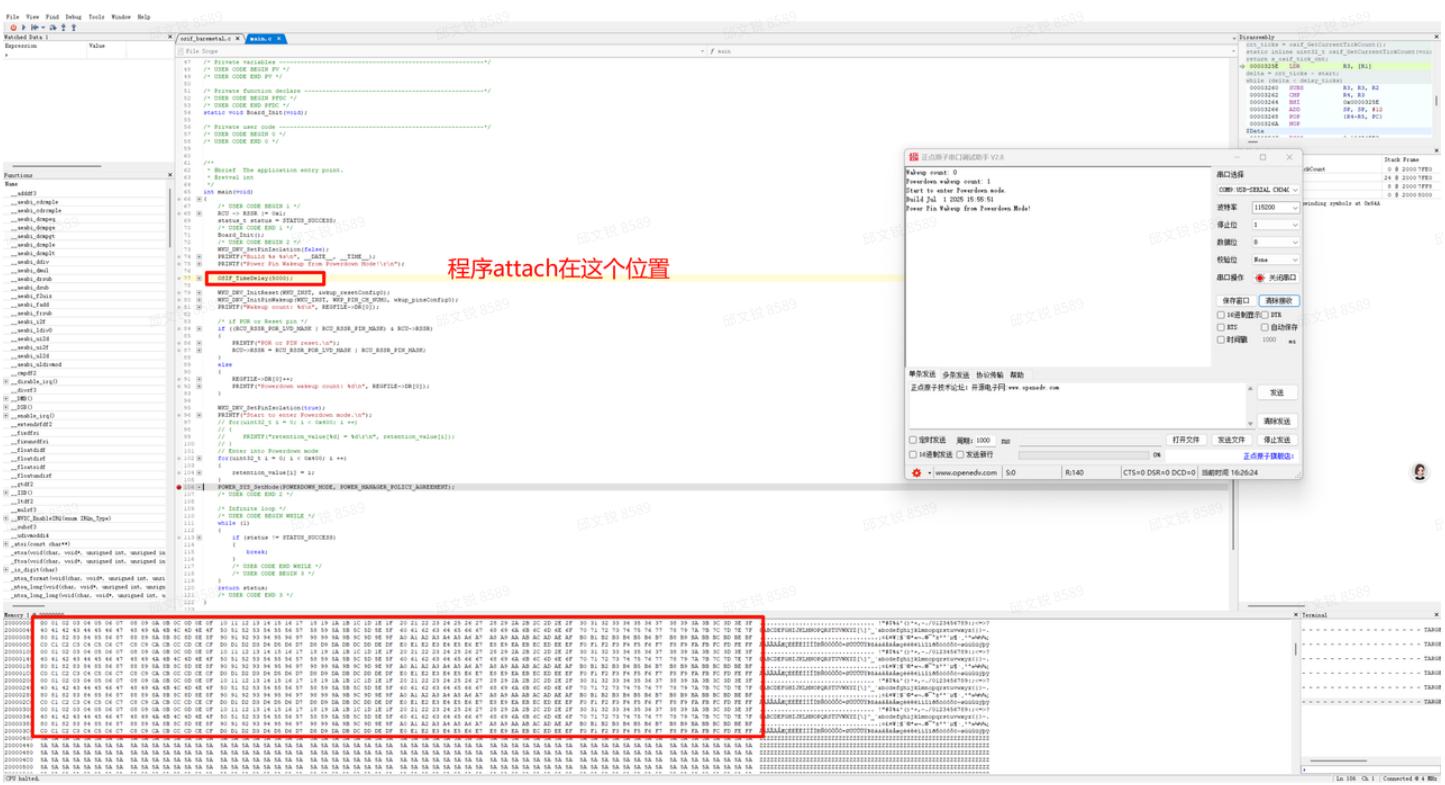
```

    volatile uint8_t retention_value[0x400] __attribute__((section(".retention_ram")));
}

```

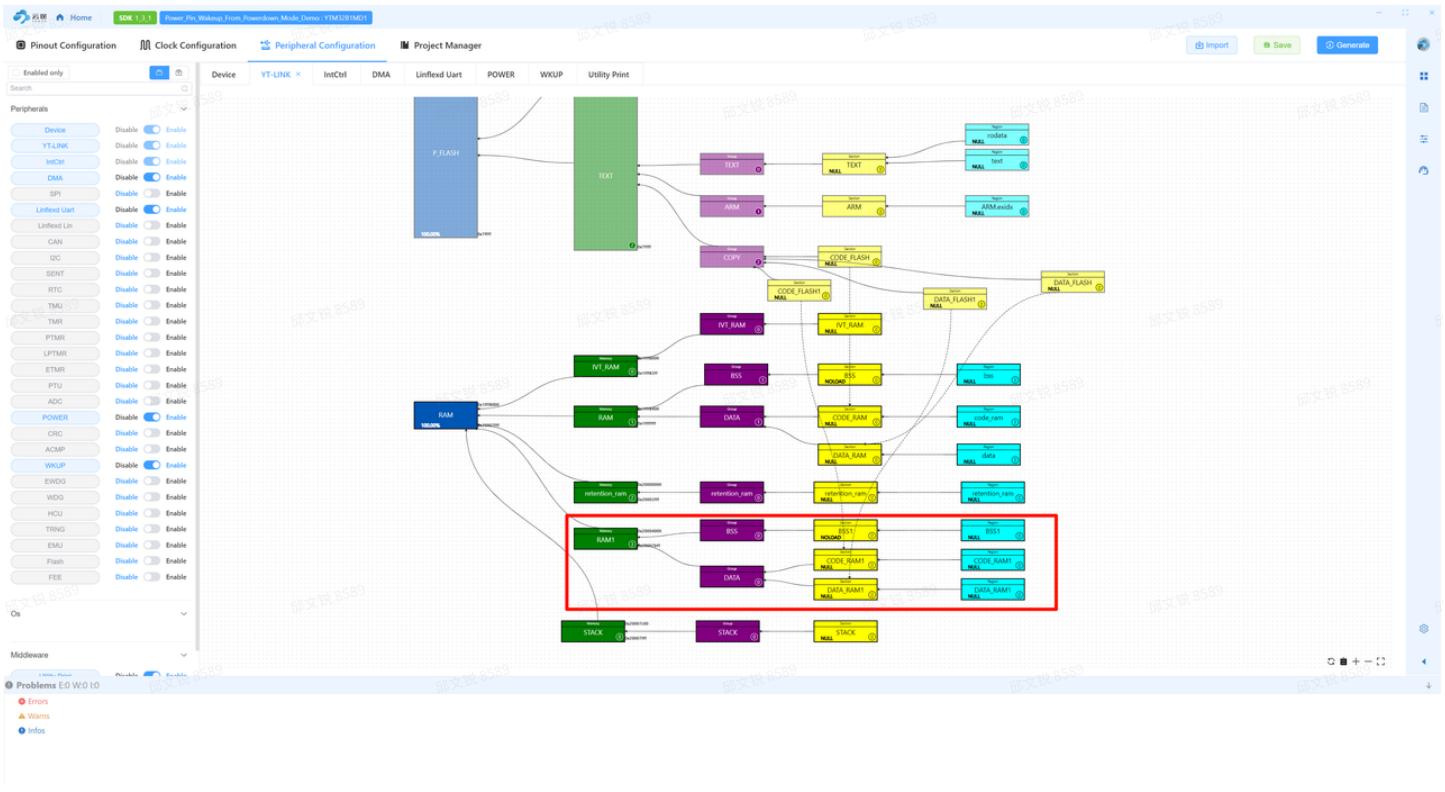
The screenshot shows the Visual Studio Code interface with the main.c file open. A red box highlights the line of code where the variable retention\_value is declared. The code is part of a function that initializes a timer and then enters a loop where it prints the retention value every second.

5. 测试，程序下载后等5s进入Powerdown，按下按键2唤醒后attach，查看0x20000000这段内存，实测可以保持



## 2. 将RAM1利用起来

如下图所示将RAM1像主RAM一样分段，包含BSS, CODE\_RAM, DATA\_RAM. 并且设置COPY\_From属性



在Ram\_Init1文件里可以看到增加了三个RAM1的初始化段，可以实现和主ram一样的功能。

```

const RamCopyLayoutType CopyLayout[4] = {
    {
        .RamStart=CODE_RAM_ram_start,
        .RomStart=CODE_RAM_rom_start,
        .RomEnd=CODE_RAM_rom_end,
        .InitType=INIT_NORMAL,
    },
    {
        .RamStart=DATA_RAM_ram_start,
        .RomStart=DATA_RAM_rom_start,
        .RomEnd=DATA_RAM_rom_end,
        .InitType=INIT_NORMAL,
    },
    {
        .RamStart=DATA_RAM1_ram_start,
        .RomStart=DATA_RAM1_rom_start,
        .RomEnd=DATA_RAM1_rom_end,
        .InitType=INIT_NORMAL,
    },
    {
        .RamStart=CODE_RAM1_ram_start,
        .RomStart=CODE_RAM1_rom_start,
        .RomEnd=CODE_RAM1_rom_end,
        .InitType=INIT_NORMAL,
    }
};

const RamZeroLayoutType ZeroLayout[2] = {
    {
        .RamStart=BSS_start,
        .RomEnd=BSS_end,
        .InitType=INIT_NORMAL,
    },
    {
        .RamStart=BSS1_start,
        .RomEnd=BSS1_end,
        .InitType=INIT_NORMAL,
    }
};

```

在各个RAM段定义变量进行测试，主RAM写满了，需要把变量手动声明定义到RAM1，如下图：

### 3. 工程

