

## 32 位 微控制器

## HC32F460 系列的 DMA 控制器

### 适用对象

系列	产品型号
HC32F460	HC32F460JEU A
	HC32F460JETA
	HC32F460KEUA
	HC32F460KETA
	HC32F460PETB

# 目 录

<b>1</b>	<b>摘要 .....</b>	<b>3</b>
<b>2</b>	<b>DMA 简介 .....</b>	<b>3</b>
<b>3</b>	<b>HC32F460 系列的 DMA.....</b>	<b>4</b>
3.1	简介 .....	4
3.2	说明 .....	4
3.2.1	寄存器介绍 .....	4
3.2.2	工作流程介绍 .....	6
<b>4</b>	<b>样例代码 .....</b>	<b>10</b>
4.1	代码介绍 .....	10
4.2	代码运行 .....	12
<b>5</b>	<b>版本信息 &amp; 联系方式 .....</b>	<b>13</b>

## 1 摘要

本篇应用笔记主要介绍如何使用 HC32F460 系列芯片的 DMA 模块传输数据。

## 2 DMA 简介

### 什么是 DMA？

DMA（直接内存访问控制器）功能块可以不通过 CPU 高速传输数据。使用 DMA 能提高系统性能。

### DMA 的重要特征？

DMA 独立于 CPU 总线的总线，所以即便是在使用 CPU 总线的时候，DMA 也可进行传输操作。

## 3 HC32F460 系列的 DMA

### 3.1 简介

HC32F460 系列 MCU 内部集成 DMAC 模块，能够在 CPU 不参与的情况下实现存储器之间，存储器和外围功能模块之间以及外围功能模块之间的数据交换。

### 3.2 说明

DMAC 总线独立于 CPU 总线，按照 AMBA AHB-Lite 总线协议传输。

拥有 2 个 DMA 控制单元，共 8 个独立通道，可以独立操作不同的 DMA 传输功能。两个控制单元受不同处理器控制，可以同时独立使用。

每个通道的启动资源通过独立的触发源选择寄存器配置。

每次请求传输一个数据块，数据块最小为 1 个数据，最多为 1024 个数据。每个数据的宽度可配置为 8bit，16bit，32bit。

源地址和目标地址可以独立配置为固定、自增、自减、循环或指定偏移量的跳转。

可产生 3 种中断：块传输完成中断，传输完成中断，传输错误中断。每种中断都可配置是否屏蔽。其中块传输完成，传输完成可作为事件输出，作为其他外围模块的触发源。

支持连锁传输功能，可实现一次请求传输多个数据块。

支持外部事件触发通道重置。

不使用时可设置进入模块停止状态以降低功耗。

#### 3.2.1 寄存器介绍

- 1) DMA\_EN：DMA 使能寄存器，使能或关闭 DMA 模块。
- 2) DMA\_CHEN：通道使能寄存器，使能或关闭 DMA 通道，bit0~3 分别对应一个通道。
- 3) DMA\_INSTAT0~1：中断状态寄存器（传输请求溢出错误中断、传输错误中断、块传输完成中断、传输完成中断）。
- 4) DMA\_INTMASK0~1：中断屏蔽寄存器，配置各中断是否屏蔽。
- 5) DMA\_INTCLR0~1：中断复位寄存器，清空中断状态标志位。

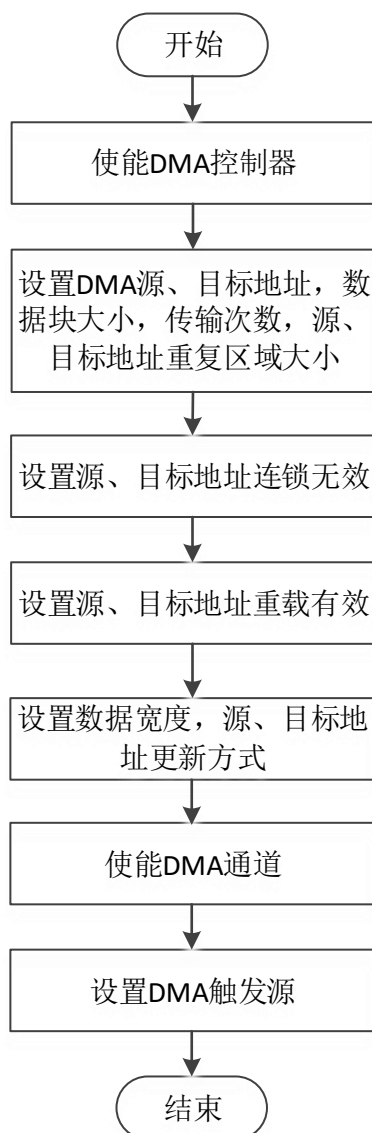
- 6) DMA\_RCFGCTL: 通道重置寄存器, 配置 DMA 重置后的相关参数, 包括: 剩余传输次数计数方式、目标/源地址重置方式、通道选择、链式传输等。
- 7) DMA\_CHSTAT: 通道状态观测寄存器。
- 8) DMA\_TRGSEL0~3: 触发源选择寄存器, 配置各通道启动传输的触发源, 配置前需打开 PWR\_FCG0 寄存器的 PTDIS 位。
- 9) DMA\_TRGSELRC: 通道重置触发源选择寄存器, 配置启动通道重置的触发源。
- 10) DMA\_SAR0~3: 源地址寄存器, 配置传输源地址。
- 11) DMA\_DAR0~3: 目标地址寄存器, 配置传输目标地址。
- 12) DMA\_DTCTL0~3: 数据控制寄存器, 配置传输次数和数据块大小。
- 13) DMA\_RPT0~3: 重复区域大小寄存器, 配置源地址和目标地址重复区域大小。
- 14) DMA\_RPTBB0~3: 重复区域大小寄存器 B, 配置源地址和目标地址重复区域大小。
- 15) DMA\_SNSEQCTL0~3: 源设备不连续地址传输控制寄存器, 配置源地址跳转的地址偏移和源地址跳转的数据量
- 16) DMA\_SNSEQCTLB0~3: 源设备不连续地址传输控制寄存器 B, 配置源不连续区域地址间距和源地址跳转的数据量
- 17) DMA\_DNSEQCTL0~3: 目标设备不连续地址传输控制寄存器, 配置目标地址跳转的地址偏移量和数据量
- 18) DMA\_DNSEQCTLB0~3: 目标设备不连续地址传输控制寄存器 B, 配置目标不连续区域地址间距和目标地址跳转数据量
- 19) DMA\_LLP0~3: 链指针寄存器, 配置链指针
- 20) DMA\_CHxCTL(x=0~3):通道控制寄存器

### 3.2.2 工作流程介绍

本章节主要介绍 DMA 传输模式的设置和运行流程。

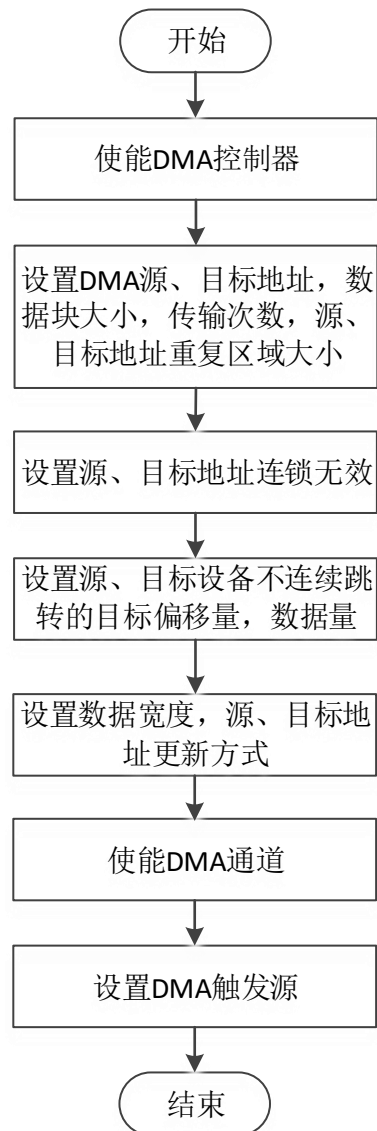
#### 1) 重载传输

该传输可以配置源地址、目标地址在增加/减少至寄存器配置的重复区域大小时重新返回至最初的地址设定值。重复区域的大小由寄存器 DMA\_RPT 和 DMA\_CHxCTL.HSIZE 的设定值决定。



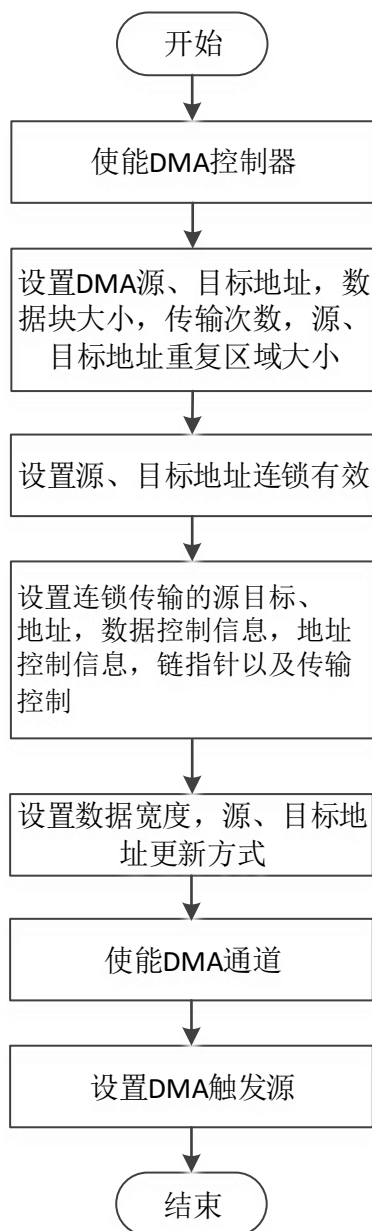
## 2) 不连续传输

该传输可以传输指定数量的数据后，地址将跳过指定偏移量，当地址重载与不连续跳转的条件同时满足时，执行地址重载。



### 3) 连锁传输

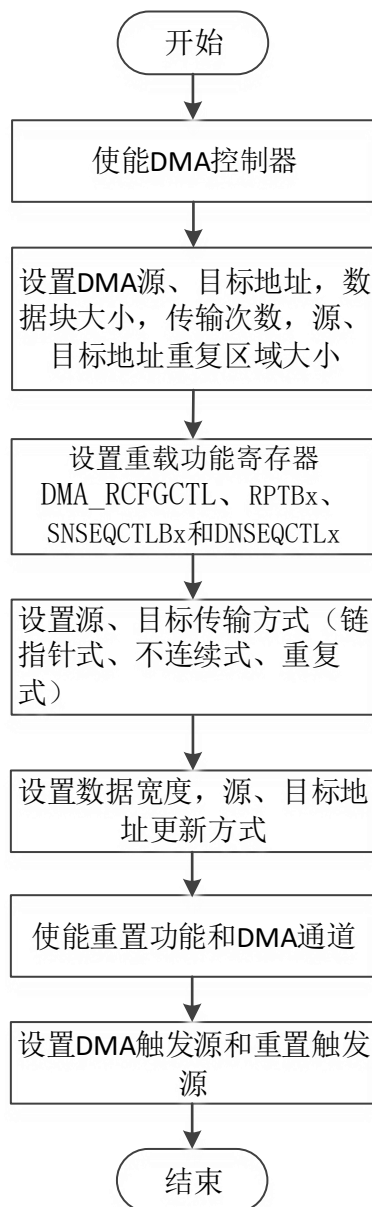
该传输当一个描述符的最后一次传输结束时，LLP 指定的下一个描述符将被从存储器中载入通道配置寄存器。等待下一次传输请求输入，开始新描述符的第一次传输。或者根据寄存器 DMA\_CHxCTLx.LLPRUN 的设置，在载入新描述符后直接开始第一次传输。





#### 4) 通道重置传输

通道重置功能，是指通过外围电路的事件请求来修改通道内部状态寄存器，重新配置下一次数据的传输方式。



#### 5) 传输提前终止

传输过程中通道使能寄存器 DMA\_CHEN.CHENx 保持有效，非连锁传输时，数据控制寄存器 DMA\_DTCTLx 设定的传输次数完成后自动置为无效，连锁传输时，最后一次传输的传输次数完成后自动置为无效。传输过程中如果软件写 DMA\_CHEN.CHENx 为 0，则 DMA 将在完成当次数据读写后终止传输。

## 4 样例代码

### 4.1 代码介绍

用户可根据上述的工作流程编写自己的代码来学习验证该模块，也可以直接通过华大半导体的网站下载到设备驱动库（Device Driver Library, DDL）的样例代码并使用其中的 DMA 的样例进行验证。

以下部分简要介绍本 AN 基于 DDL 的 DMA 模块样例 dmac\_reload\_address 代码所涉及的各项配置。

#### 1) 初始化 LED:

```
/* Initialize LED */  
LedInit();
```

#### 2) 初始化 DMA 配置:

```
/* Set data block size. */  
stcDmaCfg.u16BlockSize = DMA_BLKSIZE;  
/* Set transfer count. */  
stcDmaCfg.u16TransferCnt = DMA_TRNCNT;  
/* Set source & destination address. */  
stcDmaCfg.u32SrcAddr = (uint32_t)(&u32SrcBuf[0]);  
stcDmaCfg.u32DesAddr = (uint32_t)(&u32DstBuf[0]);  
/* Set repeat size. */  
stcDmaCfg.u16SrcRptSize = DMA_SRPT_SIZE;  
stcDmaCfg.u16DesRptSize = DMA_DRPT_SIZE;  
  
/* Disable linked list transfer. */  
stcDmaCfg.stcDmaChCfg.enLlpEn = Disable;  
/* Enable repeat function. */  
stcDmaCfg.stcDmaChCfg.enSrcRptEn = Enable;  
stcDmaCfg.stcDmaChCfg.enDesRptEn = Enable;  
/* Set source & destination address mode. */  
stcDmaCfg.stcDmaChCfg.enSrcInc = AddressIncrease;  
stcDmaCfg.stcDmaChCfg.enDesInc = AddressIncrease;  
/* Enable interrupt. */  
stcDmaCfg.stcDmaChCfg.enIntEn = Enable;  
/* Set data width 32bit. */  
stcDmaCfg.stcDmaChCfg.enTrnWidth = Dma32Bit;
```

3) 使能 DMA 外设时钟:

```
/* Enable DMA clock. */  
if(DMA_UNIT == M4_DMA1)  
{  
    PWC_Fcg0PeriphClockCmd(PWC_FCG0_PERIPH_DMA1,Enable);  
}  
else if(DMA_UNIT == M4_DMA2)  
{  
    PWC_Fcg0PeriphClockCmd(PWC_FCG0_PERIPH_DMA2,Enable);  
}
```

4) 使能和初始化 DMA:

```
/* Enable DMA1. */  
DMA_Cmd(DMA_UNIT,Enable);  
/* Initialize DMA. */  
DMA_InitChannel(DMA_UNIT, DMA_CH, &stcDmaCfg);  
/* Enable DMA1 channel0. */  
DMA_ChannelCmd(DMA_UNIT, DMA_CH,Enable);  
/* Clear DMA transfer complete interrupt flag. */  
DMA_ClearIrqFlag(DMA_UNIT, DMA_CH,TrnCpltIrq);
```

5) 设置 DMA 触发源、触发 DMA:

```
/* Enable PTDIS(AOS) clock*/  
PWC_Fcg0PeriphClockCmd(PWC_FCG0_PERIPH_PTDIS,Enable);  
DMA_SetTriggerSrc(DMA_UNIT, DMA_CH, EVT_AOS_STRG);  
AOS_SW_Trigger();
```

6) 比较 DMA 源、目标缓存数据:

```
u8CmpRet = memcmp(u32DstBuf, u32ExpectDstBufData, sizeof(u32DstBuf));  
if(0 == u8CmpRet)  
{  
    LED1_ON(); /* Meet the expected */  
}  
else  
{  
    LED0_ON(); /* Don't meet the expected */  
}
```

## 4.2 代码运行

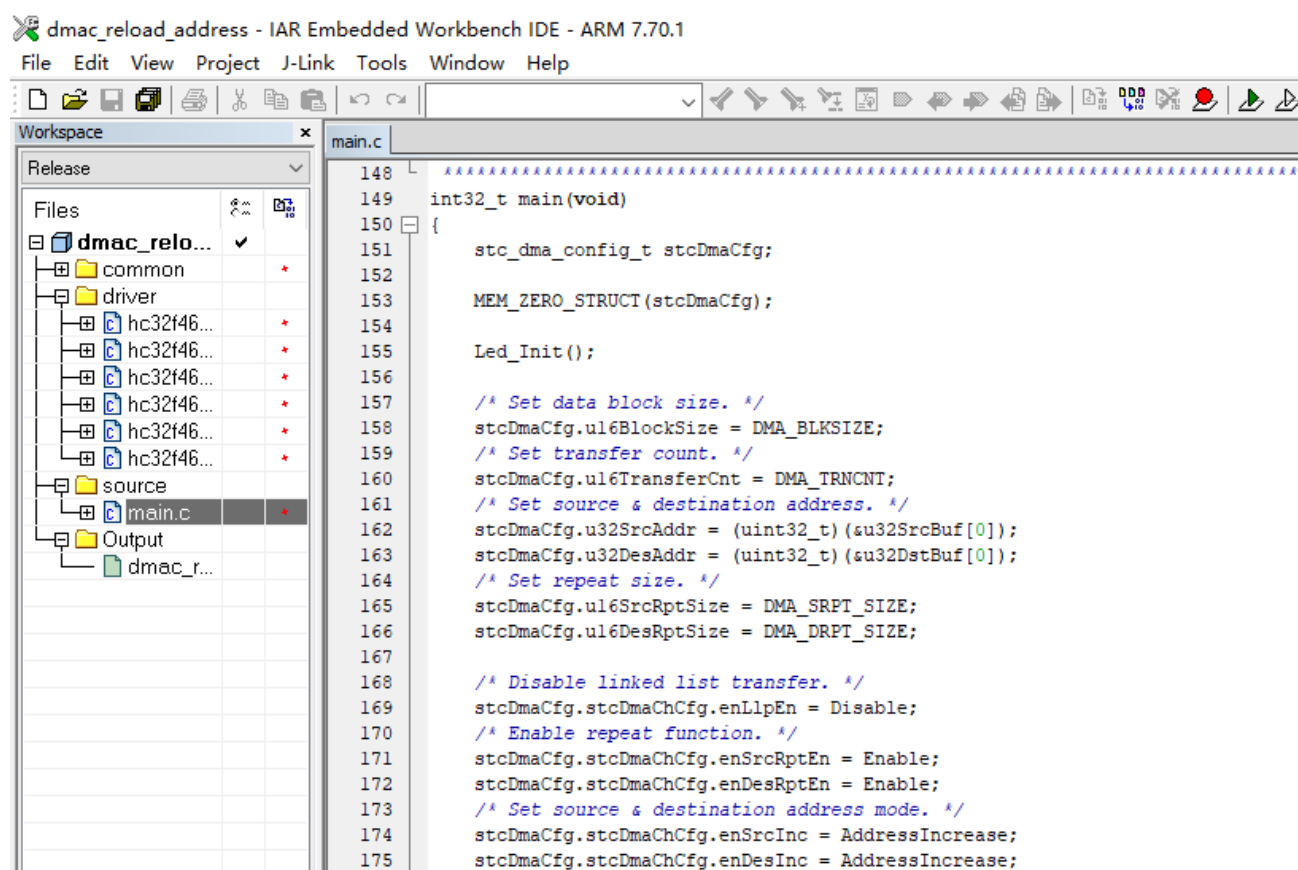
用户可以通过华大半导体的网站下载到 HC32F460 的 DDL 的样例代码



（dmac\_reload\_address），并配合评估用板（EV-HC32F460-LQFP100-050-V1.1）运行相关代码学习使用 DMA 模块。

以下部分主要介绍如何在评估板上运行 DMA 样例代码并观察结果：

- 确认安装正确的 IAR EWARM v7.7 工具（请从 IAR 官方网站下载相应的安装包，并参考用户手册进行安装）。
- 从华大半导体网站下载 HC32F460 DDL 代码。
- 下载并运行 dmac\dmac\_reload\_address\中的工程文件：

1) 打开 dmac\_reload\_address\工程，并打开‘main.c’如下视图：



- 2) 点击  重新编译整个项目。
- 3) 点击  将代码下载到评估板上，全速运行。
- 4) 绿色 LED 灯点亮。

## 5 版本信息 & 联系方式

日期	版本	修改记录
2019/3/20	Rev1.0	初版发布



---

如果您在购买与使用过程中有任何意见或建议，请随时与我们联系。

Email: [mcu@hdsc.com.cn](mailto:mcu@hdsc.com.cn)

网址: <http://www.hdsc.com.cn/mcu.htm>

通信地址: 上海市张江高科园区碧波路 572 弄 39 号

邮编: 201203

---

