



32-bit Microcontrollers

DMA Controller for HC 32 F 460 Series

Applicable objects

Series	Product Model
HC32F460	HC32F460JEUA
	HC32F460JETA
	HC32F460KEUA
	HC32F460KETA
	HC32F460PETB

Table of Contents Table of Contents

1	Abstract	3
2	DMA Introduction	3
3	DMA of HC32F460 Series.....	4
3.1	Introduction	4
3.2	Description	4
3.2.1	Register Introduction	4
3.2.2	Workflow Introduction.....	6
4	Sample Code	10
4.1	Code Introduction.....	10
4.2	Code Run	12
5	Version Information & Contact	13

1 Abstract

This application note introduces how to transfer data using the DMA module of HC32F460 series chip.

2 DMA Introduction

What is DMA?

The DMA (Direct Memory Access Controller) function block allows high-speed data transfer without going through the CPU. Using DMA can improve system performance.

What are the important features of DMA?

DMA is bus independent of the CPU bus, so DMA can perform transfer operations even when the CPU bus is in use.

3 DMA of HC32F460 series

3.1 Introduction

The HC32F460 series MCUs have an internal DMAC module that enables data exchange between memories, between memories and peripheral function modules, and between peripheral function modules without CPU involvement.

3.2 Description

The DMAC bus is independent of the CPU bus and is transmitted according to the AMBA AHB-Lite bus protocol.

There are 2 DMA control units with 8 independent channels, which can operate different DMA transfer functions independently. The two control units are controlled by different processors and can be used independently at the same time.

The start-up resources for each channel are configured through separate trigger source selection registers.

One block of data is transmitted per request, with a minimum of 1 data block and a maximum of 1024 data blocks. The width of each data block can be configured as 8bit, 16bit, 32bit.

The source and destination addresses can be independently configured as fixed, self-increasing, self-decreasing, cyclic, or hopping at specified offsets.

Three types of interrupts can be generated: block transfer completion interrupt, transfer completion interrupt, and transfer error interrupt. Each of these interrupts can be configured to be masked or not. Among them, block transfer completion and transfer completion can be output as events and used as trigger sources for other peripheral modules.

Support chain transmission function, which enables transmission of multiple data blocks in one request.

Support external events to trigger channel reset.

It can be set to enter the module stop state when not in use to reduce power consumption.

3.2.1 Register Introduction

1) DMA_EN: DMA enable register to enable or disable the DMA module.

- 2) DMA_CHEN: Channel enable register, enable or disable DMA channel, bits 0~3 correspond to one channel respectively.
- 3) DMA_INSTAT0~1 : Interrupt status registers (transfer request overflow error interrupt, transfer error interrupt, block transfer completion interrupt, transfer completion interrupt).
- 4) DMA_INTMASK0~1: Interrupt mask register, configure whether each interrupt is masked or not.
- 5) DMA_INTCLR0~1 : Interrupt reset register, clear the interrupt status flag bit.

- 6) DMA_RCFGCTL: Channel reset register, configure relevant parameters after DMA reset, including: remaining transfer count method, destination/source address reset method, channel selection, chain transfer, etc.
- 7) DMA_CHSTAT: Channel status observation register.
- 8) DMA_TRGSEL0~3: Trigger source selection register, configure the trigger source for each channel to start transmission, need to open before configuration
The PTDIS bit of the PWR_FCG0 register.
- 9) DMA_TRGSELRC: Channel reset trigger source selection register, configure the trigger source to start channel reset.
- 10) DMA_SAR0~3: Source address registers, configure the transmission source address.
- 11) DMA_DAR0~3: Destination address registers, configure the transmission destination address.
- 12) DMA_DTCTL0~3: Data control registers, configure the number of transfers and data block size.
- 13) DMA_RPT0~3: Repeat area size registers, configure the repeat area size of source and destination addresses.
- 14) DMA_RPTBB0~3: Repeat area size register B, configure the repeat area size of source and destination address.
- 15) DMA_SNSEQCTL 0~3: Source device discontinuous address transfer control register, configure the address offset of source address hopping and the amount of data of source address hopping
- 16) DMA_SNSEQCTLB0~3: Source device discontinuous address transfer control register B, configure the source discontinuous area address spacing and the amount of data for source address hopping
- 17) DMA_DNSEQCTL 0~3: Target device discontinuous address transfer control registers, configure the address offset and data amount for target address hopping
- 18) DMA_DNSEQCTLB0~3: Target device discontinuous address transfer control register B, configure target discontinuous area address spacing and target address hopping data amount
- 19) DMA_LLPO~3: Chain pointer register, configure chain pointer

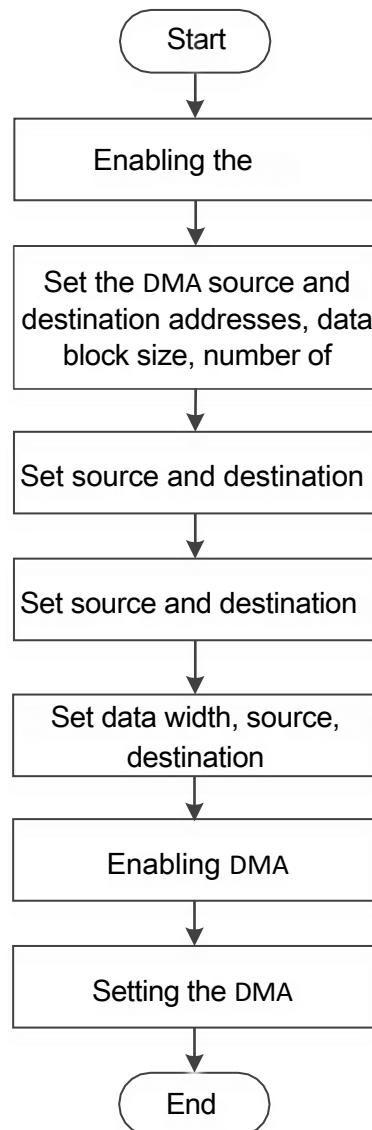
20) DMA_CHxCTL(x=0~3):Channel control register

3.2.2 Workflow Introduction

This section describes the setup and operation flow of DMA transfer mode.

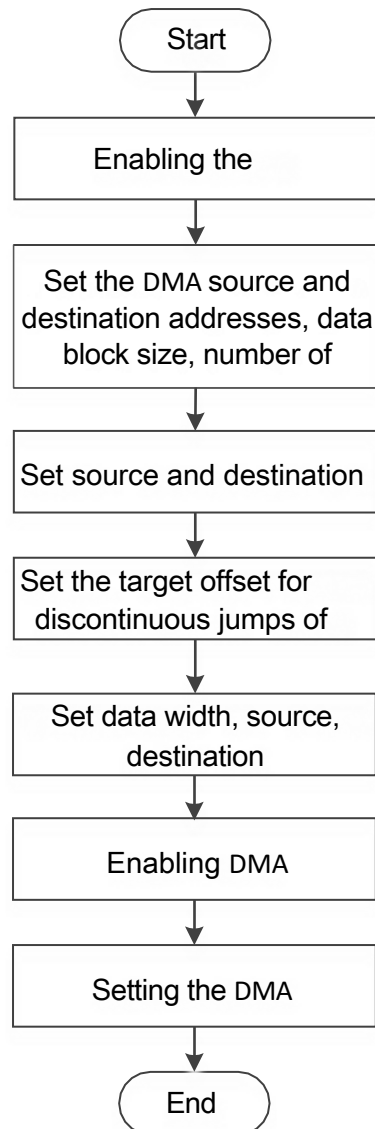
1) Heavy load transmission

This transfer configures the source address, the destination address to revert to the original address setting when it increases/decreases to the size of the repeat area configured in the register. The size of the repeat area is determined by the set values of the registers DMA_RPT and DMA_CHxCTL.HSIZE.



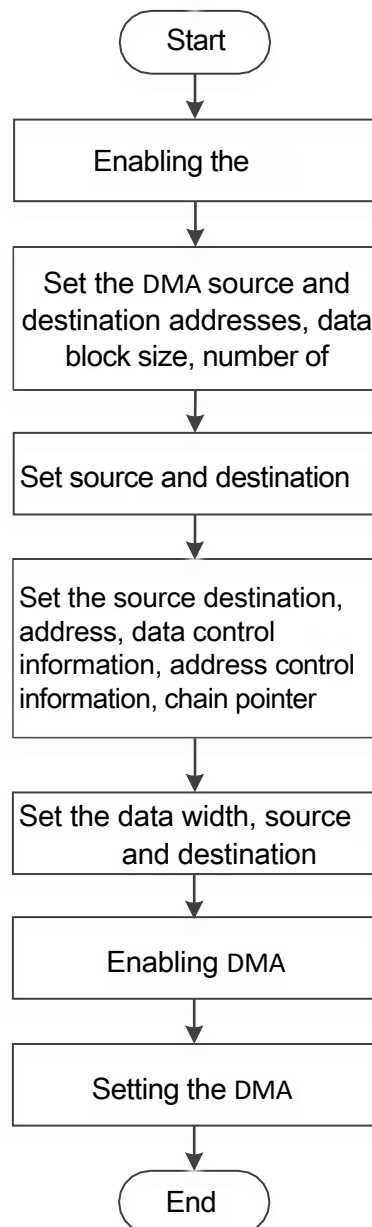
2) Discontinuous transmission

This transfer can transfer a specified amount of data after which the address will skip the specified offset. When the conditions of address reload and discontinuous hopping are satisfied at the same time, address reload is executed.



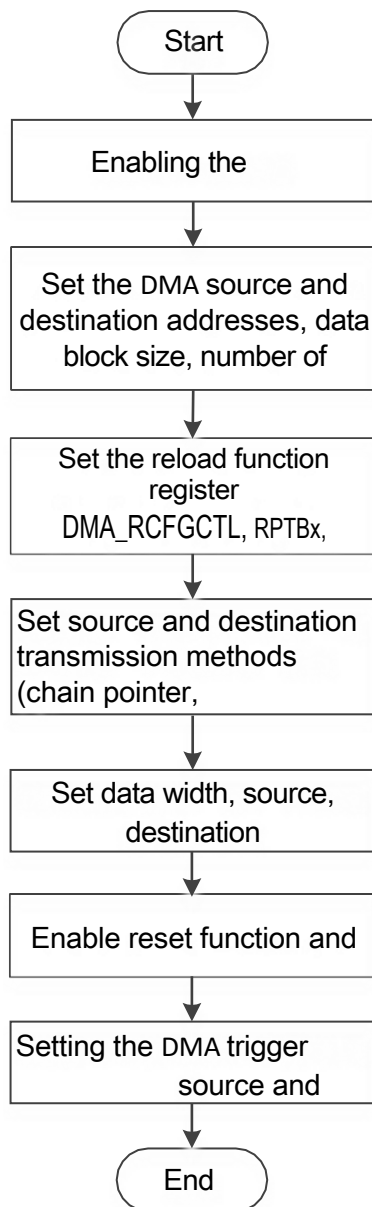
3) Chain Transfer

This transfer When the last transfer of a descriptor is completed, the next descriptor specified by LLP is loaded from memory into the channel configuration register. Wait for the next transfer request input to start the first transfer of the new descriptor. Or, depending on the setting of register DMA_CHxCTLx.LLPRUN, the first transfer is started directly after the new descriptor is loaded.



4) Channel Reset Transmission

The channel reset function is a way to reconfigure the next data transmission by modifying the internal status register of the channel through an event request from the peripheral circuit.



5) Early termination of transmission

CHENx remains valid during the transfer, and the number of transfers set in the data control register DMA_DTCTLx is automatically set to invalid when the number of transfers is completed for non-chain transfers, and automatically set to invalid when the number of transfers for the last transfer is completed for chain transfers. If the software write DMA_CHEN.CHENx is 0 during the transfer, the DMA will terminate the transfer after

completing the current data read/write.

4 Sample Code

4.1 Code Introduction

Users can write their own code to learn and verify the module according to the above workflow, or download the sample code of Device Driver Library (DDL) directly through the website of UW Semiconductors and use the sample of DMA in it for verification.

The following section briefly describes the configuration involved in the `dmac_reload_address` code of this sample AN DDL-based DMA module.

1) Initialization LED:

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

2) Initialize DMA configuration:

```
/* 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) Enables the DMA peripheral clock:

```
/* 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) Enables and initializes 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) Set DMA trigger source, trigger 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) Compare DMA source and destination cache data:

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

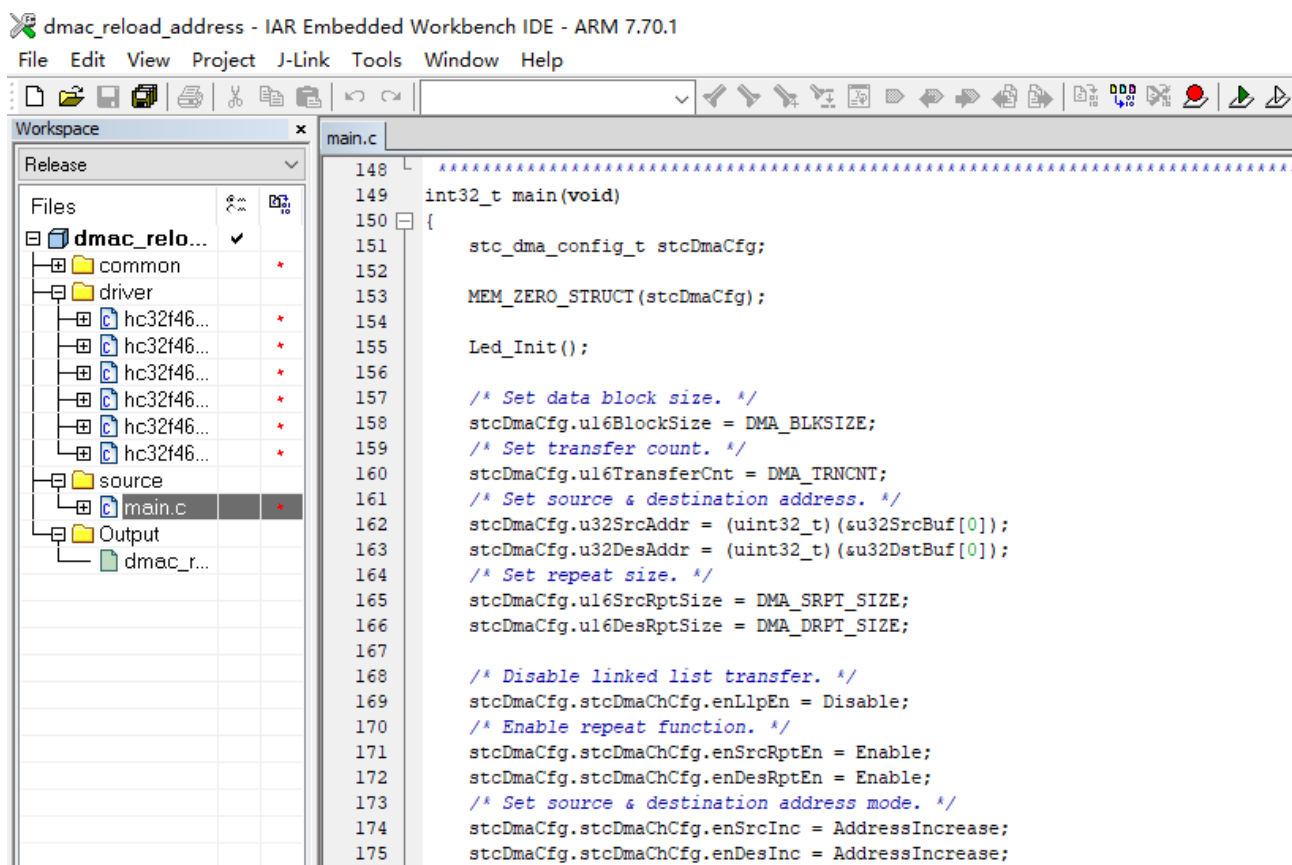
4.2 Code Run



Users can download the sample code of HC32F460 DDL through the website of UW Semiconductors.

(dmac_reload_address) and run the relevant code with the evaluation board (EV-HC32F460-LQFP100-050-V1.1) to learn to use the DMA module.

The following section describes how to run the DMA sample code on the evaluation board and observe the results:

- Verify that the correct IAR EWARM v7.7 tool is installed (please download the appropriate installation package from the official IAR website and refer to the user manual for installation).
- Download the HC32F460 DDL code from the UW Semiconductors website.
- Download and run the project file in dmac\ dmac_reload_address\ at
- 1) Open the dmac_reload_address\ project and open the 'main.c' view as follows:



- 2) Click  to recompile the entire project.
- 3) Click  Download the code to the evaluation board and run it at full speed.

4) The green LED lights up.

5 Version Information & Contact

Date	Versions	Modify records
2019/3/20	Rev1.0	Initial Release



If you have any comments or suggestions in the process of purchase and use, please feel free to contact us.

Email: mcu@hdsc.com.cn

Website: <http://www.hdsc.com.cn/mcu.htm>

Address: 39, Lane 572, Bibo Road, Zhangjiang Hi-

Tech Park, Shanghai, 201203, P.R. China

