XHSC 小华半导体 XIAOHUA SEMICONDUCTOR

32-bit Microcontrollers

DMA Controller for HC32F460 Series

Applicable objects:

F Series	HC32F460

declaration Ming Dynasty (1368-1644)

table of contents table of contents

declaration .... Ming

2

table of contents .... Table of Contents

3

1 Abstract 4

2 Introduction to DMA 5

3 DMA f o r HC32F460 series 6

3.1 Introduction... 6

3.2 Description.. 6

3.2.1 Introduction to Registers 6

3.2.2 Workflow Introduction... 8

3.3 Cautions.. 12

4 Sample Code 13

4.1 Code Introduction 13

4.2 Code Run 15

Version Information & Contacts.. 16

1 summaries

This application note describes how to transfer data using the DMA module of the HC32F460 series chips.

2 Introduction to DMA

What is DMA?

The DMA (Direct Memory Access Controller) function block transfers data at high speeds without going through the CPU. Using DMA improves system performance.

Important features of DMA?

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

3 DMA for HC32F460 series

3.1 summary

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

3.2 clarification

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 a total of 8 independent channels, which can be operated independently of the different DMA transfer functions. The two control units are controlled by different processors and can be used independently at the same time.

The startup 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 can be configured as 8bit, 16bit, 32bit.

The source and destination addresses can be independently configured as fixed, self-incrementing, self-decrementing, cyclic, or jumps with specified offsets.

Three types of interrupts can be generated: block transfer completion interrupt, transfer completion interrupt, and transfer error interrupt. Each interrupt can be configured to be blocked or not. The block transfer completion and transfer completion can be used as event outputs as triggers for other peripheral modules.

Support chain transmission function, which can realize transmission of multiple data blocks in one request. Supports external event triggered channel reset.

When not in use, it can be set to enter the module stop state to reduce power consumption.

3.2.1 Introduction to Registers

1. DMA_EN: DMA enable register, enables or disables the DMA module.
2. DMA_CHEN: Channel enable register, enable or disable DMA channel, bit0~3 corresponds 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 interrupt status flag bit.

6. DMA_RCFGCTL: Channel reset register, configures the relevant parameters after DMA reset, including: number of remaining transfers
   counting method, destination/source address reset method, channel selection, chaining, 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, the AOS bit of PWR_FCG0 register needs to be opened before configuration.
9. DMA_TRGSELRC: Channel Reset Trigger Source Selection Register, configures the trigger source that initiates channel reset.
10. DMA_SAR0~3: source address registers, configure the transmission source address.
11. DMA_DAR0~3: Destination Address Register, 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 region size registers, configure source and destination address repeat region size.
14. DMA_RPTBB0~3: Repeat Region Size Register B, configures the repeat region size of source and destination addresses.
15. DMA_SNSEQCTL0~3: source device discontinuous address transfer control registers, configure the address offset for source address hopping and the amount of data for source address hopping.
16. DMA_SNSEQCTLB0~3: Source Device Discontinuous Address Transmission Control Register B. Configure the source discontinuous area address spacing and the amount of data for source address hopping.
17. DMA_DNSEQCTL0~3: Destination Device Discontinuous Address Transfer Control Register, configures the address offset and data amount for destination address hopping.
18. DMA_DNSEQCTLB0~3: Target Device Discontinuous Address Transmission Control Register B. Configure the target discontinuous area address spacing and the amount of target address hopping data.
19. DMA_LLP0~3: chain pointer registers, configure the chain pointer.

20. DMA_CHxCTL (x=0~3): channel control register.
21. DMA_MONSARx , DMA_MONDARx , DMA_MONDTCTLx , DMA_MONRPTx . DMA_MONSNSEQCTLx, DMA_MONDNSEQCTLx: Channel Monitor Register, updated after every transfer request completed by the DMA.

3.2.2 Workflow Introduction

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

1. heavy load transmission

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

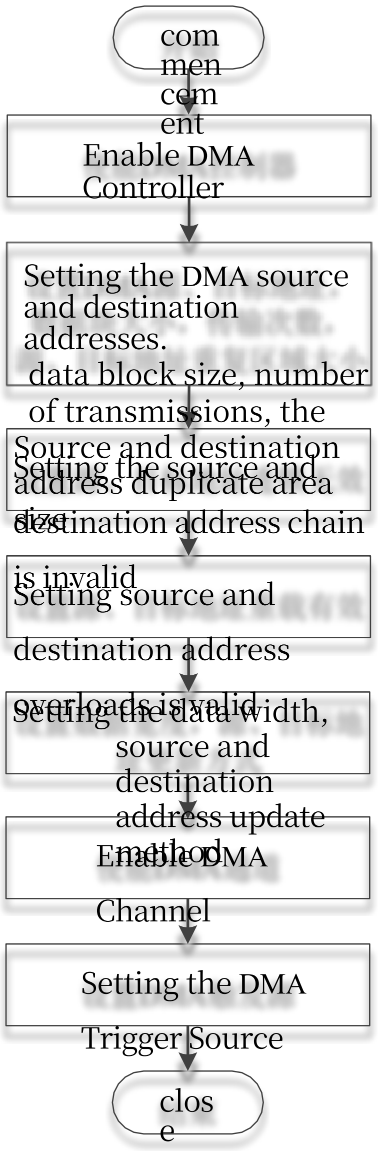

com men cem ent
Enable DMA Controller
Setting the DMA source and destination addresses. data block size, number of transmissions, the
Source and destination address duplicate area size
Setting the source and destination address chain is invalid
Setting source and destination address overloads is valid
Setting the data width, source and destination address update method
Enable DMA Channel
Setting the DMA Trigger Source
clos e

Figure 3-1 Heavy Duty Transmission Operation Flow

2. discontinuous transmission

This transmission can transmit a specified amount of data after which the address will skip the specified offset. address reloading is performed when the conditions of address reloading and discontinuous skip are both met.

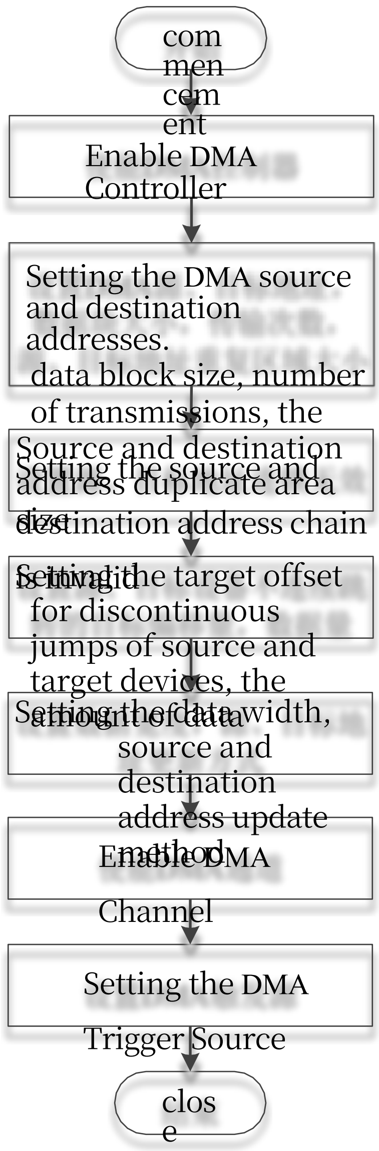
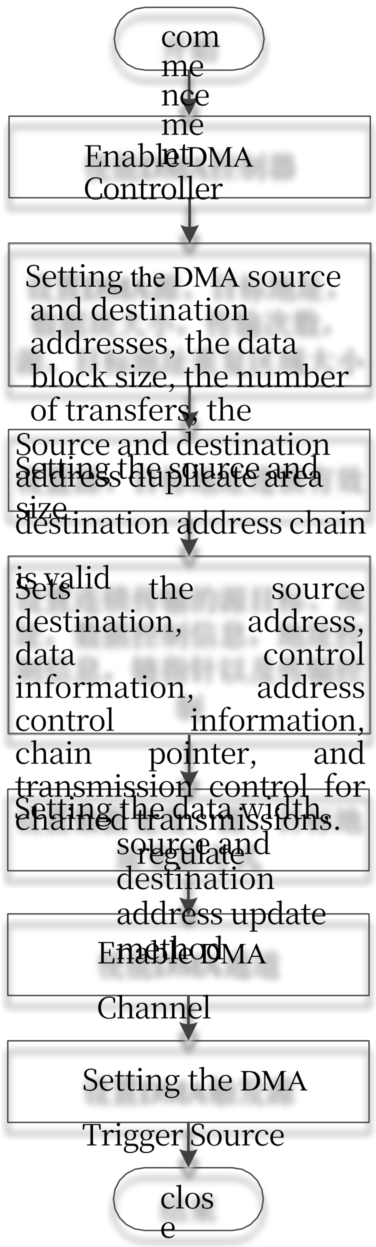

commencement
Enable DMA Controller
Setting the DMA source and destination addresses. data block size, number of transmissions, the Source and destination address duplicate area size
Setting the source and destination address chain is invalid
Setting the target offset for discontinuous jumps of source and target devices, the amount of data
Setting the data width, source and destination address update method
Enable DMA Channel
Setting the DMA Trigger Source
close

Figure 3-2 Discontinuous transmission operation flow

3. chain transmission

This transmission When the last transmission of a descriptor ends, the next descriptor specified by the LLP is loaded from memory into the channel configuration register. Wait for the next transfer request input to begin the

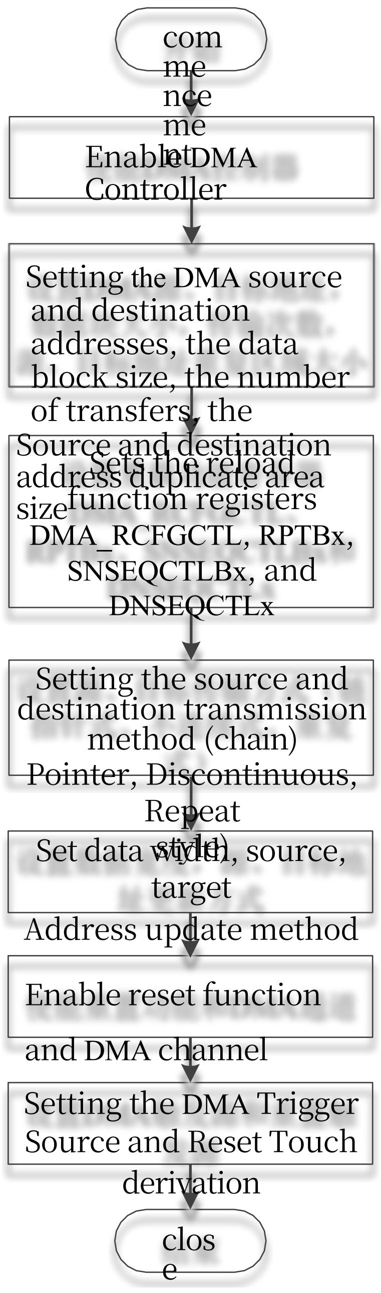

commencement
Enable DMA Controller
Setting the DMA source and destination addresses, the data block size, the number of transfers, the
Source and destination address duplicate area size
Setting the source and destination address chain is valid
Sets the source destination, address, data control information, address control information, chain pointer, and transmission control for chained transmissions.
Setting the data width, regulate
source and destination address update method
Enable DMA Channel
Setting the DMA Trigger Source
close

first transfer of the new descriptor. Or, depending on the setting of register DMA_CHxCTLx.LLPRUN, the first transfer is started directly after loading the new descriptor.

Figure 3-3 Chain Transfer Operation Flow

4. Channel Reset Transfer

The channel reset function refers to the modification of the channel's internal status registers through event requests from the peripheral circuitry to reconfigure the next data transfer.



commencement
Enable DMA Controller
Setting the DMA source and destination addresses, the data block size, the number of transfers, the Source and destination address duplicate area size
Sets the reload function registers DMA_RCFGCTL, RPTBx, SNSEQCTLBx, and DNSEQCTLx
Setting the source and destination transmission method (chain) Pointer, Discontinuous, Repeat style)
Set data width, source, target Address update method
Enable reset function and DMA channel
Setting the DMA Trigger Source and Reset Touch derivation
close

Figure 3-4 Channel Reset Transmission Run Flow

5. premature termination of transmission

The channel enable register DMA_CHEN.CHENx remains active during transmission. For non-chained transmission, the number of transmissions set in the data control register DMA_DTCTLx is automatically invalidated upon completion of the transmission count, and for chained transmission, the number of transmissions for the last transmission is automatically invalidated upon completion of the transmission count. If software write DMA_CHEN.CHENx is 0 during transmission, the DMA will terminate the transmission after completing the current data read/write.

3.3 caveat

1. When the transfer count of the DMA channel is set to non-zero, it is the actual number of transfers, and for each block of data transferred, the transfer count is reduced by 1, when decremented to 0, the channel transmission license bit is automatically cleared (channel shutdown) when transmission is completed. A transmission count of 0 indicates an infinite number of transmissions without generating a transmission completion interrupt or clearing the channel transmission license bit.
2. DMA When multiple channels are initiated to request the same bus interface, they are executed in priority order, but the channel in transmission is not interrupted, and the higher priority channel is not initiated until the current block of the current channel has been transmitted. The priority order is CH1>CH2>CH3>CH4.
3. Zeroing CHEN terminates the transfer prematurely; the DMA does not internally save the transfer state at the time of termination. If CHEN is set to allow the channel again without resetting the Channel Configuration Register (Descriptor) state, the DMA will retransmit the terminated block of data after the transfer request is entered, rather than continue the transfer at the end of the period.
4. The DMA registers only support 32bit reads and writes.
5. The channel enable CHEN register is automatically cleared by the hardware with the corresponding channel license bit after the transmission is completed. Therefore, when other channel enable operations are performed on this CHEN register, the Read Modify Write (RMW) operation instruction may inadvertently enable this channel (transmission completion channel).
6. A DMA unit has channels that cannot be configured with other channels of the unit while block transfers are in progress.

4 sample code (computing)

4.1 Code Introduction

Users can write their own code according to the above workflow to learn and verify the module, or they can directly download the sample code of Device Driver Library (DDL) from Semiconductor's official website and verify it with the sample of DMA. The following sections provide a brief overview of the configurations involved in the sample dmac_reload_address code for this AN DDL-based DMA module.

1. Initialize the LED:

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

2. Initialize the 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]);
stcDmaCfg. 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 interrup. */
stcDmaCfg.stcDmaChCfg.enIntEn = Enable;
/* Set data width 32bit. */
stcDmaCfg.stcDmaChCfg.enTrnWidth = Dma32Bit;
```

3. Enable 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 the 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. Sets the DMA trigger source, triggers the DMA:

```
/* Enable AOS clock*/
PWC_Fcg0PeriphClockCmd(PWC_FCG0_PERIPH_AOS,Enable);
DMA_SetTriggerSrc(DMA_UNIT, DMA_CH, EVT_AOS_STRG); AOS_SW_
Trigger();
```

6. Compare DMA source and target cache data:

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

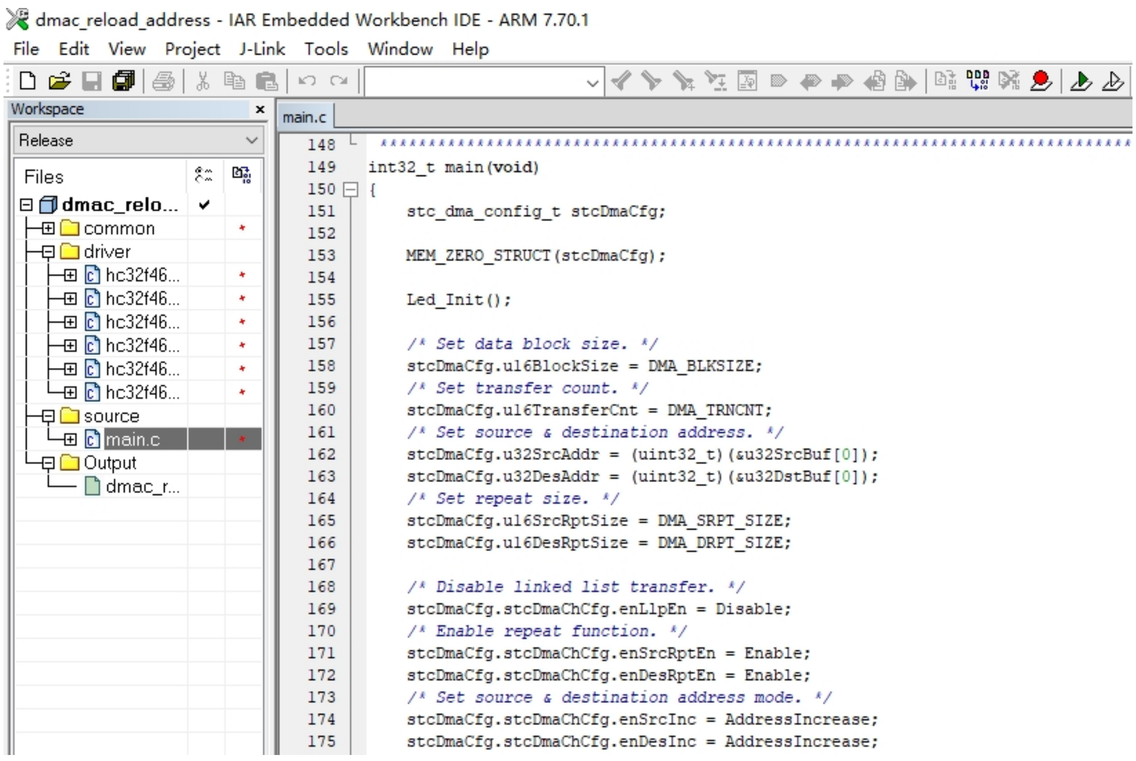
4.2 code running

Users can download the sample code (dmac_reload_address) of DDL for HC32F460 from Semiconductor's website and run the code with the evaluation board (EV-HC32F460-LQFP100-050-V1.1) to learn how to use the DMA module.

The following sections describe 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 (download the appropriate installation package from the official IAR web site and refer to the user manual for installation).
- Download HC32F460 DDL code from Semiconductor website.
- Download and run the project file in dmac\ dmac_reload_address\:

1. Open the dmac_reload_address\ project and open 'main.c' in the following view:



```
********************************************************************************
int32_t main(void)
{
    stc_dma_config_t stcDmaCfg;

    MEM_ZERO_STRUCT(stcDmaCfg);

    Led_Init();

    /* 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;
```

2. Click ▦ to recompile the entire project.
3. Click ▶ to download the code to the evaluation board and run it at full speed.
4. The green LED is lit.

Version Information & Contacts

dates	releases	edit a record
2019/03/20	Rev1.0	First Edition Release.
2020/08/07	Rev1.1	1) Add a description of the DMA_MONxx channel monitor register; 2) Updated to support models.
2022/05/27	Rev1.2	Adding DMA Notes.

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

Email: mcu@xhsc.com.cn

Website: http://www.xhsc.com.cn

Address: 10/F, Block A, No. 1867 Zhongke Road,
Pudong New Area, Shanghai, 201210, P.R. China