Programming the NS32CG160 On-Chip DMA Controller

National Semiconductor
Application Note 721
Aharon Ostrer
August 1990

1.0 INTRODUCTION

The NS32CG160 Integrated System Processor features an on-chip 2-channel DMA Controller (DMAC). The DMAC is capable of transferring blocks of data between memory and I/O devices with minimal CPU intervention. Each channel is independently programmable for operating in one of three modes. These are Single transfer operation, Double-buffer operation, and Auto-initialize operation.

This application note describes each of the three modes, and presents assembly language programs for activating the on-chip DMAC to operate in each mode.

2.0 DESCRIPTION

NS32CG160's on-chip DMA Controller (DMAC) supports 2 channels for transferring blocks of data between memory and I/O devices. The memory address, block size and type of operation, are set up in advance by software which writes to the DMAC's dedicated control registers. Once a DMA channel is programmed and enabled for block transfer, the DMAC responds to DMA channel requests on the corresponding DMA request ($\overline{DRQ}$) pin by transferring the next data element associated with that block. A block transfer normally completes once the specified number of bytes have been transferred. Each channel is independently programmable to operate in one of the following modes:

- In single transfer mode, the DMA terminates the operation once the specified block transfer is complete.
- In double-buffer mode, an alternate set of address and count registers is prepared while the current block transfer is in progress, and the DMAC automatically switches to use them when the current transfer is completed.
- In auto-initialize mode, completion of a block transfer causes the DMAC to automatically transfer a memory block, using the alternate address and byte count prespecified for it.

3.0 SETTING UP THE THREE MODES

The following three assembly programs may be used to set up channel 0 for the particular mode of DMAC operation. For the DMAC registers description refer to the NS32CG160 data sheet. The programs use the following parameters:

Parameter	Description
block1__strt	Specifies the data block start address, using the ADC register. This may be any address in the user accessible address space (from 0x000000 to 0xFFFFFF).
block1__cnt	Specifies the number of bytes to be transferred, using the BLTC register. This should be a multiple of 2 if the bus width specified in bits 7 and 8 (BW) of the MODE register of the respective DMA channel is 16-bit wide.
block2__strt	Specifies the alternate data block start address, using the ADR register. This may be any address in the user accessible address space (from 0x0000000 to 0xFFFFFF).
block2__cnt	Specifies the alternate number of bytes to be transferred, using the BLTR register. This should be a multiple of 2 if the bus width specified in bits 7 and 8 (BW) of the MODE register of the respective DMA channel is 16-bit wide.
optype	Specifies the operating mode, using the MODE register. Bit 0 specifies auto-initialize operation when set, and single transfer or double-buffer when reset. Bit 3 (DIR) specifies read (0) or write (1) memory cycle. Bits 7 and 8 (BW) specify 8-bit (00) or 16-bit (01) bus width. Bit 9 (AD) enables (1) or disables (0) the incrementing of the source or destination data block address. Bits 10–14 (BLT) specify the block length after which the DMAC will relinquish the bus even if a request is still pending. Bits 1, 2, 4–6 must always be set to zero. Bits 15 through 31 are reserved. Example 1: single transfer, 16-bit bus, read from I/O, write to memory, unlimited number of back-to-back transfers optype = 0x00000288 Example 2: double-buffer, 8-bit bus, read from I/O, write to memory, do not increment memory address, relinquish the bus after every byte transfer optype = 0x00000408 Example 3: auto-initialize, 16-bit bus, read from memory, write to I/O, unlimited number of transfers optype = 0x00000281

 TL/EE11065 RRD-B30M75/Printed in U. S. A.

3.1 Single Transfer Operation

The DMAC transfers one block of data as specified in the ADC and BLTC registers and relinquishes the bus after reaching the terminal count (BLTC = 0). An interrupt is issued (if enabled) to the CPU.

```
.set adc, 0xfffff020            #for channel l use 0xfffff040
.set bltc, 0xfffff030           #for channel l use 0xfffff050
.set mode, 0xfffff038           #for channel l use 0xfffff058
.set cntl, 0xfffff03C           #for channel l use 0xfffff05C
.set imsk, 0xfffff014


.text
movw $0,cntl                    #disable DMA channel
movd $blockl_strt,adc           #write block start address to ADC register
movd $blockl_cnt,bltc           #write byte count to BLTC register
movw $optype,mode               #enable single transfer operation
movw $0x8l,imsk                 #enable high priority interrupt on terminal
                                #count. You can use M0VW l, IMSK to enable
                                #low priority interrupt on terminal count.
                                #These values are for channel 0.
                                #For channel l use respectively:
                                #M0VW $0x90, IMSK for high priority int.
                                #M0VW $0xl0, IMSK for low priority int.
                                #To disable interrupt on terminal count reset
                                #bit 0 (for channel 0) or bit 4 (for channel l)
                                #in the IMSK register.
                                #In this case the DMAC will not
                                #issue interrupt upon transfer completion.
                                #It will set bit 0 (for channel 0) or bit 4
                                #(for channel l) in the STAT register
                                #and relinquish the bus.

movw $l,cntl                    #enable DMA channel
```

3.2 Double-Buffer Operation

The DMAC transfers a block of data according to the parameters specified in the ADC and BLTC registers. If the INPUT DATA VALID (VLD) bit in the CNTL register is set, the DMAC reloads the ADC and BLTC registers from the ADR and BLTR registers, respectively, resets the VLD bit and transfers another block of data. This mode is similar to the auto-initialize operation except that there the VLD bit is not checked. If the VLD bit is not set when the terminal count is reached (BLTC = 0), the DMAC relinquishes the bus. This is an overrun condition and an interrupt is issued (if enabled) to the CPU.

```
.set adc, 0xfffff020            #for channel 1 use 0xfffff040
.set bltc, 0xfffff030           #for channel 1 use 0xfffff050
.set adr, 0xfffff024            #for channel 1 use 0xfffff044
.set bltr, 0xfffff034           #for channel 1 use 0xfffff054
.set mode, 0xfffff038           #for channel 1 use 0xfffff058
.set cntl, 0xfffff03C           #for channel 1 use 0xfffff05C
.set imsk, 0xfffff014

.text
movw $0,cntl                    #disable DMA channel
movd $block1_strt,adc           #write block start address to ADC register
movd $block1_cnt,bltc           #write byte count to BLTC register
movw $optype,mode               #enable double-buffer operation
movw $0x84,imsk                 #enable high priority interrupt on channel
                                #overrun. You can use MOVW 4, IMSK
                                #to enable
                                #low priority interrupt on channel overrun.
                                #These values are for channel 0.
                                #For channel 1 use respectively:
                                #MOVW $0xC0, IMSK for high priority int.
                                #MOVW $0x40, IMSK for low priority int.
                                #To disable interrupt on channel overrun reset
                                #bit 2 (for channel 0) or bit 6 (for channel 1)
                                #in the IMSK register
                                #In this case the DMAC will not
                                #issue interrupt upon channel overrun.
                                #It will set bit 2 (for channel 0) or bit 6
                                #(for channel 1) in the STAT register
                                #and relinquish the bus.
movd $block2_strt,adr           #write alternate block start address
                                #to ADR register
movd $block2_cnt,bltr           #write alternate byte count to BLTR
                                #register. It must not be of zero value.
movw $3,cntl                    #set INPUT DATA VALID bit in CNTL register
                                #and enable DMA channel.
                                #Notice that the DMA channel enabling could
                                #be done even before preparing the
                                #alternate block address and byte count,
                                #using MOVW 1,CNTL.
```

3.3 Auto-Initialize Operation

The DMAC transfers a block of data according the parameters specified in the ADC and BLTC registers. It then automatically reloads them from the ADR and BLTR registers, respectively, and transfers another block of data. This mode of operation differs from double-buffer operation because the ADC and BLTC registers automatically perform reloading regardless of the value of the INPUT DATA VALID bit in the CNTL register. The auto-initialize operation is useful for DRAM refresh.

```
.set adc, 0xfffff020          #for channel l use 0xfffff040
.set bltc, 0xfffff030         #for channel l use 0xfffff050
.set adr, 0xfffff024          #for channel l use 0xfffff044
.set bltr, 0xfffff034         #for channel l use 0xfffff054
.set mode, 0xfffff38          #for channel l use 0xfffff058
.set cntl, 0xfffff03C         #for channel l use 0xfffff05C
.set imsk, 0xfffff0l4

 .text
movw $0,cntl                  #disable DMA channel
movd $blockl_strt,adc         #write block start address to ADC register
movd $blockl_cnt,bltc         #write byte count to BLTC register
movd $block2_strt,adr         #write alternate block start address
                              #to ADR register
movd $block2_cnt,bltr         #write alternate byte count to BLTR
                              #register. It must not be of zero value.
movw $optype,mode             #enable auto-initialize operation
movw $0x8l,imsk               #enable high priority interrupt on terminal
                              #count. You can use MOVW l, IMSK to enable
                              #low priority interrupt on terminal count.
                              #These values are for channel 0.
                              #For channel l use respectively:
                              #MOVW $0x90, IMSK for high priority int.
                              #MOVW $0xl0, IMSK for low priority int.
                              #To disable interrupt on terminal count reset
                              #bit 0 (for channel 0) or bit 4 (for channel l)
                              #in the IMSK register
                              #In this case the DMAC won't
                              #issue interrupt upon transfer completion.
                              #It will set bit 0 (for channel 0) or bit 4
                              #(for channel l) in the STAT register
                              #and relinquish the bus.
movw $l,cntl                  #enable DMA channel
```


National Semiconductor Corporation
2900 Semiconductor Drive
P.O. Box 58090
Santa Clara, CA 95052-8090
Tel: 1(800) 272-9959
TWX: (910) 339-9240

National Semiconductor GmbH
Livry-Gargan-Str. 10
D-82256 Fürstenfeldbruck
Germany
Tel: (81-41) 35-0
Telex: 527649
Fax: (81-41) 35-1

National Semiconductor Japan Ltd.
Sumitomo Chemical
Engineering Center
Bldg. 7F
1-7-1, Nakase, Mihama-Ku
Chiba-City,
Ciba Prefecture 261
Tel: (043) 299-2300
Fax: (043) 299-2500

National Semiconductor Hong Kong Ltd.
13th Floor, Straight Block,
Ocean Centre, 5 Canton Rd.
Tsimshatsui, Kowloon
Hong Kong
Tel: (852) 2737-1600
Fax: (852) 2736-9960

National Semiconductores Do Brazil Ltda.
Rue Deputado Lacorda Franco
120-3A
Sao Paulo-SP
Brazil 05418-000
Tel: (55-11) 212-5066
Telex: 391-1131931 NSBR BR
Fax: (55-11) 212-1181

National Semiconductor (Australia) Pty, Ltd.
Building 16
Business Park Drive
Monash Business Park
Nottinghill, Melbourne
Victoria 3168 Australia
Tel: (3) 558-9999
Fax: (3) 558-9998