Handling of On-Chip DMAC Interrupts in the NS32CG160

National Semiconductor
Application Note 699
Zohar Peleg
December 1990

INTRODUCTION

The on-chip Interrupt Control Unit (ICU) of the NS32CG160 manages up to 15 levels of interrupt requests. Two of these levels, 6 and 14 may be requested internally by the on-chip DMAC, or externally by issuing the level number on the $\overline{\text{IR}}$0–3 input pins. The on-chip DMAC may request an interrupt due to 3 possible events on each of its 2 channels. Each event is associated with a status bit that may be read and cleared by the software. The status bits are sticky, and will keep issuing an interrupt request until they are cleared by the software.

After receiving a DMAC interrupt, the user must perform the following tasks:

1. Detect the cause of the interrupt.
2. Clear the corresponding status bit(s).
3. Handle the event and (if required) resume channel's operation.

This application note describes how to perform these three tasks.

BACKGROUND

There are 3 events that can cause an interrupt in each DMAC channel. These are:

1. TC—Terminal Count. This occurs when the channel's transfer is completed by a terminal count condition (BLTC register reaches zero).
2. EOT—End Of Transfer. This occurs when the transfer is externally terminated by the assertion of $\overline{\text{EOT}}$ signal.
3. OVR—Channel OverRun. This occurs in non-autoinitialize mode when the current transfer is completed, and the parameters for the next transfer are not ready (VLD bit in the CNTL register is zero).

Each of these events has a corresponding status bit in the DMAC status register called STAT *(Figure 1)*, and an enable bit in the DMAC interrupt mask register called IMSK *(Figure 2)*. The status bit is set when the event has occurred. An interrupt will be requested when both the status bit and corresponding enable bit are set. A status bit is cleared by writing "1" into it. Writing "0" does not change the bit.

The DMAC interrupt request may have a priority level of 6 or 14. This priority level is programmed by setting the DMAC Interrupt Priority bit (DIP) in the IMSK register (DIP = 0 means the priority level is 6, DIP = 1 means the priority level is 14).

When a channel is stopped due to EOT or OVR (when unmasked), the channel enable bit (CHEN) in the control register (CNTL) is cleared. *(Figure 3)*. Channel operation is resumed by setting the CHEN bit to "1". This can be done only after clearing or masking the pending bit(s).

Interrupts of priorities 6 and 14 may also be caused by external events. Though it is not recommended to do this, there are ways to detect when this occurs. Note that the more sources there are for an interrupt priority level, the more software checks are required to detect the cause of the interrupt.

	31 8	7							0	
STAT	Reserved	CHAC	OVR	EOT	TC	CHAC	OVR	EOT	TC	FFFFF014
		← Channel 1 →				← Channel 0 →				

FIGURE 1. DMAC Status Register

	31 8	7							0	
IMSK	Reserved	DIP	OVR	EOT	TC	0	OVR	EOT	TC	FFFFF010
			← Ch. 1 →				← Ch. 0 →			

FIGURE 2. DMAC Mask Register

	31 2	1	0	
CNTL-0	Reserved	VLD	CHEN	FFFFF03C

	31 2	1	0	
CNTL-1	Reserved	VLD	CHEN	FFFFF05C

FIGURE 3. DMAC Control Registers

DETECTION OF INTERRUPT SOURCE

This section lists the steps the user may follow to detect the different interrupt sources. This includes the six interrupt sources described above plus any external interrupt with the same priority level.

1. If nesting of higher level interrupts is desired, set I bit in PSR.
2. Read the DMAC's IMSK register. (Address h'FFFFF014)
3. Test the DIP bit. If it does not match the interrupt level in service, then it is an external interrupt request (this check is redundant if the DMAC's interrupt priority is not due to change, or if there is no external request assigned for that interrupt priority level).
4. Read the DMAC's STAT register. (Address h'FFFFF010)
5. Screen out all masked events. Retain a copy of all the unmasked pending bits in a general purpose register (the rest of this application note refers to this register as the "unmasked-pending word").
6. If the Unmasked-Pending word is "0", then it is an external vector. This check is redundant if there is no external request assigned for that level.
7. Write the Unmasked-Pending word back to the DMAC status register. This clears all the unmasked pending bits. Note that the masked bits should not be cleared here, since they may be handled by another piece of software which is not called by an interrupt.
8. Test all the relevant bits in the Unmasked-Pending word. Perform the service routines for each of the pending events (see the next section). Re-initialize the DMAC channel(s) according to the specific application and return from the interrupt.

Note that if a new pending bit is set in the status register after the DMAC's STAT register is read, then it will not be cleared by writing the Unmasked-Pending word back to the DMAC status register. The pending bit will remain set, and will issue another interrupt immediately after returning from the current interrupt.

SERVICE ROUTINES

This section describes the implications that should be considered when handling the interrupt events.

EOT (End of Transfer) Handling

The transfer is externally terminated by the assertion of $\overline{\text{EOT}}$ signal. That means that the external condition for the transfer's completion is met, and the channel is free for the next DMAC task. The handler may initialize the channel for its next task and resume its operation.

OVR (Channel OVerRun) Handling

Channel Overrun occurs in non-autoinitialize mode when the current transfer is completed, and the parameters for the next transfer are not ready (VLD bit in the CNTL register is zero). The channel is halted.

In single transfer operation, this indicates that the channel's task is over, and the channel is free for another task.

In double buffer operation, this indicates that the DMAC is stopped since the parameters for the next transfer are not ready yet, and the handler has to prepare the parameters, to set the VLD bit and to resume the channel's operation. It may also consider the data of the last transfer as ready, and start processing it.

TC (Terminal Count) Handling

TC condition occurs when the BLTC reaches zero. This event does not disable the channel regardless of the value of the corresponding bit in IMSK.

In single transfer operation the TC will always be accompanied by OVR, which will stop the channel's operation. In that case the TC is redundant, and can be masked if OVR is not masked. It indicates that the previous transfer is complete, the current one has started, and that it is time to go ahead and prepare the next buffer in advance, in order to avoid channel overrun. The TC will be accompained by OVR, so if OVR is found set, then the handling of TC may be included in the OVR handling, and the check of TC event may be skipped.

In autoinitialize operation the TC indicates completion of another transfer, and may be used to count number of transfers, and, if required, to stop the channel's operation when a desired number of transfers is reached.

In all cases TC indicates that the data of the last transfer is ready in its destination, and may be further processed.

IMPLEMENTATION IN ASSEMBLY

Following is an assembly code implementation of how to handle interrupts with priority level six. Comments inserted within the code describe how it works.

```
#
# Example of on-chip DMAC interrupts handling
#
.set dma_imsk, h'fffff014
.set dma_stat, h'fffff010
.set dma0_cntl, h'fffff03c
.set dmal_cntl, h'fffff05c

int_6:
             bispsrw $h'0800         # set I bit in PSR to enable
                                     # nesting of higher level interrupt
             movd    dma_imsk, r0    # Store dma interrupt mask.
             cbitb   $7, r0          # Test and clear interrupt priority.
             bfs     ext_int_6       # If set - this is external int.
             andd    dmb_stat, r0    # Record all unmasked pending events.
             cmpd    $0, r0          # Are there any?
             beq     ext_int_6
             movd    r0, dma_stat    # clear all unmasked pending events.

chk_ovr0:    tbitb   $2, r0          # Check if DMAC0 OVR event.
             bfc     chk_tc0
             jsr     dma0_ovr
chk_tc0:     tbitb   $0, ro          # Check if DMAC0 TC event.
             bfc     chk_eot0
             jsr     dma0_tc
chk_eot0:    tbitb   $1, r0          # Check if DMAC0 EOT event.
             bfc     chk_ovrl
             jsr     dma0_eot
chk_ovrl:    tbitb   $6, r0          # Check if DMACl OVR event.
             bfc     chk_tcl
             jsr     dmal_ovr
chk_tcl:     tbitb   $4, r0          # Check if DMACl TC event.
             bfc     chk_eotl
             jsr     dmal_tc
chk_eotl:    tbitb   $5, r0          # Check if DMAC EOT event.
             bfc     chk_out
             jsr     dmal_eot
chk_out:     reti
#
# If some of the events are guaranteed to be masked, the above check
# can be optimized by not testing the non relevant bits.
# If TC and OVR are both unmasked then OVR implies also TC,
# or no TC implies no OVR and that can be used to optimize
# the check as following: (example for ch-0 only)
#
# chk_tc0:   tbitb   $0, r0          # Check if DMAC0 TC event.
#            bfc     chk_eot0        # no TC → no OVR
#            tbitb   $2, r0          # if TC - is there also OVR?
#            bfc     jmp_tc0         # no → only TC.
#            jsr     dma0_ovr        # yes - handle OVR. Consider
#                                    # TC implied in OVR.
#            br      chk_eot0        # Skip TC handler.
# jmp_tc0:   jsr     dma0_tc
# chk_eot0:  tbitb   $1, r0          # Check if DMAC0 EOT event.
#            bfc     chk_tcl
#            jsr     dma0_eot
```

IMPLEMENTATION IN ASSEMBLY (Continued)

```
dma0_ovr:
# Body of service routine for DMAC0 over-run.
# Prepare the next buffer parameters.
# To resume channel operation use:

             movd    $3, dma0_ctl     # set channel enable bit and
                                      # input data valid bit of dma0

# If handling TC event is implied in handling OVR event, use
# the following to skip the TC check:

             movd    $chk_eot0, tos   # Change the return address in
                                      # the stuck to skip the TC check
             ret 0
dma0_tc:
#
# Body of service routine for DMAC0 Terminal Count.
# To resume channel operation use:

             movd    $1, dma0_ctl     # set channel enable bit of dma0
             ret     0

dma0_eot:
#
# Body of service routine for DMAC0 external End Of Transfer.
# Initialize channel for next DMAC task.
# To resume channel operation use:
             movd    $1, dma0_ctl     # set channel enable bit of dma0
             ret     0

dma1_ovr:
#
# Body of service routine for DMAC1 over-run.
# Prepare the next buffer parameters.
# To resume channel operation use:

             movd    $3, dma1_ctl     # set channel enable bit and
                                      # input data valid bit of dma1

# If handling TC event is implied in handling OVR event, use
# the following to skip the TC check:

             movd    $chk_eot1, tos   # Change the return address in
                                      # the stuck to skip the TC check
             ret 0
dma1_tc:
#
# Body of service routine for DMAC1 Terminal Court.
# to resume channel operation use:
             movd    $1, dma1_ctl     # set channel enable bit of dma1
             ret     0
dma1_eot:
#
# Body of service routine for DMAC1 external End Of Transfer.
# Initialize channel for next DMAC task.
# to resume channel operation use:
             movd    $1, dma1_ctl     # set channel enable bit of dma1
             ret     0
ext_int_6:
#
# Body of service routine for priority level 6 external interrupt.
#
             reti
```

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