SGS-THOMSON MICROELECTRONICS

APPLICATION NOTE

3-PHASE MOTOR DRIVE USING THE ST9 MULTI-FUNCTION TIMER AND DMA

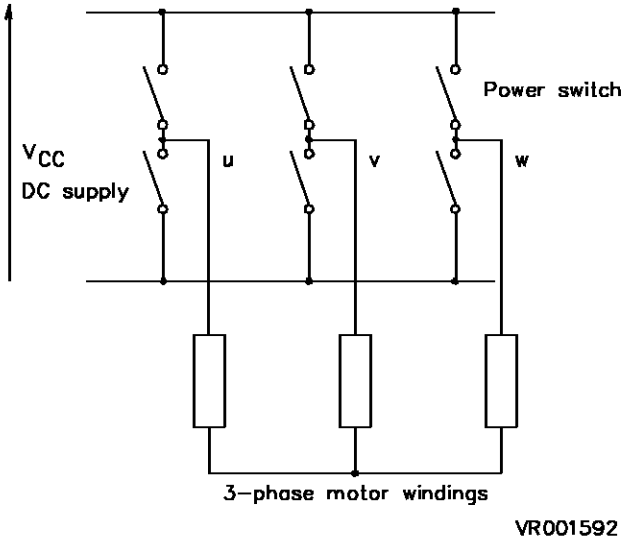
B. Saby/P. Guillemin

INTRODUCTION

3-phase induction motors are growing in popularity. Nevertheless, their maximum efficiency is achieved with a variable voltage and variable frequency drive through a bridge inverter (see Figure 1). The 6 power switches of the inverter require complex command sequences in order to approximate the 3 sine waves with a good accuracy. This is generally achieved by using a dedicated analog or digital IC, supervised by a standard microcontroller.

This application note presents an innovative single-chip solution taking advantage of the on-chip DMA of the ST9 family of microcontrollers. The DMA channel of the ST9 multi-function timers can be diverted to a 8-bit I/O port. This allows the ST9 to perform as pattern generator. Different patterns are used to drive the 6 power switches in order to generate various voltages and frequencies.

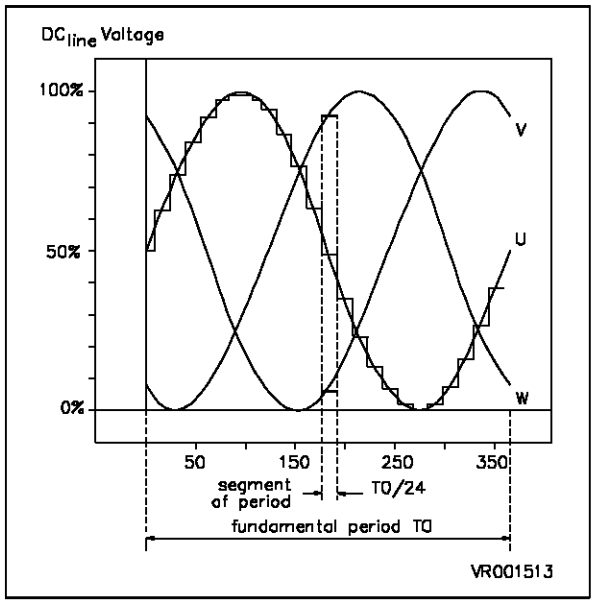
Figure 1. 3-Phase bridge inverter



Note : For a better understanding of this Application Note, it is strongly recommended to refer to the technical note “External DMA mode: I/O data transfer synchronized by Timer” for a detailed description of the DMA data transfer. Chapter 10 of “ST9 Technical Manual” provides all the details about the timer’s internal registers, interrupt and DMA flags. Chapter 9 of this manual provides the details about the DMA I/O port configuration.

PULSE WIDTH MODULATION (PWM) GENERATION

Figure 2. Staircase approximation of the 3 phase sine waves



The elementary switching period is subdivided into elementary time slots during which all 6 switches of the inverter are in a given state (“on” or “off”). The state of the switches can be represented with 6 bits stored in a byte: 1 = “on”, 0 = “off”. Hence, a switching period can be represented with a pattern, i.e. a sequence of bytes corresponding to the different time slots (42 time slots in our example).

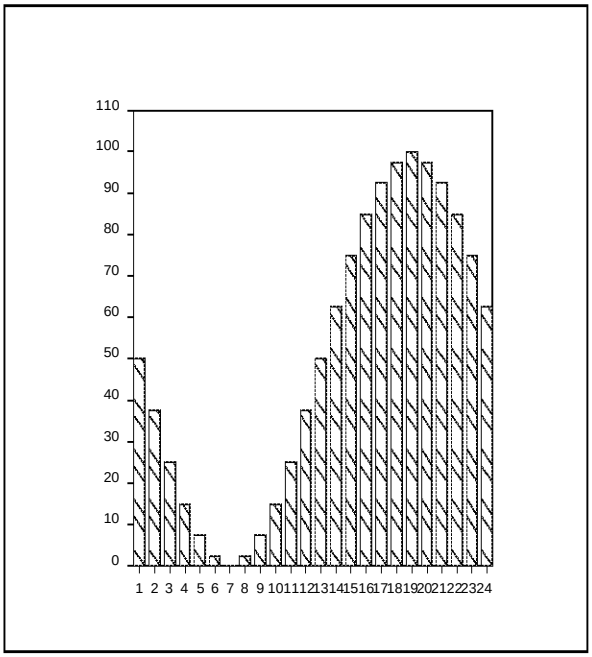
Each segment of the 3-phase sine waves can be represented with a defined average voltage during this segment on each of the 3 phases, and therefore with a given pattern. There must be as many different patterns as period segments (24 segments in our example).

Note: the number of period segments must be a multiple of 3, as the 3 sine waves present the same amplitude with a phase shift of one third of a period between each other.

The 3 sine waves are digitalized, using a “staircase approximation” (see Figure 2).

In the present example, there are 24 period segments (24 “stairs”) for one period of the approximated sine wave. Each segment corresponds to an average voltage value during the segment duration (see Figure 3): this voltage is generated by the power switches of each half-bridge of the inverter, using a PWM technique. The duration of each segment is a multiple of the elementary PWM switching period. Therefore, it is very easy to set the period of the generated sine waves by modifying the segment duration.

Figure 3. Period Segments and Average Voltage (in % of V_{CC})



During a period segment, this pattern can be repeated several times; then the period segments can have a variable duration, a multiple of the elementary switching period. Thus it is possible to generate the same sine waves (i.e. same voltage) with different frequencies (see Figure 4 and Figure 5).

Figure 4. One pattern repetition per segment

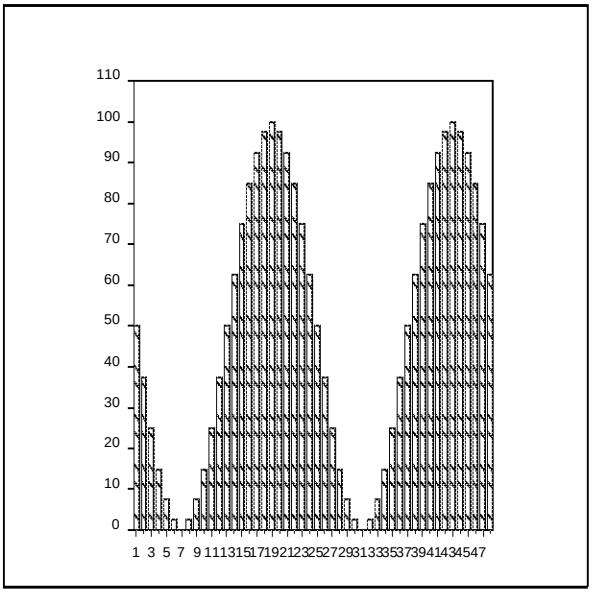


Figure 5. Two patterns per segment

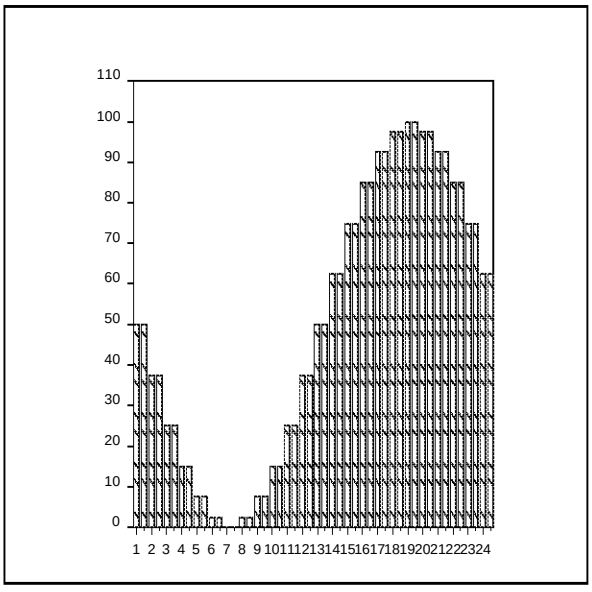


Table 1. Sine wave subdivision summary

time slot duration (same state for all 6switches): 4.75 µs

number of time slots per elementary switching period: 42 time slots

elementary switching period (one pattern): 199.5 µs

period segment duration (average voltage constant on the 3 phases:

n patterns (n = 1,2,3...), i.e. n x 199.5 µs

sine wave period: m = 24 period segments = 24 x n x 199.5 µs = n x 4.788 ms

sine wave frequency: $\frac{1}{n \times 4.788}$ kHz

for n = 1: the frequency is F = 208.86 Hz

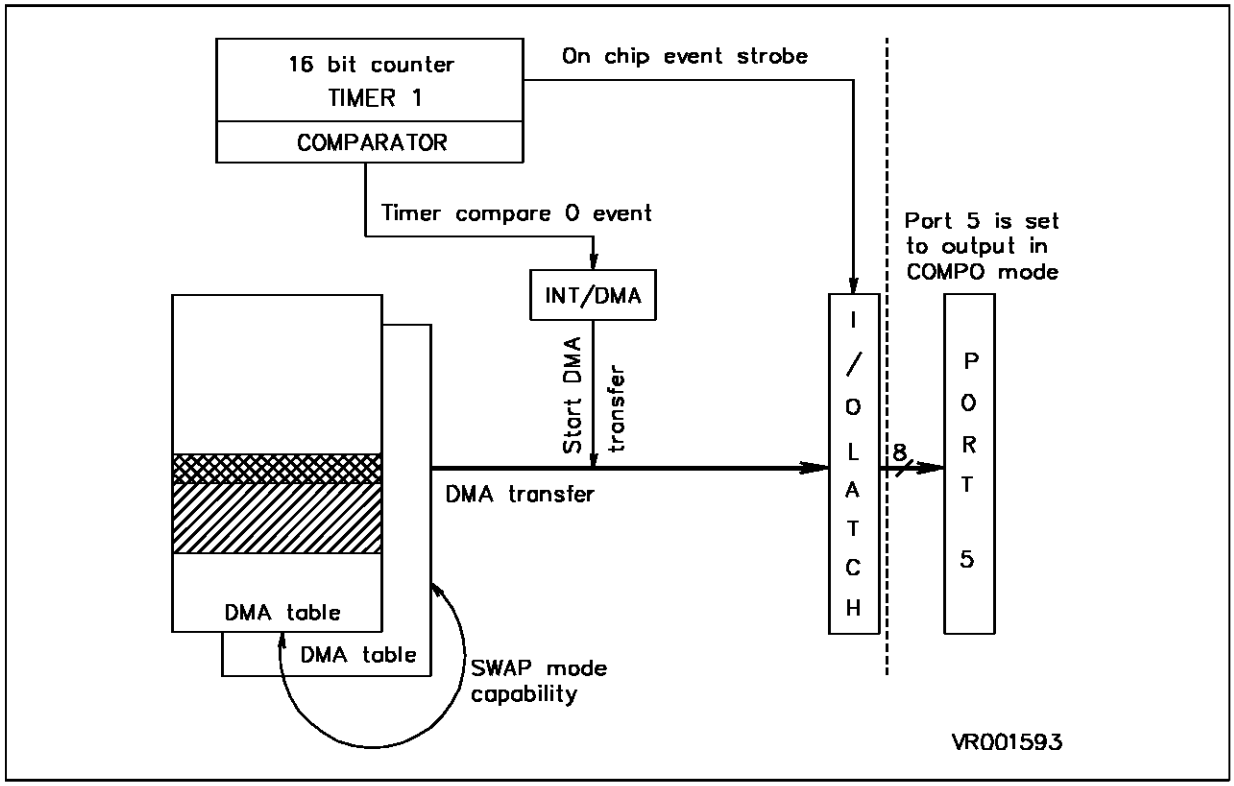
for n = 20: the frequency is F = 10.44 Hz

ST9 DIRECT MEMORY ACCESS (DMA)

One of the unique features of the ST9 family of microcontrollers consists of a DMA capability between its memory and its on-chip peripherals, including the Multifunction Timers. On top of that, one 8-bit I/O port can be coupled with one timer's DMA channel for fast data transfers between memory and the I/O port with minimum CPU overhead. The data transfers are scheduled by the timer (see Figure 6).

This DMA feature allows the transfer of a complete pattern of bytes without any software intervention. Two registers are used by the DMA logic: the DMA address pointer holds the address of the pattern stored in the program memory of the ST9 and is incremented after each DMA transfer, thus pointing to the next byte of the pattern; the DMA counter holds the total number of bytes in a pattern (here: 42 bytes) and is decremented after each DMA transfer.

Figure 6. DMA output on I/O port with timer compare mode



When the DMA counter is decremented down to zero, an "End of Block" interrupt request is generated. During the interrupt servicing routine, the DMA address pointer and DMA counter should be reloaded with new values corresponding to the next pattern (or the same pattern, if it is repeated several times). As this operation takes some time, it can create an undesired delay in the process of data transfer to the I/O port: in our application example, a DMA cycle should occur every time slot, i.e. every 4.75 µs.

In order to achieve this data throughput without imposing a stringent interrupt response time for the End of Block interrupt, an extra feature of the DMA channel of the timer unit is used: the swap mode. When operating in swap mode, the DMA channel uses 2 address pointers and 2 counters in the following way: while a pair of pointer/counter is used for DMA transfers, the other pair can be loaded with the appropriate values corresponding to the next pattern to be transfered. Once the last byte of the current pattern is transfered (DMA counter decremented down to zero), an interrupt request is generated, as in the regular mode; in addition, the DMA channel automatically switches to the other pair of DMA pointer/counter and therefore is immediately ready to start a new sequence of DMA transfers.

During the interrupt service routine, the software reloads the “old” pair of pointer/counter with the next pattern address and length. Using the swap mode provides a major advantage in this application because the DMA data transfers are never interrupted: this is essential to achieve a good accuracy in the 3 sine waves reconstitution.

The major advantage of using the DMA is that the ST9 microcontroller is available for other tasks of control or calculation, as the DMA operation does not require any software intervention except during the End of Block interrupt, i.e. every 200µs.

ST9 SOFTWARE

In order to generate the 3-phase sine waves, the following software routines are needed:

- a routine to initialize the timer and the I/O port in the desired configuration. Note: on the ST9030 used in the present example, the DMA I/O operation is available on Port 5 coupled with the Multifunction timer 1 (cf. the technical note “External DMA mode: I/O data transfer synchronized by Timer”).

- a routine to start the timer and the DMA operation when the 3-phase motor must be started.

- a routine to stop the timer and DMA when the motor must be stopped.

- the DMA End of Block interrupt routine where the DMA pointer and counter are reloaded with new values.

Two kinds of information are also needed in order to generate 3-phase sine waves at a given voltage and frequency:

Information for defining the voltage:

- the number of period segments per sine wave period (24 segments in our example). Each segment requires a specific pattern of 42 bytes stored in the ST9030’s program memory (ROM memory).

Table 2. Descriptor structure

Address:	
N	m = segments per period
N + 1 N + 2	Address of pattern #1 (2 bytes)
...	
N + 1*m	Address of pattern #m (2 bytes)

- the addresses of the different patterns. There are 24 patterns (one per period segment) for a given voltage of the sine waves, i.e. 24 x 42 = 1008 bytes.

All this information is stored in a small ROM table named a descriptor. There is one descriptor table for each given voltage/frequency operation of the motor. The descriptor structure is as follows (see Figure 7):

- the first byte is the number of period segments for a complete sine wave period: m = 24 in this example.

- the following bytes are the addresses of the patterns (16-bit words), beginning with the address of pattern #1 and finishing with the address of pattern #m.

INFORMATION FOR DEFINING THE FREQUENCY:

- the number of repetitions of an elementary switching period (represented by one pattern) during a period segment.

- the timer duration defining the DMA frequency.

This information is stored in a table in ROM called FREQ_TABLE.

When operating the motor at a given voltage and frequency, all the needed parameters are loaded from the corresponding descriptor and from the frequency table. The following registers are used in the software example:

curr_speed: holds the address N (16-bit) of the current descriptor.

patt_count: holds the pattern repetition number n (first byte of the descriptor) and is decremented upon each repetition of the current pattern.

patt_nb: holds the total period segments number m (second byte of the pattern) and is decremented upon each new segment.

patt_point: points to the descriptor position where the address of the current pattern (16-bit) is stored. It is incremented by two upon each new segment in order to point to the next pattern address.

next_cmp_t1: holds the next value for the comparizon event on Timer 1; fixes the DMA frequency and therefore associated with the repetition number, the frequency of the three-phase sine waves.

rep_nb: holds the pattern repeptition number. This is an image of patt_count.

dma_buff0: first DMA address pointer (16-bit). When starting the motor, this register pair is loaded with the address of the first pattern (pointed by patt_point). It is incremented upon each DMA transfer.

dma_count0: first DMA counter (16-bit). When starting a new pattern with dma_buff0, this register pair is loaded with the number of bytes in the pattern (42 bytes in this example).

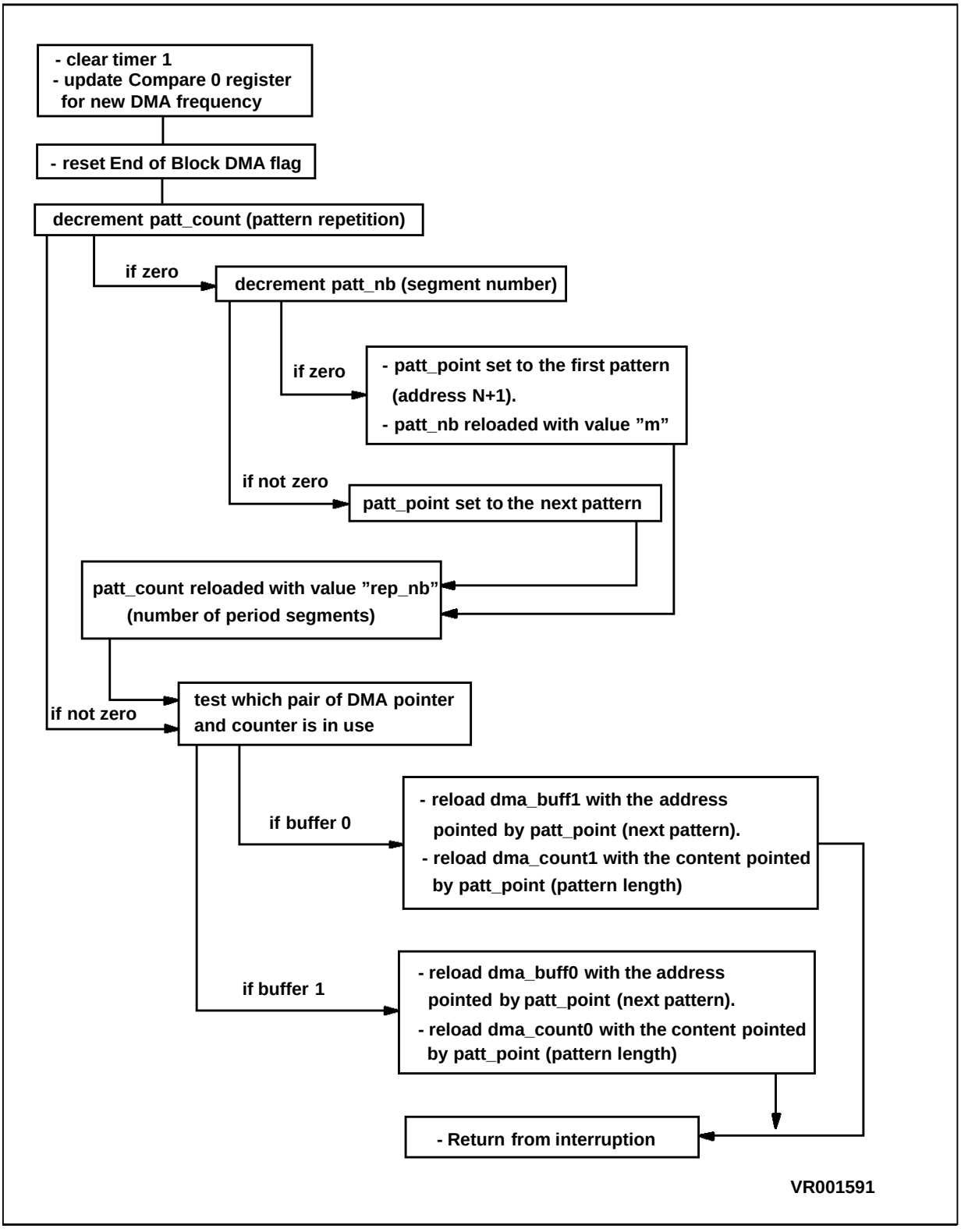
dma_buff1: second DMA address pointer.

dma_count1: second DMA counter.

The timer is operated in “count up” mode. The desired value for the time-slots between 2 consecutive DMA transfers (4.75µs) is loaded into the COMPARE 0 register of the timer; a “clear on COMPARE 0” is also enabled, so it will restart from the beginning upon each successful COMPARE 0. The DMA channel is set to operate in conjunction with the COMPARE 0 event.

When transfering data, the DMA channel toggles between dma_buff0/dma_count0 and dma_buff1/dma_count1, as indicated by bit 2 of the the DCPR register of the timer. When servicing the DMA End of Block interrupt, the routine must check which of the 2 DMA pointers is in use and reload the other one, as detailed in Figure 8, and then reset the interrupt flag and return to the main program.

Figure 7. DMA End of Block interrupt routine



SUMMARY

Using the ST9 DMA on I/O port to drive a 3-phase induction motor presents several advantages compared to conventional solutions. First, the ST9 is able to generate the 6 command sequences for the bridge inverter without any additional hardware or dedicated circuits, just by using its standard features.

In addition, the patterns used to set up the command sequences are prepared by the user. This gives the possibility to generate sophisticated command sequences for any particular purpose, as, for instance, to reduce the noise by dephasing the switching of the currents in the 3 phases.

First tests have shown that the DMA operation, including the interrupt routines, accounts for approximatively 35 % to 40 % of the total available CPU time of the ST9 when operating at its maximum speed (12 MHz internal clock). Conventionnal microcontrollers should spend almost all their time at transfering all the bytes of the patterns to the I/O port at a rate of one byte every 4.75 µs. The ST9 microcontrollers can do this using their DMA channel, with 60 % or more of their processing power available for other tasks such as speed regulation, temperature supervision, keyboard input. This solution is cost effective, even when compared with a low end microcontroller plus a dedicated circuit.

In our example, a complete set of patterns (for one given voltage) occupies approximatively 1 K bytes in the ROM memory. This is low, especially when considering that standard ST9 ROM sizes range from 8 K to 32 K bytes.

As a conclusion, it can be said that this cost effective solution provides new technical possibilities in the 3-phase motor drive, thanks to the DMA feature and the flexibility of the ST9 microcontroller family.

Bibliography:

- Versatile and cost effective induction motor drive with three phase digital generation, B.MAURICE/JM.BOURGEOIS/B.SABY, PCIM 1991, Nürnberg.

- External DMA mode: I/O data transfer synchronized by Timer, P.GUILLEMIN, technical note, SGS-THOMSON Microelectronics 1990.

- ST9 family 8/16 bit MCU Technical Manual, SGS-THOMSON Microelectronics 1990.

Annex A. Software example

```
     .title “Three-phase motor control with DMA I/O”
     .list

;The timer 1 is programmed in COMPARE0 DMA channel EXT mode

;******************************** *
;*INTERRUPT VECTOR ADDRESSES      *
;******************************** *

CORE_IT_VECT     :=    00h         ; Core interrupt vectors
T1_IT_VECT       :=    08h         ; Timer 1 interrupt vectors

COMP_IT_VECT     :=    6           ; COMPARE event interrupt address
T1_LEVEL         :=    1           ; Timer 1 priority level

;****************************************** *
;    Register working groups definition       *
;****************************************** *

     BK_0 :=     0 * 2             ;working group 0
     BK_1 :=     1 * 2             ;working group 1
     BK_2 :=     2 * 2             ;working group 2
     BK_3 :=     3 * 2             ;working group 3
     BK_4 :=     4 * 2             ;working group 4
     BK_5 :=     5 * 2             ;working group 5
     BK_6 :=     6 * 2             ;working group 6
     BK_7 :=     7 * 2             ;working group 7
     BK_8 :=     8 * 2             ;working group 8
     BK_9 :=     9 * 2             ;working group 9
     BK_A :=     10 * 2            ;working group A
     BK_B :=     11 * 2            ;working group B
     BK_C :=     12 * 2            ;working group C
     BK_D :=     13 * 2            ;working group D
     BK_E :=     14 * 2            ;working group E
     BK_F :=     15 * 2            ;working group F

;******************************** *
;    Stack definition              *
;******************************** *
     SSTACK      :=    0E0h        ;system stack: group C and D
     USTACK      :=    0C0h        ;user stack: group B
```

```
;****************************************
;    DMA and pattern control registers   *
;****************************************

;    DMA pointers and counters (swap mode)
;    ************************************

     T1_DMA      :=    BK_8              ;Timer 1 DMA group
     LG_DMA      =     42                ;length of DMA
     AD_DMA      :=    082H              ;DMA pointer reg. nb
     CT_DMA      :=    08AH              ;DMA counter reg. nb

     dma_buff0   =     rr2               ;buffer 0 pointer
     DMA_buff0   :==   RR#AD_DMA

     dma_count0  =     rr10              ;buffer 0 counter
     DMA_count0  :==   RR#CT_DMA

     dma_buff1   =     rr6               ;buffer 1 pointer
     DMA_buff1   :==   RR#AD_DMA+4

     dma_count1  =     rr14              ;buffer 1 counter
     DMA_count1  :==   RR#CT_DMA+4

;    Patterns pointers and counters
;    ******************************
     Speed       :=    BK_A              ;working group 10

     CURR_SPEED  :==   RR#0A0h           ;speed descriptor
     curr_speed  =     rr0               ;speed descriptor

     PATT_POINT  :==   RR#0A4h           ;pattern pointer in the list
     patt_point  =     rr4               ;pattern pointer in the list

     REP_nb      :==   R#0A7h            ;pattern repetition number
     rep_nb      =     r7

     PAT_COUNT   ==    R#0A8h            ;current pattern repetition
     patt_count  =     r8                ;current pattern repetition

     pATT_NB     :==   R#0A9h            ;number of patterns for 1 period
     patt_nb     =     r9                ;number of patterns for 1 period
```

```
     NEXT_CMP_T1 :==   R#0AEh      ; next value of Timer 1
     next_cmp_t1 =     r14         ; Compare 0 register

;************************
;*START of PROGRAM       *
;************************

START_OF_CODE    :=    20h         ; start address program

;****************************************** *
;*Declaration of the interrupt vectors table   *
;****************************************** *

     .text                         ; start of program

     .org        CORE_IT_VECT      ; Core interrupt vector
                                   ; *********************
     .word       RESET_START       ; power on interrupt vector

     .org T1_IT_VECT               ; Timer 1 interrupt vectors
                                   ; ************************
     .org        T1_IT_VECT + 6    ; unused addressses
     .word       COMPARE0          ; Timer 1 compare 0 interrupt

;************************
;    Timer 0 int vectors *
;************************

     ;timer 0 interrupt vector
     .word       R_UDFLW_T0        ;underflow timer 0

;************************
;*Start of main module   *
;************************
     .org        START_OF_CODE     ;start of code
```

```
RESET_START::
     clr  P5DR                    ; port 5: all zeros
     ld   MODER,#11100000b        ; CLOCK MODE REGISTER
                                  ; internal stack
                                  ; no clock prescaling
     ld   CICR,#10001111b         ; CENTRAL INTERRUPT CONTROL REGISTER
                                  ; priority level = 7
                                  ; Nested Arbitration mode
                                  ; disable interrupt
                                  ; enable counters
                                  ; At reset, Global Counter Enable bit is
                                  ; active.
     spm                          ; use program memory

     ld   SSPLR,#SSTACK           ; load system stack pointer
     ld   USPLR,#USTACK           ; load user stack pointer

     call TIMER_1                 ; Timer 1 initialization in DMA mode
     call INIT_IO                 ; Port 5 initialization in DMA mode
     ei                           ; enable all interrupts

;***********************
;*    MAIN PROGRAM     *
;***********************

     loop {

; During the main loop, the motor can be started by loading the address of
; the appropriate descriptor into “curr_speed” register pair and calling
; the “START_T1” routine.
; It can be stopped by calling the “STOP_T1” routine.

          }
```

```
;*****************************************************************
;*               initialize TIMER 1                              *
;*****************************************************************


proc TIMER_1     {

     srp  #BK_F                    ; select paged working reg.
     spp  #T1D_PG                  ; select timer 1 reg. page
     ld   t_tmr,#oe0               ; Disable output B
                                   ; Enable out A
                                   ; Internal clock
                                   ; Countinuous mode
     ld   t_tcr,#( ccl | ccmp0 | udc)          ; clear counter
                                               ; count up
                                               ; clear on compare 0
     clr  t_icr                                ; No action on input pins
     clr  t_prsr                               ; No prescaling
     ld   t_oacr,#( ou_nop | c1_nop | c0_tog ) ; Toggle OUTPUT0 on
                                               ; Compare 0
     ld   t_obcr,#( c0_nop | c1_nop | ou_nop ) ; No action on OUTPUT1
     clr  t_flagr

     spp  #T1C_PG                  ; Timer 1 Control page reg.
     ld   t1_dcpr,#CT_DMA          ; DMA count. reg. base addres
     ld   t1_dapr,#AD_DMA          ; DMA add. reg. base addres

     ld   t1_ivr,#T1_IT_VECT                   ; load interrupt vect.
     ld   t1_idcr,#( T1_LEVEL | dctd | swen)   ; DMA compare,
                                               ; swap enable

     spp  #T1D_PG                              ; Timer 1 Data page
     ld   t_idmr,#( gtien | cm0i | cm0d )      ; Compare 0 INT and DMA
     ldw  t_reg0r,#0                           ; reg 0
}
;*************************************************************************
;*        TIMER 1 COMPARE 0 INTERRUPT ROUTINE                            *
;*        DMA Interrupt End of block                                     *
;*************************************************************************
COMPARE0:
 begin    [PPR,RP0R]   {          ; save page pointer
                                  ; save register pointer
```

```
    srp  #Speed                    ; speed control working regs.
    spp  #T1D_PG                   ; timer 1 data page
    or   T_TCR,#ccl                ; Clear counter
    ld   T_CMP0LR,next_cmp_t1      ; update compare 0 register
                                   ; for new freq. in slope
                                   ; generation
    spp  #T1C_PG                   ; timer 1 control page
    and  T1_IDCR,#~(cme)           ; reset EOB DMA condition
    dec  patt_count                ; decrement repetition counter
    if   [SETZ] {                              ; when finished,

         dec    patt_nb                        ; see next pattern.
         if     [ SETZ ]    {                  ; if end of table
                ldw   patt_point,curr_speed    ; restart from the beginning
                ld    patt_nb,(patt_point)     ; read number

         } else {
                incw  patt_point               ; skip first byte
         }
         incw   patt_point                     ; see next pattern.
         ld     patt_count,rep_nb              ; reload pattern rep. count
    }
    tm   T1_DCPR,#00000100b             ; test if buffer 0 in use
    if   [SETZ] {                       ; if buffer 0,...
         ldw    DMA_buff1,(patt_point)  ; load pattern address in buffer 1
         ldw    DMA_count1,#LG_DMA      ; load count
         } else {                       ; if buffer 1,...
         ldw    DMA_buff0,(patt_point)  ; load pattern address in buffer0
         ldw    DMA_count0,#LG_DMA      ; load count
    }
}   ; ..... end begin

    iret                           ; return from interrupt
```

```
;*************************************************************************
;               I/O port initialization                                  *
;*************************************************************************
;    Set port 5 to I/O DMA mode
;    *************************

proc INIT_IO     [ PPR,RP0R ]      {

     srp  #BK_F                    ; select paged working register
     spp  #P5C_PG                  ; Port 5 control register page
                                   ; Port 5 in DMA mode
                                   ; Port 5 Handshake disabled
                                   ; Dma on Compare 0 channel
;               76543210
     ld   p5c0r,#00000000b
     ld   p5c1r,#11111111b
     ld   p5c2r,#00000000b

     ld   hdc5r,#( hsdis | den | ddw | dcm0 )

}
;***********************
;*Stop Timer 1 routine *
;***********************

proc STOP_T1 [PPR]{

     spp  #T1D_PG                  ;Timer 1 data page

     clr  T_TCR                    ;stop timer
     clr  T_FLAGR                  ;clear pending interrupts
     clr  P5DR                     ;port 5 = 0 to stop the motor
}
```

```
;****************************
;*Start Timer 1 routine     *
;****************************

proc START_T1 [RP0R,PPR]{

     srp  #BK_F
     spp  #T1C_PG                  ;timer 1 control page

     ld   t1_dcpr,#CT_DMA          ; dma counter register base addres
     ld   t1_dapr,#AD_DMA          ; dma address register base addres

     srp  #Speed                   ;motor control registers

     ldw  patt_point,curr_speed    ; start from the beginning of
                                   ; the descriptor
     incw patt_point               ; skip first byte
     ld   patt_nb,(patt_point)     ; read number of patterns
     incw patt_point               ; see first pattern.
     ldw  dma_buff0,(patt_point)   ; load pattern address into first
                                   ; DMA buffer
     ldw  dma_count0,#LG_DMA ; load byte count (42 bytes)

     ld   patt_count,(curr_speed)  ; load pattern repetition count
     dec  patt_count               ; decrement repetition counter
     if   [SETZ]{                        ; if only one repetition per
                                         ; segment,

          dec    patt_nb                 ; see next pattern.
          incw   patt_point              ; point to next pattern.
          incw   patt_point
          ld     patt_count,(curr_speed) ;reload pattern repetition
                                         ; counter

}
```

```
     ldw  dma_buff1,(patt_point)   ; load the second pattern address
                                   ; into buffer 1
     ldw  dma_count1,#LG_DMA ;  load byte count

     spp  #T1D_PG                  ;Timer 1 data page

     clr  T_FLAGR                  ;clear pending interrupts

     or   T_TCR,#( cen | ccl | ccmp0 | udc)    ; counter enable bit
                                               ; clear counter
                                               ; count up
                                               ; clear on compare 0
}

;***********************************************************************
;*                Procedure Set_freq
;* Input:        - offset_fil contains frequency location in FREQ_TABLE
;* Output:       - rep_nb and next_cmpt1l updated from FREQUENCY TABLE
;* Modified:     - rep_nb, PPR, next_cmp_t1l
;***********************************************************************
proc Set_freq    {
     beginw      [ curr_slope ]    {
                 ldw   curr_slope,#FREQ_TABLE  ; Pointer on FREQ. table
                 clr   offset_fih
                 addw  curr_slope,offset_fi
                 addw  curr_slope,offset_fi
                 ld    rep_nb,(curr_slope)+    ; load number of repetition
                 ld    next_cmp_t1,(curr_slope); for new Timer 1 compare 0
                                               ;reg.
     }
}
```

Annex B. Pattern Definition Example

```
;****************************************** *
;*New pattern: 100 % of Vcc: sine centered    *
;****************************************** *
.global  PATT_10B
 PATT_10B:
     .byte       24                ;24 different patterns for one period
     .word       P_10B_A, P_10B_B, P_10B_C, P_10B_D
     .word       P_10B_E, P_10B_F, P_10B_G, P_10B_H
     .word       P_10B_I, P_10B_J, P_10B_K, P_10B_L
     .word       P_10B_M, P_10B_N, P_10B_O, P_10B_P
     .word       P_10B_Q, P_10B_R, P_10B_S, P_10B_T
     .word       P_10B_U, P_10B_V, P_10B_W, P_10B_X

;1.0 1
;******
 P_10B_A:
     .byte       000101b
     .byte       100101b
     .byte       100101b
          .
          .
          .
     .byte       100101b
     .byte       100101b
     .byte       100101b
     .byte       000101b

;1.0 2
;******
 P_10B_B:
     .byte       010101b
     .byte       000101b
     .byte       100101b
          .
          .
          .
```

```
;1.0 24
;******
 P_10B_X:
     .byte       000101b
     .byte       100101b
          .
          .
          .
```

Annex C. Frequency Table Definition

```
;****************************************
;* Define global and external references *
;****************************************

.global          FREQ_TABLE

;************************************************************************
;*                       FREQUENCIES TABLE
;*
;* This table give all the possible frequencies available according to the
;* number of pattern repetition and to the Timer 1 Compare 0 value.
;* Take care to the Timer 1 Compare value: according to the Timer
;* programmation
;* the Compare 0 register must be loaded with 0 (instead of 1) to reach the
;* minimal counting value (= 250ns). So the Timer duration given in the
;* following table is ( DMA frequency / Timer resolution ) - 1.
;* This table must accessed giving the frequency location within the table,
;* for location 6 correspond to 153 Hz with a timer Compare 0 value egal to
;* 3.25 s
;************************************************************************

FREQ_TABLE:                          ; pattern repetition + TIMER 1 duration
     .byte  1, (4750/250) - 1        ; 208   Hz              4.75 s
     .byte  1, (5000/250) - 1        ; 198   Hz from 208 Hz: 5.00 s
     .byte  1, (5250/250) - 1        ; 189   Hz   “  208 Hz: 5.25 s
     .byte  1, (5500/250) - 1        ; 180   Hz   “  208 Hz: 5.50 s
     .byte  1, (5750/250) - 1        ; 172   Hz   “  208 Hz: 5.75 s
     .byte  1, (6000/250) - 1        ; 165   Hz   “  208 Hz: 6.00 s
     .byte  1, (6250/250) - 1        ; 159   Hz   “  208 Hz: 6.25 s
     .byte  2, (3250/250) - 1        ; 153   Hz   “  104 Hz: 3.25 s
     .byte  2, (3500/250) - 1        ; 142   Hz   “  104 Hz: 3.50 s
     .byte  2, (3750/250) - 1        ; 132   Hz   “  104 Hz: 3.75 s
     .byte  2, (4000/250) - 1        ; 124   Hz   “  104 Hz: 4.00 s
     .byte  2, (4250/250) - 1        ; 117   Hz   “  104 Hz: 4.25 s
     .byte  2, (4500/250) - 1        ; 110   Hz   “  104 Hz: 4.50 s
     .byte  2, (4750/250) - 1        ; 104   Hz              4.75 s
     .byte  2, (5000/250) - 1        ;  99   Hz   “  104 Hz: 5.00 s
     .byte  2, (5250/250) - 1        ;  94   Hz   “  104 Hz: 5.25 s
     .byte  2, (5500/250) - 1        ;  90   Hz   “  104 Hz: 5.50 s
     .byte  2, (5750/250) - 1        ;  86   Hz   “  104 Hz: 5.75 s
     .byte  3, (4000/250) - 1        ;  82   Hz   “   70 Hz: 4.00 s
     .byte  3, (4250/250) - 1        ;  78   Hz   “   70 Hz: 4.25 s
```

```
.byte  3, (4500/250) - 1      ;  73   Hz   “   70 Hz: 4.50 s
.byte  3, (4750/250) - 1      ;  70   Hz             4.75 s(21)
.byte  3, (5000/250) - 1      ;  66   Hz   “   70 Hz: 5.00 s
.byte  3, (5250/250) - 1      ;  63   Hz   “   70 Hz: 5.25 s
.byte  4, (4000/250) - 1      ;  62   Hz   “   52 Hz: 4.00 s
.byte  4, (4250/250) - 1      ;  58   Hz   “   52 Hz: 4.25 s
.byte  4, (4500/250) - 1      ;  55   Hz   “   52 Hz: 4.50 s
                .
                .
                .
.byte  11, (4750/250) - 1     ;  19   Hz        4.75 s      (42)
.byte  12, (4750/250) - 1     ;  17.4 Hz        4.75 s      (43)
.byte  13, (4750/250) - 1     ;  16   Hz        4.75 s      (44)
.byte  14, (4750/250) - 1     ;  15   Hz        4.75 s      (45)
.byte  15, (4750/250) - 1     ;  14   Hz        4.75 s      (46)
.byte  16, (4750/250) - 1     ;  13   Hz        4.75 s      (47)
.byte  17, (4750/250) - 1     ;  12   Hz        4.75 s      (48)
.byte  18, (4750/250) - 1     ;  11.6 Hz        4.75 s      (49)
.byte  19, (4750/250) - 1     ;  11   Hz        4.75 s      (50)
.byte  20, (4750/250) - 1     ;  10.4 Hz        4.75 s      (51)
.byte  22, (4750/250) - 1     ;   9   Hz        4.75 s      (52)
.byte  26, (4750/250) - 1     ;   8   Hz        4.75 s      (53)
.byte  30, (4750/250) - 1     ;   7   Hz        4.75 s      (54)
.byte  35, (4750/250) - 1     ;   6   Hz        4.75 s      (55)
.byte  42, (4750/250) - 1     ;   5   Hz        4.75 s      (56)
.byte  52, (4750/250) - 1     ;   4   Hz        4.75 s      (57)
.byte  70, (4750/250) - 1     ;   3   Hz        4.75 s      (58)
.byte 104, (4750/250) - 1     ;   2   Hz        4.75 s      (59)
.byte 209, (4750/250) - 1     ;   1   Hz        4.75 s      (60)
```

Annex D. Glossary

Bridge inverter	Power converter generating 3-phase sine waves from a DC power supply (see Figure 1, Page 3).
COMPARE 0	Register pair of the ST9 timer to which the timer's contents are compared upon each increment; a successful compare event triggers the DMA transfer to the I/O port.
Descriptor	Small table containing all the parameters for a given voltage and frequency of the 3-phase sine waves (see Figure 7, page 9).
DMA	Direct Memory Access, a technique used in microprocessor systems where the peripherals can transfer data to and from the memory without requiring any software intervention.
DMA counter	Register pair containing the number of remaining bytes to be transfered by the DMA channel.
DMA pointer	Register pair containing the address of the next byte to be transfered by the DMA channel.
Elementary switching period	Repetition period of the PWM.
End of Block interrupt	Interrupt request generated by the DMA control logic once a complete pattern of bytes has been transfered.
I/O port	8-bit parallel port of the ST9 microcontroller; one of these I/O ports can be coupled to the DMA channel of a timer.

Pattern	A sequence of bytes describing an elementary switching period.
Period segment	A part of the sine wave period during which the average voltage is kept constant on all 3 phases.
PWM	Pulse Width Modulation, a technique used to generate an average voltage by switching the bridge output alternatively to Vcc and ground.
ROM	Read Only Memory, non-volatile memory of the ST9 microcontroller into which the ST9 program and the patterns are stored.
Swap mode	An extra feature of the ST9 DMA channel that allows continuous data transfers.
Time slot	Time interval between two DMA transfers during which the 6 switches of the inverter remain in a constant state.

THE SOFTWARE INCLUDED IN THIS NOTE IS FOR GUIDANCE ONLY. SGS-THOMSON SHALL NOT BE HELD LIABLE FOR ANY DIRECT, INDIRECT OR CONSEQUENTIAL DAMAGES WITH RESPECT TO ANY CLAIMS ARISING FROM