SGS-THOMSON MICROELECTRONICS

APPLICATION NOTE

FREQUENCY DOUBLER DEMONSTRATION SYSTEM

Myriam Chabaud and Alan Dunworth

INTRODUCTION

This Application Note is intended to provide the interested designer of ST9 system applications with a further insight into methods of exploiting the powerful capabilities offered by the ST9 chip in conjunction with the ST9 Technical Manual (Ref. 3). For this purpose we present full software and broad hardware details of a complete system application (Appendices A and B).

BASIC SPECIFICATION OF A FREQUENCY DOUBLER SYSTEM.

An analogue signal (speech signal) is sampled at a fixed rate and digitized sample values are stored in internal RAM. After a delay period, equal to the period of the lowest frequency component in the input signal, the samples are read from RAM at twice the input sampling frequency and converted to analogue form by using Pulse Width modulation techniques coupled with external filtering.

This process is illustrated by the waveforms shown in Figure 1 which show the nature and timing of the output signal resulting from sampling and frequency doubling the given input signal.

USE OF ST9 SYSTEM RESOURCES.

The demonstration application makes use of the following basic ST9 system resources:

- A/D Convertor
- Multifunction Timer/Counter
- Internal RAM (Register File)
- Input/Output Ports

The A/D convertor

One only of the eight available analogue input channels of the ST9 A/D Convertor is used in this application. The input speech signal, band-restricted by filtering to remove frequency components below a lower, or above an upper limit, is fed to the input channel. The A/D Convertor is operated in continuous scanning mode, each conversion being started by an internal On Chip Event signal generated by the Multifunction Timer underflow event.

USE OF ST9 SYSTEM RESOURCES (Continued)

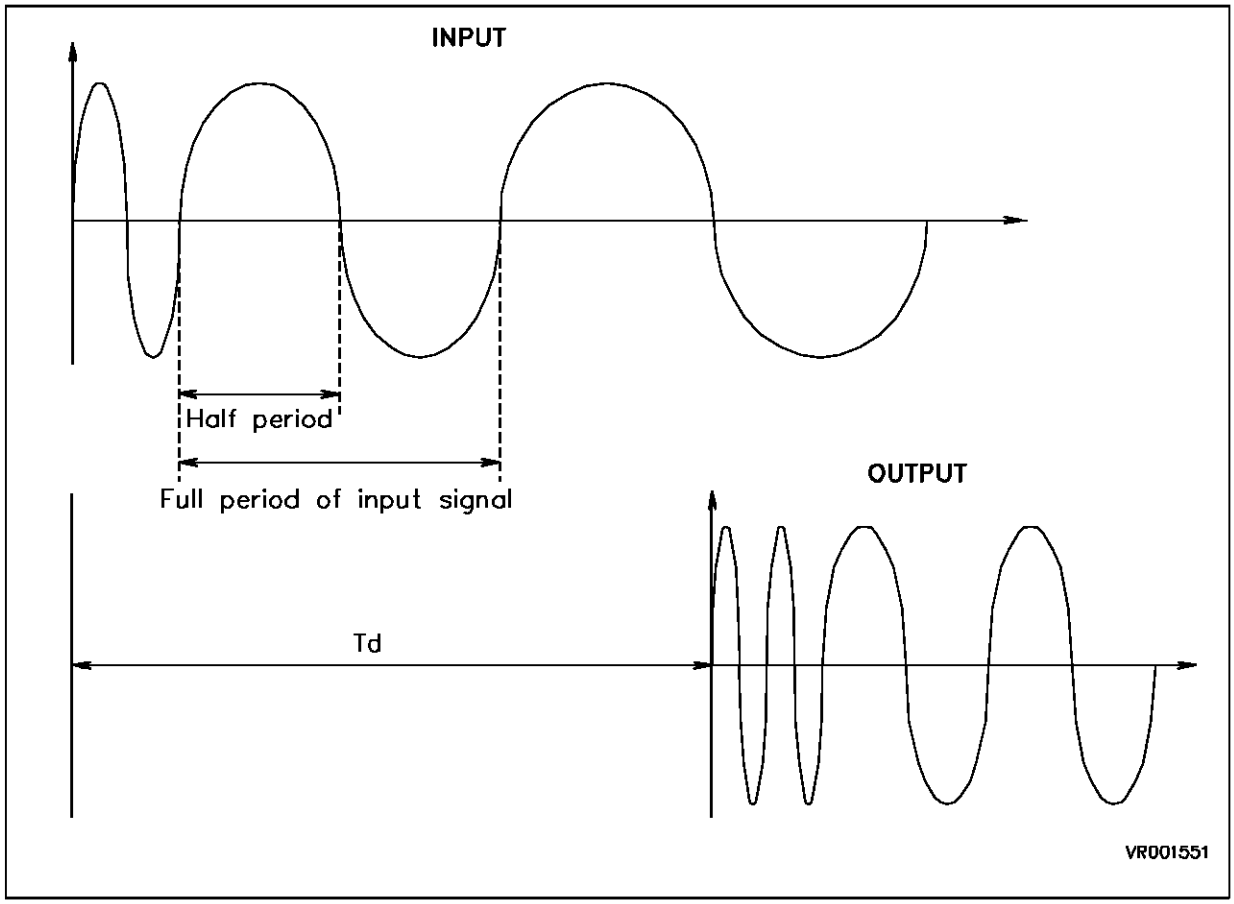
The Multifunction Timer/Counter

The Multifunction Timer is operated in continuous mode with count down from a fixed value of 508, each underflow resulting in the generation of an On-Chip Event signal and reload of the fixed initial counter value (508). The initial count value, prescaler, and clock rate are chosen to give a sampling rate of ~7.8kHz.

Comparison register 1, CMP1, is used to obtain the double sample output rate by successivelyloading this register with the maximum count value (508), and one half the maximum count value (254). In this way two CMP1 event pulses, and the related sample output, will be obtained for each Timer count-down period, i.e. for each input sampling event.

Comparison register 0, CMP0, is used to obtain the Pulse-Width Modulated output. This is achieved by successively loading this register with a value equal to 508 - S, where S is the sample value, or 254 - S. Timer output OUT1 is then set to a “one” value by an OVF or CMP1 event, i.e. effectively at count values 508 or 254, and reset to a “zero” value by a CMP0 event. This results in an output pulse train in which the output is set to “one” for the period of time it takes for the counter to count down from 508 to (508 - S), or from 254 to (254 - S). In either case the “ON” period of the output pulse is proportional to the sample value, S, (range 0 to 254).

Figure 1. Illustration of Frequency Doubling

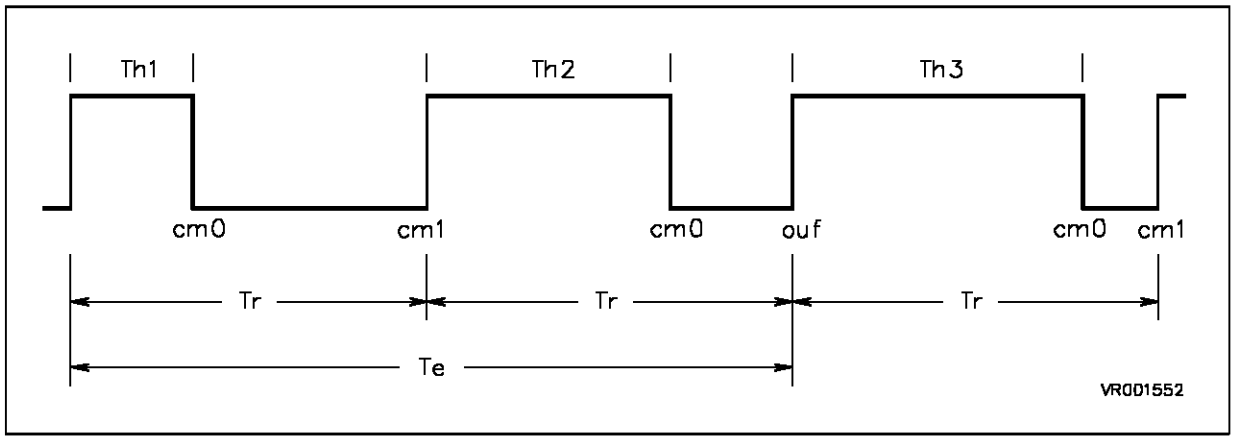


USE OF ST9 SYSTEM RESOURCES (Continued)

The output signal is delayed by an amount T_d relative to the input signal. The minimum input frequency must be no lower than $1/T_d$ since there must be at least three zero-crossings during the period T_d for correct restitution of the double frequency output signal.

Figure 2 shows the Timing control events with T_r, the output sampling rate equal to $T_e/2$, i.e. the input sampling period, T_e, divided by two. In addition, it will be noticed how the sequence of values, T_{hn}, the mark periods of the output, are controlled by the CMP0 values.

Figure 2.Timer Output Controlled by 3 Counter Events



Use of Internal RAM (the Register File).

The ST9030 has 224 bytes of available internal RAM storage, addressable as registers. Two hundred locations have been allocated to the storage of 200 8-bit digitized samples, which leaves 24 register locations for the stack or for storage of temporary values.

The delay of T_d, required before output samples can be read out, is equal to the time required to read and store 100 samples. During this time there must be at least three zero-crossings for the lowest frequency component in the input signal (see Figure 1). For a sampling rate of 7.8kHz this defines the minimum input frequency to be F_{min} ~78Hz.

Use of Port terminals for special input/output.

Two of the Port pins must be used for the analogue signal input and for the PWM output signal. These pins can be initialized for A/D input or Alternate Function Output as described in the initialization section.

ASSOCIATED ANALOGUE CIRCUITRY.

The sampling frequency, F_e, for the input analogue signal is determined by the Timer parameters to be 7.8kHz., which, by Shannon's Theorem implies that the maximun input frequency be limited to no more than half this value, i.e. 3.9kHz. The input circuit must hence include a low-pass filter to avoid any resultant aliasing.

The input frequency must also be limited at the lower frequency end. This arises because the maximum number of samples which can be stored is limited to 200, and furthermore a minimum delay must exist between the input signal and the reconstituted, frequency-doubled output. This minimum delay, equivalent to the period of the lowest frequency component present in the input signal, in conjunction with the sample size limit, implies that the minimum frequency must be no lower than $F_e/100$, i.e. 78Hz.

Effectively, therefore, the input signal must be fed to the A/D sampling input via a Band-Pass filter. In our Application, the filter used had -3db cut-off frequencies of 85Hz and 1.5kHz.

The output signal from the ST9 consists of a PWM pulse train at a frequency of $2xF_e$. The use of a Low Pass filter is hence indicated to recover the required output signal. In our example a high-order Pass-band filter with -3db frequencies of 200Hz and 3kHz was used (Appendix A).

Full details of the design considerations and performance characteristics of the input and output filters do not fall within the scope of this short note.

PROGRAM DETAILS.

The software associated with this design example comprises the following four components:

(i) Initialization of ST9 core and on-chip peripherals.

(ii) The Main Program.

(iii) The Interrupt routine controlling the input sampling and storage of data (`SOUND_IN`).

(iv) The Interrupt routine controlling the output of frequency-doubled samples (`SOUND_OUT`).

Core Initialization.

The ST9 is initialized in the following manner:

(i) The User and System Stacks are set up in Internal RAM,

(ii) The Internal Clock frequency is set to 12MHz,

(iii) The Priority level of the RESET interrupt is set to 7

Initialization of the Input/Output Ports.

Two Input/Outputs only are required, corresponding to the Analog input and output. The corresponding Port pins are initialized either as an A/D input or as an Alternate Function output, i.e. they are linked to ST9 internal signals, as follows:

(i) T0OUTB: Output B from Timer 0,

This output was chosen as it can be activated by the CMP0 event of Timer 0

(ii) AIN7: Input No. 7 of the A/D Convertor

Note: only the one (out of 8 possible) analogue entries is used.

PROGRAM DETAILS (Continued)

Initialization of the A/D Convertor.

* the software activation bit is set to disable,
* an A/D Conversion is triggered by an OCE (On-Chip Event),
* the Interrupt Vector Address is set up at Table Address 20h in EPROM,
* the Analog Watch-Dog Interrupt is disabled,
* the Interrupt generated by "End of Conversion" is validated.

Initialization of the Timer/Counter.

Timer 0 is initialized as follows:

* Count-down mode is selected,
* Continuous Sampling Mode is enabled,
* the internal Clock is selected (4MHz).
* Output, OUTB is:
  - . initialized to 1
  - . Set to 0 by the CMP0 event,
  - . Set to 1 by the CMP1 and OVF events,
* the OCE signal is generated on Counter Underflow,
* the Vector Interrupt Address is specified as 30h in EPROM
* the Interrupt Priority level is set to 6.

MAIN PROGRAM

The Main program, (see Appendix B for a full listing), is automatically entered on System Reset since the address, 38h, has been loaded in the Interrupt Vector Table at locations 0 and 1. Program Main first initializes the Clock, stacks, Multifunction Timer 0, A/D Convertor, and Ports, using the sub-routine `periph_init`. The RAM table pointer is initialized together with pointers and counters which are used to record the number of input waveform zero-crossings, and the number of times the output waveform has been repeated (N.B. there are two repeated output periods for each complete input period).

At this point the `Timer 0 start` Macro is executed which causes the counter to start counting down towards zero from an initial value of 508. Each time the counter clears to zero an On-Chip Event signal will be generated internally in the ST9 chip which will initiate the next A/D input sample conversion.

The main program loops around label "`here`" and the two following statements until such time as 100 input samples have been acquired, as indicated by equality between `cpt_in`, the input sample counter and `#ptr_moy` (64 Hex). The main program then proceeds to set bit 1 in working register 10 (myflags.1 is equivalent to the `start_out` flag in Figure 3).

From this point on the Main program loops around the statements following "`there`" until such time as a System Reset is applied. Each Timer 0 OCE pulse initiates an input sample conversion, and each successful CMP0 comparison event initiates an output sample, these two operations being effected by the Interrupt subroutines `SOUND_IN` and `SOUND_OUT`, respectively.

The overall working of the Main program may be readily visualized by reference to Figure 3.

THE SOUND_IN INTERRUPT ROUTINE.

The organization of the SOUND_IN routine is illustrated by the flow-diagram of Figure 4, and the program details are shown in Appendix B.

This routine is entered whenever the A/D Convertor raises an End of Conversion Interrupt, and will thus occur shortly after the OCE pulse produced by Timer 0 counting down to zero.

After saving the current CPU context (working register pointer and page registers) and selecting the A/D system register page, this routine loads the current input sample into RAM, resetting the RAM table pointer if the end of Table has been reached.

The routine then resets the Timer interrupt pending flags and exits after restoring the CPU context.

Figure 3. Flow Chart of the Main Program

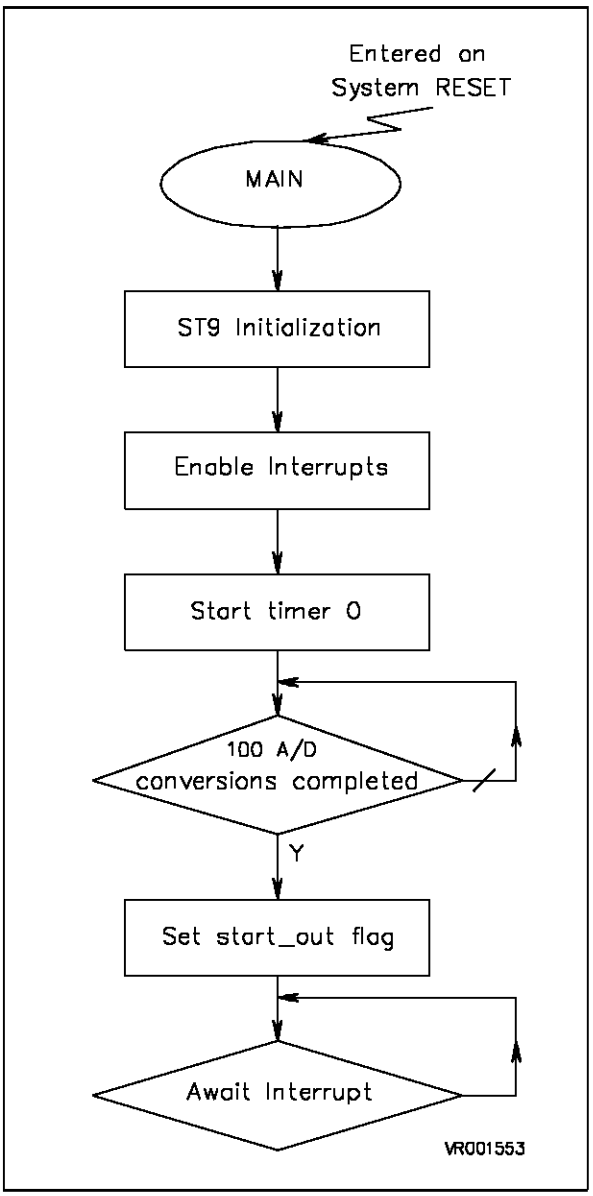
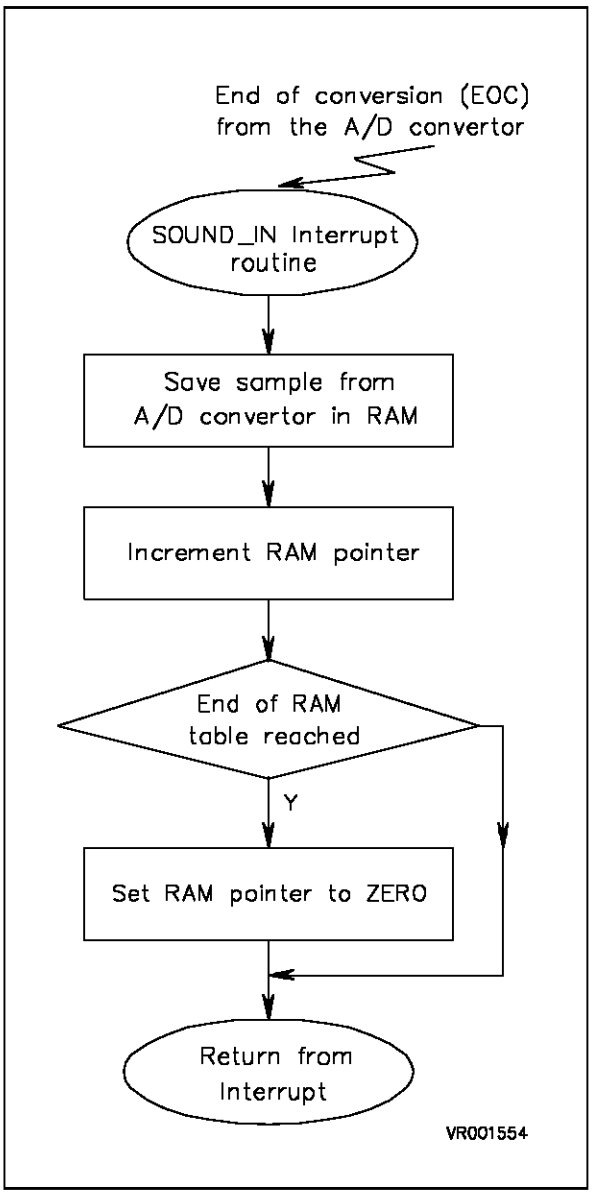


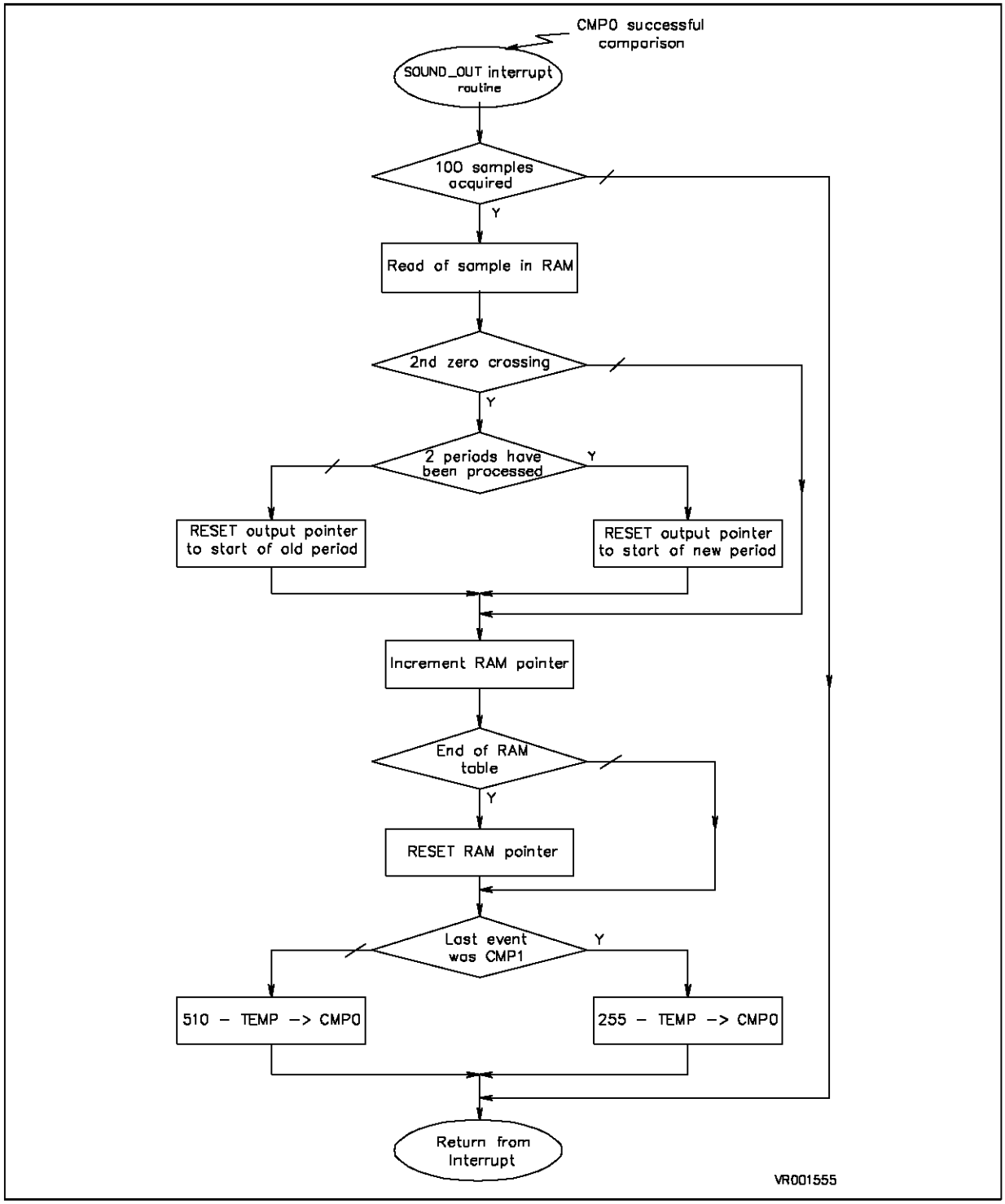
Figure 4. Flow Chart of the SOUND_IN Routine



THE SOUND_OUT INTERRUPT ROUTINE.

The organization of the SOUND_OUT routine is illustrated by the flow-diagram of Figure 5, and the program details are shown in Appendix B.

Figure 5. Flow Chart of the SOUND_OUT Routine



THE SOUND_OUT INTERRUPT ROUTINE (Continued)

This Interrupt routine is entered after each successful CMP0 comparison, i.e. at the conclusion of the "ON" period of the output pulse train (refer to Figure 2).

After saving the current context (working register pointer and page registers) and specifying a set of current working registers, this routine tests to see whether at least 100 samples have been accumulated since the last System Reset. If this is not the case an immediate exit is made by means of a branch to `end_out`. Otherwise the routine proceeds by copying the current output sample value into working registers r8 and r9, These registers comprise respectively the higher and lower components, `tamph` (normally zero) and `tampl`, of the 16-bit extended version (`tampw`) of the basic 8-bit sampled input value.

The next step is to establish whether a zero-crossing has occurred. This is established if the "zero" value (actually "zero" equals the mid-range value of 128) lies between the values of the current and the previous sample values. If a zero-crossing has occurred a further test is made to establish whether this is the first or second such zero-crossing, i.e. whether we have encountered the end of an input sample half- or full-period (refer to Figure 1). If the end of a complete period has been reached a further test is made to see if this complete period has been outputted once, in which case a further copy of the same period is required, or twice, in which case we can proceed to the next input period. The appropriate counter values are updated and a branch is made to `next_sample`.

At this point the sample value is saved for the next zero-crossing test, and the value of the register, CMP1 is loaded with 508 or 254 as appropriate.

Register CMP0 is then loaded with a value of 508 - S, or 254 - S, where S is the current sample. The appropriate choice for the CMP1 and CMP0 values is made on the basis of whether the previous CMP1 value was 508 or 254. In the former case (508) CMP1 is loaded with 254, and CMP0 with 254 - S. In the latter case (254) CMP1 is loaded with 508 and CMP0 is loaded with 508 - S.

Finally the Timer 0 pending event flags are reset, the CPU contest is restored, and exit made by the `IRET` instruction.

REFERENCES

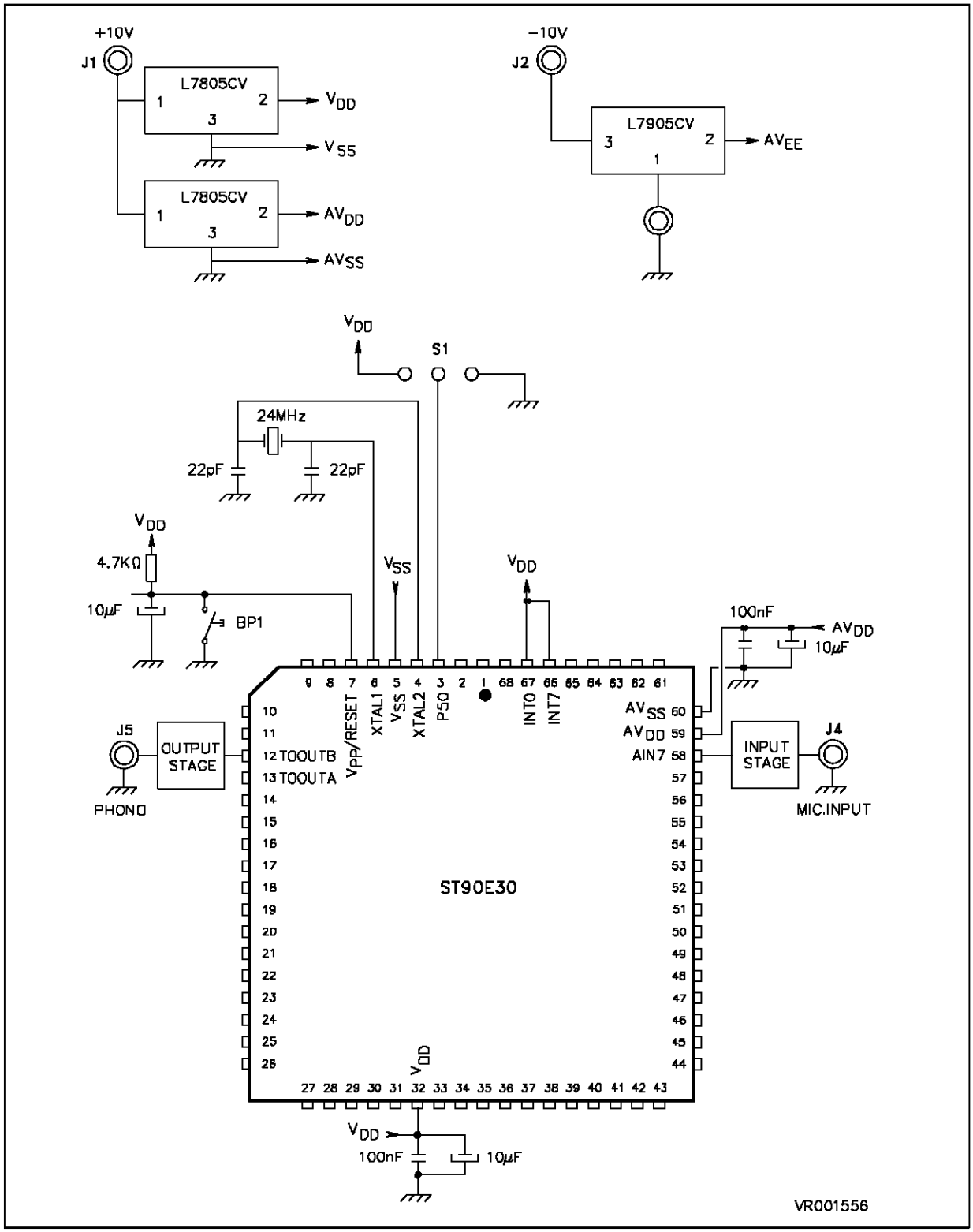
(1) Application Note AN413, "Initialization of the ST9",
Pierre Guillemin and Alan Dunworth, SGS-THOMSON, Rousset.

(2) "ST9 Technical Manual",
SGS-THOMSON Microelectronics.

(3) Application Note AN411, "SYMBOLS.INC, ST9 Register Address and Content Names",
Pierre Guillemin, SGS-THOMSON Microelectronics.

APPENDIX A. CIRCUIT SCHEMATICS

Figure 6 . Frequency Doubler Demonstration System Overview



APPENDIX A. CIRCUIT SCHEMATICS (Continued)

Figure 7. Input Stage Overview

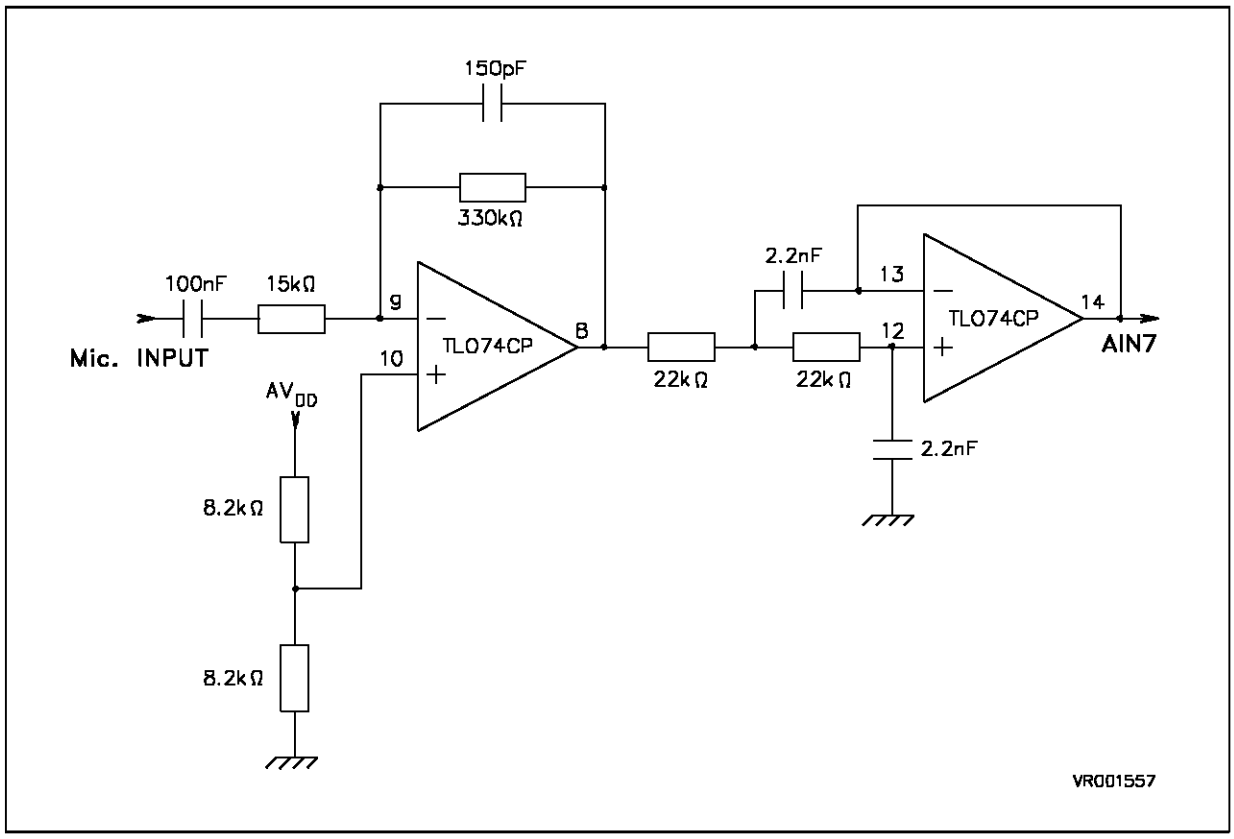
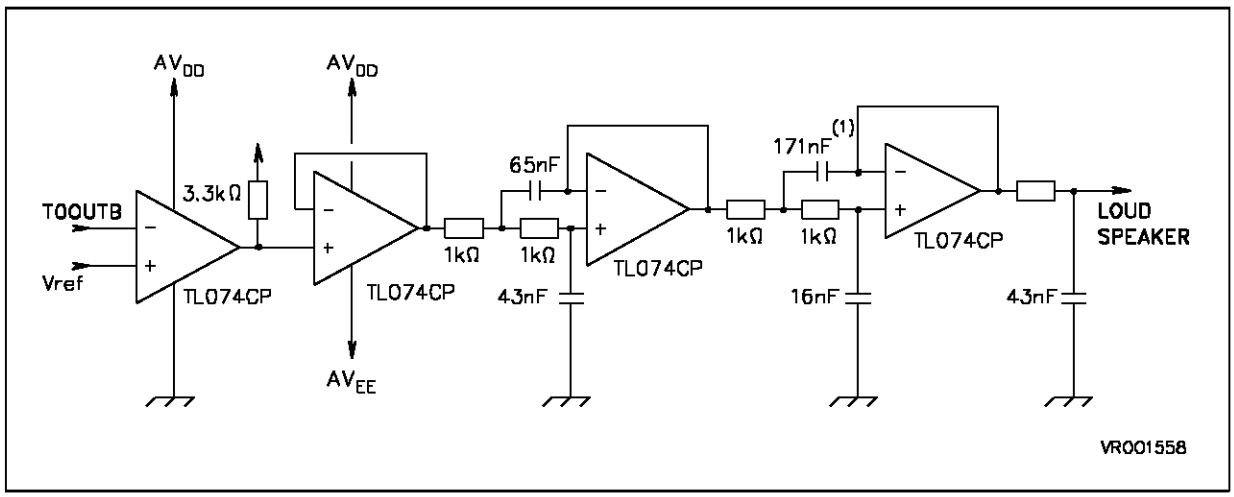


Figure 8. Output Stage



Note 1: This value must be generated

APPENDIX B. PROGRAM LISTING (for ST9030)

```
;*************************************************************************
;
;          ST90E30  DEMOBOARD
;                DEMO.ST9
;*************************************************************************
; REGISTER DEFINITIONS
;*********************

memo_sample      =     r15   ; R0DFh  To detect a zero crossing.
pos_zc           =     r14   ; R0DEh  The first sample location of a period.
cpt_zc           =     r13   ; R0DDh  Zero Crossing counter .
cpt_out          =     r12   ; R0DCh  To scan table in output.
cpt_in           =     r11   ; R0DBh  To scan table in input.
myflags          =     r10   ; R0DAh  My 8 flags.
tampl            =     r9    ; R0D9h
tamph            =     r8    ; R0D8h
tampw            =     rr8   ; R0D8h-R0D9h

tamp_sub         =     rr6   ; R0D6h-R0D7h
cpt_repeat       =     r5    ; Period repeat counter.

sys_stack        =     0D4h  ; !!! system stack is limited by bank 0 registers.

;                R0C7h       ; The end of the system stack.

;*********************
; CONSTANTS DEFINITION
;*********************

work_reg_page0   =     (0Dh*2)
work_reg_page1   =     (0Dh*2)+1

max_count        =     508   ; The maximum count of timer 0.
mid_count        =     (max_count/2)

zc_value         =     07Fh  ; To detect a zero crossing.

;                      R0C8h   The last register of table.
table            =     00h   ; The first register of table.
```

APPENDIX B. PROGRAM LISTING (Continued)

```
ptr_moy          =     100   ; The middle of the table.

ptr_max          =     200   ; Table contains 200 registers.
                             ; Input signal frequency min = 100 Hz .

P_debug          =     R229  ; Port 5 is used for debug.

ad_vect          =     020h  ; Start of A/D vector table.

t0_vect          =     030h  ; Start of Timer 0 vector table.
t0_cmp_vect      =     036h  ; Timer 0 compare event interrupts.


; Flags of MYFLAGS register.
;———————————————-
start_out        =     1     ; 1 => sound output can begin with the 100
                             ; first samples in table.
mask_start_out   =     (1<-start_out)


;******************
; MACROS DEFINITION
;******************

.macro t0start
     spp  #T0D_PG
     or   T_IDMR,#gtien      ; T0 Global interrupt mask disabled.
     or   T_TCR,#cen         ; Counter enabled.
.endm

.macro t0stop
     spp  #T0D_PG
     and  T_TCR,#~cen        ; Counter disabled.
     and  T_IDMR,#~gtiem     ; T0 Global interrupt mask enabled.
.endm
```

APPENDIX B. PROGRAM LISTING (Continued)

```
******************
; INTERRUPT VECTORS
;******************

power_on::
     .text

     .word       main        ; RESET vector.
     .word       main        ; Divide by 0 vector.

     .org ad_vect
     .word       main        ; Analog Watchdog Request not used.
     .word       sound_in    ; A/D End Of Conversion Request.

     .org t0_cmp_vect
     .word       sound_out   ; Compare 0 of timer 0 request.

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

main::

     ld   MODER,#11100000b   ; Ext clock prescale by 2.
                             ; Internal system and user stacks.
     ldw  SSPR,#sys_stack    ; System stack pointer.
     spm                     ; Data memory selected.

     spp  #WDT_PG
     ld   wcr,#wden          ; Watch dog mode disabled, no wait states.
     ld   EIMR,#0            ; Mask all channels interrupts.
                             ; At reset, Global Counter Enable bit is active.

     srp0 #work_reg_page0
     srp1 #work_reg_page1

     call periph_init        ; Initialization of ports, timer0, ADC.

     clr  cpt_in             ; To input samples in table.
```

APPENDIX B. PROGRAM LISTING (Continued)

```
     clr  cpt_out            ; To output samples from table.

     ld   cpt_repeat,#2      ; Period repeat counter.
     ldw  tampw,#381         ; The first value.
     clr  memo_sample
     clr  myflags

     ei

     t0start                 ; T0 ,in PWM mode, will effect D/A conversion.

here::
     cp   cpt_in,#ptr_moy
     jrne here

     bset myflags.1          ; To permit output of the 100 first samples.

there::
     wfi
     jr   there

;****************
; INITIALIZATIONS
;****************

periph_init::
; ——- T0_OUTB.

     spp  #P3C_PG
     ld   P3C0R,#00001000b   ; port3: b3:af,pp,ttl.
     ld   P3C1R,#0ffh        ; others:out,pp,ttl
     ld   P3C2R,#0

; ——- AIN7 input.

     spp  #P4C_PG
     ld   P4C0R,#10000000b   ; port4: b7:af,od,ttl.
     ld   P4C1R,#11111111b   ; others:out,pp,ttl
     ld   P4C2R,#10000000b
```

APPENDIX B. PROGRAM LISTING (Continued)

```
; tim0 init ****************************************
;         d/a 8 bit in pwm
;
;         outB is preset to 1.
;                set by OUF and CMP1.
;                reset by CMP0.
;
;         prescale=0,continuous mode.
;         On Chip Event (OCE) generated by OUF.
;         sound_out interrupt generated by CMP0.
;************************************************

init_t0::

     spp  #T0D_PG            ; T0 data page
                             ; (Xtal clock / 6) => 4 MHz (at reset).
     srp  #(15*2)            ; To access bank F with "r" addressing mode.

     ldw  t_reg0r,#max_count ; 508 counts at 4MHz => 127 micro sec. (7.8 kHz).
     ldw  t_cmp1r,#max_count ; CMP1 value will change for
                             ; 4 times between 2 acquisitions.
     ldw  t_cmp0r,#(381)     ; 508 - 127.
     ld   t_tcr,#00000000b   ; Timer 0 stops, software down count.
     ld   t_tmr,#10000000b   ; OUTB enabled, OUTA disabled.
                             ; Retrigger mode enabled.
                             ; Continuous mode selected.
     ld   t_icr,#0           ; Nop on inputs.
     ld   t_oacr,#11111101b  ; OUTA is disabled.
     ld   t_obcr,#10000011b  ; OUTB is preset to 1,reset by cmp0,
                             ; OUTB is set by ouf and cmp1.
                             ; OCE (a single pulse) is generated by ouf.
                             ; Flags register is cleared at reset.

     ld   t_idmr,#00000100b  ; Only cmp0 interrupt will be enabled.

     spp  #T0C_PG            ; Timer 0 control page.

     ld   t0_ivr,#t0_vect    ; Timer 0 interrupt vector table.
     ld   t0_idcr,#0C6h      ; Priority level 6.
```

APPENDIX B. PROGRAM LISTING (for ST9030) (Continued)

```
; A/D converter ***************************************************************
;         speech or sound input
;
;         Start Conversion is triggered by On Chip Event signal (OCE).
;         ADC frequency = 7.8 kHz.
;
;         Continuous scanning channel 7.
;         Interrupt on End Of Conversion (EOC).
;*****************************************************************************

init_ad::

     spp  #AD0_PG
     ld   AD_CLR,#11001100b  ; OCE starts conversion (single mode).
                             ; Power up, only the channel 7 is converted.
     ld   AD_IVR,#ad_vect    ; Vector pointing the A/D int. routine starting
                             ; address.
     ld   AD_ICR,#00100110b  ; Enables the End Of Conversion interrupt request.
                             ; Masks the Analog Watchdog interrupt request.
                             ; Priority level 6.
     ret
```

APPENDIX B. PROGRAM LISTING (Continued)

```
; ************************************************
; Sound acquisition routine.
; Called by A/D End Of Conversion.
; A/D EOC occurs every 127µs (Fin = 7.8 kHz)
; digital value -> table(cpt_in),
; ************************************************

sound_in::

     pushw RPP               ; Register pointer pair.
     push  PPR               ; Page pointer register.

     srp0  #work_reg_page0   ; Selects the Working registers bank 0.
     srp1 #work_reg_page1    ; Selects the Working registers bank 1.

     spp  #AD0_PG            ; ADC page.
     ld   tampl,AD_D6R       ; Load sample (from A/D data register)
                             ; in RAM table.

     ld   table(cpt_in),tampl
     inc  cpt_in

     cp   cpt_in,#ptr_max
     jrne skip_cpt_in

     clr  cpt_in             ; When the end of the table is encountered.

skip_cpt_in::
     and  AD_ICR,#~(awd+ecv)
                             ; Reset of the request flags.
     pop  PPR
     popw RPP
     iret
```

APPENDIX B. PROGRAM LISTING (Continued)

```
;*************************************************************************
; sound generation routine.
; called by cmp0 of timer 0.
; this routine will be executed every 64 micro sec.
; cmp0 register of timer 0 is here initialized for D/A conversion in PWM mode.
; *************************************************************************
sound_out::

     pushw RPP               ;  Register pointer pair.
     push  PPR               ;  Page pointer register.

     spp  #T0D_PG

     srp0 #work_reg_page0
     srp1 #work_reg_page1

     tcm  myflags,#mask_start_out
     jrne end_out

     clr  tamph
     ld   tampl,table(cpt_out); Load sample value from table.

; ——— test if zero crossing.
     cp   memo_sample,#zc_value
     jrmi small              ; If memo_sample < zc_value.

     cp   tampl,#zc_value
     jrpl end_zc             ; If tamp and memo_sample > zc_value.
     jr   end_test_zc        ; If tamp > zc_value and memo_sample < zc_value.

small::
     cp   tampl,#zc_value
     jrmi end_zc             ; If tamp and memo_sample < zc_value.

; ——— there is a zero_crossing.
end_test_zc::
     djnz cpt_zc,end_zc      ; Test if it’s the second zero crossing.

     ld   cpt_zc,#2          ; To count again 2 zero crossing.
```

APPENDIX B. PROGRAM LISTING (Continued)

```
     djnz cpt_repeat,again   ; Period repeat counter.
     ld   cpt_repeat,#2
     ld   pos_zc,cpt_out     ; Save the zero crossing position.
     jr   end_zc

again::
     ld   cpt_out,pos_zc     ; Load the zc position to repeat a period.

     ; ——— zero crossing has been treated.
end_zc::

     inc   cpt_out

test_cpt_out::
     cp   cpt_out,#ptr_max
     jrne next_sample

     clr  cpt_out

next_sample::
     ld   memo_sample,tampl  ; Save the sample for the next zero crossing test.

     subw T_CMP1R,#mid_count ; The next value to set OUT0B.
     jrne init_cmp0
     ldw  T_CMP1R,#max_count

init_cmp0::
     ldw  tamp_sub,T_CMP1R
     subw tamp_sub,tampw
     ldw  T_CMP0R,tamp_sub   ; The next value to reset OUT0B.

end_out::
     clr  T_FLAGR            ; Resets the timer0 event flags.

     pop  PPR
     popw RPP
     iret
```

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 USE OF THE SOFTWARE.