DIGITAL 3-PHASE GENERATION ST9 DEMONSTRATION SOFTWARE

by P.GUILLEMIN

1) Introduction:

Due to the progress of microcontrollers, accurate 3-phase signals can now be generated in digital way by using three-phase PWM signals. The ST9 is well suited for this application. In fact, an internal Direct Memory Access (DMA) channel offers a capability of continuous high speed flow of data on an 8-bit output port allowing direct drive of a three-phase half-bridge through a very simple hardware interface. This method keeps most of the ST9 power, more than 75% of the CPU time is free for other tasks. Furthermore, this digital approach permits the creation of variable frequencies and voltages.

Based on this ST9 concept for digital 3-phase generation, this note describe demonstration software which is associated with a hardware demonstration board including a keyboard.

The goals of this starter kit are to provide a fast evaluation tool of this new concept, to give an easy way to evaluate the functioning of a motor (according to the voltage and frequencies applied to it) in an application, and to allow a fast design time by using this software as a basis for the application.

This note describes examples of waveforms already implemented and gives the recipe to customize the function of the demonstration board via the keyboard in order to select new motor functions.

2) Reference on digital 3_phase generation:

The principle of 3-phase generation is explained in the application note:"Versatile and cost effective induction motor drive with digital three-phase generation" (cf page 1). Two main parameters are controlled by the software.

2.1) Motor voltage (ie Modulation depth):

The fundamental period of the three-phase waveform is shared into 24 segments. During one segment, the voltage applied to the motor (which is a percentage of the DC line voltage) can be described by the PWM duty cycle. A table describing the duty cycle value is associated to each of these segments. These tables (PATTERN) contain the list of the power switches switching instants. A set of 24 patterns define a complete three-phase sinewave period.

2.2) Three-phase frequency:

The fundamental period of the three-phase waveform is fixed by the number of data values necessary to define a period and the rhythm of changing this data.
Two solutions can be used to modify this frequency:

- by repeating each pattern from 1 to 20 (and more) times,
- by modifying the rhythm of the power switches command.

3) Demonstration board Software organization:

3.1) Flexible programming, Software Black Box:

From a hardware point of view, this application can be considered as a black box between one human interface (the keyboard to enter commands) and an external output driven by the three phase waveforms.

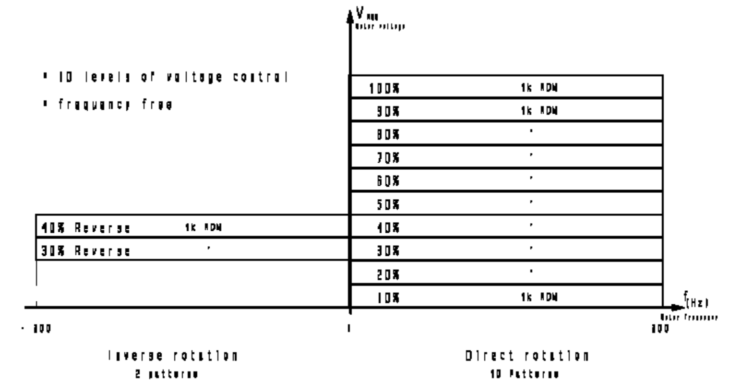

Fig. 1: demonstration board hardware principle

The same black box principle has been used to develop the software. A set of user modifiable tables allow the customization of the keyboard in order to run the motor into different ways.

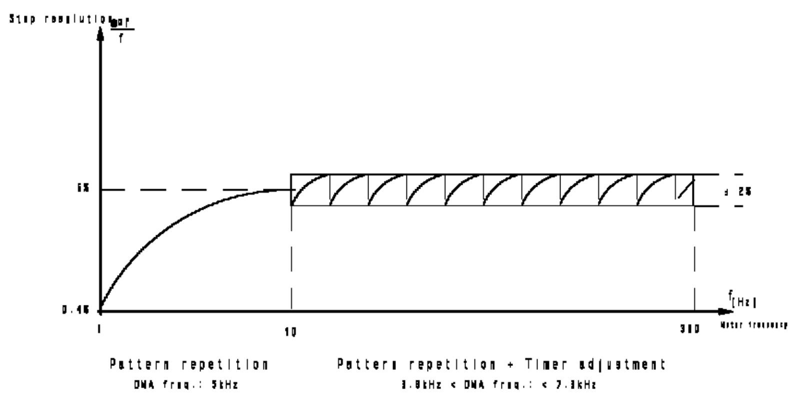

Fig. 2: Software black box principle

Three functions to run or stop the motor can be selected from the keyboard:

- generation of 9 predefined frequencies with their associated voltages,
- generation of ramp (frequency and/or voltage evolution) for example: speeding-up the motor, braking the motor, reversing the motor rotation, using keys ”*” and ”#”.
- stop the motor using key ”0”.

The keyboard customization, allowing the definition of new speeds and voltages to be applied to the motor is made by the modification of tables. These table give the possibility to:

- choose, from a pattern library, several voltages to be applied to the motor,
- choose, from a frequencies table, the speed of the motor,
- define two ramps.

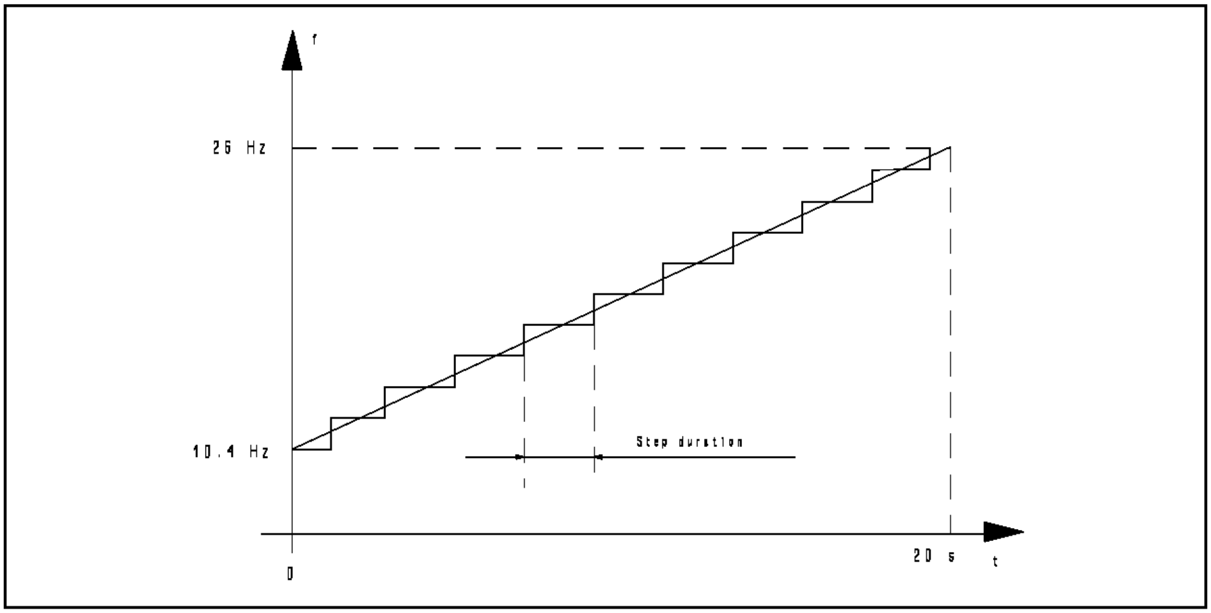

Fig. 3: Set of configuration tables

3.2) Frequencies and Voltage definition libraries:

3.2.1) Voltage definition: PATTERN library.(annex A)

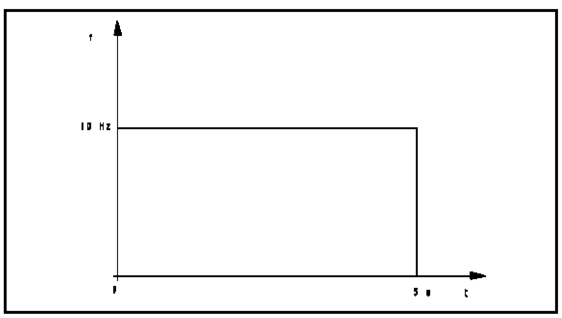

Fig. 4: Example of voltage motor adjustment

As explained in paragraph 2.1, a set of 24 patterns containing the PWM duty cycle definition for each pattern (42 bytes) is necessary to define one complete period. In order to define several voltage level, several patterns are necessary.

Each of these pattern sets is approximatively 1k bytes long. According to the program memory size of the ST9 used, the user can implement several of these voltage levels (up to 7 with a ST90E30, up to 15 with a ST90E36, and more with a romless device).

A software library including several pattern sets defines different voltages (modulation depth) and waveforms (sinewave, trapezewave, ...). These pattern sets, called PATT_XXY (XX for the modulation depth, Y for the pattern structure) may be extracted from the library to define the voltage in the keyboard customization. The content of the patterns library of this demonstration board software is described in annex A.

3.2.2) Frequency definition table: FREQ_TABLE: (Annex B.)

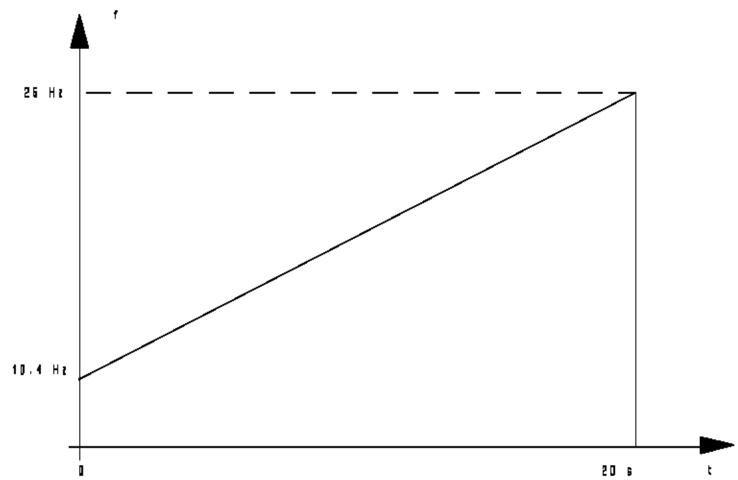

Fig. 5: Frequency adjustment

The rotation speed of the motor is defined by the number of repetitions of each pattern associated with the duration Timer DMA frequency).

The ”FREQ_TABLE” gives the list of all predefined frequencies by with for each one:

- the value of the pattern repetition number,
- the Timer DMA frequency.

A specific frequency can be accessed by giving its location within the FREQ_TABLE.

The modification of the repetition number of each pattern and the timer adjustment define frequencies step by step from 1 Hz to 300 Hz. Typically the step increment is 6%.

3.3) Keyboard customization: KEY_TABLE:

The keyboard can be customized by using a specific table called **KEY_TABLE**. This table can be shared into two distinct sections:

- key 1 to 9 define a pair of values giving the voltage and the frequency supplying the motor.
- key ”*” and ”#” are reserved for the ramp generation and for the alternate rotation of the motor,

3.3.1) Voltage and Frequency assignment (Keys 1 to 9):

To assign to each key from 1 to 9 one voltage and one frequency, the following sequence within SPEED_TABLE has to be repeated:

- Waveform frequency location: number read in the frequency table and corresponding to the chosen frequency (see annex B),
- Modulation depth = pattern address (see annex A).

The above two data value must be replaced by 0FFh and 0FFFFh for the keys which are not assigned.

example:

```
.byte   44          ;16.7 Hz                  key (1)
.word   PATT_04A    ;40% of modulation depth

.byte   0ffh        ; unaffected key
.word   0ffffh
```

Fig. 6: Voltage and Frequency assignment

3.3.2) Ramp definition assignment (Keys "*" and "#"):

The key "*" is used to generate a washing cycle (defined by the table WASHING_TABLE) followed by a ramp (defined by the table SLOPE).

The key "#" generates another ramp (defined by the table SLOPE_1).

example:

```
.byte   0ffh             ;Washing cycle
.word   WASHING_TABLE    ;on key (*)
.byte   0ffh             ;2nd ramp
.word   SLOPE_1          ;on key "#"
.byte   0ffh             ;1st ramp after
.word   SLOPE            ;Washing cycle
```

Fig. 7: Ramp definition assignment

3.3.3) Complete KEY_TABLE example:

```
KEY_TABLE:                  Comments                          Key number
     .byte   44             ;16.7 Hz
     .word   PATT_04A       ;$ for 40% of modulation depth    (1)
     .byte   39             ;25 Hz
     .word   PATT_06A       ;$ for 60% of modulation depth    (2)
     .byte   32             ;40 Hz
     .word   PATT_06A       ;$ for 60% of modulation depth    (3)
     .byte   28             ;50 Hz
     .word   PATT_08A       ;$ for 80% of modulation depth    (4)
     .byte   25             ;60 Hz
     .word   PATT_10B       ;$ for 100% of modulation depth   (5)
     .byte   39             ;16.7 Hz
     .word   PATT_08A       ;$ for 80% of modulation depth    (6)
     .byte   32             ;40 Hz
     .word   PATT_08A       ;$ for 80% of modulation depth    (7)
     .byte   28             ;50 Hz
     .word   PATT_10B       ;$ for 100% of modulation depth   (8)
     .byte   0ffh           ;
     .word   0ffffh         ;unassigned key                   (9)
     .byte   0ffh           ;
     .word   WASHING_TABLE  ;WASHING sequence                 (*)
     .byte   0ffh           ;
     .word   SLOPE_1        ;2nd ramp generation              (#)
     .byte   0ffh           ;washing cycle continuated
     .word   SLOPE          ;with 1st ramp generation
```

3.4) RAMP description table:

The available ramps on key (*) and (#) are described by using a ramp descriptor. This software can generate the following ramps:

- constant rotation speed with predefined voltage and duration,
- ramped rotation speed with predefined voltage and duration,
- washing cycle sequence,
- complete ramp generation.

3.4.1) Ramp descriptor:

This descriptor defines for one elementary ramp, the voltage applied to the motor during the ramp, the starting frequency, the number of steps to reach the new frequency, the duration of each step and the direction of the evolution (positive or negative slope).

The frequency evolution is done step by step from a starting frequency value during several steps by a continuous scanning of the frequency table (defining each discrete frequency).

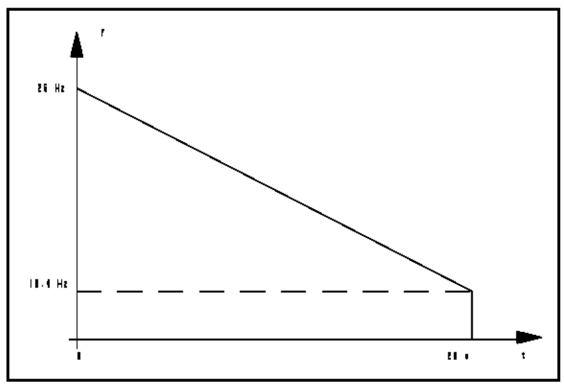

Fig. 8: Frequency stepped evolution

Four parameters are used to completely describe an elementary ramp:

- the pattern address (defining the voltage applied to the motor). This pattern address has to be chosen within the PATTERN library (see annex A).
- the number of steps. This defines the number of steps to be read from the frequency table between two frequencies. If this number is equal to 1, the generated frequency will be stable (no evolution).
- duration of each steps. This duration must be given in ms so the range for this duration is [1-65535] ms
- the starting frequency location within FREQ_TABLE.

By default, the frequency will be increased to reach a higher frequency. A mask allows the choice of a positive or a negative slope. Adding 080h to this value will decrease the frequency to reach a lower value (according to FREQ_TABLE and to the number of steps).

3.4.2) Example of ramp definition:

3.4.2.1) Fixed motor rotation:

The following sequence:

```
.word     PATT_03A
.byte     1
.word     5000
.byte     51
```

will generate a stable frequency during 5 seconds with a speed rotation of 10.4 Hz using a 30% voltage pattern.
For this particular case, if the step duration (here 5000 for 5 seconds) is replace by 0ffffh, the step duration (ie the motor rotation duration) will be infinite. The motor can be stopped only by pressing the key ”0”.

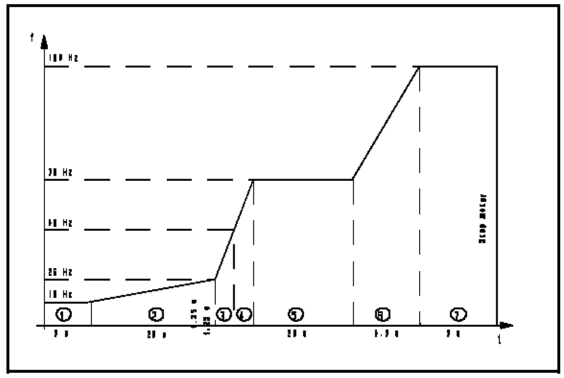

Fig. 9: Frequency level generation

3.4.2.2) Positive ramp generation:

The following sequence:

```
.word     PATT_04A
.byte     13
.word     1538
.byte     51
```

will generate a positive ramp using a 40% voltage pattern with 13 steps from 10.4Hz to 26 Hz. Each step duration is 1.5s in order to reach 20 seconds duration for the whole ramp.

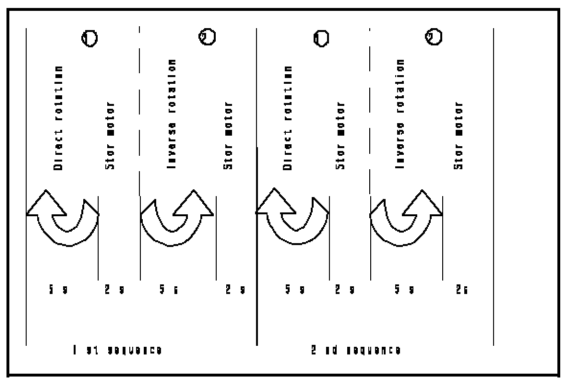

Fig. 10: Positive ramp generation

3.4.2.3) Negative ramp generation:

The following sequence:

```
.word     PATT_04A
.byte     13 + 80h
.word     1538
.byte     51
```

will generate a negative ramp using a 40% voltage pattern with 13 steps from 10.4Hz to 26 Hz. Each step duration is 1.5s in order to reach 20 seconds duration for the whole ramp.

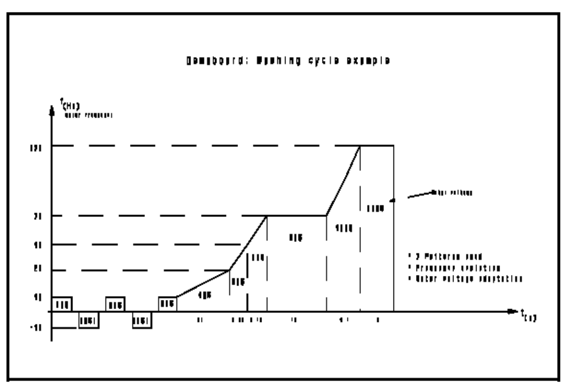

Fig. 11: Negative ramp generation

3.4.2.4) Complete ramp generation:

Several of these descriptors can be added sequentially to create a complete ramp.

The following description table will generate a complete frequency evolution with several ramp and stable levels.

```
SLOPE:
      .word PATT_30A
  1   .byte 1
      .word 5000
      .byte 51
      .word PATT_40A
  2   .byte 13
      .word 1538
      .byte 51
      .word PATT_60A
  3   .byte 11
      .word 114
      .byte 38
      .word PATT_80A
  4   .byte 7
      .word 178
      .byte 27
      .word PATT_80A
  5   .byte 1
      .word 20000
      .byte 21
      .word PATT_100B
  6   .byte 14
      .word 300
      .byte 20
      .word PATT_100B
  7   .byte 1
      .word 5000
      .byte 7
```

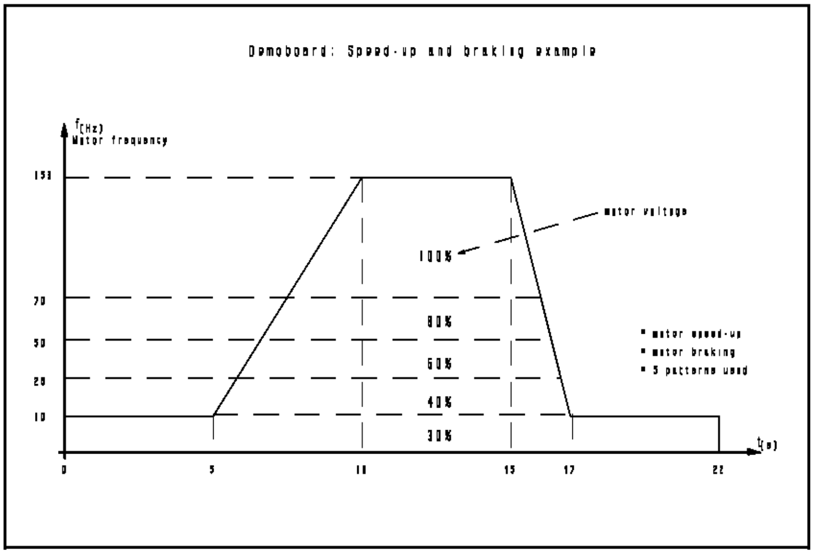

Fig. 12: Complete ramp generation

3.4.3) Direct and Inverse rotation sequence:

This table allows the definition of a sequence with which the motor will evoluate alternatively from one direction to the other direction.

The inverse rotation of the motor is directly generated by software with a specific pattern set in which the two phases are exchanged. Inside the PATTERN library, the PATT_03AI can be used for this purpose.

The table description associated with this function is located on the key ”*” and the evolution will be followed by the ramp generation describe by the SLOPE table.

The table associated is built around:

```
.byte     sequence repetition number
.byte     frequency for the direct rotation
.word     pattern name: direct rotation
.byte     duration for direct rotation
.byte     duration of motor stop
.byte     frequency for the reverse rotation
.word     pattern name: reverse rotation
.byte     duration for reverse rotation
.byte     duration of motor stop
```

Example of Direct/Inverse rotation:

The following table:

```
          .byte     2
┌    .byte     51
│1   .word     PATT_30A
│    .word     5000
└    .word     2000
┌    .byte     51
│2   .word     PATT_30AI
│    .word     5000
└    .word     2000
```

will generate the rotation of the motor as shown in Fig. 12.

4) Demonstration board example:

Annex C describes the three differents examples implemented in the demonstration board software:

- 9 speeds and voltages
- 1 washing cycle associated with a ramp
- 1 braking cycle

Annex D gives the electronic schematic of the demonstration board.

Annex E shows the ST9 configuration file which can be modified by the user to create new actions on the motor.

5) Summary:

For evaluation and new application design, saving time and cost is today the main target in development. This software allows the easy reach of this objective. In fact, this versatile demonstration software provides a good tool to evaluate the three-phase motor concept, to evaluate it in an application and to save time by using this software for the final application.

Furthermore, the ST9, well suited for this application thanks to its DMA capability, remains free to manage other tasks.

Bibliography/references:

- **"Versatile and cost effective induction motor drive with digital three-phase generation"**
Bruno MAURICE - Jean-Marie BOURGEOIS - Bernard SABY Central Applications Laboratory ROUSSET.

- **"3-phase motor drive using the ST9 multi-function Timer and DMA"**
Bernard SABY - Pierre GUILLEMIN Central Applications Laboratory ROUSSET.

- **"External DMA mode: I/O data transfer synchronized by ST9 Timer"**
Pierre GUILLEMIN Central Applications Laboratory ROUSSET.

- **"Concept description of Induction Motor Drive with digital 3-phase generation"**
Bruno MAURICE Central Applications Laboratory ROUSSET

- **"Digital 3-phase Induction Motor Drive with ST9, demoboard software description"**
Pierre GUILLEMIN Central Applications Laboratory ROUSSET

- **"New isolated Gate Drive for Power Mosfet and IGBT"**
Jean-Marie BOURGEOIS E.P.E.-Firenze/Italy Setp. 1991

- **"Industrial and Computer Peripheral ICs databook"**
1st edition SGS-THOMSON Microelectronics

- **"ST9 Family 8/16 bit MCU databook"**
1st edition SGS-THOMSON Microelectronics

- **"ST9 Family 8/16 bit MCU Programming Manual"**
2nd edition SGS-THOMSON Microelectronics

- **"ST9 Family 8/16 bit MCU Technical Manual"**
1st edition SGS-THOMSON Microelectronics

ANNEX A: Available patterns for modulation depth

Name	Address	Voltage (%)	Structure
Patt_02d.obj	PATT_02D	20 %	synchronisation started
Patt_02e.obj	PATT_02E	20 %	Centered
Patt_02f.obj	PATT_02F	20 %	Doubled
Patt_02g.obj	PATT_02G	20 %	Centered with DC component
Patt_02h.obj	PATT_02H	20 %	centered forced 3rd harmonic
Patt_03A.obj	PATT_03A	30 %	Doubled
Patt_03AI.obj	PATT_03AI	30 %	Doubled Inversed rotation
Patt_04a.obj	PATT_04A	40 %	Doubled
Patt_06a.obj	PATT_06A	60 %	Doubled
Patt_08a.obj	PATT_08A	80 %	Doubled
Patt_10b.obj	PATT_10B	100 %	Centered
Patt_10c.obj	PATT_10C	100 %	Doubled
Patt_12a.obj	PATT_12A	125 %	Centered trapezoidal
Patt_12b.obj	PATT_12B	125 %	Doubled trapezoidal

ANNEX B: List of available frequencies (note [1])

Frequency Index	Frequency (in Hz)	Repetition number	Timer duration (in µs)
0	208	1	4.75
1	198	1	5.00
2	189	1	5.25
3	180	1	5.50
4	172	1	5.75
5	165	1	6.00
6	159	1	6.25
7	153	2	3.25
8	142	2	3.50
9	132	2	3.75
10	124	2	4.00
11	117	2	4.25
12	110	2	4.50
13	104	2	4.75
14	99	2	5.00
15	94	2	5.25
16	90	2	5.50
17	86	2	5.72
18	82	3	4.00
19	78	3	4.25
20	73	3	4.50
21	70	3	4.75
22	66	3	5.00
23	63	3	5.25
24	62	4	4.00
25	58	4	4.25
26	55	4	4.50
27	52	4	4.75
28	49	4	5.00
29	47	4	5.25
30	44	5	4.50

Frequency Index	Frequency (in Hz)	Repetition number	Timer duration (in µs)
31	42	5	4.75
32	40	5	5.00
33	36	6	4.50
34	35	6	4.75
35	33	6	5.00
36	31	7	4.50
37	30	7	4.75
38	28	7	5.00
39	26	8	4.75
40	23	9	4.75
41	20.9	10	4.75
42	19	11	4.75
43	17.4	12	4.75
44	16	13	4.75
45	15	14	4.75
46	14	15	4.75
47	13	16	4.75
48	12	17	4.75
49	11.6	18	4.75
50	11	19	4.75
51	10.4	20	4.75
52	9	22	4.75
53	8	26	4.75
54	7	30	4.75
55	6	35	4.75
56	5	42	4.75
57	4	52	4.75
58	3	70	4.75
59	2	104	4.75
60	1	209	4.75

Note[1]: The ST9 is driven with an internal clock of 12 MHz.

ANNEX C: Demonstration board example

Standard Keyboard configuration:

KEY	FREQ.(Hz)	PATTERN	
		Voltage (%)	Structure
1	3	30	Doubled
2	10.4	30	Doubled
3	26	40	Doubled
4	35	80	Doubled
5	52	80	Doubled
6	52	100	Centered
7	70	80	Doubled
8	70	100	Centered
9	104	100	Doubled
*	Washing cycle	See SLOPE description	
#	Braking cycle	See SLOPE_1 description	
0	Stop		

Washing cycle plus ramp generation (SLOPE):

Braking cycle (SLOPE_1):

ANNEX D: Demoboard Schematics

ANNEX E: ST9 demoboard configuration software

```
.sbttl  ”Frequencies, keyboard, slope description table for 3Φ motor”
.list

; new date: October 29 th
; last rev: October 01 st

               .pl            65             ; number of lines per page
               .list
;              .list          me             ; enable macro expansion control
;              .list          bex            ; enable continuation of code on next line
;              .nlist         line           ; disable source line number control
;              .nlist         loc            ; disable current location counter control
;              .nlist         code           ; disable binary code control
;              .nlist         src            ; disable source line control
;              .nlist         com            ; disable comment control
;              .nlist         md             ; disable macro definition control
;              .nlist         mc             ; disable macro call control
;              .nlist


;*****************************************************************************
;*                    SPEED AND SLOPE DESCRIPTION TABLES
;*****************************************************************************
;****************************************
;*     Connection with other modules *
;****************************************

.extern                PATT_03A, PATT_03AI
.extern                PATT_04A
.extern                PATT_06A
.extern                PATT_08A
.extern                PATT_10B

.global                SPEED_TABLE, WASHING_TABLE, SLOPE, End_washing_table
.global                End_slope, End_slope_1

;*******************************************************************************
;*                             FREQUENCIES TABLE                            *
;*                                                                          *
;*     Here is the list of all the frequency available for this application      *
;* this table give the frequency location and the frequency location within the   *
;* FREQUENCY_TABLE located in MOTOR.ST9 source file.                                *
;* This table must be used to fill the SPEED_TABLE with the frequency location    *
;*                                                                          *
;*    ╔══════════════════════════════════════════════════════════════════╗   *
;*    ║  198   189    180    172     165    159    153    142   ║          *
;*    ║  (1)   (2)    (3)    (4)     (5)    (6)    (7)    (8)   ║          *
;*    ║                                                         ║          *
;*    ║  132   124    117    110     104    99     94     90    ║          *
;*    ║  (9)   (10)   (11)   (12)    (13)   (14)   (15)   (16)  ║          *
;*    ║                                                         ║          *
;*    ║  86    82     78     73      70     66     63     62    ║          *
;*    ║ (17)   (18)   (19)   (20)    (21)   (22)   (23)   (24)  ║          *
;*    ║                                                         ║          *
;*    ║  58    55     52     49      47     44     42     40    ║          *
;*    ║ (25)   (26)   (27)   (28)    (29)   (30)   (31)   (32)  ║          *
;*    ║                                                         ║          *
;*    ║  36    35     33     31      30     28     26     23    ║          *
;*    ║ (33)   (34)   (35)   (36)    (37)   (38)   (39)   (40)  ║          *
;*    ║                                                         ║          *
;*    ║ 20.9   19     17.4   16      15     14     13     12    ║          *
;*    ║ (41)   (42)   (43)   (44)    (45)   (46)   (47)   (48)  ║          *
;*    ║                                                         ║          *
;*    ║ 11.6   11     10.4   9       8      7      6      5     ║          *
;*    ║ (49)   (50)   (51)   (52)    (53)   (54)   (55)   (56)  ║          *
;*    ║                                                         ║          *
;*    ║  4     3      2      1                                  ║          *
;*    ║ (57)   (58)   (59)   (60)                               ║          *
;*    ╚══════════════════════════════════════════════════════════════════╝   *
;*                                                                          *
;*******************************************************************************
```

```
;********************************************************************************
;*                   KEY_TABLE: Keyboard table                              *
;*      This table must be updated in order to modify the assignment between the  *
;* keyboard and the frequency and the voltage applied to the motor         *
;* You must give (using the following example) for each key the name of the      *
;* pattern (defining the voltage applied on the motor) and the frequency           *
;* location within FREQ_TABLE (defining the pattern repetition number and the      *
;* timer Compare 0 event                                                           *
;* Non used key will be detected by 0ffffh instead of a real address               *
;* Two keys (* and #) must considered as reserved by the software for WASHING      *
;* SLOPE demonstration

KEY_TABLE:

  .byte        58             ;3 Hz
  .word        PATT_03A       ;$ for  30% of Vcc          (1)
  .byte        51             ;10.5 Hz
  .word        PATT_03A       ;$ for  30% of Vcc          (2)
  .byte        39             ;26.1 Hz
  .word        PATT_04A       ;$ for  40% of Vcc          (3)
  .byte        34             ;34.7 Hz
  .word        PATT_08A       ;$ for 80% of Vcc           (4)
  .byte        27             ;52 Hz
  .word        PATT_08A       ;$ for  80% of Vcc          (5)
  .byte        27             ;52 Hz
  .word        PATT_10B       ;$ for 100% of Vcc          (6)
  .byte        21             ;70 Hz
  .word        PATT_08A       ;$ for 80% of Vcc           (7)
  .byte        24             ;70 Hz
  .word        PATT_10B       ;$ for 100% of Vcc          (8)
  .byte        13             ;104 Hz
  .word        PATT_10B       ;$ for 100% of Vcc          (9)
  .byte        0ffh
  .word        WASHING_TABLE  ;Washing sequence           (*)
  .byte        0ffh
  .word        SLOPE_1               ;Slope generation           (#)
  .byte        0ffh
  .word        SLOPE          ;Washing sequence continuted

;*******************************************************************************
;*                             WASHING TABLE DESCRIPTION                       *
;*******************************************************************************

WASHING_TABLE:
  .byte        2                     ; Sequence repetition number
  .byte        51                    ; 10.4 Hz
  .word        PATT_03A              ; Direct rotation
       .word   5000           ; Pattern duration: 5 seconds (in ms)
       .word   2000           ; Motor stop duration
  .byte        51                    ; 10.4 Hz
  .word        PATT_03AI             ; Inverse rotation
       .word   5000           ; Pattern duration
       .word   2000           ; Motor stop duration
End_washing_table:

;*******************************************************************************
;*                             SLOPE DESCRIPTION TABLE                            *
;*******************************************************************************

SLOPE:
  .word        PATT_03A              ; 1st frequency
       .byte   1                     ; one step ==> no evolution
       .word   5000                  ; 5 seconds ( in ms)
       .byte   51                    ; 10.4 Hz
  .word        PATT_04A              ; 2st frequency
       .byte   13                    ; number of step
       .word   1538                  ; 1.538 seconds (in ms) per step
       .byte   51                    ; from 10 Hz to 26 Hz
  .word        PATT_06A              ; 3rd frequency
       .byte   11                    ; number of step
       .word   114                   ; 0.114 seconds (in ms) per step
       .byte   38                    ; from 28 Hz to 49 Hz
  .word        PATT_08A              ; 4th frequency
       .byte   7                     ; number of step
       .word   178                   ; 0.178 seconds ( in ms) per step
       .byte   27                    ; from 52 Hz to 70 Hz
  .word        PATT_08A              ; 5th frequency
       .byte   1                     ; one step = no evolution
       .word   20000                 ; 20 seconds (in ms)
       .byte   21                    ; 70 Hz
  .word        PATT_10B              ; 6th frequency
       .byte   14                    ; number of step
       .word   300                   ; 0.300 seconds (in ms ) per step
       .byte   20                    ; from 73 Hz to 153 Hz
  .word        PATT_10B              ; 7th frequency
```

```
       .byte   1                      ; one step = no evolution
       .word   5000                   ; 5 seconds (in ms)
       .byte   7                      ; 153 Hz
End_slope:
  .word        End_slope                     ; end slope generation ==> stop motor


SLOPE_1:
  .word        PATT_03A               ; 1st frequency
       .byte   1                      ; one step ==> no evolution
       .word   5000                   ; 5 seconds ( in ms)
       .byte   51                     ; 10.4 Hz
  .word        PATT_04A               ; 2st frequency
       .byte   13                     ; number of step
       .word   111                    ; 0.111 seconds (in ms) per step
       .byte   51                     ; from 10 Hz to 26.1 Hz
  .word        PATT_06A               ; 3rd frequency
       .byte   11                     ; number of step
       .word   111                    ; 0.111 seconds (in ms) per step
       .byte   38                     ; from 28 Hz to 49.6 Hz
  .word        PATT_08A               ; 4th frequency
       .byte   7                      ; number of step
       .word   111                    ; 0.111 seconds ( in ms) per step
       .byte   27                     ; from 52 Hz to 70 Hz
  .word        PATT_10B               ; 5th frequency
       .byte   14                     ; number of step
       .word   111                    ; 0.111 seconds (in ms ) per step
       .byte   20                     ; from 73 Hz to 153 Hz
  .word        PATT_10B               ; 7th frequency
       .byte   1                      ; one step = no evolution
       .word   5000                   ; 5 seconds (in ms)
       .byte   7                      ; 153 Hz
  .word        PATT_10B               ; 8th: decreasing step
       .byte   14 + 80h               ; number of step
       .word   45                     ; 45 ms
       .byte   8                      ; from 142 Hz to 70 Hz
  .word        PATT_08A               ; 9th: decreasing step
       .byte   7 + 80h                       ; number of step
       .word   45                     ; 45 ms
       .byte   22                     ; from 66 Hz to 49.6 Hz
  .word        PATT_06A               ; 10 th: decreasing step
       .byte   10 + 80h               ; number of step
       .word   45                     ; 45 ms
       .byte   29                     ; from 47 Hz to 28 Hz
  .word        PATT_04A               ; 11 th: decreasing step
       .byte   13 + 80h               ; number of step
       .word   45                     ; 45 ms
       .byte   39                     ; from 26.1 Hz to 10.4 Hz
  .word        PATT_03A               ; 12 th: decreasing step
       .byte   1                      ; one step = no evolution
       .word   5000                   ; 5 second (in ms)
       .byte   51                     ; 10.4 Hz
End_slope_1:
  .word        End_slope_1                   ; end slope generation ==> stop motor

;************************ End of motor software configuration ******************
```

Table of contents

TABLE OF FIGURES

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.