Hitachi America, Ltd.

Application Note

H8/3003

Cristian Tomescu

H8/3003 Stepper Motor Control

INTRODUCTION

The H8/300H Family of microcontrollers offers many improvements to the H8/300 line in terms of CPU performance as well as additional and enhanced on-chip functionality. Of particular interest for this application note, the H8/300H Family features full-fledged DMA capabilities, a versatile timer network (ITU), and a Timing Pattern Controller (TPC). Direct memory transfers can occur between 2 memory areas or between I/O register space and memory. These transfers can be triggered upon a variety of interrupt-driven events in the timer network and in other peripherals as well. The TPC can be set up to output up to 16 square wave patterns upon selectable trigger signals from the ITU network. These features make the H8/300H microcontrollers ideal for controlling the operation of multiple-phase DC and stepper motors found in embedded systems involved in motion control.

In general, a variety of alternatives are available for motion control, such as linear and rotary solenoids, servo motors, AC and DC motors, and stepper motors. For applications that require accurate positioning and moderate positioning speed at a reasonable cost, a stepper motor implementation is the ideal choice. Such applications range from printers to tape drives, floppy disc drives, medical instrumentation, robotics equipment, and other digitally controlled positioning systems.

This application note will show how an H8/3003 can be utilized to control the operation of a stepping motor. Although the H8/3003 was the chosen device, any member of the H8/300H Family could be used in the process. A simple motion control embedded system can be designed utilizing only a H8/300H microcontroller, a stepper motor, and the motor-specific required control circuitry (consisting of power FET transistors, clamping diodes, and current-limiting high-power resistors).

STEPPER MOTOR CHARACTERISTICS

Before stepping into the design example, a brief synopsis of a stepper motor features is warranted. Figure 1 depicts an example of how a simple, 2-phase stepper motor is built. It consists of a disc centered and mounted on a shaft which is free to rotate between a pair of stator poles. The disc consists of a series of permanent magnets radially distributed on its surface. Two electromagnetic coils are wound around each stator pole. Energizing the coils with the proper polarities will generate a magnetic field pattern to which the permanent magnets on the rotor disc will try to align producing a torque. As a result, the disc will rotate until the stator poles are aligned with the next permanent magnet, hence taking a "step". The numbers of radial permanent magnets on the disc determine the stepping angle of the rotor, hence the number of steps.

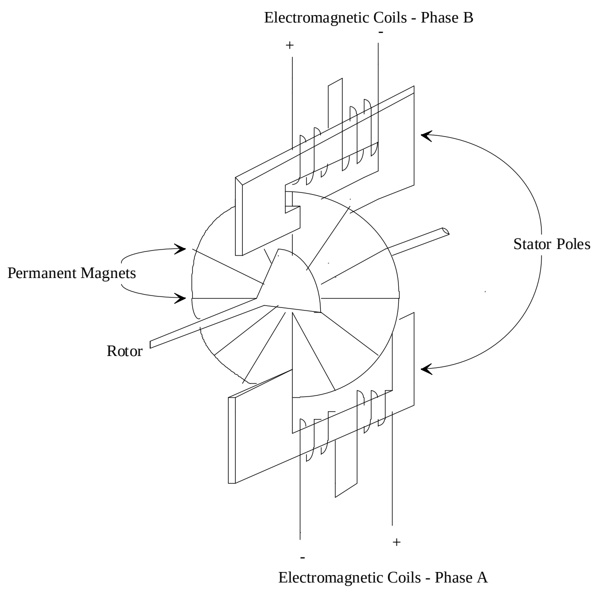

Electromagnetic Coils - Phase B
+
-
Stator Poles
Permanent Magnets
Rotor
-
+
Electromagnetic Coils - Phase A

Figure 1

The electromagnetic coils in figure 1 must be energized in a particular fashion in order to allow the rotor to take incremental steps. The voltage drop across each coil determines how fast the current will build in the coil windings, hence the speed of the motor. An example of the needed circuitry in order to provide alternating voltage drops across the electromagnetic coils is shown in figure 2.

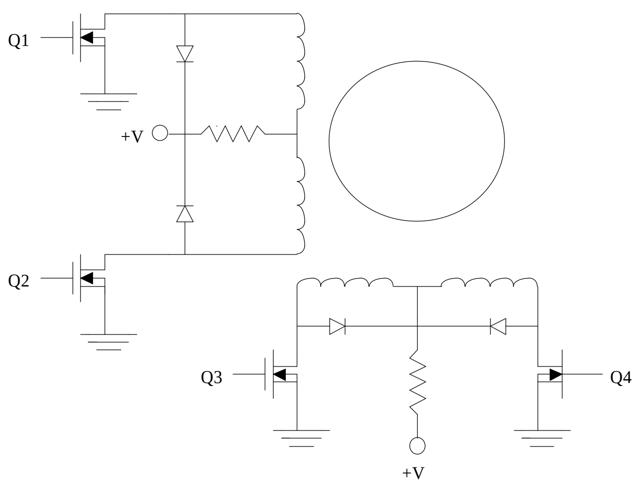


Figure 2

In order to cause the motor shaft to rotate in a clockwise fashion one step at a time, the 4 MOSFET transistors must be turned on and off in the 4 step sequence shown in table 1 below. For constant step rotation, this sequence must be repeated continuously.

Step	Q1	Q2	Q3	Q4
1	On	Off	On	Off
2	On	Off	Off	On
3	Off	On	Off	On
4	Off	On	On	Off

Table 1

For counterclockwise rotor motion, the step sequence shown above must simply be reversed. The resistors in figure 2 provide coil protection by limiting the current to the specified maximum in accordance with the motor specifications. The 4 clamp diodes prevent the voltage across the inductive winding to reach levels that would damage the MOSFET switching transistors as they are turned off. The speed of the motor depends upon 2 factors: the +V voltage level and/or the time this voltage is applied at the coils.

First, we consider the voltage level. Applying a voltage at the +V point will produce a current flow through the coils if the corresponding transistor is biased. However, due to the coil inductance, the current will reach its steady state value in a finite time; its value will depend upon the motor winding series resistance, the coil inductance, and the applied voltage, according to the formula:

$$\mathbf{i(t) = V/R(1-e^{(-R/L)})}$$

Figure 3 shows how this current is built up as a function of time for different voltage values. The higher the voltage, the quicker the current will build up and reach its steady value, and the faster the rotor will move to the next step position.

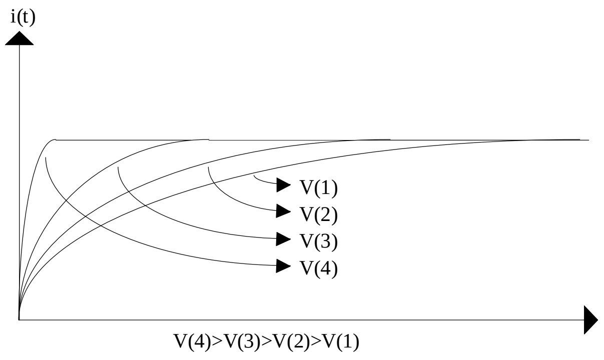


Figure 3

Second, we consider the time the voltage is applied at the electromagnetic coils. As mentioned above, the +V voltage will create a steady state current as long as the transistor associated with the coil is turned on. The shorter the MOSFETs are active for each step shown in figure 3, the quicker will the motor go through each step. Therefore, by successively decreasing the active time of the MOSFETs, the motor performs an accelerated rotational motion. Vice versa, as the transistors active time is gradually increased, the motor will decelerate. Of course, there is a minimum active time and voltage that must be supplied in order for the motor to properly rotate, and it depends upon the motor torque characteristics. Figure 4 shows the alternating states of the 4 transistors in order to perform the motor full-stepping process.

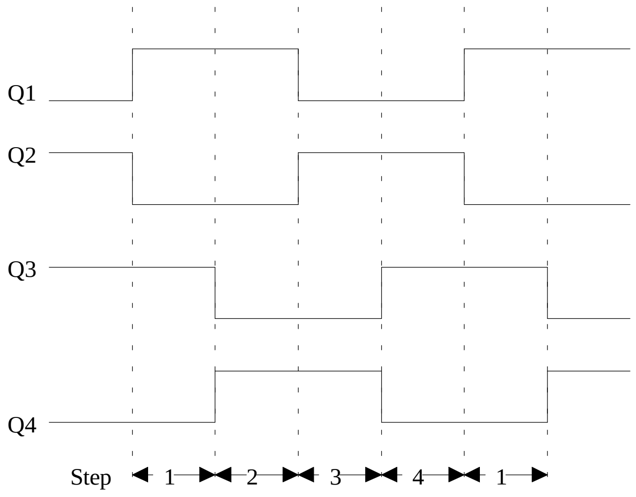


Figure 4

These excitation waveforms correspond to a 2-phase 4 excitation process. The input at Q1 corresponds to phase A, the Q3 input to phase B, the Q2 input to the inverse of phase A, and the input at Q4 to the inverse of phase B. Since phase A falls while phase B is low, figure 4 depicts the excitation waveforms for a clockwise (CW) motion. For counter clockwise (CCW) rotation, phase A must fall while phase B is high.

As a final aspect to be considered, each stepper motor, being a mechanical device, has limitations on how much current can be supplied to the coils, and for how long in order for the shaft to rotate each step. A typical relationship between the motor torque, current and the stepping rate (in pulses per second) is shown in figure 5. In order to overcome inertia, the motor must be supplied a minimum amount of current in order to take one step. During continuous stepping with a constant voltage supply, the dynamic torque developed decreases with increasing stepping rate. There is also a maximum stepping rate to which the motor will respond; overdriving this rate will cause rotor vibration and no stepping will be performed.

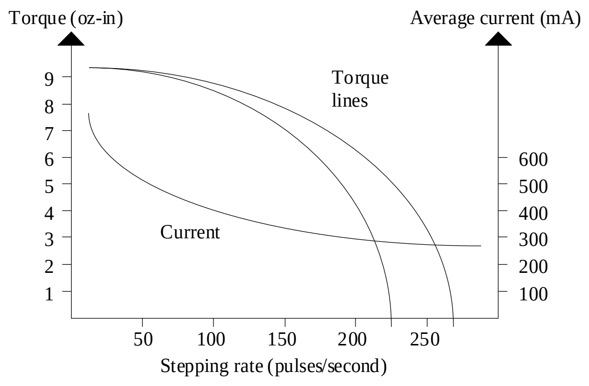


Figure 5

DESIGN EXAMPLE

As stated in the introduction, the H8/3003 microcontroller is utilized to provide the stepper motor control waveforms. The stepper motor used required 2-phase 4 excitation waveforms, and was of a so-called bifilar type. This term refers to the type of windings used in the 2 stator coils; specifically, bifilar windings contain 2 coils in each stator half. It is important to note that although a bifilar-wound motor contains 4 coils (or phases), it is essentially operated as a 2-phase motor. The motor used is manufactured by Hitachi, Ltd. and it is of a permanent magnet type; its part number is KP6P8-701. Appendix A shows its standard specifications and its torque-versus-pulse rate curve measured at an excitation voltage of +12V.

The hardware utilized for this project consisted of the following main parts:

1. The H8/3003 compact evaluation board.
2. A specially wirewrapped motor interface board used to boost the motor control signals, provide the motor power supply requirements.
3. The KP6P8-701 permanent-magnet stepper motor.

The H8/3003 is supplying the initial 4 excitation waveforms via the 4 TPC output lines TP 12-15. TP12 produces the phase A waveform, TP13 the phase B waveform, TP14 the inverted phase A waveform, and TP15 the inverted phase B waveform. The motor interface board schematic is shown in figure 6. The original motor TPC excitation signals are driven through AND gates, and are available as long as 5V power is supplied to the system. Next, they are input to the gates of 4 MOSFET transistors (type 2SK1095), alternately switching them on and off. These transistors act as a drain for the current generated across the electromagnetic coils of the motor. Four protection diodes of the type HRP-22 are used to keep the voltage drop between the drain and the source to 0.55V. Finally, power is supplied to the motor windings from a +12V power supply through a pair of low-resistance, high power dissipation loads. Figure 7 shows the waveform output from the H8/3003 in order to produce successive motor steps. Since the MOSFET's switching time is extremely fast (50-400ns depending upon the drain current), a dead-time (t) is inserted between inverting phase signals (A and A-, B and B-) so that A- and B- go low a little bit before A and B are turned on.

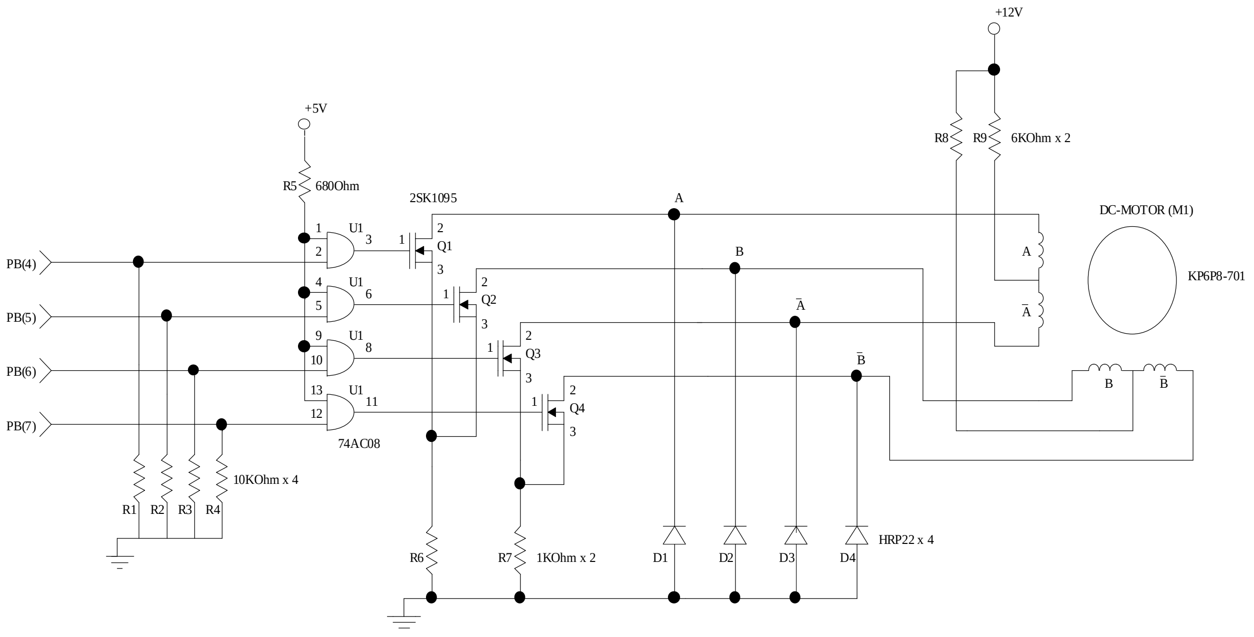


Figure 6

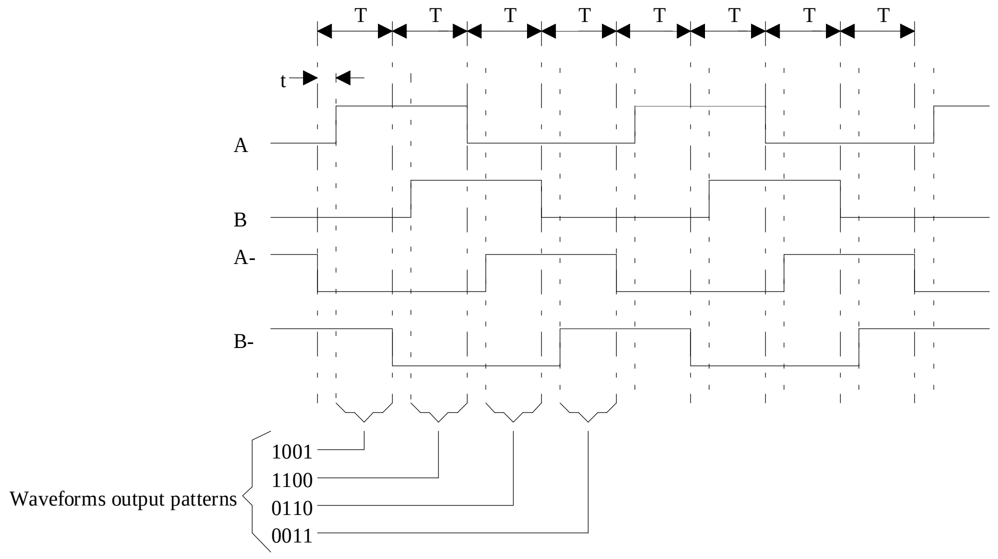


Figure 7

If the stepper motor is driven with the waveforms depicted in figure 7, a constant clockwise stepping motion is generated. If the waveform period T is less, the motor will step through quicker, while if it is longer, so will be the stepping intervals. If T is gradually decreased, the shaft will perform an accelerated rotational motion, while if it is increased, it will decelerate. The full-stepping clockwise sequence corresponds to the 4 data patterns shown on the bottom left in figure 7. This application note provides 2 stepping examples:

<u>**Example 1.**</u> A constant clockwise (CW) and counter-clockwise (CCW) stepping motion at 2 different speeds. The rotor will start stepping clockwise at a slow speed until it rotates 45 degrees, after which it will stop for a set amount of time. Then it will resume stepping for another 45 degrees, and it will stop again. This pattern will repeat 4 times (until a full 360 degrees rotation is completed). Then, the same process will re-start in a counter-clockwise motion (see figure 8), but at a faster speed.

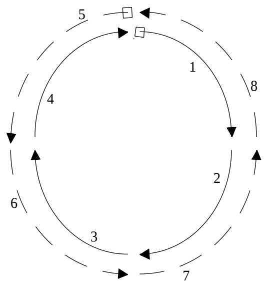


Figure 8

<u>Operation Description:</u>

Channel 0 of ITU is utilized for output compare timing operation. General register A (GRA0) is loaded with the non-overlap time "t", and upon compare-match between GRA0 and the 16-bit timer counter, 2 operations occur:

1. The next output data pattern is loaded from memory table into the Next Data Register B (NDRBof the TPC module.
2. The waveform (step) pulse period T is transferred from its memory location into the General Register B (GRB0).

Upon compare-match between GRB0 and the 16-bit timer counter, the current waveform pattern data is transferred from NDRB into I/O Port B Data Register (PBDR), and thus generated as voltage to the motor coils **after the next compare-match A occurs**. Channels 0A and 0B of the DMA module are setup in the following fashion: DMA0A is used to cyclically transfer the next output pattern data from memory into NDRB, and it is set in the repeat mode of operation. DMA0B is used to load the waveform step time T from its memory location into GRB0. The TPC uses its upper outputs TP 15-12 to provide the 4 excitation waveforms upon compare-match B in channel 0 of the ITU. Figure 10 shows the waveform data patterns corresponding to a CW stepping motion. Since the motor stepping angle is specified to be 7.5 degrees/step (see Appendix A), the number of counts T must be 6 for the rotor to rotate 45 degrees. To perform CCW motion, the output data pattern order must be reversed (i.e., 30-60-C0-90). Also, the CCW stepping speed will be increased by picking a smaller T-value.

<u>**Example 2.**</u> An accelerated/decelerated CW/CCW motion. The shaft will start rotating in an accelerated CW motion until it reaches a certain speed, maintains that speed for a certain amount of time, and then decelerates until it stops. After a pre-determined amount of time, the same process will start in a CCW motion. The figure 9 below illustrates this process.

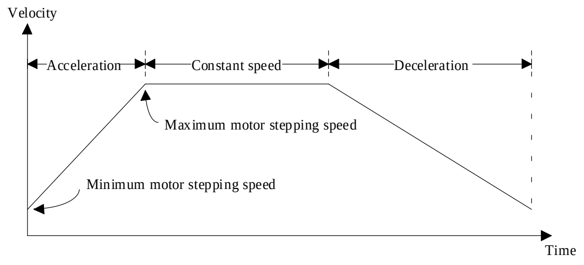


Figure 9

In order to perform both tasks, the H8/3003 is using the Direct Memory Access Controller (DMAC), the Timing Pattern Controller (TPC), and the Integrated Timer Unit (ITU). A verbal description of how the first task is accomplished by the microcontroller is provided below. Figure 10 shows the process in a block diagram format, and figure 11 depicts the waveforms outputs in relation to the ITU and TPC register contents.

<u>Operation Description:</u>

The same general approach as described above is also valid for this example. The difference consists in the fact that the stepping time T will be successively decreasing during motor acceleration, and will be successively increasing during motor deceleration.

During the constant motion, the T-value is constant as well. In this example, both motor acceleration and deceleration is accomplished in "m" successive steps (T1 - Tm). Figure 12 shows the output waveforms in relation to the contents in NDRB, PBDR, and the counting process.

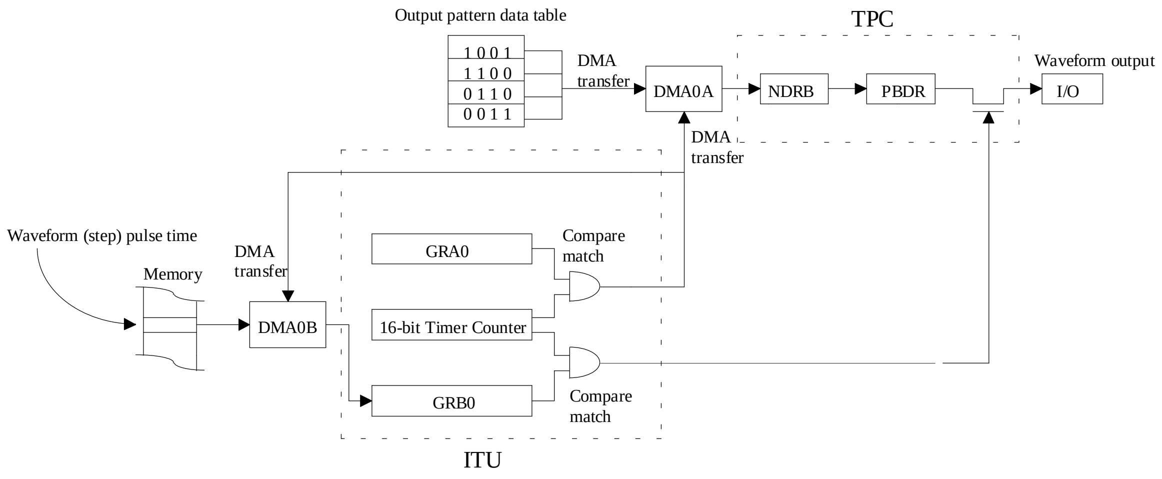


Figure 10

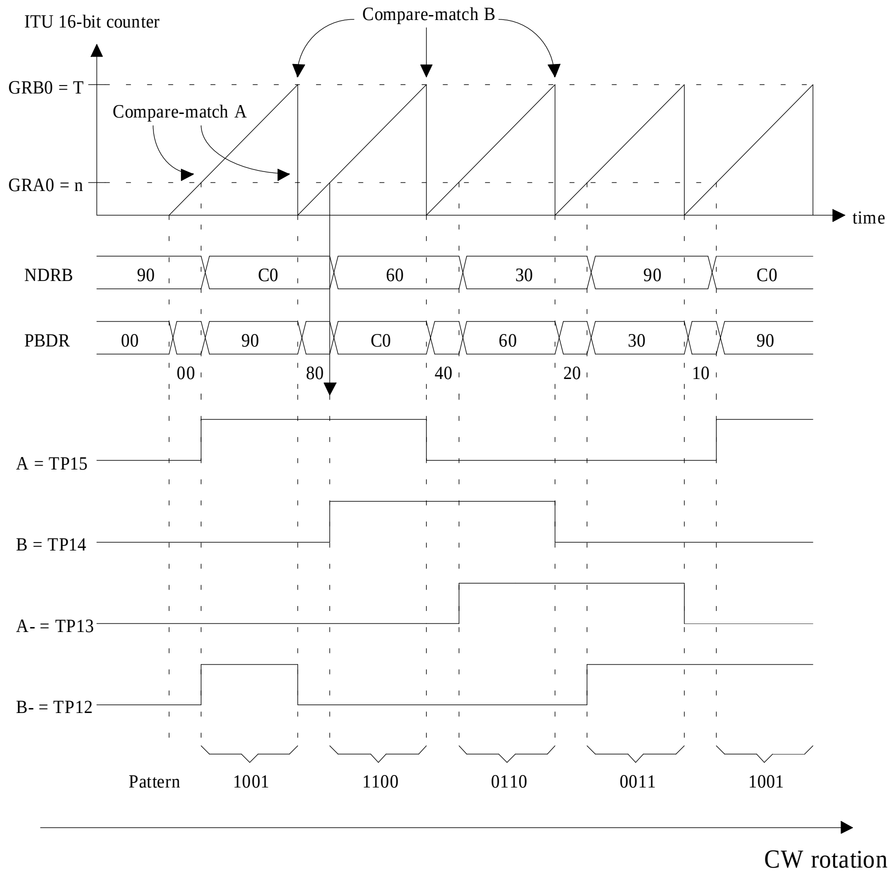


Figure 11

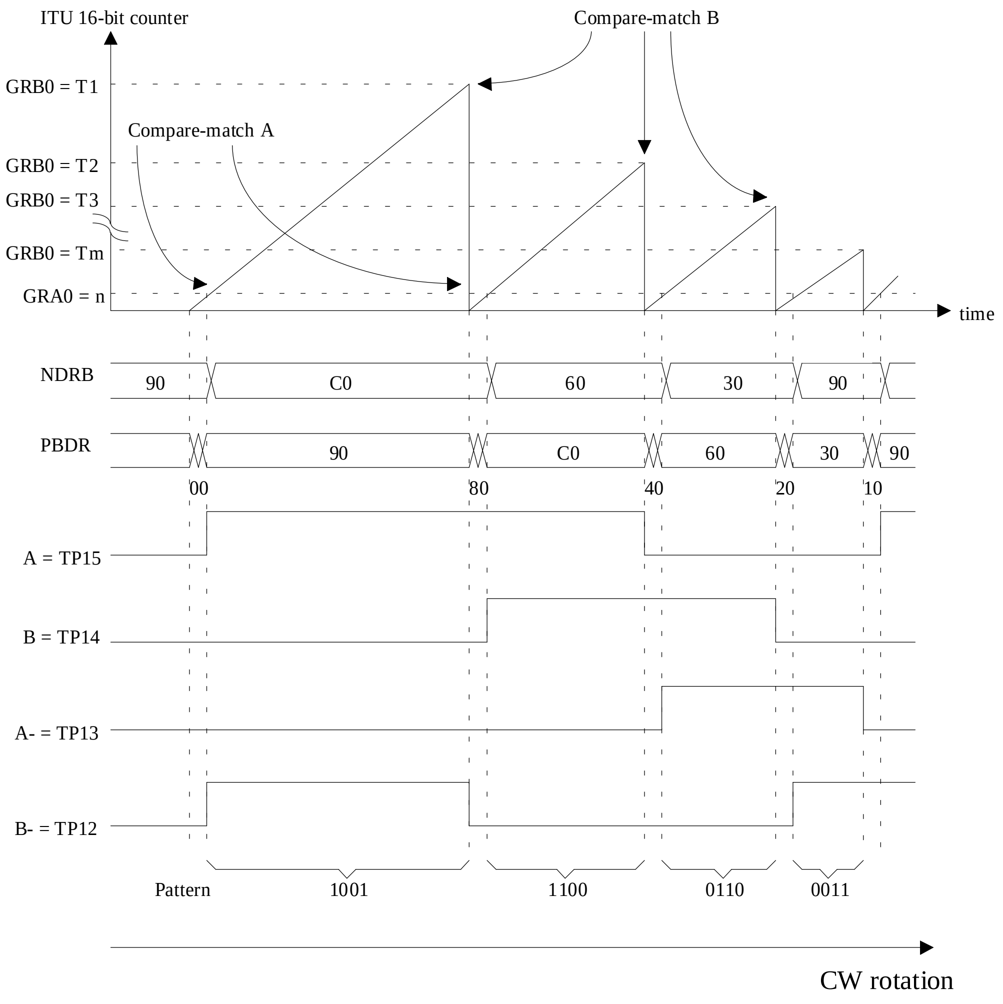


Figure 12

SOFTWARE DESCRIPTION

Two software programs are provided for each of the 2 motor excitation examples described in the previous section. Each consists of 3 main parts:

1. RAM look-up tables for the waveforms output pattern data (4 patterns), and for the stepping pulse (T) values.
2. DMAC, ITU, TPC, and general initializations.
3. An interrupt service routine consisting of several code modules executing various motor motions.

The excitation waveforms output data pattern (shown in figures 10 and 11) is common to both application examples. The stepping speeds table differ. The first example uses a table consisting of only 2 stepping pulses (T), and the second example uses a table of 30 increasing T-values. The on-chip modules initializations are identical for both examples. The TPC is initialized by configuring I/O Port B upper nibble lines (TP15-12) to outputs, and storing the first pattern data into the Next Data Register B (NDRB). Also, non-overlapping outputs in the Timing Pattern Mode Register (TPMR) are selected, and data transfer from NDRB to PBDR is enabled upon a compare-match at channel 0B of the ITU (which is set in the Timing Pattern Controller Register TPCR).

DMA channel 0A is used to transfer the next output data pattern to NDRB upon each compare-match A at channel 0 of the ITU. According to the programming instructions in the hardware manual, the Data Transfer Control Register (DTCR) is read before being written. Then, the channel is set for repeat mode data transfer, and the Memory Address Register (MAR) is set to increment through the RAM data table. The MAR is also initialized with the next address in the output pattern data table. The I/O Address Register A (IOAR0A) is initialized with the address of NDRB. The Execute Transfer Count Registers ETCR0A are loaded with the number of transfers to be executed.

DMA channel 0B is used to transfer the stepping pulse time value(s) T from RAM into General-Purpose Register B (GRB0) of channel 0 of the ITU. DTCR is again read first, and then it is set to indicate idle mode data transfer (for the first programming example), and I/O mode data transfer (for the second programming example). The end-of-transfer interrupt (DTIE) is enabled as well. MAR0B is set with the address (or the start address) of the T-values RAM table, and IOAR0B is loaded with the address of GRB0. The ETCR0B register contains the transfer count corresponding to the number of motor steps.

The ITU uses channel 0 in the output-compare operation mode. The Timer Control Register (TCR0) is configured for timer counter clearance upon compare-match with GRB0 contents, and the counting clock is set to $\Phi/8$. GRA0 is loaded with the non-overlapping counter value, and a compare-match interrupt A is issued when the contents of the timer counter match the preset contents of GRA0.

After the chip peripherals have been initialized, all interrupts are un-masked, and the ITU is started by setting the appropriate bit in the Timer Start Register TSTR. As a result, the motor starts rotating in a clockwise motion; in the first programming example, the motion is constant, while for the second example, the shaft performs an accelerated rotation. Meanwhile, the main program enters an infinite loop waiting for end-of-DMA transfer interrupts to occur, kicking the program into the interrupt service routine. This routine goes through a series of flag checking operations in order to determine what type of motion will the motor perform next.

In the first program example, the operations consist of 4 successive CW constant speed motions of 45° separated by a short pause. Then, 4 successive 45° CCW motions at a higher speed will occur, again separated by a short pause. The motion sequence is determined by the contents of registers R6L and R6H, which are updated in the interrupt service subroutines. R6L contains a value that indicates the next type of motion the rotor will perform (constant clockwise, stop, or constant counter-clockwise). R6H indicates how many times the motor will rotate clockwise or counterclockwise, and its value is decremented after each rotation sequence. After 4 CW or CCW stepping movements, R6H is re-initialized.

The first sequence in the interrupt service routine stops the clockwise movement for a timer count of H'100 by clearing the DTE bit in DTCR0A. Then, DTCR0B is re-initialized, the stopping time (corresponding to the idle DMA transfer count) is loaded into ETCR0BH, and R6H is decremented. If R6H is not zero, R6L is loaded with the next CW motion flag. If R6H is zero, R6L is loaded with the CCW motion flag, and R6H is re-initialized to 4. The constant clockwise motion interrupt subroutine resets DTCR0B, sets the 45° stepping angle by loading a 6 into ETCR0BH, and re-initializes DTCR0A as well. MAR0B is loaded with the slower stepping pulse speed, and R6L is loaded with the stop CW routine flag. The constant counterclockwise motion interrupt routine starts after 4 consecutive counterclockwise steps. DTCR0B and ETCR0BH are re-initialized, and DTCR0A is programmed to decrement MAR0A. MAR0B is loaded with the faster stepping pulse speed, and R6L is loaded with the stop CCW routine flag. Finally, the DTE bits of both DTCR0A and DTCR0B are set to start the DMA transfer process. The first program flowcharts are shown in figures 13-18.

In the second program example, the motor starts accelerating in a clockwise motion until it reaches a constant speed that is maintained for a user-specified amount of time. Then, it starts decelerating until the rotation stops. After another specified amount of time, the rotor starts the same process in a counter-clockwise rotation. The acceleration and deceleration process is achieved in 30 7.5° consecutive steps. The interrupt service routine consists of 8 subroutines. The constant rotation CW and CCW as well as the CW and CCW stop subroutines are programmed in the same fashion as in the corresponding subroutines described for the first program example. R6H is not utilized as a motion indicator flag register. The accelerated rotations are set up in 2 separate subroutines. The initializations are similar as in the constant motions' subroutines, except that DTCR0B is set for I/O transfer mode, and the transfer count loaded into ETCR0B corresponds to 30 steps. In addition, the last output data pattern is checked, and the next output data pattern is loaded into NDRB (depending upon CW or CCW acceleration movement). The decelerating rotations are setup in 2 subroutines as well (for CW and CCW deceleration). Figures 19-28 show the second program flowcharts.

FLOWCHARTS

PROGRAM 1 - MOTOR1

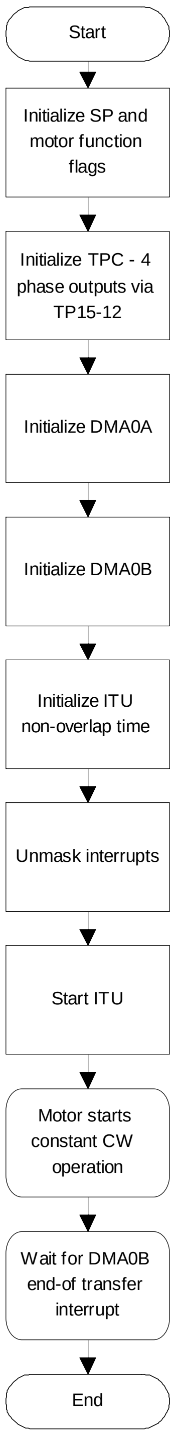

Start
Initialize SP and motor function flags
Initialize TPC - 4 phase outputs via TP15-12
Initialize DMA0A
Initialize DMA0B
Initialize ITU non-overlap time
Unmask interrupts
Start ITU
Motor starts constant CW operation
Wait for DMA0B end-of transfer interrupt
End

Figure 13

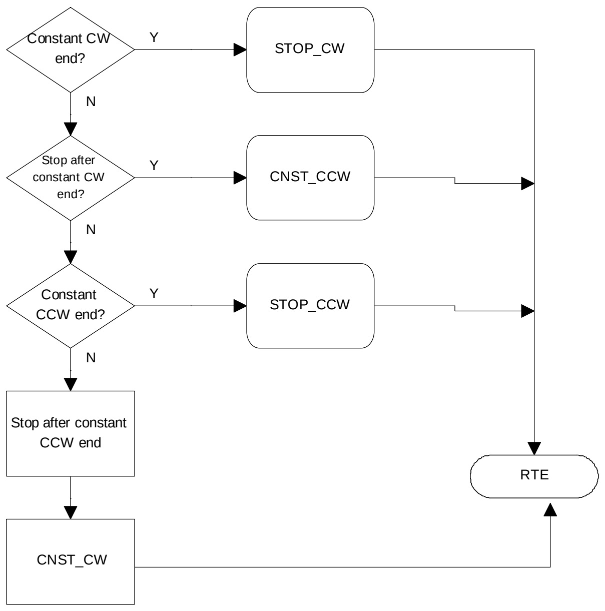
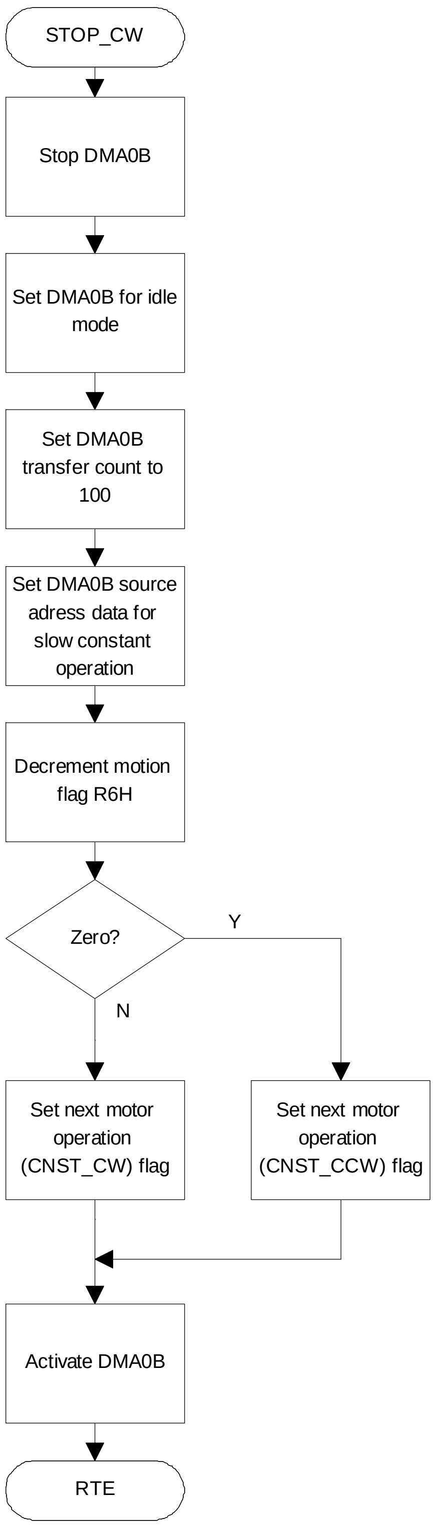


Figure 14


STOP_CW
Stop DMA0B
Set DMA0B for idle mode
Set DMA0B transfer count to 100
Set DMA0B source adress data for slow constant operation
Decrement motion flag R6H
Zero?
Y
N
Set next motor operation (CNST_CW) flag
Set next motor operation (CNST_CCW) flag
Activate DMA0B
RTE

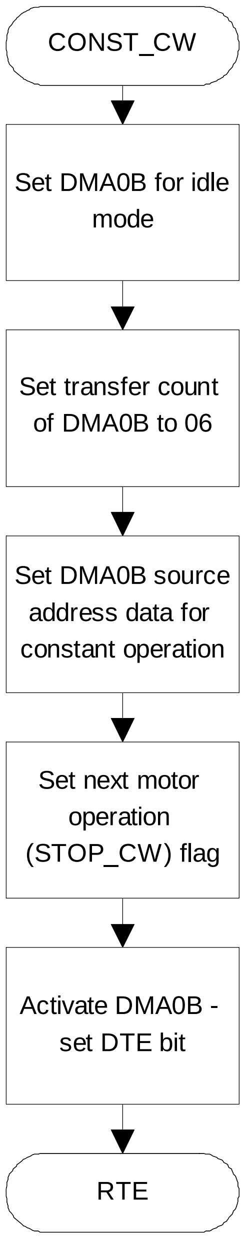

CONST_CW
Set DMA0B for idle mode
Set transfer count of DMA0B to 06
Set DMA0B source address data for constant operation
Set next motor operation (STOP_CW) flag
Activate DMA0B - set DTE bit
RTE

Figure 16

Figure 15

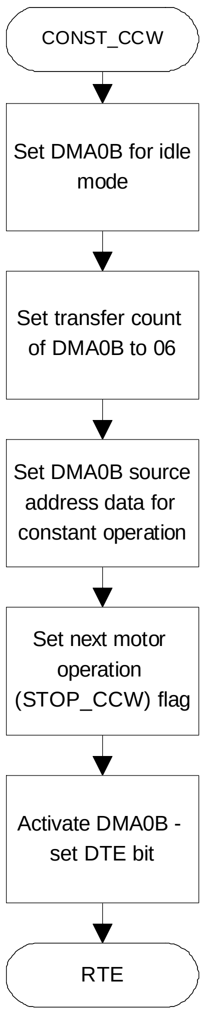

CONST_CCW
Set DMA0B for idle mode
Set transfer count of DMA0B to 06
Set DMA0B source address data for constant operation
Set next motor operation (STOP_CCW) flag
Activate DMA0B - set DTE bit
RTE

Figure 17

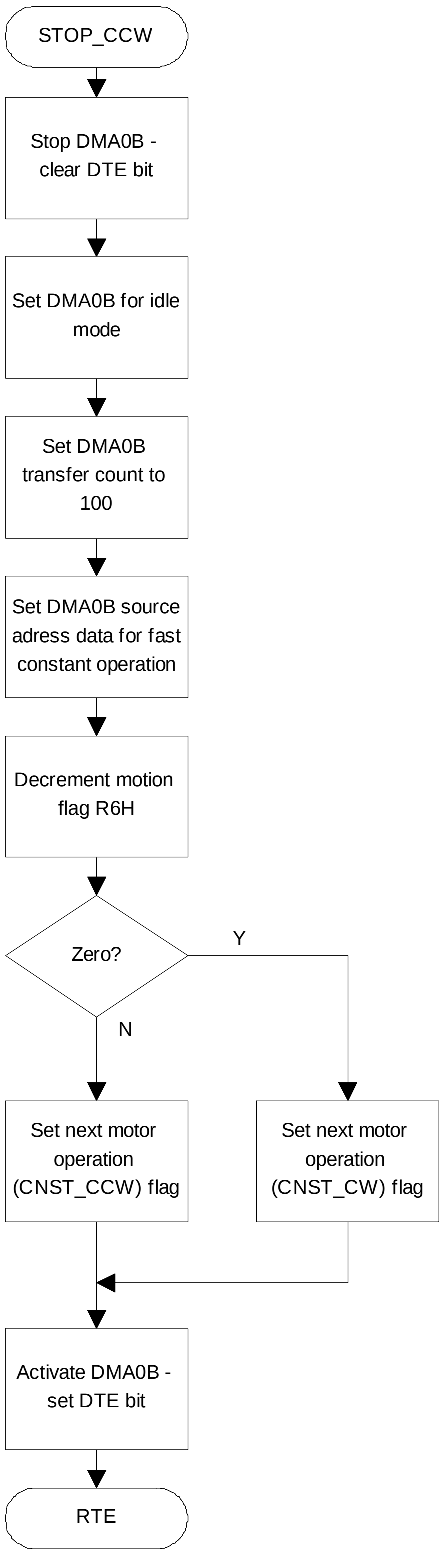

STOP_CCW
Stop DMA0B - clear DTE bit
Set DMA0B for idle mode
Set DMA0B transfer count to 100
Set DMA0B source adress data for fast constant operation
Decrement motion flag R6H
Zero?
Y
N
Set next motor operation (CNST_CCW) flag
Set next motor operation (CNST_CW) flag
Activate DMA0B - set DTE bit
RTE

Figure 18

SECOND PROGRAM - MOTOR2

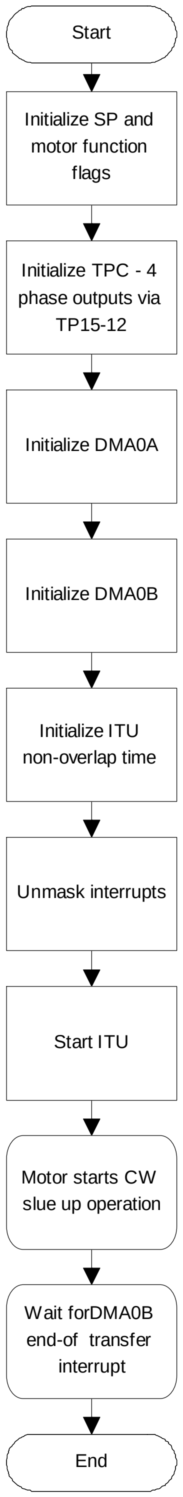

Start
Initialize SP and motor function flags
Initialize TPC - 4 phase outputs via TP15-12
Initialize DMA0A
Initialize DMA0B
Initialize ITU non-overlap time
Unmask interrupts
Start ITU
Motor starts CW slue up operation
Wait forDMA0B end-of transfer interrupt
End

Figure 19

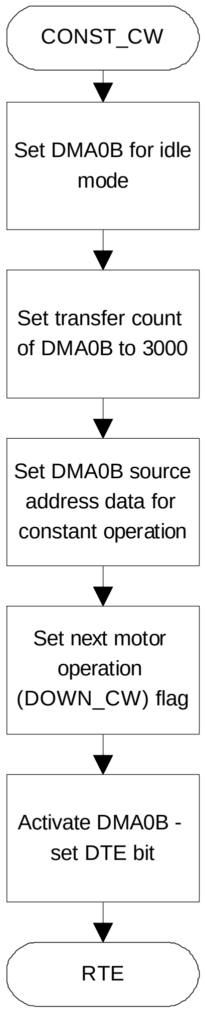

CONST_CW
Set DMA0B for idle mode
Set transfer count of DMA0B to 3000
Set DMA0B source address data for constant operation
Set next motor operation (DOWN_CW) flag
Activate DMA0B - set DTE bit
RTE

Figure 20

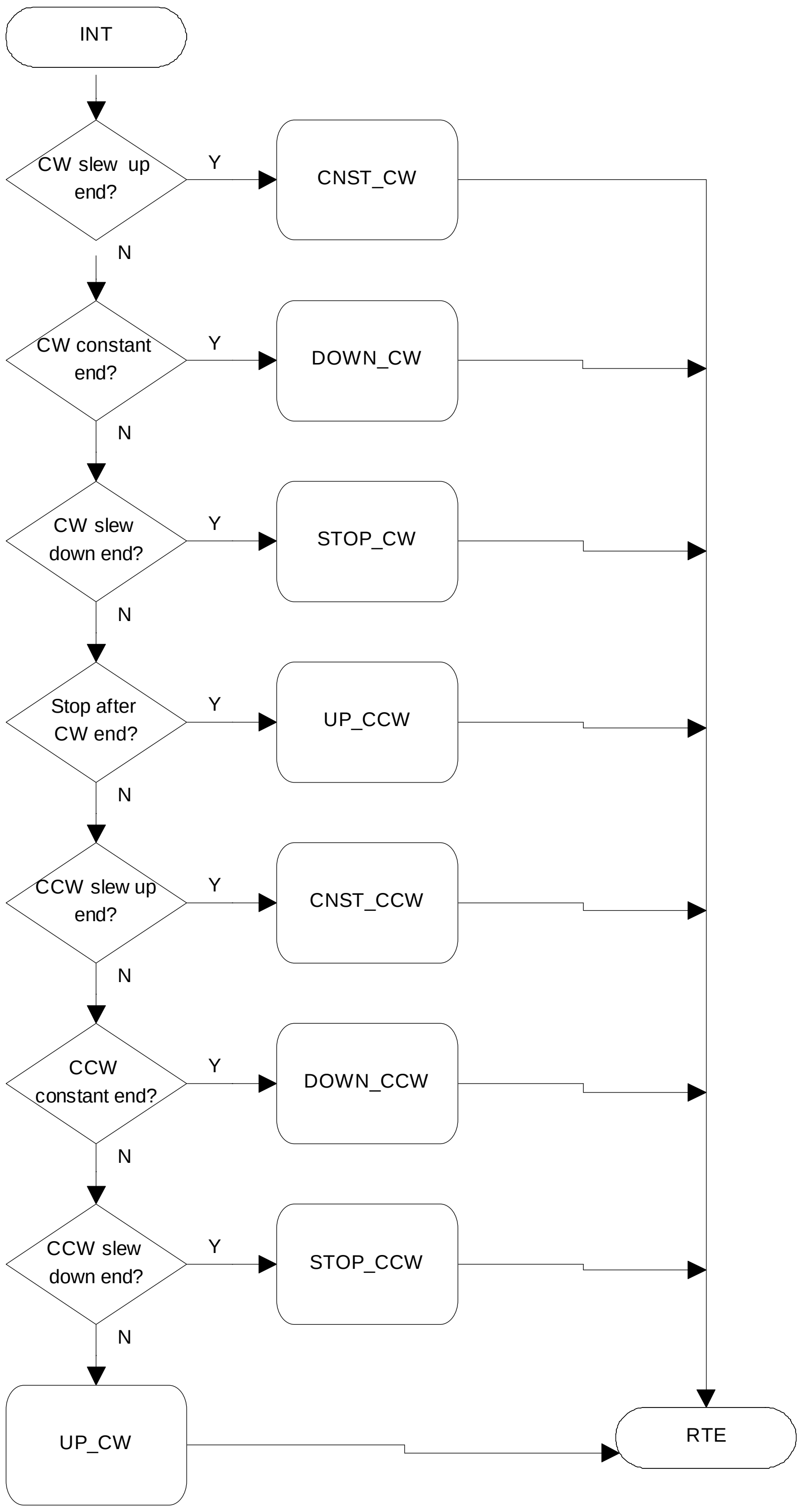

INT
CW slew up end?
Y
CNST_CW
N
CW constant end?
Y
DOWN_CW
N
CW slew down end?
Y
STOP_CW
N
Stop after CW end?
Y
UP_CCW
N
CCW slew up end?
Y
CNST_CCW
N
CCW constant end?
Y
DOWN_CCW
N
CCW slew down end?
Y
STOP_CCW
N
UP_CW
RTE

Figure 21

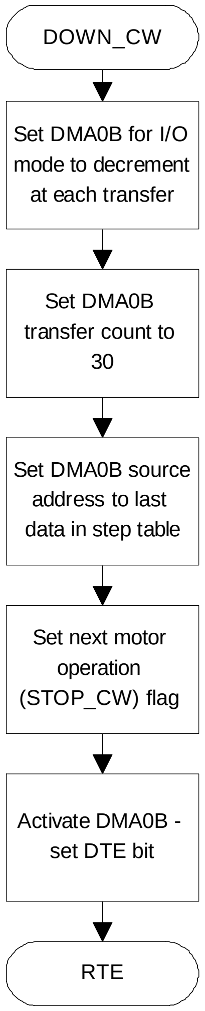

DOWN_CW
Set DMA0B for I/O mode to decrement at each transfer
Set DMA0B transfer count to 30
Set DMA0B source address to last data in step table
Set next motor operation (STOP_CW) flag
Activate DMA0B - set DTE bit
RTE

Figure 22

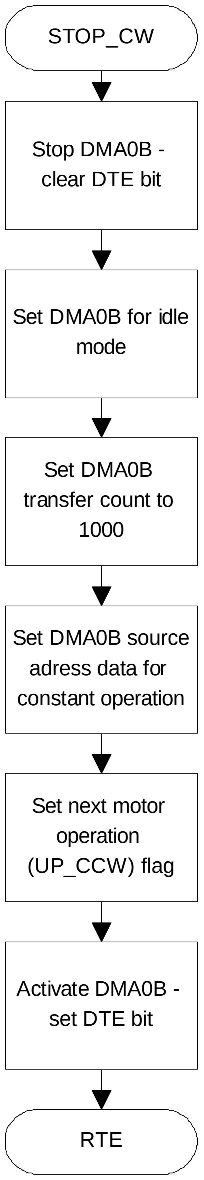

STOP_CW
Stop DMA0B - clear DTE bit
Set DMA0B for idle mode
Set DMA0B transfer count to 1000
Set DMA0B source adress data for constant operation
Set next motor operation (UP_CCW) flag
Activate DMA0B - set DTE bit
RTE

Figure 23

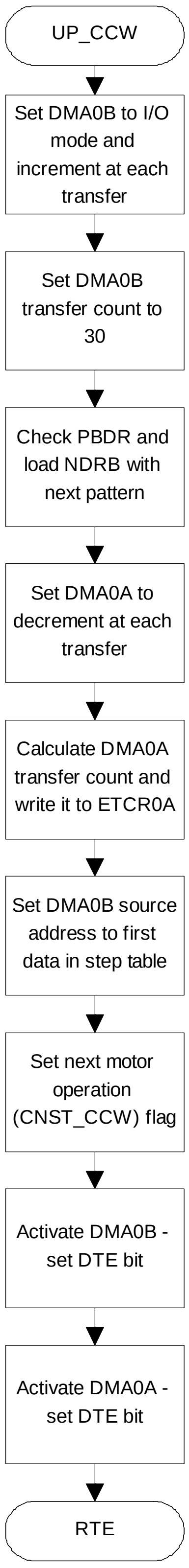

UP_CCW
Set DMA0B to I/O mode and increment at each transfer
Set DMA0B transfer count to 30
Check PBDR and load NDRB with next pattern
Set DMA0A to decrement at each transfer
Calculate DMA0A transfer count and write it to ETCR0A
Set DMA0B source address to first data in step table
Set next motor operation (CNST_CCW) flag
Activate DMA0B - set DTE bit
Activate DMA0A - set DTE bit
RTE

Figure 24

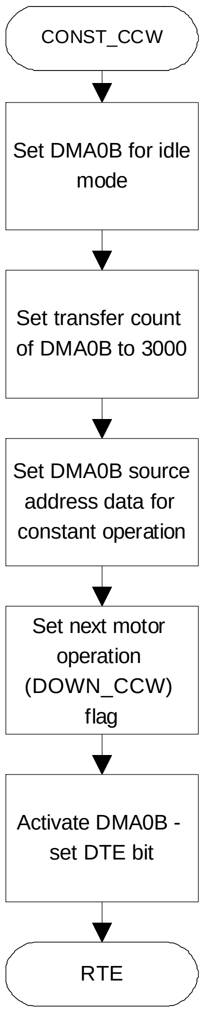

CONST_CCW
Set DMA0B for idle mode
Set transfer count of DMA0B to 3000
Set DMA0B source address data for constant operation
Set next motor operation (DOWN_CCW) flag
Activate DMA0B - set DTE bit
RTE

Figure 25

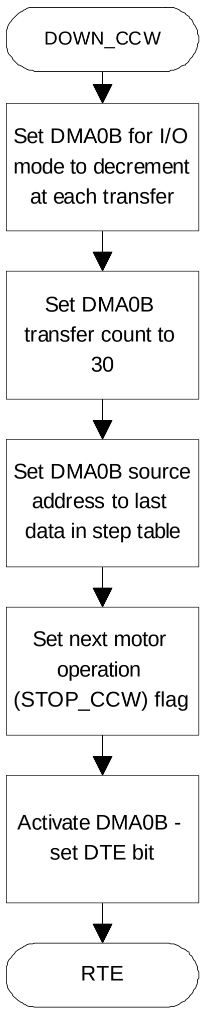

DOWN_CCW
Set DMA0B for I/O mode to decrement at each transfer
Set DMA0B transfer count to 30
Set DMA0B source address to last data in step table
Set next motor operation (STOP_CCW) flag
Activate DMA0B - set DTE bit
RTE

Figure 26

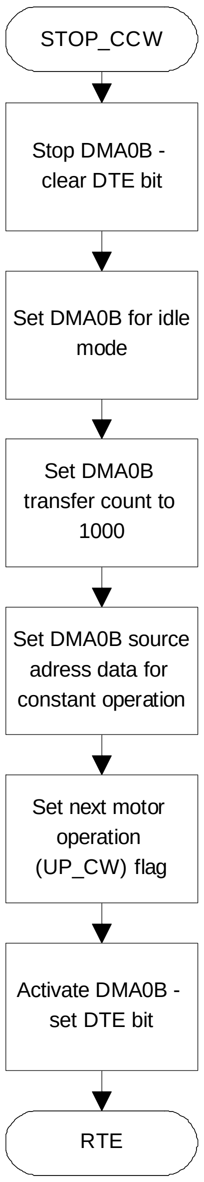

STOP_CCW
Stop DMA0B - clear DTE bit
Set DMA0B for idle mode
Set DMA0B transfer count to 1000
Set DMA0B source adress data for constant operation
Set next motor operation (UP_CW) flag
Activate DMA0B - set DTE bit
RTE

Figure 27

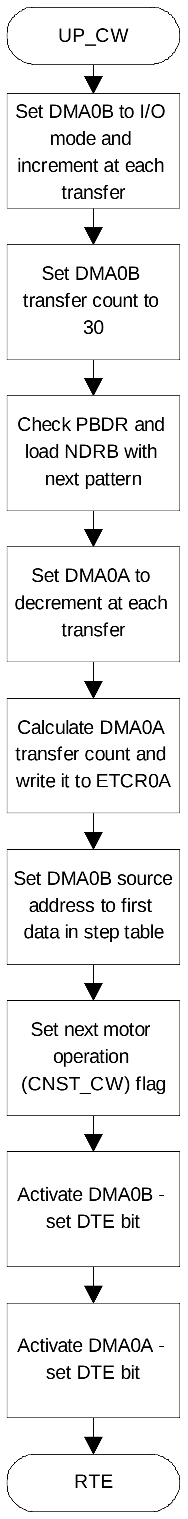

UP_CW
Set DMA0B to I/O mode and increment at each transfer
Set DMA0B transfer count to 30
Check PBDR and load NDRB with next pattern
Set DMA0A to decrement at each transfer
Calculate DMA0A transfer count and write it to ETCR0A
Set DMA0B source address to first data in step table
Set next motor operation (CNST_CW) flag
Activate DMA0B - set DTE bit
Activate DMA0A - set DTE bit
RTE

Figure 28

PROGRAM LISTINGS

FIRST PROGRAM - MOTOR1

```

                         ;**********H8/3003 STEPPER MOTOR IMPLEMENTATION**********

                         ; Symbol definitions

                         ; ITU Channel 0 Registers

       00FFFF6A          GRA0            .equ    H'FFFF6A
       0000006A          GRA0L           .equ    H'6A            ;lower address byte of GRA0
       00FFFF6C          GRB0            .equ    H'FFFF6C
       0000006C          GRB0L           .equ    H'6C            ;lower address byte of GRB0
       00FFFF64          TCR0            .equ    H'FFFF64
       00FFFF66          TIER0           .equ    H'FFFF66
       00FFFF60          TSTR            .equ    H'FFFF60

                         ; TPC Registers

       00FFFFD4          PB_DDR          .equ    H'FFFFD4
       00FFFFD6          PB_DR           .equ    H'FFFFD6
       00FFFFA0          TPMR            .equ    H'FFFFA0
       00FFFFA1          TPCR            .equ    H'FFFFA1
       00FFFFA4          NDRB            .equ    H'FFFFA4
       000000A4          NDRBL           .equ    H'A4            ;lower address byte of NDRB
       00FFFFA2          NDERB           .equ    H'FFFFA2

                         ; DMAC Channel 0 Registers

       00FFFF27          DTCR0A          .equ    H'FFFF27
       00FFFF21          MAR0AE          .equ    H'FFFF21
       00FFFF26          IOAR0A          .equ    H'FFFF26
       00FFFF24          ETCR0AH         .equ    H'FFFF24
       00FFFF25          ETCR0AL         .equ    H'FFFF25
       00FFFF2F          DTCR0B          .equ    H'FFFF2F
       00FFFF29          MAR0BE          .equ    H'FFFF29
       00FFFF2E          IOAR0B          .equ    H'FFFF2E
       00FFFF2C          ETCR0BH         .equ    H'FFFF2C

                         ; Stepping speeds

       00001200          SLOW              .equ    H'001200
       00001202          FAST              .equ    H'001202

                         ; Start Memory Location of Output Data

       00001000          OUT_DATA        .equ    H'001000

                         ; Specify Reset Address

000000                           .org    H'000000
000000 00000000                  .data.l MAIN

                         ; Specify End-of-DMA Transfer Interrupt Address

0000B4                           .org    H'0000B4
0000B4 00000000                  .data.l INT

                         ; Output Pattern Data Table

001000                           .org    H'001000
001000 90C06030                  .data.b H'90,H'C0,H'60,H'30

                         ; Waveforms Stepping Time Table

001200                           .org    H'001200
001200 FFFF5555                  .data.w H'FFFF,H'5555

                         ; Main program starts at H'002000

002000                           .org    H'002000
002000 7A0700FFFF0A      MAIN:   mov.l   #H'FFFF0A,ER7   ;initialize stackpointer
002006 FE03                      mov.b   #H'03,R6L       ;initialize mode counter
002008 F604                      mov.b   #H'04,R6H       ;initialize motion counter

                         ; Initialize TPC

00200A F8F0                      mov.b   #H'F0,R0L       ;configure PB4-7 to outputs
00200C 38D4                      mov.b   R0L,@PB_DDR
00200E 38A2                      mov.b   R0L,@NDERB      ;enable TP15-12 outputs
002010 F890                      mov.b   #H'90,R0L       ;initialize start pulse data in PB
002012 38D6                      mov.b   R0L,@PB_DR
002014 38A4                      mov.b   R0L,@NDRB       ;store pulse data
002016 F808                      mov.b   #H'08,R0L       ;select non-overlapping outputs in
002018 38A0                      mov.b   R0L,@TPMR       ;TP15-12
00201A F800                      mov.b   #H'0,R0L        ;select trigger upon ITU channel 0
```

```
00201C 38A1                   mov.b   R0L,@TPCR       ;compare-match A

                      ; Initialize DMAC0A

00201E 2827                   mov.b   @DTCR0A,R0L     ;read DTCR before initializing
002020 F890                   mov.b   #H'90,R0L       ;enable data transfer, select
002022 3827                   mov.b   R0L,@DTCR0A     ;repeat mode, and increment MAR
002024 7A0000001001           mov.l   #OUT_DATA+1,ER0 ;initialize MAR0A with next pattern
00202A 01006B80FF21           mov.l   ER0,@MAR0AE     ;output data address
002030 F8A4                   mov.b   #NDRBL,R0L      ;initialize IOAR0A with the destination
002032 3826                   mov.b   R0L,@IOAR0A     ;address (NDRB)
002034 F803                   mov.b   #H'03,R0L       ;store first transfer count
002036 3824                   mov.b   R0L,@ETCR0AH
002038 F804                   mov.b   #H'04,R0L       ;store next transfer counts
00203A 3825                   mov.b   R0L,@ETCR0AL

                      ; Initialize DMAC0B

00203C 282F                   mov.b   @DTCR0B,R0L     ;read DTCR before initializing
00203E F8D8                   mov.b   #H'D8,R0L       ;enable data transfer, select idle
002040 382F                   mov.b   R0L,@DTCR0B     ;mode, and transfer-end interrupt
002042 7A0000001200           mov.l   #SLOW,ER0       ;initialize MAR0B with the address
002048 01006B80FF29           mov.l   ER0,@MAR0BE     ;of the slow stepping pulse value
00204E F86C                   mov.b   #GRB0L,R0L      ;initialize IOAR0B with the destination
002050 382E                   mov.b   R0L,@IOAR0B     ;address (GRB)
002052 79000008               mov.w   #D'8,R0         ;store transfer count
002056 6B80FF2C               mov.w   R0,@ETCR0BH

                      ; Initialize ITU

00205A F843                   mov.b   #H'43,R0L       ;counter cleared by GRB compare
00205C 3864                   mov.b   R0L,@TCR0       ;match, and clock/8
00205E F801                   mov.b   #H'01,R0L       ;enable compare-match A interrupt
002060 3866                   mov.b   R0L,@TIER0
002062 7900000A               mov.w   #D'10,R0        ;initialize GRA0 with non-overlap time
002066 6B80FF6A               mov.w   R0,@GRA0

                      ; Unmask interrupts

00206A 0700                   ldc.b   #H'0,CCR

                      ; Start ITU

00206C 7A0000145855           mov.l   #H'145855,ER0   ;initialize motor
002072 1B70           COUNT:  dec.l   #1,ER0
002074 46FC                   bne     COUNT
002076 F9E1                   mov.b   #H'E1,R1L       ;start ITU
002078 3960                   mov.b   R1L,@TSTR

                      ; Wait for end-of-transfer interrupt

00207A 40FE           LOOP:   bra     LOOP
00207C 0000                   nop
00207E 0000                   nop
002080 0000                   nop
002082 0000                   nop

                      ; DMAC End-of Transfer Interrupt Service Routine

                      ; This interrupt routine consists of 8 sequential subroutines:

                      ; STOP_CW  - stops rotation for a while (count=100)
                      ; CNST_CW  - executes constant clockwise (CW) rotation (count=6);
                      ; CNST_CCW - executes constant CCW rotation (count=6)
                      ; STOP_CCW - stops rotation for a while (count=100)


002084 AE03           INT:    cmp.b   #H'03,R6L       ;stop for a while if constant CW
002086 58700012               beq     STOP_CW         ;completed

00208A AE01                   cmp.b   #H'01,R6L       ;perform next constant CW if 3 CCW
00208C 58700044               beq     CNST_CW         ;or a CW stop period completed

002090 AE81                   cmp.b   #H'81,R6L       ;perform next constant CCW if 3 CW
002092 58700068               beq     CNST_CCW        ;or a CCW stop period completed

002096 AE83                   cmp.b   #H'83,R6L       ;stop for a while if constant CCW
002098 5870008C               beq     STOP_CCW

                      ;****************** Stop CW Routine *******************

00209C                STOP_CW:
00209C 7F277270               bclr.b  #7,@DTCR0A      ;stop DMA transfer of pattern data,
0020A0 F858                   mov.b   #H'58,R0L       ;and set for idle mode DMA transfer
0020A2 382F                   mov.b   R0L,@DTCR0B
0020A4 79000064               mov.w   #D'100,R0       ;set transfer count
0020A8 6B80FF2C               mov.w   R0,@ETCR0BH
0020AC 7A0000001200           mov.l   #SLOW,ER0       ;store data address to MAR
0020B2 01006B80FF29           mov.l   ER0,@MAR0BE
0020B8 1A06                   dec.b   R6H             ;decrement motion flag
0020BA 5870000A               beq     START_CCW       ;if 0, set flag for CCW motion
0020BE FE01                   mov.b   #H'01,R6L       ;set CW operation flag
```

```
177 0020C0 7F2F7070          177            bset.b  #7,@DTCR0B      ;start DMA operation in channel B
179 0020C8                   179    START_CCW:
180 0020C8 F604              180            mov.b   #H'04,R6H       ;update motion counter flag
181 0020CA FE81              181            mov.b   #H'81,R6L       ;set CCW operation flag
182 0020CC 7F2F7070          182            bset.b  #7,@DTCR0B      ;start DMA operation in channel B
183 0020D0 5800008C          183            bra     EXIT
184                          184
185                          185    ;****************** Constant CW Routine ********************
186                          186
187 0020D4                   187    CNST_CW:
188 0020D4 F858              188            mov.b   #H'58,R0L       ;set for idle mode DMA transfer
189 0020D6 382F              189            mov.b   R0L,@DTCR0B
190 0020D8 79000008          190            mov.w   #D'8,R0        ;set transfer count
191 0020DC 6B80FF2C          191            mov.w   R0,@ETCR0BH
192 0020E0 F810              192            mov.b   #H'10,R0L       ;enable data transfer, select
193 0020E2 3827              193            mov.b   R0L,@DTCR0A     ;repeat mode, and increment MAR
194 0020E4 7A0000001200      194            mov.l   #SLOW,ER0       ;initialize MAR0B with the address
195 0020EA 01006B80FF29      195            mov.l   ER0,@MAR0BE     ;of the CW stepping pulse time
196 0020F0 FE03              196            mov.b   #H'03,R6L       ;set next motor operation flag
197 0020F2 7F2F7070          197            bset.b  #7,@DTCR0B      ;start both DMA channels
198 0020F6 7F277070          198            bset.b  #7,@DTCR0A
199 0020FA 58000062          199            bra     EXIT
200                          200
201                          201    ;***************** Constant CCW Routine ********************
202                          202
203 0020FE                   203    CNST_CCW:
204 0020FE F858              204            mov.b   #H'58,R0L       ;set for idle mode DMA transfer
205 002100 382F              205            mov.b   R0L,@DTCR0B
206 002102 79000008          206            mov.w   #D'8,R0         ;set transfer count
207 002106 6B80FF2C          207            mov.w   R0,@ETCR0BH
208 00210A F830              208            mov.b   #H'30,R0L       ;enable data transfer, select
209 00210C 3827              209            mov.b   R0L,@DTCR0A     ;repeat mode, and decrement MAR
210 00210E 7A0000001202      210            mov.l   #FAST,ER0       ;initialize MAR0B with the address
211 002114 01006B80FF29      211            mov.l   ER0,@MAR0BE     ;of the slow stepping pulse time
212 00211A FE83              212            mov.b   #H'83,R6L       ;set next motor operation flag
213 00211C 7F2F7070          213            bset.b  #7,@DTCR0B      ;start both DMA channels
214 002120 7F277070          214            bset.b  #7,@DTCR0A
215 002124 58000038          215            bra     EXIT
216                          216
217                          217    ;******************* Stop CCW Routine ********************
218                          218
219 002128                   219    STOP_CCW:
220 002128 7F277270          220            bclr.b  #7,@DTCR0A      ;stop DMA transfer of pattern data
221 00212C F858              221            mov.b   #H'58,R0L       ;set for idle mode DMA transfer
222 00212E 382F              222            mov.b   R0L,@DTCR0B
223 002130 79000064          223            mov.w   #D'100,R0       ;set transfer count
224 002134 6B80FF2C          224            mov.w   R0,@ETCR0BH
225 002138 7A0000001202      225            mov.l   #FAST,ER0       ;store data address to MAR
226 00213E 01006B80FF29      226            mov.l   ER0,@MAR0BE
227 002144 1A06              227            dec.b   R6H             ;decrement motion flag
228 002146 5870000A          228            beq     START_CW        ;if 0, set flag for CW motion
229 00214A FE81              229            mov.b   #H'81,R6L       ;set CCW operation flag
230 00214C 7F2F7070          230            bset.b  #7,@DTCR0B      ;start DMA operation in channel B
231 002150 5800000C          231            bra     EXIT
232 002154                   232    START_CW:
233 002154 F604              233            mov.b   #H'04,R6H       ;update motion counter flag
234 002156 FE01              234            mov.b   #H'01,R6L       ;set CW operation flag
235 002158 7F2F7070          235            bset.b  #7,@DTCR0B      ;start DMA operation in channel B
236 00215C 58000000          236            bra     EXIT
237                          237
238 002160 5670              238    EXIT:   rte                     ;return from interrupt service routine
239                          239            .end
*****TOTAL ERRORS       0
*****TOTAL WARNINGS     0
```

SECOND PROGRAM - MOTOR2

```
*** H8/300 ASSEMBLER Ver.3.2E ***   03/30/94 09:48:48                                                                PAGE     1
PROGRAM NAME =

    1                             1
    2                             2    ;**********H8/3003 STEPPER MOTOR IMPLEMENTATION**********
    3                             3
    4                             4    ; Symbol definitions
    5                             5
    6                             6    ; ITU Channel 0 Registers
    7                             7
    8        00FFFF6A             8    GRA0            .equ    H'FFFF6A
    9        0000006A             9    GRA0L           .equ    H'6A            ;lower address byte of GRA0
   10        00FFFF6C            10    GRB0            .equ    H'FFFF6C
   11        0000006C            11    GRB0L           .equ    H'6C            ;lower address byte of GRB0
   12        00FFFF64            12    TCR0            .equ    H'FFFF64
   13        00FFFF66            13    TIER0           .equ    H'FFFF66
   14        00FFFF60            14    TSTR            .equ    H'FFFF60
   15                            15
   16                            16    ; TPC Registers
   17                            17
   18        00FFFFD4            18    PB_DDR          .equ    H'FFFFD4
   19        00FFFFD6            19    PB_DR           .equ    H'FFFFD6
   20        00FFFFA0            20    TPMR            .equ    H'FFFFA0
   21        00FFFFA1            21    TPCR            .equ    H'FFFFA1
   22        00FFFFA4            22    NDRB            .equ    H'FFFFA4
   23        000000A4            23    NDRBL           .equ    H'A4            ;lower address byte of NDRB
   24        00FFFFA2            24    NDERB           .equ    H'FFFFA2
   25                            25
   26                            26    ; DMAC Channel 0 Registers
   27                            27
   28        00FFFF27            28    DTCR0A          .equ    H'FFFF27
   29        00FFFF21            29    MAR0AE          .equ    H'FFFF21
   30        00FFFF26            30    IOAR0A          .equ    H'FFFF26
   31        00FFFF24            31    ETCR0AH         .equ    H'FFFF24
   32        00FFFF25            32    ETCR0AL         .equ    H'FFFF25
   33        00FFFF2F            33    DTCR0B          .equ    H'FFFF2F
   34        00FFFF29            34    MAR0BE          .equ    H'FFFF29
   35        00FFFF2E            35    IOAR0B          .equ    H'FFFF2E
   36        00FFFF2C            36    ETCR0BH         .equ    H'FFFF2C
   37                            37
   38                            38    ; Slue Speed
   39                            39
   40        00001200            40    UP              .equ    H'001200
   41        0000123A            41    DOWN            .equ    H'00123A
   42        0000123C            42    CONSTANT        .equ    H'00123C
   43                            43
   44                            44    ; Start Memory Location of Output Data
   45                            45
   46        00001000            46    OUT_DATA        .equ    H'001000
   47                            47
   48                            48    ; Specify Reset Address
   49                            49
   50 000000                     50            .org    H'000000
   51 000000 00000000            51            .data.l MAIN
   52                            52
   53                            53    ; Specify End-of-DMA Transfer Interrupt Address
   54                            54
   55 0000B4                     55            .org    H'0000B4
   56 0000B4 00000000            56            .data.l INT
   57                            57
   58                            58    ; Output Pattern Data Table
   59                            59
   60 001000                     60            .org    H'001000
   61 001000 90C06030            61            .data.b H'90,H'C0,H'60,H'30
   62                            62
   63                            63    ; Output Waveforms Period Table
   64                            64
   65                            65    ; Slew Up and Down Data Table
   66                            66
   67 001200                     67            .org    H'001200
   68                            68    ;*************************** UP **************************
   69 001200 F4246EA3550147B3    69            .data.w D'62500,D'28323,D'21761,D'18355,D'16175
      001208 3F2F
   70 00120A 3922348C30EA2DF2    70            .data.w D'14626,D'13452,D'12522,D'11762,D'11125
      001212 2B75
   71 001214 2956277F25E22474    71            .data.w D'10582,D'10111,D'9698,D'9332,D'9005,D'8709
      00121C 232D2205
   72 001220 20F920041F241E55    72            .data.w D'8441,D'8196,D'7972,D'7765,D'7573,D'7395
      001228 1D951CE3
   73 00122C 1C3D1BA11B0F1A86    73            .data.w D'7229,D'7073,D'6927,D'6790,D'6661,D'6539
      001234 1A05198B
   74 001238 191718A9            74            .data.w D'6423,D'6313
   75                            75    ;************************* DOWN ************************
   76                            76
   77                            77    ; Constant Waveforms Period
   78                            78
   79                            79    ;*********************** CONSTANT **********************
   80 00123C 18A9                80            .data.w D'6313
```

```
                            
                            ; Main program starts at H'002000
                            
002000                             .org    H'002000
002000 7A0700FFFF0A         MAIN:   mov.l   #H'FFFF0A,ER7   ;initialize stackpointer
002006 FE01                         mov.b   #H'01,R6L       ;initialize mode counter
                            
                            ; Initialize TPC
                            
002008 F8F0                         mov.b   #H'F0,R0L       ;configure PB4-7 to outputs
00200A 38D4                         mov.b   R0L,@PB_DDR
00200C 38A2                         mov.b   R0L,@NDERB      ;enable TP15-12 outputs
00200E F890                         mov.b   #H'90,R0L       ;initialize start pulse data in PB
002010 38D6                         mov.b   R0L,@PB_DR
002012 38A4                         mov.b   R0L,@NDRB       ;store pulse data
002014 F808                         mov.b   #H'08,R0L       ;select non-overlapping outputs in
002016 38A0                         mov.b   R0L,@TPMR       ;TP15-12
002018 F800                         mov.b   #H'0,R0L        ;select trigger upon ITU channel 0
00201A 38A1                         mov.b   R0L,@TPCR       ;compare-match A
                            
                            ; Initialize DMAC0A
                            
00201C 2827                         mov.b   @DTCR0A,R0L     ;read DTCR before initializing
00201E F890                         mov.b   #H'90,R0L       ;enable data transfer, select
002020 3827                         mov.b   R0L,@DTCR0A     ;repeat mode, and increment MAR
002022 7A0000001001                 mov.l   #OUT_DATA+1,ER0 ;initialize MAR0A with next pattern
002028 01006B80FF21                 mov.l   ER0,@MAR0AE     ;output data address
00202E F8A4                         mov.b   #NDRBL,R0L      ;initialize IOAR0A with the destination
002030 3826                         mov.b   R0L,@IOAR0A     ;address (NDRB)
002032 F803                         mov.b   #H'03,R0L       ;store first transfer count
002034 3824                         mov.b   R0L,@ETCR0AH
002036 F804                         mov.b   #H'04,R0L       ;store next transfer counts
002038 3825                         mov.b   R0L,@ETCR0AL
                            
                            ; Initialize DMAC0B
                            
00203A 282F                         mov.b   @DTCR0B,R0L     ;read DTCR before initializing
00203C F8C8                         mov.b   #H'C8,R0L       ;enable data transfer, select I/O
00203E 382F                         mov.b   R0L,@DTCR0B     ;mode, and transfer-end interrupt
002040 7A0000001200                 mov.l   #UP,ER0         ;initialize MAR0B with the start
002046 01006B80FF29                 mov.l   ER0,@MAR0BE     ;address in the slue up data table
00204C F86C                         mov.b   #GRB0L,R0L      ;initialize IOAR0B with the destination
00204E 382E                         mov.b   R0L,@IOAR0B     ;address (GRB)
002050 7900001E                     mov.w   #D'30,R0        ;store slue up transfer count
002054 6B80FF2C                     mov.w   R0,@ETCR0BH
                            
                            ; Initialize ITU
                            
002058 F843                         mov.b   #H'43,R0L       ;counter cleared by GRB compare
00205A 3864                         mov.b   R0L,@TCR0       ;match, and clock/8
00205C F801                         mov.b   #H'01,R0L       ;enable compare-match A interrupt
00205E 3866                         mov.b   R0L,@TIER0
002060 7900000A                     mov.w   #D'10,R0        ;initialize GRA0 with non-overlap time
002064 6B80FF6A                     mov.w   R0,@GRA0
                            
                            ; Unmask interrupts
                            
002068 0700                         ldc.b   #H'0,CCR
                            
                            ; Start ITU
                            
00206A 7A0000145855                 mov.l   #H'145855,ER0   ;initialize motor
002070 1B70                 COUNT:  dec.l   #1,ER0
002072 46FC                         bne     COUNT
002074 F9E1                         mov.b   #H'E1,R1L       ;start ITU
002076 3960                         mov.b   R1L,@TSTR
                            
                            ; Wait for end-of-transfer interrupt
                            
002078 40FE                 LOOP:   bra     LOOP
00207A 0000                         nop
00207C 0000                         nop
00207E 0000                         nop
002080 0000                         nop
                            
                            ; DMAC End-of Transfer Interrupt Service Routine
                            
                            ; This interrupt routine consists of 8 sequential subroutines:
                            
                            ; CNST_CW  - executes constant clockwise (CW) rotation (count=3000)
                            ; DOWN_CW  - executes CW slue down (count=60)
                            ; STOP_CW  - stops rotation for a while (count=1000)
                            ; UP_CCW   - executes counter clockwise (CCW) slue up (count=60)
                            ; CNST_CCW - executes constant CCW rotation (count=3000)
                            ; DOWN_CCW - executes CCW slue down (count=60)
                            ; STOP_CCW - stops rotation for a while (count=1000)
                            ; UP_CW    - executes CW slue up (count=60)
                            
002082 AE01                 INT:    cmp.b   #H'01,R6L       ;perform constant CW rotation
002084 5870002A                     beq     CNST_CW         ;if slue up completed
                            
002088 AE02                         cmp.b   #H'02,R6L       ;perform slue down CW if
```

```
00208A 58700046                 beq     DOWN_CW         ;constant rotation completed

00208E AE03                     cmp.b   #H'03,R6L       ;stop for a while if CW slue
002090 58700062                 beq     STOP_CW         ;down completed

002094 AE00                     cmp.b   #H'0,R6L        ;perform slue up CCW if stop
002096 58700082                 beq     UP_CCW          ;period completed

00209A AE81                     cmp.b   #H'81,R6L       ;perform constant CCW rotation
00209C 587000E2                 beq     CNST_CCW        ;if slue up completed

0020A0 AE82                     cmp.b   #H'82,R6L       ;perform CCW slue down if
0020A2 587000FE                 beq     DOWN_CCW        ;constant rotation completed

0020A6 AE83                     cmp.b   #H'83,R6L       ;stop for a while if CCW slue
0020A8 5870011A                 beq     STOP_CCW        ;down completed

0020AC AE80                     cmp.b   #H'80,R6L       ;perform next CW slue up if
0020AE 5870013A                 beq     UP_CW           ;stop period completed

                        ;************ Constant CW Rotation Routine *****************

0020B2                  CNST_CW:
0020B2 F858                     mov.b   #H'58,R0L       ;set for idle mode DMA transfer
0020B4 382F                     mov.b   R0L,@DTCR0B
0020B6 79000BB8                 mov.w   #D'3000,R0      ;set transfer count
0020BA 6B80FF2C                 mov.w   R0,@ETCR0BH
0020BE 7A000000123C             mov.l   #CONSTANT,ER0   ;store waveform period data address
0020C4 01006B80FF29             mov.l   ER0,@MAR0BE     ;into MAR
0020CA FE02                     mov.b   #H'02,R6L       ;set next motor operation flag
0020CC 7F2F7070                 bset.b  #7,@DTCR0B      ;start DMA transfer in channel B
0020D0 5800017E                 bra     EXIT

                        ;**************** Slue Down CW Routine *********************

0020D4                  DOWN_CW:
0020D4 F868                     mov.b   #H'68,R0L       ;set for I/O mode DMA transfer and
0020D6 382F                     mov.b   R0L,@DTCR0B     ;decrement MAR
0020D8 7900001E                 mov.w   #D'30,R0        ;set transfer count
0020DC 6B80FF2C                 mov.w   R0,@ETCR0BH
0020E0 7A000000123A             mov.l   #DOWN,ER0       ;store start address of waveform
0020E6 01006B80FF29             mov.l   ER0,@MAR0BE     ;period table
0020EC FE03                     mov.b   #H'03,R6L       ;set next motor operation flag
0020EE 7F2F7070                 bset.b  #7,@DTCR0B      ;start DMA process in channel B
0020F2 5800015C                 bra     EXIT

                        ;******************** Stop CW Routine **********************

0020F6                  STOP_CW:
0020F6 7F277270                 bclr.b  #7,@DTCR0A      ;stop DMA transfer of pattern data
0020FA F858                     mov.b   #H'58,R0L       ;set for idle mode DMA transfer
0020FC 382F                     mov.b   R0L,@DTCR0B
0020FE 790003E8                 mov.w   #D'1000,R0      ;set transfer count
002102 6B80FF2C                 mov.w   R0,@ETCR0BH
002106 7A000000123C             mov.l   #CONSTANT,ER0   ;store data address to MAR
00210C 01006B80FF29             mov.l   ER0,@MAR0BE
002112 FE00                     mov.b   #H'00,R6L       ;set next motor operation flag
002114 7F2F7070                 bset.b  #7,@DTCR0B      ;start DMA operation in channel B
002118 58000136                 bra     EXIT

                        ;****************** Slue Up CCW Routine ********************

00211C                  UP_CCW:
00211C F848                     mov.b   #H'48,R0L       ;set for I/O mode DMA transfer and
00211E 382F                     mov.b   R0L,@DTCR0B     ;increment MAR
002120 7900001E                 mov.w   #D'30,R0        ;set transfer count
002124 6B80FF2C                 mov.w   R0,@ETCR0BH
002128 7EA47370                 btst.b  #7,@NDRB        ;check output data pattern
00212C 58600010                 bne     P1_2_CCW        ;if not zero, pattern 1 or 2
002130 7EA47360                 btst.b  #6,@NDRB        ;check pattern again
002134 58600020                 bne     P3_CCW          ;if not zero, pattern 3
002138 F860                     mov.b   #H'60,R0L       ;if zero, pattern 4
00213A 38A4                     mov.b   R0L,@NDRB       ;load NDRB with pattern 3
00213C 5800001C                 bra     CONT
002140                  P1_2_CCW:
002140 7EA47360                 btst.b  #6,@NDRB        ;pattern 1 or 2?
002144 58600008                 bne     P2_CCW          ;if not zero, pattern 2
002148 F830                     mov.b   #H'30,R0L       ;if zero, pattern 1
00214A 38A4                     mov.b   R0L,@NDRB       ;load NDRB with pattern 4
00214C 5800000C                 bra     CONT
002150 F890             P2_CCW: mov.b   #H'90,R0L       ;pattern 2
002152 38A4                     mov.b   R0L,@NDRB       ;load NDRB with pattern 1
002154 58000004                 bra     CONT
002158 F8C0             P3_CCW: mov.b   #H'C0,R0L       ;pattern 3
00215A 38A4                     mov.b   R0L,@NDRB       ;load NDRB with pattern 2
00215C F8B0             CONT:   mov.b   #H'B0,R0L       ;enable data transfer, select
00215E 3827                     mov.b   R0L,@DTCR0A     ;repeat mode, and decrement MAR
002160 2024                     mov.b   @ETCR0AH,R0H    ;store current transfer count into R0H
002162 F805                     mov.b   #D'5,R0L
002164 1808                     sub.b   R0H,R0L         ;determine new transfer count
002166 3824                     mov.b   R0L,@ETCR0AH
002168 7A0000001200             mov.l   #UP,ER0         ;initialize MAR0B with the start address
```

```
00216E 01006B80FF29          mov.l   ER0,@MAR0BE     ;of the slue up data table
002174 FE81                  mov.b   #H'81,R6L       ;set next motor operation flag
002176 7F2F7070              bset.b  #7,@DTCR0B      ;start both DMA channels
00217A 7F277070              bset.b  #7,@DTCR0A
00217E 580000D0              bra     EXIT

                     ;************ Constant CCW Rotation Routine *****************

002182               CNST_CCW:
002182 F858                  mov.b   #H'58,R0L       ;set for idle mode DMA transfer
002184 382F                  mov.b   R0L,@DTCR0B
002186 79000BB8              mov.w   #D'3000,R0      ;set transfer count
00218A 6B80FF2C              mov.w   R0,@ETCR0BH
00218E 7A000000123C          mov.l   #CONSTANT,ER0   ;store waveform period data address
002194 01006B80FF29          mov.l   ER0,@MAR0BE     ;into MAR
00219A FE82                  mov.b   #H'82,R6L       ;set next motor operation flag
00219C 7F2F7070              bset.b  #7,@DTCR0B      ;start DMA transfer in channel B
0021A0 580000AE              bra     EXIT

                     ;**************** Slue Down CCW Routine *********************

0021A4               DOWN_CCW:
0021A4 F868                  mov.b   #H'68,R0L       ;set for I/O mode DMA transfer and
0021A6 382F                  mov.b   R0L,@DTCR0B     ;decrement MAR
0021A8 7900001E              mov.w   #D'30,R0        ;set transfer count
0021AC 6B80FF2C              mov.w   R0,@ETCR0BH
0021B0 7A000000123A          mov.l   #DOWN,ER0       ;store start address of waveform
0021B6 01006B80FF29          mov.l   ER0,@MAR0BE     ;period table
0021BC FE83                  mov.b   #H'83,R6L       ;set next motor operation flag
0021BE 7F2F7070              bset.b  #7,@DTCR0B      ;start DMA process in channel B
0021C2 5800008C              bra     EXIT

                     ;******************** Stop CCW Routine **********************

0021C6               STOP_CCW:
0021C6 7F277270              bclr.b  #7,@DTCR0A      ;stop DMA transfer of pattern data
0021CA F858                  mov.b   #H'58,R0L       ;set for idle mode DMA transfer
0021CC 382F                  mov.b   R0L,@DTCR0B
0021CE 790003E8              mov.w   #D'1000,R0      ;set transfer count
0021D2 6B80FF2C              mov.w   R0,@ETCR0BH
0021D6 7A000000123C          mov.l   #CONSTANT,ER0   ;store data address to MAR
0021DC 01006B80FF29          mov.l   ER0,@MAR0BE
0021E2 FE80                  mov.b   #H'80,R6L       ;set next motor operation flag
0021E4 7F2F7070              bset.b  #7,@DTCR0B      ;start DMA operation in channel B
0021E8 58000066              bra     EXIT

                     ;****************** Slue Up CW Routine ********************

0021EC               UP_CW:
0021EC F848                  mov.b   #H'48,R0L       ;set for I/O mode DMA transfer and
0021EE 382F                  mov.b   R0L,@DTCR0B     ;increment MAR
0021F0 7900001E              mov.w   #D'30,R0        ;set transfer count
0021F4 6B80FF2C              mov.w   R0,@ETCR0BH
0021F8 7EA47370              btst.b  #7,@NDRB        ;check output data pattern
0021FC 58600010              bne     P1_2_CW         ;if not zero, pattern 1 or 2
002200 7EA47360              btst.b  #6,@NDRB        ;check pattern again
002204 58600020              bne     P3_CW           ;if not zero, pattern 3
002208 F890                  mov.b   #H'90,R0L       ;if zero, pattern 1
00220A 38A4                  mov.b   R0L,@NDRB       ;load NDRB with pattern 3
00220C 5800001C              bra     CONT1
002210               P1_2_CW:
002210 7EA47360              btst.b  #6,@NDRB        ;pattern 1 or 2?
002214 58600008              bne     P2_CW           ;if not zero, pattern 2
002218 F8C0                  mov.b   #H'C0,R0L       ;if zero, pattern 1
00221A 38A4                  mov.b   R0L,@NDRB       ;load NDRB with pattern 2
00221C 5800000C              bra     CONT1
002220 F860          P2_CW:  mov.b   #H'60,R0L       ;pattern 2
002222 38A4                  mov.b   R0L,@NDRB       ;load NDRB with pattern 3
002224 58000004              bra     CONT1
002228 F830          P3_CW:  mov.b   #H'30,R0L       ;pattern 3
00222A 38A4                  mov.b   R0L,@NDRB       ;load NDRB with pattern 4
00222C F890          CONT1:  mov.b   #H'90,R0L       ;enable data transfer, select
00222E 3827                  mov.b   R0L,@DTCR0A     ;repeat mode, and increment MAR
002230 2024                  mov.b   @ETCR0AH,R0H    ;store current transfer count into R0H
002232 F805                  mov.b   #D'5,R0L
002234 1808                  sub.b   R0H,R0L         ;determine new transfer count
002236 3824                  mov.b   R0L,@ETCR0AH
002238 7A0000001200          mov.l   #UP,ER0         ;initialize MAR0B with the start address
00223E 01006B80FF29          mov.l   ER0,@MAR0BE     ;of the slue up data table
002244 FE01                  mov.b   #H'01,R6L       ;set next motor operation flag
002246 7F2F7070              bset.b  #7,@DTCR0B      ;start both DMA channels
00224A 7F277070              bset.b  #7,@DTCR0A
00224E 58000000              bra     EXIT

002252 5670          EXIT:   rte                     ;return from interrupt service routine
                             .end
*****TOTAL ERRORS       0
*****TOTAL WARNINGS     0
```

APPENDIX A

Motor Standard Specifications

Model	KP6P8
Phase	4
Step Angle (degrees/step)	7.5
Voltage (V)	12
Current (Amps/phase)	0.33
Resistance (Ohms/phase)	36
Inductance (mH/phase)	28
Holding Torque (gf • cm)	1100
Detent Torque (gf • cm)	160
Rotor Inertia (g • cm^2)	23.7
Weight (kg)	0.25
Insulation Class	E
Ambient Temperature (°C)	-10 ~ 45
Temperature Rise (degrees)	70
Lead Specification	AWG #22

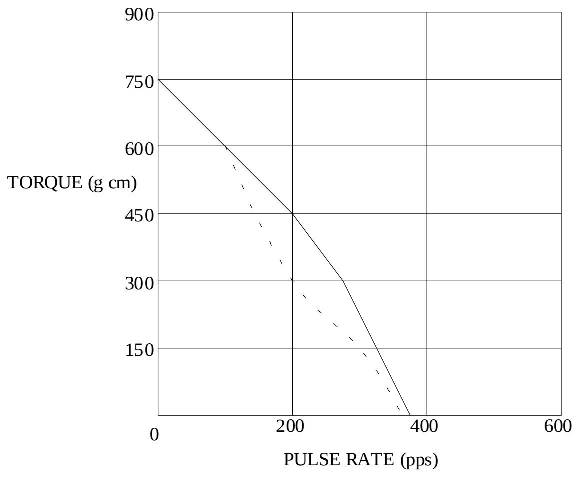


Figure 29