H8/300

Using a H8/300 to control a stepper motor

HITACHI

AE-0057
Rev 0
September 1994
Cristian Tomescu

Introduction

The H8/300 family of microcontrollers are multi-purpose, 8-bit data path devices used in a multitude of applications that require a low-cost controller with adequate performance and a high mix of peripheral integration. In particular, versatile timer modules with built-in comparators and capture functions able to run at various frequencies are needed in virtually most control applications. All H8/300 family members possess a 16-bit free-running counter used primarily to generate various frequency and duty-cycle rectangular waveforms through its 2 output channels. Even if an application requires more than 2 control output signals, the timer can still be utilized in determining the pulse width of the output waveforms, while the signals can be generated at various I/O lines.   An example of such situations can be found in various motion control applications that require multiple phase excitation waveforms for activation.

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 H8/330 can be utilized to control the operation of a stepping motor. Although the H8/330 was the chosen device, any member of the H8/300 Family could be used in the process (with the obvious exception of the H8/310 and H8/3101 that lack a timer module). A simple motion control embedded system can be designed utilizing only a H8/330 controller, a stepper motor, and the motor-specific required control circuitry (consisting of power FET transistors, clamping diodes, and current-limiting high-power resistors).

Using a H8/300 to control a stepper motor

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.

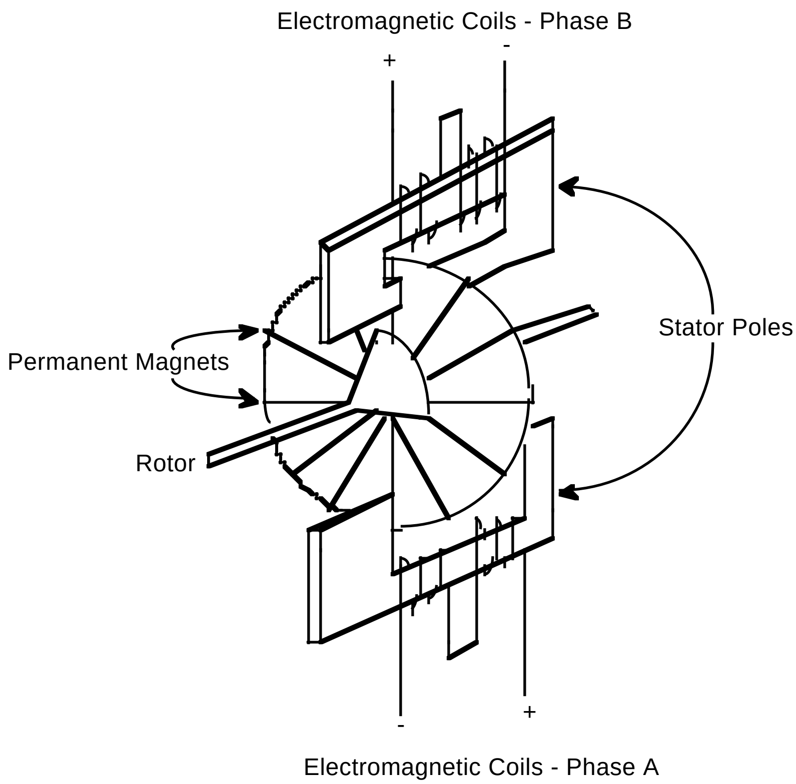

Figure 1

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.

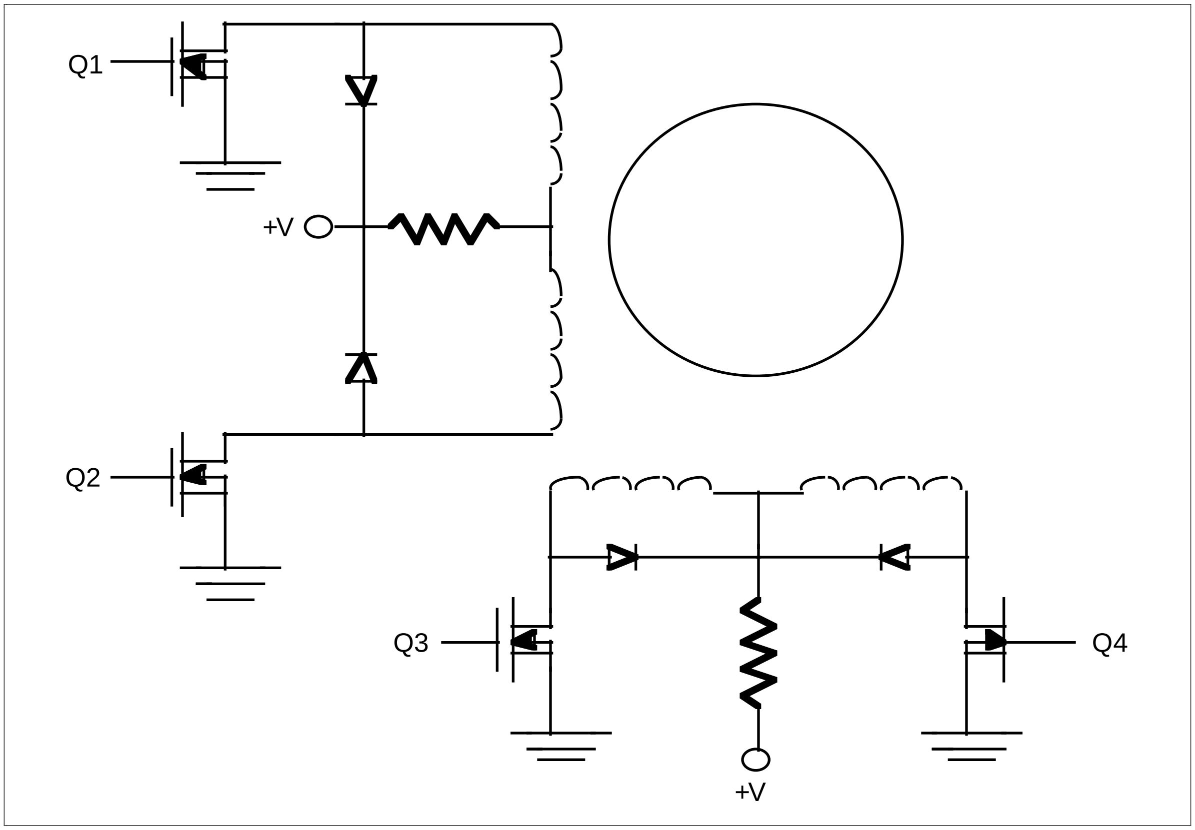

Figure 2

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.

Table 1

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

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:

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

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.

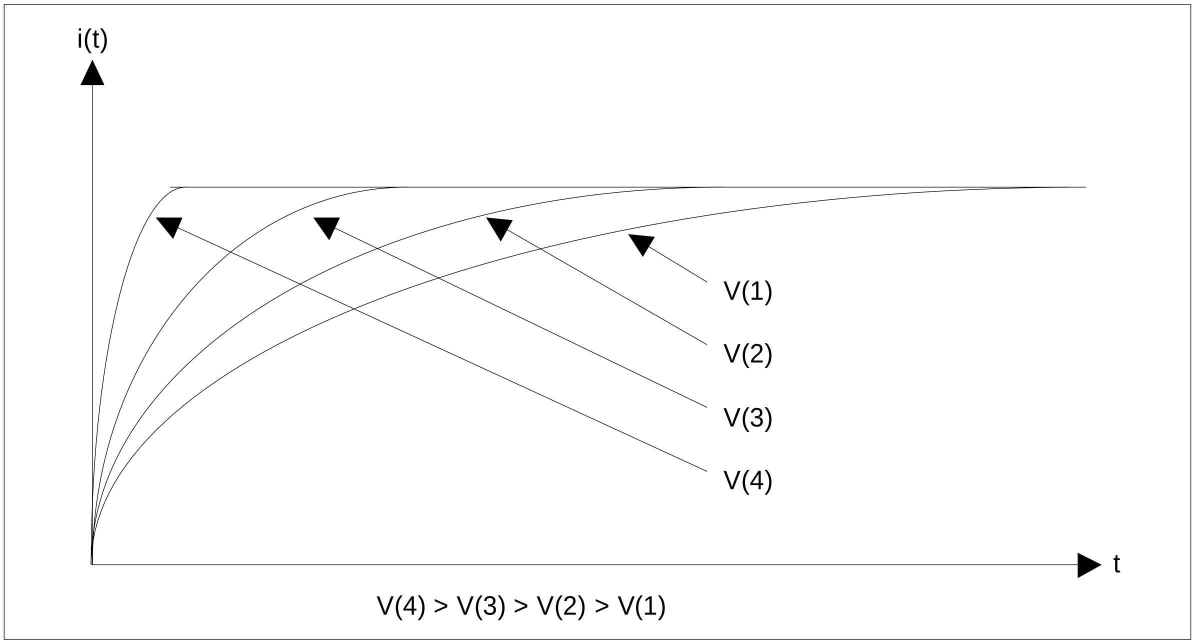

Figure 3

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.

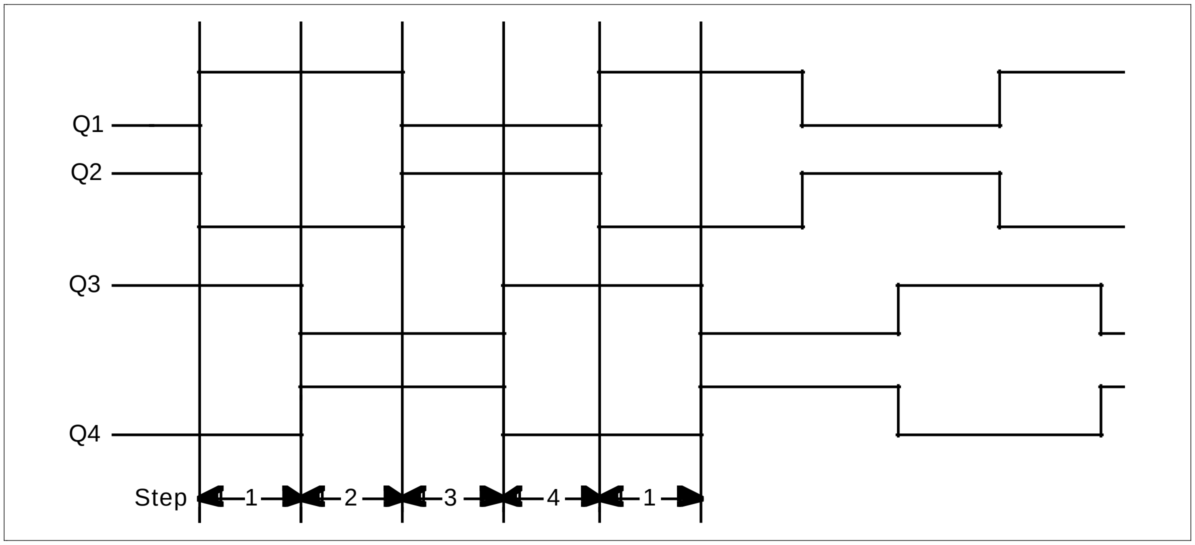

Figure 4

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.

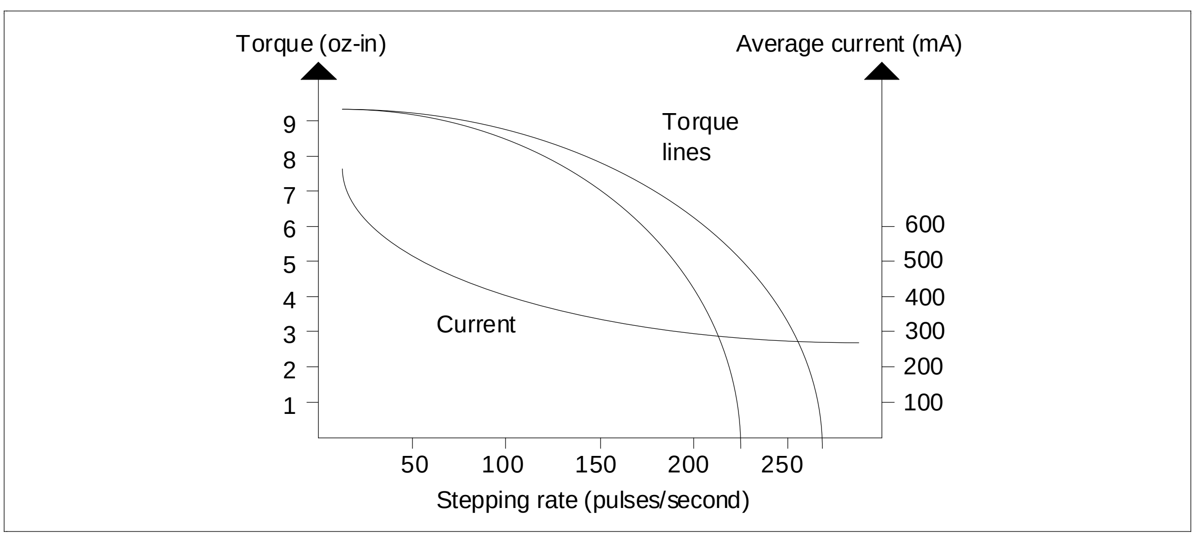

Figure 5

Using a H8/300 to control a stepper motor

Design Example

As stated in the introduction, the H8/330 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 H8/330 evaluation board.
2. A specially wire-wrapped motor interface circuitry board used to boost the motor control signals, and provide the motor power supply requirements.
3. The KP6P8-701 permanent-magnet stepper motor.

The H8/330 is supplying the initial 4 excitation waveforms via the 4 I/O output lines $P8_{3-6}$. $P8_3$ produces the phase A waveform, $P8_4$ the phase B waveform, $P8_5$ the inverted phase A waveform, and $P8_6$ the inverted phase B waveform. The motor interface board schematic is shown in figure 6. The initial motor 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 H8/330 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.

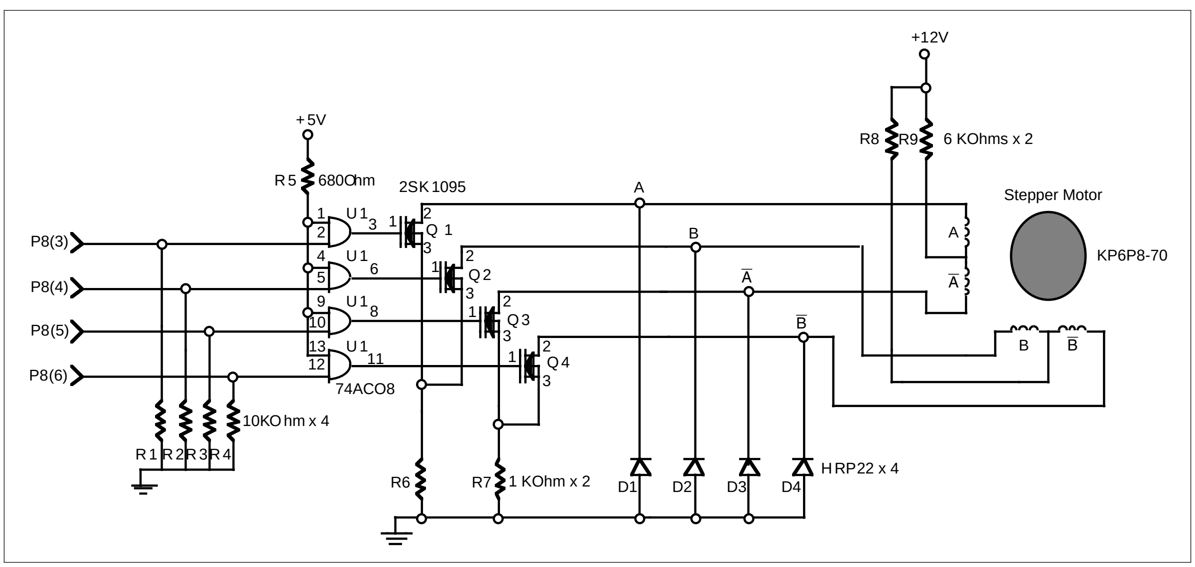

Figure 6

HITACHI

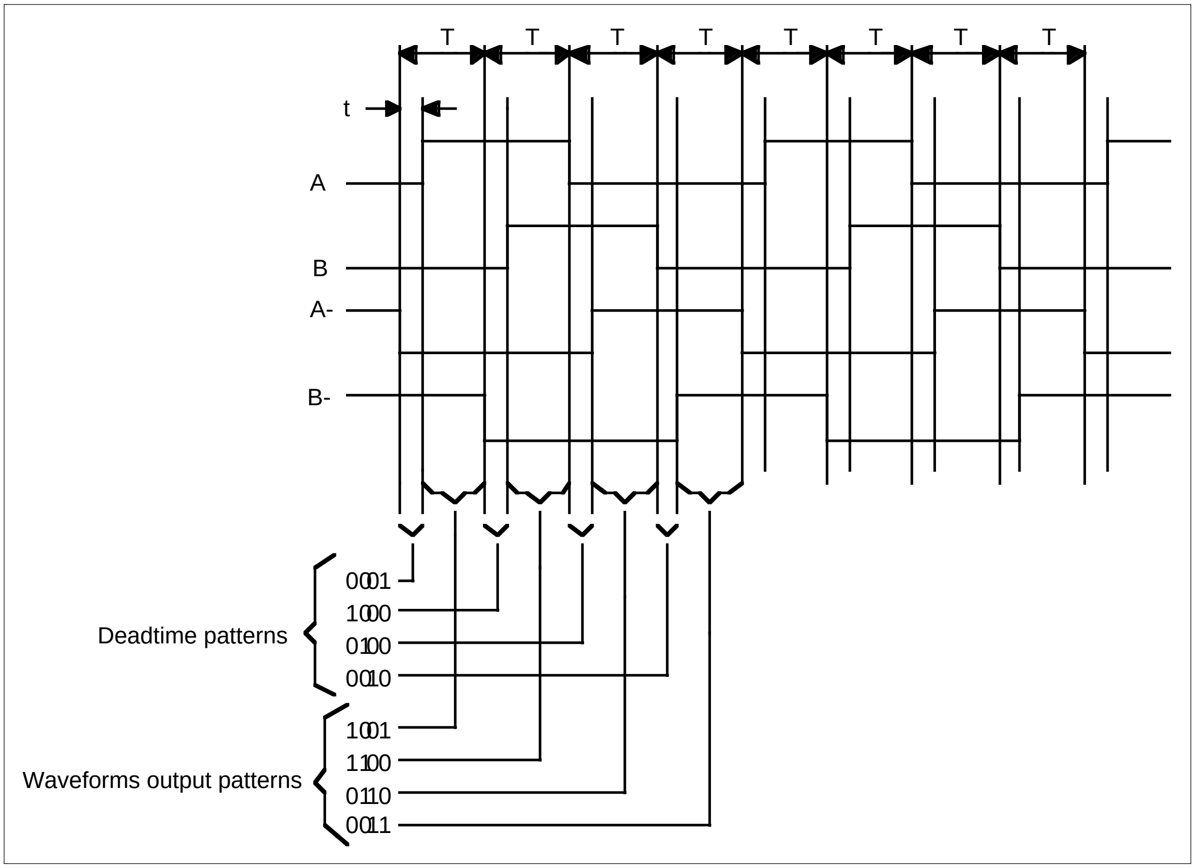

Figure 7

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 operation is implemented by using the 16-bit free-running timer of the H8/330 to time out via the $P8_{3-6}$ output-configured pins the waveform pattern shown in figure 7. The A comparator inside the timer module alternatively holds the timer counts corresponding to the non-overlap and excitation times. The first non-overlap pattern time and the first excitation pattern time are output as a result of polling the output-compare A flag. The next non-overlap pattern is output when a compare-match A interrupt occurs, and the next excitation pattern time is issued by polling when the compare-match A flag is set. This process then continues ad infinitum and is illustrated in Figure 8.

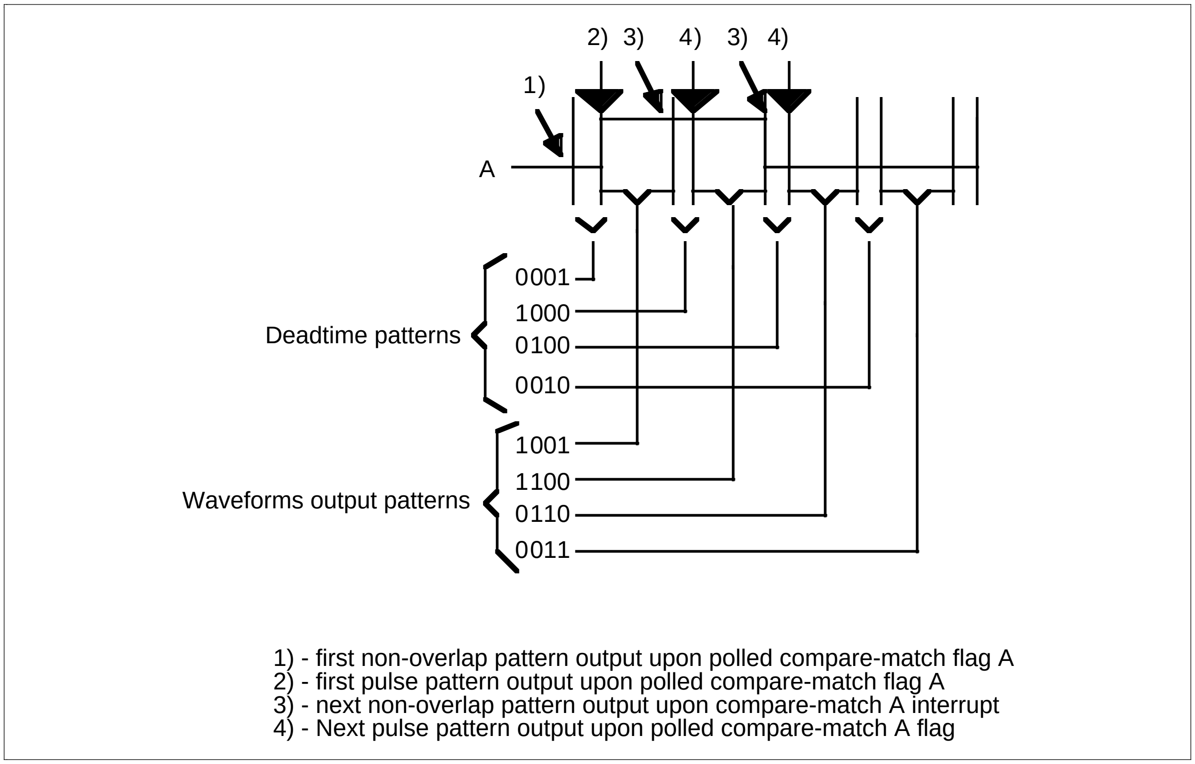
2) 3) 4) 3) 4)
1)
A
Deadtime patterns
0001
1000
0100
0010
Waveforms output patterns
1001
1100
0110
0011
1) - first non-overlap pattern output upon polled compare-match flag A
2) - first pulse pattern output upon polled compare-match flag A
3) - next non-overlap pattern output upon compare-match A interrupt
4) - Next pulse pattern output upon polled compare-match A flag

Figure 8

Software Description

The code controlling the process explained in the previous section consists of 4 main parts:

1. The non-overlap and excitation pattern look-up table.
2. Initializations of CPU, constants, I/O Port, and 16-bit timer.
3. The first 2 loops that control the timing of the first non-overlap and excitation patterns.
4. The compare-match A interrupt service routine that provides the timing for the next non-overlap and excitation patterns.

The look-up table consists of 8 hexadecimal values corresponding to the 4 excitation waveforms patterns. The 4 non-overlap patterns alternate with the 4 pulse patterns.

After the CPU is initialized, 4 constant values that are to be used later in the program are initialized. The non-overlap time count and the pulse output time count are stored in R1H and R2 respectively. These values are loaded into the Output Compare Match Register A (OCRA) of the 16-bit Free Running Counter Timer. R3L contains the pattern count corresponding to the number of output patterns that repeat in a cycle (8). R1L contains the initial count that starts up the whole process.

The 16-bit timer is set to clear the contents of the free-running counter register (FRC) when a compare-match between OCRA and FRC occurs. Also, the timer clock is set to Φ/8. The I/O Port 8 data direction register (P8DDR) is configured for outputs only, and the data register (P8DR) is cleared.

The process starts by loading the startup count into the OCRA, and polling the compare-match A flag (OCFA) until it is set. Then, the first value in the look-up table (corresponding to the first non-overlap pattern) is output at the I/O Port $8_{3\text{-}6}$ lines. The OCRA is loaded with the non-overlap count value in R1H, the OCFA flag is cleared, and the pattern count in register R3L is decremented. The next loop polls the OCFA flag again, and upon flag set, the first pulse output pattern is output. Next, the OCRA is updated with the pulse count value in R2, OCFA is cleared, the pattern count R3L decremented, and the output-compare interrupt A enable flag (OCIAE) in the Timer Interrupt Enable Register (TIER) is set. The program then enters an infinite loop that basically waits for the interrupt to occur.

The interrupt routine is serviced when the contents of OCRA match the contents of FRC. Then, the next non-overlap pattern is output at the $P8_{3\text{-}6}$ lines, the OCRA is updated, the OCIAE and OCFA flags are cleared, and the pattern count is decremented. Loop 3 within the interrupt service routine polls the OCFA flag, and upon compare-match, the next pulse pattern is output. If the pattern count is not zero, the OCRA is loaded with the pulse count, the OCIAE is set, and the program exits the interrupt service routine and enters the wait loop again. If the pattern count is 0, the first pattern in the look-up table is loaded again and the process continues. Figures 9 and 10 show the main program and the interrupt service routine flowcharts. The program listing follows the flowcharts.

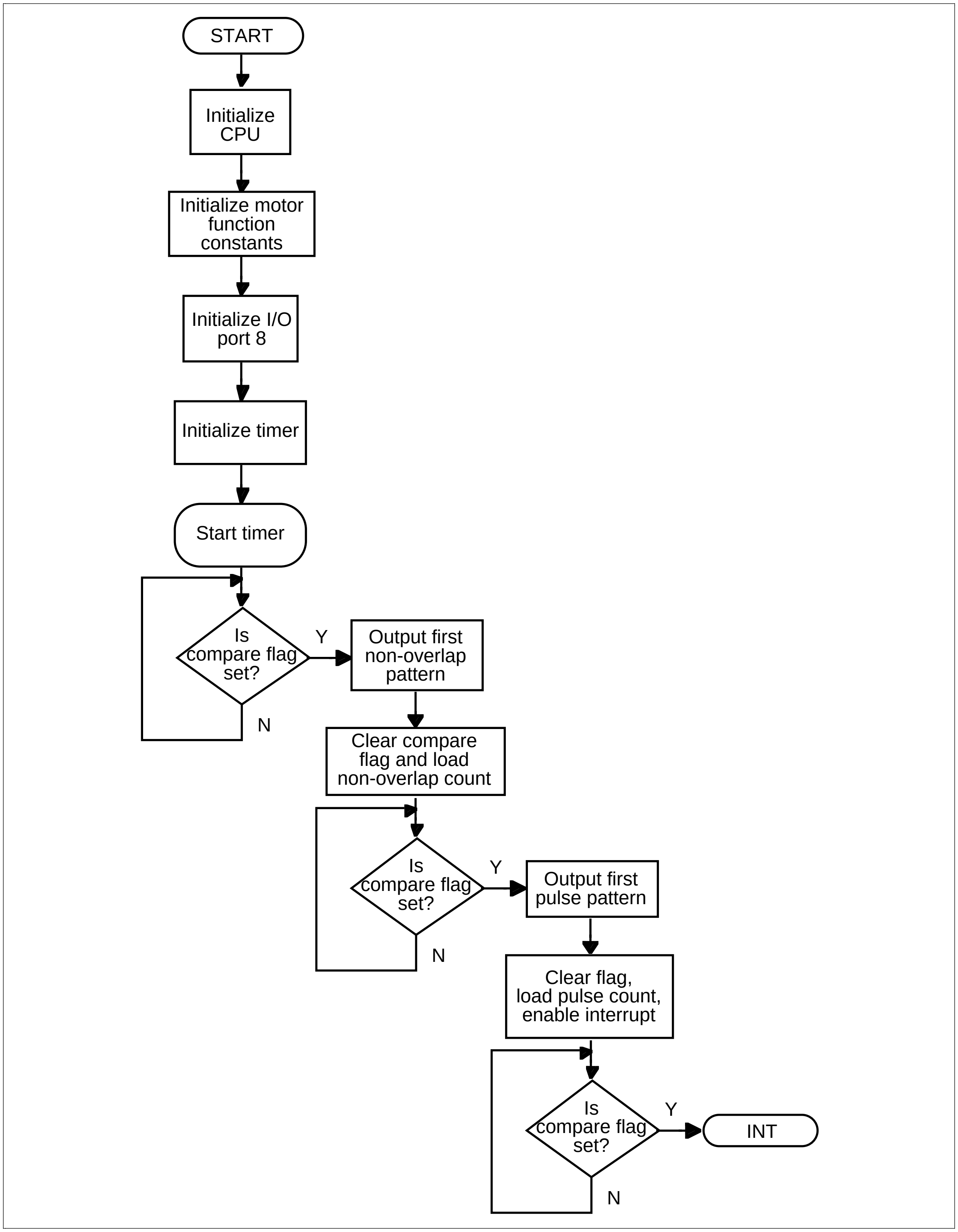
START
Initialize CPU
Initialize motor function constants
Initialize I/O port 8
Initialize timer
Start timer
Is compare flag set?
Y
N
Output first non-overlap pattern
Clear compare flag and load non-overlap count
Is compare flag set?
Y
N
Output first pulse pattern
Clear flag, load pulse count, enable interrupt
Is compare flag set?
Y
N
INT

Figure 9

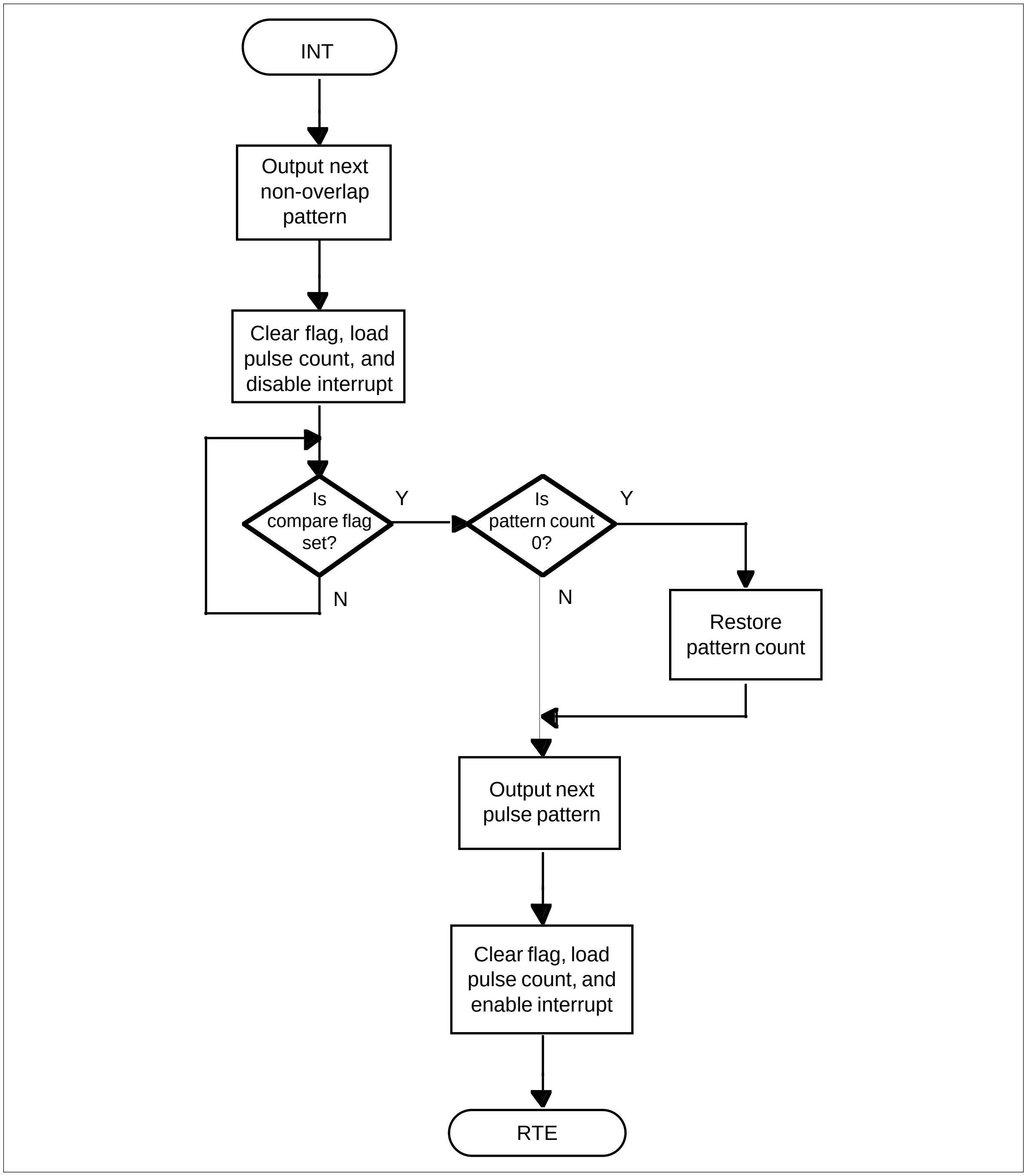
INT
Output next non-overlap pattern
Clear flag, load pulse count, and disable interrupt
Is compare flag set?
Y
N
Is pattern count 0?
Y
N
Restore pattern count
Output next pulse pattern
Clear flag, load pulse count, and enable interrupt
RTE

Figure 10

Program listing - motor.lst

```
Microtec Research ASMH83    Version 1.0A    Sep 08 11:11:35 1994     Page 1


Command line:  C:\MRI\ASMH83\ASMH83.EXE -l m2.src
Line        Addr
            FF90                        TIER    .EQU    H'FF90
            FF91                        TCSR    .EQU    H'FF91
            FF92                        FRC     .EQU    H'FF92
            FF94                        OCRA    .EQU    H'FF94
            FF96                        TCR     .EQU    H'FF96
            FF97                        TOCR    .EQU    H'FF97
            FFBD                        P8DDR   .EQU    H'FFBD
            FFBF                        P8DR    .EQU    H'FFBF

                                                .ORG    H'0020
            0020 025C                           .DATA.W INT

                                                .ORG    H'100
            0100 88C8 C0E0                      .DATA.B H'88,H'C8,H'C0,H'E0
            0104 A0B0 9098                      .DATA.B H'A0,H'B0,H'90,H'98

                                                .ORG    H'200
            0200 7907 FF80              START:  MOV.W   #H'FF80,R7      ;SET
STACKPOINTER
            0204 0700                           LDC     #0,CCR
            0206 7906 0100                      MOV.W   #H'100,R6       ;POINT TO
OUTPUT PATTERN MEMORY TABLE
            020A F9FF                           MOV.B   #H'FF,R1L       ;STARTUP
COUNT
            020C F11F                           MOV.B   #H'1F,R1H       ;NON-OVERLAP
COUNT
            020E 7902 0800                      MOV.W   #H'0800,R2      ;PULSE COUNT
            0212 FB08                           MOV.B   #H'08,R3L       ;PATTERN
COUNT

                                        ;       I/O PORT 8 SETUP

            0214 F8FF                           MOV.B   #H'FF,R0L       ;PORT 8 IS
OUTPUT
            0216 38BD                           MOV.B   R0L,@P8DDR
            0218 F800                           MOV.B   #H'0,R0L
```

```
021A 38BF                   MOV.B   R0L,@P8DR
                    ;       TIMER INITIALIZATION
021C F801                   MOV.B   #H'01,R0L       ;CLEAR FRC AT COMPARE-MATCH A
021E 3891                   MOV.B   R0L,@TCSR
0220 3896                   MOV.B   R0L,@TCR        ;TIMER CLOCK IS /8
                    ;       START TIMER WITH STARTUP COUNT
0222 3994                   MOV.B   R1L,@OCRA
0224 7900 FFFF              MOV.W   #H'FFFF,R0
0228 6B80 FF92              MOV.W   R0,@FRC
022C 7E91 7330      LOOP1:  BTST.B  #3,@TCSR        ;STARTUP LOOP
0230 47FA                   BEQ     LOOP1
0232 6C6D                   MOV.B   @R6+,R5L        ;OUTPUT FIRST NON-OVERLAP PATTERN
0234 3DBF                   MOV.B   R5L,@P8DR
0236 1A0B                   DEC     R3L
0238 3194                   MOV.B   R1H,@OCRA       ;LOAD OCRA W/ NON-OVERLAP COUNT
023A 7F91 7230              BCLR.B  #3,@TCSR        ;CLEAR COMPARE-MATCH FLAG
023E 7E91 7330      LOOP2:  BTST.B  #3,@TCSR        ;FIRST NON-OVERLAP LOOP
0242 47FA                   BEQ     LOOP2
0244 6C6D                   MOV.B   @R6+,R5L        ;OUTPUT FIRST PULSE PATTERN
0246 3DBF                   MOV.B   R5L,@P8DR
0248 1A0B                   DEC     R3L
024A 6B82 FF94              MOV.W   R2,@OCRA        ;LOAD OCRA W/ PULSE COUNT
024E 7F91 7230              BCLR.B  #3,@TCSR        ;CLEAR COMPARE-MATCH FLAG
0252 7F90 7030              BSET.B  #3,@TIER        ;ENABLE COMPARE-MATCH INTERRUPT
0256 40FE           WAIT:   BRA     WAIT            ;WAIT FOR INTERRUPT
0258 0000                   NOP
025A 0000                   NOP
```

```
                         ;       INT INTERRUPT SERVICE ROUTINE

        025C 1A0B        INT:    DEC     R3L
        025E 6C6D                MOV.B   @R6+,R5L      ;NEXT NON-OVERLAP OUTPUT
        0260 3DBF                MOV.B   R5L,@P8DR
        0262 3194                MOV.B   R1H,@OCRA     ;NON-OVERLAP COUNT
        0264 7F90 7230           BCLR.B  #3,@TIER      ;DISABLE INTERRUPT
        0268 7F91 7230           BCLR.B  #3,@TCSR      ;CLEAR COMPARE-MATCH FLAG
        026C 7E91 7330   LOOP3:  BTST.B  #3,@TCSR      ;NON-OVERLAP LOOP
        0270 47FA                BEQ     LOOP3
        0272 1A0B                DEC     R3L
        0274 4606                BNE     CONT
        0276 FB08                MOV.B   #H'08,R3L     ;RESTORE PATTERN COUNT
        0278 7906 0100           MOV.W   #H'100,R6     ;POINT TO START PATTERN AGAIN
        027C 6C6D        CONT:   MOV.B   @R6+,R5L      ;NEXT PULSE OUTPUT
        027E 3DBF                MOV.B   R5L,@P8DR
        0280 6B83 FF94           MOV.W   R3,@OCRA      ;LOAD OCRA W/ PULSE COUNT
        0284 7F91 7230           BCLR.B  #3,@TCSR      ;CLEAR COMPARE-MATCH FLAG
        0288 7F90 7030           BSET.B  #3,@TIER      ;ENABLE COMPARE-MATCH INTERRUPT
        028C 5670                RTE
                                 .END
```

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

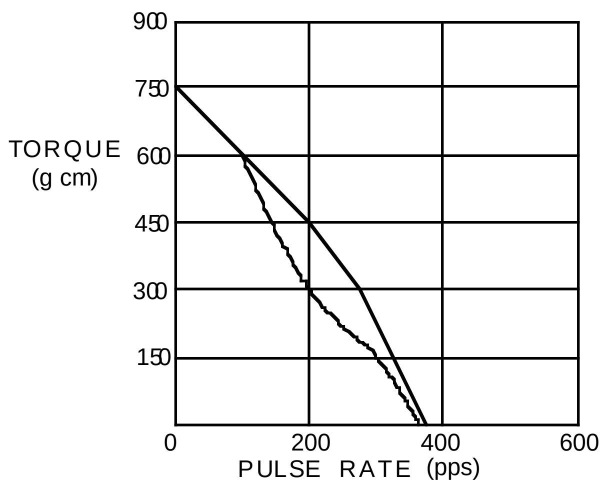

Figure 11