Driving Unipolar Stepper Motors Using C51/C251

C51 Microcontrollers

Application Note

Introduction

Stepper motors are commonly used in accurate motion control. They allow to control any motion with high precision by counting the number of steps applied to the motor. Most of systems controlling stepper motors are embedded systems such as printer, scanner or floppy disk drive. This application note describes how to drive a unipolar stepper motor with the Programmable Counter Array of an Atmel C51/C251 microcontroller.

Description

C51/C251 microcontroller output pins cannot directly drive stepper motors. These have to be powered before being applied to the stepper motor.

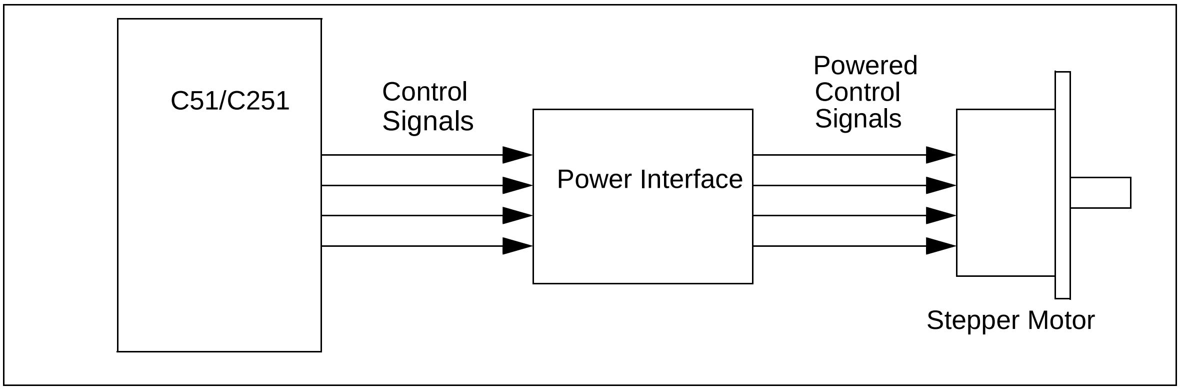
This document explains uses the Programmable Counter Array (PCA) of the microcontroller to generate the control signals to the Power Interface. The Power Interface allows the microcontroller to drive enough current into coils of a stepper motor.

There are two advantages to using PCA. First of all, PCA provides greater accuracy than toggling pins in software because the toggle occurs before the interrupt request is serviced. Thus, interrupt response time does not affect the accuracy of the output. Secondly the microcontroller CPU is left free for application task execution while the PCA drives stepper motors.

There are two major types of stepper motors: Permanent magnet stepper motors (unipolar stepper motors and bipolar stepper motors) and variable reluctance stepper motors (hybrid stepper motors).

Rev. 4214B–8051–12/02

Figure 1. System Configuration



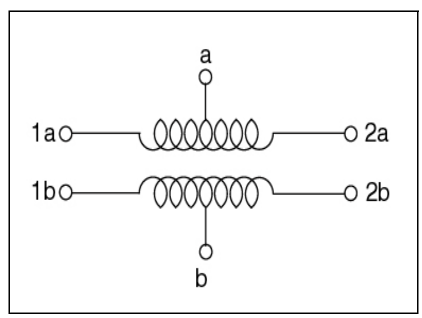
Identification of Stepper Motor

There are several types of stepper motors, these cannot be driven in the same way. In this application note, we have chosen to drive a ***unipolar stepper*** motor (see Figure 2). For more information you will find schemes to identify the other types of stepper motors.

Unipolar Stepper Motor

Unipolar stepper motors are characterised by their center-tapped windings.

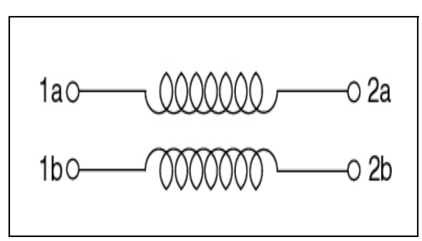
Figure 2. Unipolar Stepper Motors Coils



Bipolar Stepper Motor

Bipolar stepper motors are designed with separate coils.

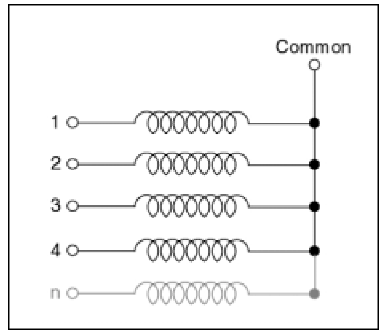
Figure 3. Bipolar Stepper Motor Coils



Variable Reluctance

Variable reluctance stepper motor (also called hybrid motors) are characterised by one common lead.

Figure 4. Hybrid Stepper Motor Coils



Driving Unipolar Stepper Motors

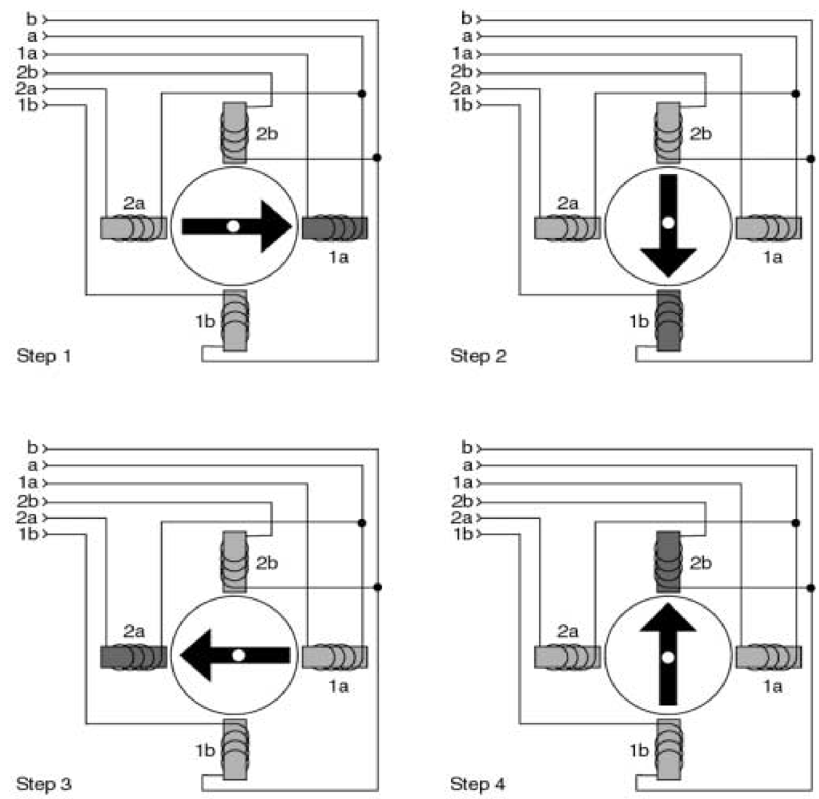
There are three ways to drive unipolar stepper motors (one phase on, two phase on or half step), each one has some advantages and disavantages.

One Phase On Mode (Full Step mode)

Table 1. One Phase On Sequence

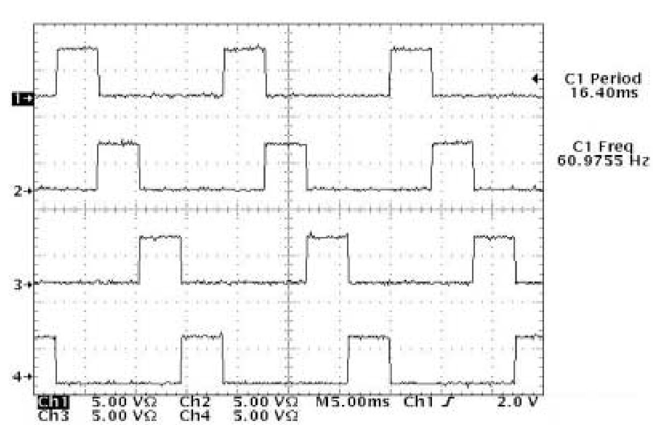
Step	1a	1b	2a	2b
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	1

Figure 5. One Phase Steps



In one phase mode, each successive coil is energized in turn. One phase mode produces smooth rotations and the lowest power comsumption of the three modes. Steps are applied in order from one to four. After step four, the sequence is repeated from step one. Applying steps from one to four makes the motor run clockwise, reversing the order of step from four to one will make the motor run counter-clockwise.

Figure 6. One Phase On Steps Sequence

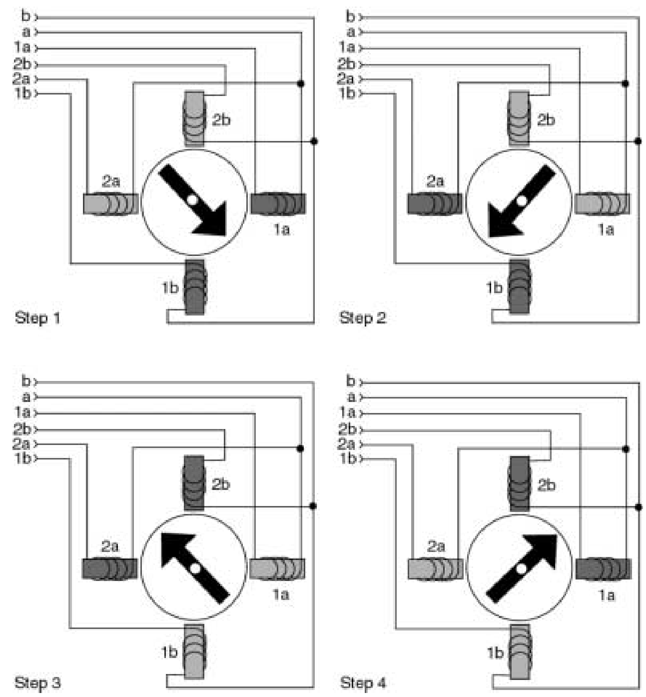


Two Phases On Mode (Alternate Full step Mode)

Table 2. Two Phases On Sequence

Step	1a	1b	2a	2b
1	1	0	0	1
2	1	1	0	0
3	0	1	1	0
4	0	0	1	1

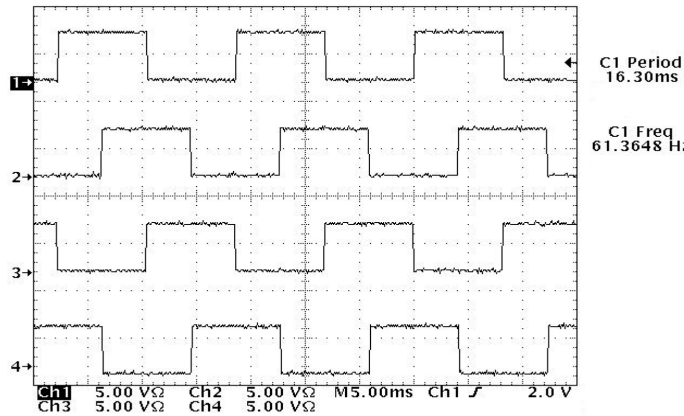
Figure 7. Two Phases On Steps



In two phase mode, successive pairs of adjacent coils are energised in turn, motion is not as smooth as in one phase mode, power comsumption is more important but it produces greater torque.

As in one phase mode, applying the steps in order makes the stepper motor run clockwise and reversing order makes it turn counter-clockwise.

Figure 8. Two Phases On Steps Sequence



Half Step Mode

Table 3. Half Step Sequence

Step	1a	1b	2a	2b
1	1	0	0	1
2	1	0	0	0
3	1	1	0	0
4	0	1	0	0
5	0	1	1	0
6	0	0	1	0
7	0	0	1	1
8	0	0	0	1

The half step sequence is a mix of one phase on and two phase on sequences. The main advantage of this mode is to increase by two the nominal number of steps of your stepper motor. By example, an unipolar stepper motor of 24 steps of 15 degrees each "becomes", when we use half step mode, a stepper motor of 48 steps of 7.5 degrees.

Figure 9. Half Step Sequence

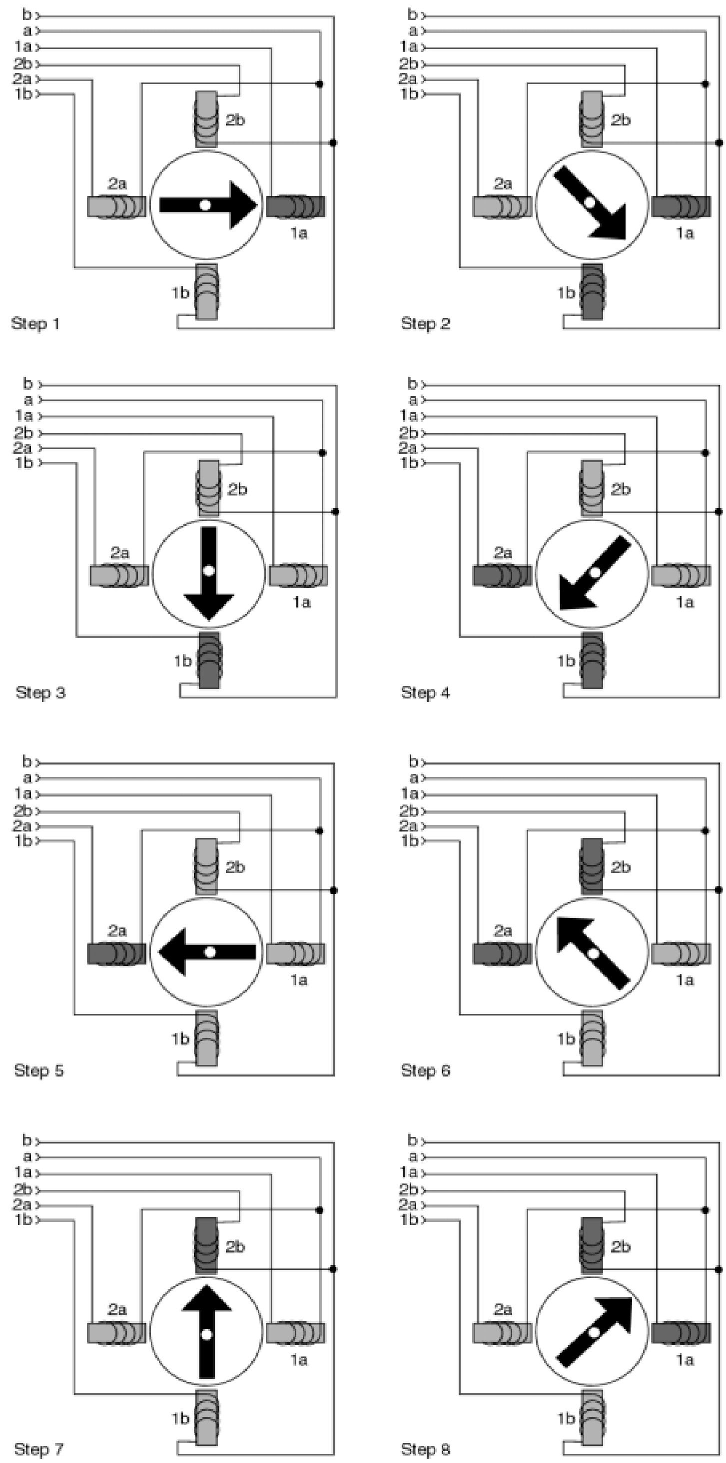
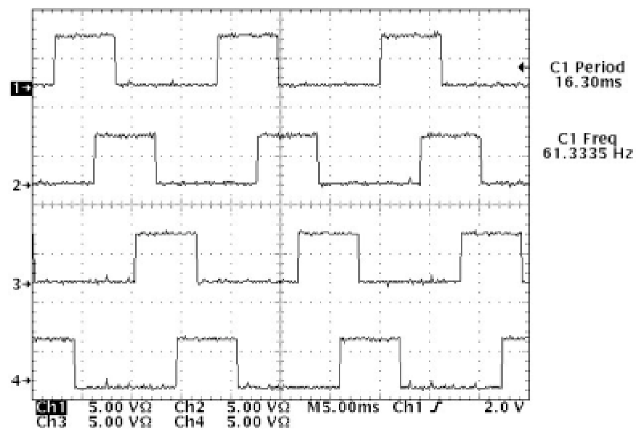


Figure 10. Half Step Sequence



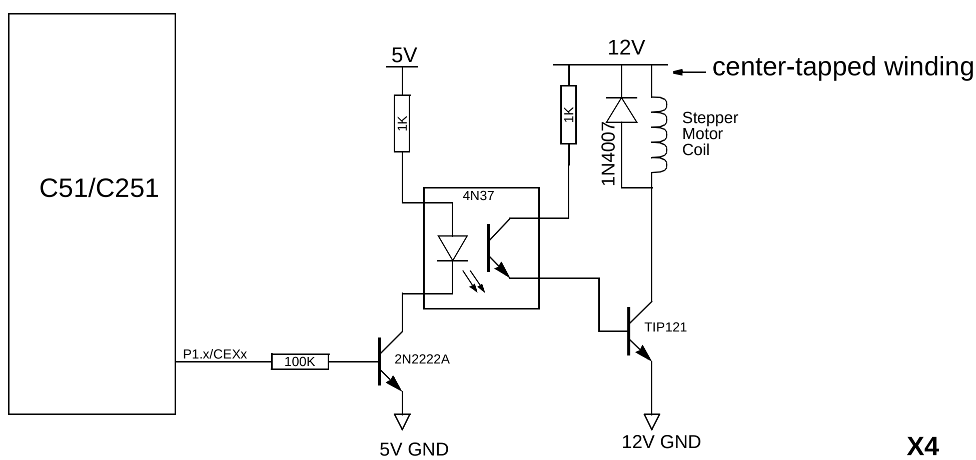
Hardware

The following schematics shows the power interface between the Atmel C51/C251 and a stepper motor.

The four PCA pins must have the following hardware connected.

Coils are connected with both center-tapped windings connected to 12V power supply.

Figure 11. Power Interface



In this configuration, the stepper motor is opto-isolated from the microcontroller with a high protection level.

The 2N2222A transistor helps to drive enough current in 4N37 led (via 1kΩ resistor).

Stepper motor power is given by 12V power supply via TIP121 transistor.

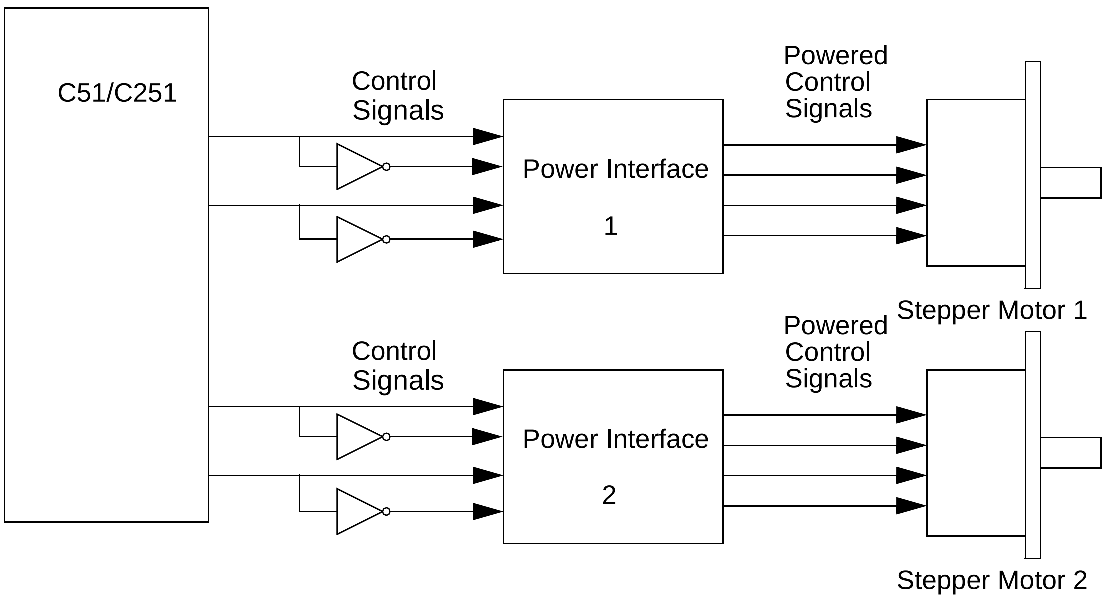
Diode on stepper motor coil is used to prevent inductive kicks produced when coil is turned off.

Driving Two Stepper Motors with Only One PCA

It is possible to drive two stepper motors at the same time with only four PCA channels. For this, we will use the the Two Phase On mode (see Table 2 and Figure 6).

In this mode, signals 1 and 2 are respectively opposite waveforms of signals 3 and 4.

Thus, it is possible to have the four signals needed for one stepper motor with only two PCA channels and two logical inverters.



Software

The software of this application note (Given in appendix A), written in C language allows to make a motor turn NB_LOOP loops clockwise (or counter-clockwise) direction in half step mode.

NB_LOOP and clockwise (or counter-clockwise) are defined in constant variable at the beginning of code, and must be defined before compilation.

The user also needs to define (via constant variable) the number of steps of the motor.

The unipolar stepper motor will be driven with PCA via Power Interface described in section 3.

- The following software initializes CEX0 to CEX3 of Programmable Counter Array in High Speed Output Mode

- Timer 0 is configured in 8 bit auto-reload mode

- Clock of the PCA is given by overflow of Timer 0.

- PCA interrupt vector at address 0x0033h is also used.

If speed precision is not very important in the application it is possible to use Fosc/12 or Fosc/4 for PCA clock input, it will leave Timer 0 free for the application itself.

The speed of the stepper motor may be calculated with the formula:

$$SPEED = \frac{60}{\frac{NB\ STEP}{8} * \frac{12}{Fosc} * 0xFFFF * [\ 0xFF - TH0\ ]}$$

SPEED = Speed in rotations per minute

NBSTEP = Number of step of the motor, in general written on the stepper motor itself (48, 96, 200..)

Fosc = Frequency of the oscillator in Hertz.

For example, when using stepper motor with 200 steps and a 12 Mhz oscillator and loading THO with 0xFC, the speed is 12.2 rotations per minute.

With the same formula THO is found by:

$$TH0 = 0xFF - \left[\frac{40 * Fosc}{NB\ STEP * 0xFFFF * SPEED} \right]$$

The table below lists the values of speed for different oscillator frequencies and nominal number of steps of the motor, when using Fosc / 12 and Fosc / 4 as PCA clock input.

Rot/Min	Fosc (MHz)	Nb of Step	
210.941	11.0592	96	Fosc / 4
70.314	11.0592	96	Fosc / 12
101.252	11.0592	200	Fosc / 4
33.751	11.0592	200	Fosc / 12
178.705	12	96	Fosc / 4
76.295	12	96	Fosc / 12
109.865	12	200	Fosc / 4
36.622	12	200	Fosc / 12
305.180	16	96	Fosc / 4
101.723	16	96	Fosc / 12
146.487	16	200	Fosc / 4
48.829	16	200	Fosc / 12

Value for other oscillator frequency or number of steps can be found easily by arithmetic operation.

For example, at Fosc = 12 MHz and clock input = Fosc / 12 , NBSTEP = 96 we found 76.295 rotations per minute.

If another motor of 48 steps is used instead, we found now 2*76.295 rotations per minute.

Note: Above values are theoretical for high speed rotation, motors are limited by needed torque. Sometimes it will be necessary to start the motor slowly and accelerate it after few seconds.

Appendix A: Software

MAIN.C

```
/*C**************************************************************************************
* NAME : Stepper.c
*----------------------------------------------------------------------------------------
* CREATED_BY : Eric TINLOT
* CREATION_DATE : 05/02/01
*----------------------------------------------------------------------------------------
* PURPOSE : Driving unipolar stepper motor in half step mode using PCA.
***************************************************************************************/

/*_____ I N C L U D E S ______________________________________________________________*/

#include "89C51RD2.H"    /* sfr definition file */


/*_____ D E C L A R A T I O N_________________________________________________________*/

/*_____ M A C R O S __________________________________________________________________*/
#define ENABLE_ALL_IT   ( EA = 1 )
#define PCA_ON          ( CCON |= 0x40)
#define PCA_OFF         ( CCON &= 0xBF)

/*_____ D E F I N I T I O N ___________________________________________________________*/

#define NB_LOOP 10/* place here the number of loop desired */

       /* select rotation direction here        */
#define CLOCKWISE

#define NB_STEP 48 /* define by stepping motor itself    */

#define P_HIGH 0xFF
#define P_LOW  0x00

long int nb_it=0;




/*F**************************************************************************************
* NAME :ccaxh_reload_value
*----------------------------------------------------------------------------------------
* AUTHOR :Eric TINLOT
* DATE :05/02/01
*----------------------------------------------------------------------------------------
* PURPOSE:Load value in PCA register CCAPxH
* INPUTS:value for CCAP0H,CCAP1H,CCAP2H,CCAP3H
```

```
* OUTPUTS:void
*********************************************************************************/
void ccapxh_reload_value(char h0, char h1, char h2, char h3)
{
CCAP0H =  h0;
CCAP1H =  h1;
CCAP2H =  h2;
CCAP3H =  h3;
}

/*F*********************************************************************************
* NAME  :timer0_init
*-------------------------------------------------------------------------------------
* AUTHOR :Eric TINLOT
* DATE   :05/02/01
*-------------------------------------------------------------------------------------
* PURPOSE:init timer 0  for PCA clock
* INPUTS :void
* OUTPUTS:void
*********************************************************************************/
void timer0_init(void)
{
TH0  = 0xB3;
TL0  = 0xB3;
TMOD = TMOD | 0x02;
TCON = TCON | 0x10;
}

/*F*********************************************************************************
* NAME   :pca_init
*-------------------------------------------------------------------------------------
* AUTHOR :Eric TINLOT
* DATE   :05/02/01
*-------------------------------------------------------------------------------------
* PURPOSE:init pca to run in High speed Output
* INPUTS :void
* OUTPUTS:void
*********************************************************************************/
void pca_init(void)
{
CCAP0L =  P_HIGH;
CCAP1L =  P_HIGH;
CCAP2L =  P_HIGH;
CCAP3L =  P_HIGH;

#ifdef CLOCKWISE
      /* pca register initialisation for CLOCKWISE rotation */
ccapxh_reload_value(0x5F,0x00,0x00,0x1F);
#else /* COUNTER_CLOCKWISE */
  /* pca register initialisation for COUNTER CLOCKWISE rotation */
```

```
ccapxh_reload_value(0x1F,0x00,0x00,0x5F);
#endif

CCAPM0 = 0x4D;    /* enable High Speed Output mode on CEX0 - CEX3 */
CCAPM1 = 0x4D;
CCAPM2 = 0x4D;
CCAPM3 = 0x4D;

/* PCA Clock Select */

CMOD   = 0x04;    /*  CMOD = 0x04 -> pca clock = timer0 overflow */
       /*  CMOD = 0x00 -> pca clock = Fosc/12 */
      /*  CMOD = 0x02 -> pca clock = Fosc/4 */
CH     = 0xFF;
CL     = 0xFF;
EC     = 0x01;
}

/*F**************************************************************************************
* NAME   :PCA - interrupt program
*---------------------------------------------------------------------------------------
* AUTHOR :Eric TINLOT
* DATE:05/02/01
*---------------------------------------------------------------------------------------
* PURPOSE:identify toggle pin and prepare pca register for next toggle
*                   nb_it variable, enable to count number of loop
* INPUTS :void
* OUTPUTS:void
***************************************************************************************/
PCA() interrupt 6 using 1   /* Int Vector at 000BH, Reg Bank 1  */
{
#ifdef CLOCKWISE

  if (CCF0==1 && CEX0==0) { nb_it++; CCF0 = 0; CCAP0H = 0xFF; }
  if (CCF1==1 && CEX1==0) { CCF1 = 0; CCAP1H = 0x3F; }
  if (CCF2==1 && CEX2==0) { CCF2 = 0; CCAP2H = 0x7F; }
  if (CCF3==1 && CEX3==0) { CCF3 = 0; CCAP3H = 0xBF; }

  if (CCF0==1 && CEX0==1) { CCF0 = 0; CCAP0H = 0x5F; }
  if (CCF1==1 && CEX1==1) { CCF1 = 0; CCAP1H = 0x9F; }
  if (CCF2==1 && CEX2==1) { CCF2 = 0; CCAP2H = 0xDF; }
  if (CCF3==1 && CEX3==1) { CCF3 = 0; CCAP3H = 0x1F; }

#else /* COUNTER_CLOCKWISE */

  if (CCF0==1 && CEX0==0) { nb_it++; CCF0 = 0; CCAP0H = 0xBF; }
  if (CCF1==1 && CEX1==0) { CCF1 = 0; CCAP1H = 0x7F; }
  if (CCF2==1 && CEX2==0) { CCF2 = 0; CCAP2H = 0x3F; }
  if (CCF3==1 && CEX3==0) { CCF3 = 0; CCAP3H = 0xFF; }
```

```
  if (CCF0==1 && CEX0==1) { CCF0 = 0; CCAP0H = 0x1F; }
  if (CCF1==1 && CEX1==1) { CCF1 = 0; CCAP1H = 0xDF; }
  if (CCF2==1 && CEX2==1) { CCF2 = 0; CCAP2H = 0x9F; }
  if (CCF3==1 && CEX3==1) { CCF3 = 0; CCAP3H = 0x5F; }

#endif
}


/*F**************************************************************************
* NAME  :main
*----------------------------------------------------------------------------
* AUTHOR:Eric TINLOT
* DATE :05/02/01
*----------------------------------------------------------------------------
* PURPOSE:call init subroutines and run stepper motor for nb_loop
* INPUTS :void
* OUTPUTS:void
**************************************************************************/
void main(void)
{

long int tocount;

tocount =  2 * (NB_STEP / 8) * NB_LOOP; /* compute number of IT needed for NB LOOP*/

timer0_init();                                   /* init Timer 0 for pca clock */
pca_init();      /* init pca to run in High speed Output */

ENABLE_ALL_IT;
PCA_ON;

do{
  }while(nb_it<=tocount); /* wait motor turn nb_loop */

PCA_OFF;

P1     = P_LOW;        /* clear all coils (no comsumption) */

do{
  }while(1);               /* endless */
}
```

Atmel Headquarters

Corporate Headquarters
2325 Orchard Parkway
San Jose, CA 95131
TEL 1(408) 441-0311
FAX 1(408) 487-2600

Europe
Atmel Sarl
Route des Arsenaux 41
Case Postale 80
CH-1705 Fribourg
Switzerland
TEL (41) 26-426-5555
FAX (41) 26-426-5500

Asia
Room 1219
Chinachem Golden Plaza
77 Mody Road Tsimhatsui
East Kowloon
Hong Kong
TEL (852) 2721-9778
FAX (852) 2722-1369

Japan
9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
TEL (81) 3-3523-3551
FAX (81) 3-3523-7581

Atmel Operations

Memory
2325 Orchard Parkway
San Jose, CA 95131
TEL 1(408) 441-0311
FAX 1(408) 436-4314

Microcontrollers
2325 Orchard Parkway
San Jose, CA 95131
TEL 1(408) 441-0311
FAX 1(408) 436-4314

La Chantrerie
BP 70602
44306 Nantes Cedex 3, France
TEL (33) 2-40-18-18-18
FAX (33) 2-40-18-19-60

ASIC/ASSP/Smart Cards
Zone Industrielle
13106 Rousset Cedex, France
TEL (33) 4-42-53-60-00
FAX (33) 4-42-53-60-01

1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906
TEL 1(719) 576-3300
FAX 1(719) 540-1759

Scottish Enterprise Technology Park
Maxwell Building
East Kilbride G75 0QR, Scotland
TEL (44) 1355-803-000
FAX (44) 1355-242-743

RF/Automotive
Theresienstrasse 2
Postfach 3535
74025 Heilbronn, Germany
TEL (49) 71-31-67-0
FAX (49) 71-31-67-2340

1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906
TEL 1(719) 576-3300
FAX 1(719) 540-1759

Biometrics/Imaging/Hi-Rel MPU/ High Speed Converters/RF Data-com
Avenue de Rochepleine
BP 123
38521 Saint-Egreve Cedex, France
TEL (33) 4-76-58-30-00
FAX (33) 4-76-58-34-80

e-mail
literature@atmel.com

Web Site
http://www.atmel.com

4214B–8051–12/02 /xM