R8C/11 Group

Control of a Brushless DC Motor

1. 0 Abstract

This document describes a control of a brushless DC motor using output compare and external interrupt functions of R8C/11 Group.

2.0 Introduction

The explanation of this document is applied in the following condition.

- MCU :R8C/11 Group
- Oscillator Frequency :20MHz
- CPU Clock :10MHz(f(XIN)/2)
- Compiler Package :NC30WA V.5.20 Release 1
- Compiler Option :-OS
  (A speed is more important than ROM capacity.)

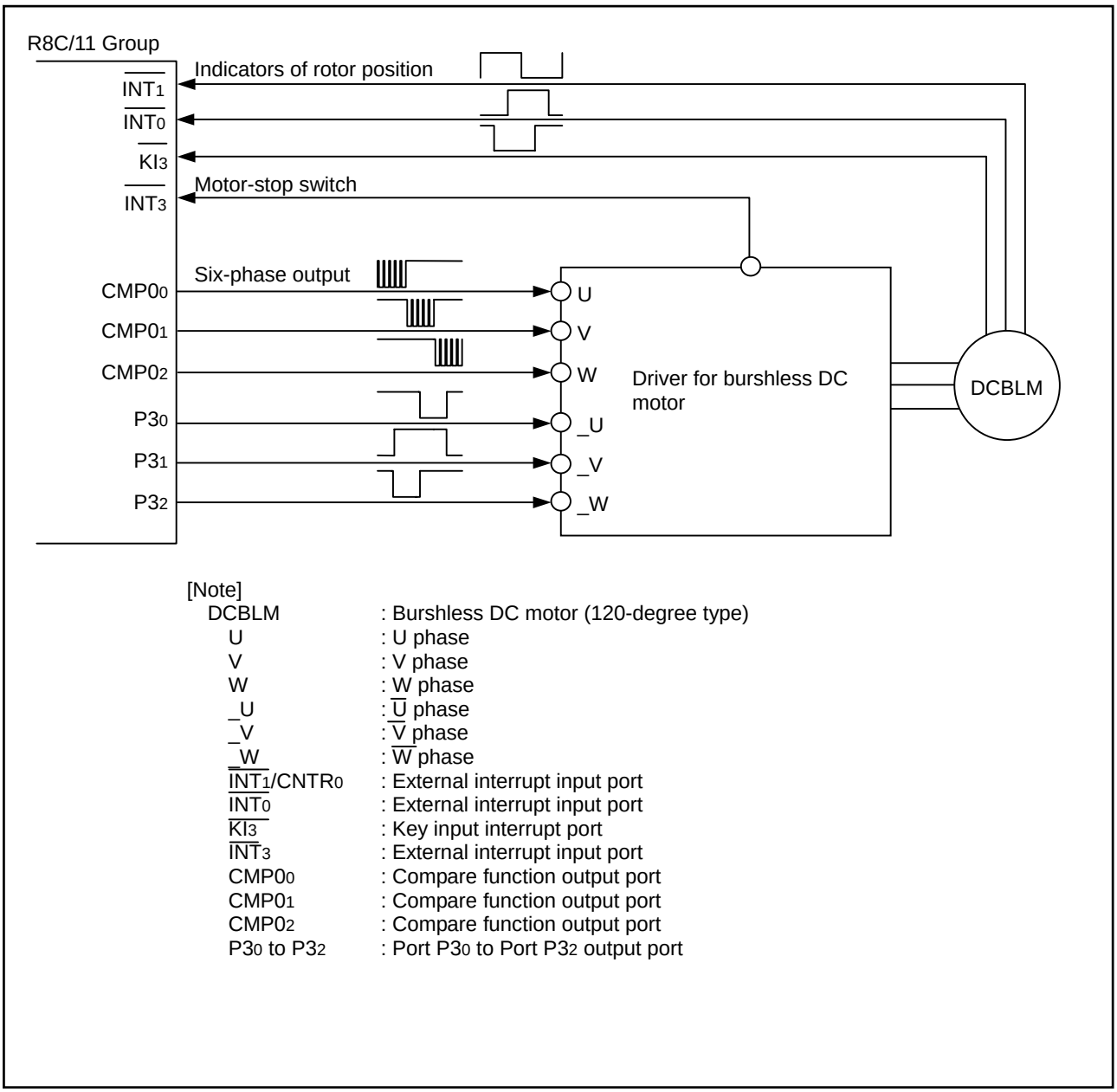
This program can also be used when operating other microcomputers within the M16C family, provided they have the same SFR (Special Function Registers) as the R8C/11 microcomputers. However, some functions may have been modified. Refer to the User's Manual for details.

Use functions covered in this Application Note only after careful evaluation.

3. 0 Control of a Brushless DC Motor

3.1 Abstract

1.The R8C/11 group is used to control a brushless DC motor in the way shown in Figure 3.1.
2.The R8C/11 group detects signals that indicate the positions of the rotor's magnetic poles and operates the motor by producing six PWM waveforms that provide control of the rotating magnetic field according to the positional signals from the motor.
3.The R8C/11 group's built-in timer generates a PWM waveform that handles chopping control for the motor.



[Note]

DCBLM	: Burshless DC motor (120-degree type)
U	: U phase
V	: V phase
W	: W phase
_U	: $\overline{U}$ phase
_V	: $\overline{V}$ phase
_W	: $\overline{W}$ phase
$\overline{INT_1}$/CNTR0	: External interrupt input port
$\overline{INT_0}$	: External interrupt input port
$\overline{KI_3}$	: Key input interrupt port
$\overline{INT_3}$	: External interrupt input port
CMP00	: Compare function output port
CMP01	: Compare function output port
CMP02	: Compare function output port
P30 to P32	: Port P30 to Port P32 output port

Figure 3.1 Set-up for Controlling a Brushless DC Motor

3.2 Specification

1.The PWM waveforms for the motor are generated by the timer and output through CMP and I/O port pins.
2.In the initial stage of the motor's operation, the motor is started by sequential switching of the excited phase on a constant cycle.
3.After switching the excited phase six times (electrical angel 1 rotation[*1]) and waiting for a certain period, control by the CPU shifts to the procedure where control of the motor is based on the rotor-positional signals from the motor.
4.The positional signals from the motor are taken in through external input port ($\overline{INT_0}$), external interrupt input port ($\overline{INT_1}$), key input interrupt port ($\overline{KI_3}$) and drive the generation of interrups.
5.These interrupts drive switching to produce a rotating magnetic field and control phase excitation through chopping.

*1 This application is premised on motor control with four magnetic poles.
1 rotating mechanical angle is 2 rotating electrical angel for motor with four poles

3.3 Descriptions of Function Used

1.As shown in Figure 3.2, timers C (output compare), external interrupt input, key input interrupt, I/O port (Port P1,P3) and timer (timer Y) of the R8C/11 Group are used to implement functions required to control a brushless DC motor.

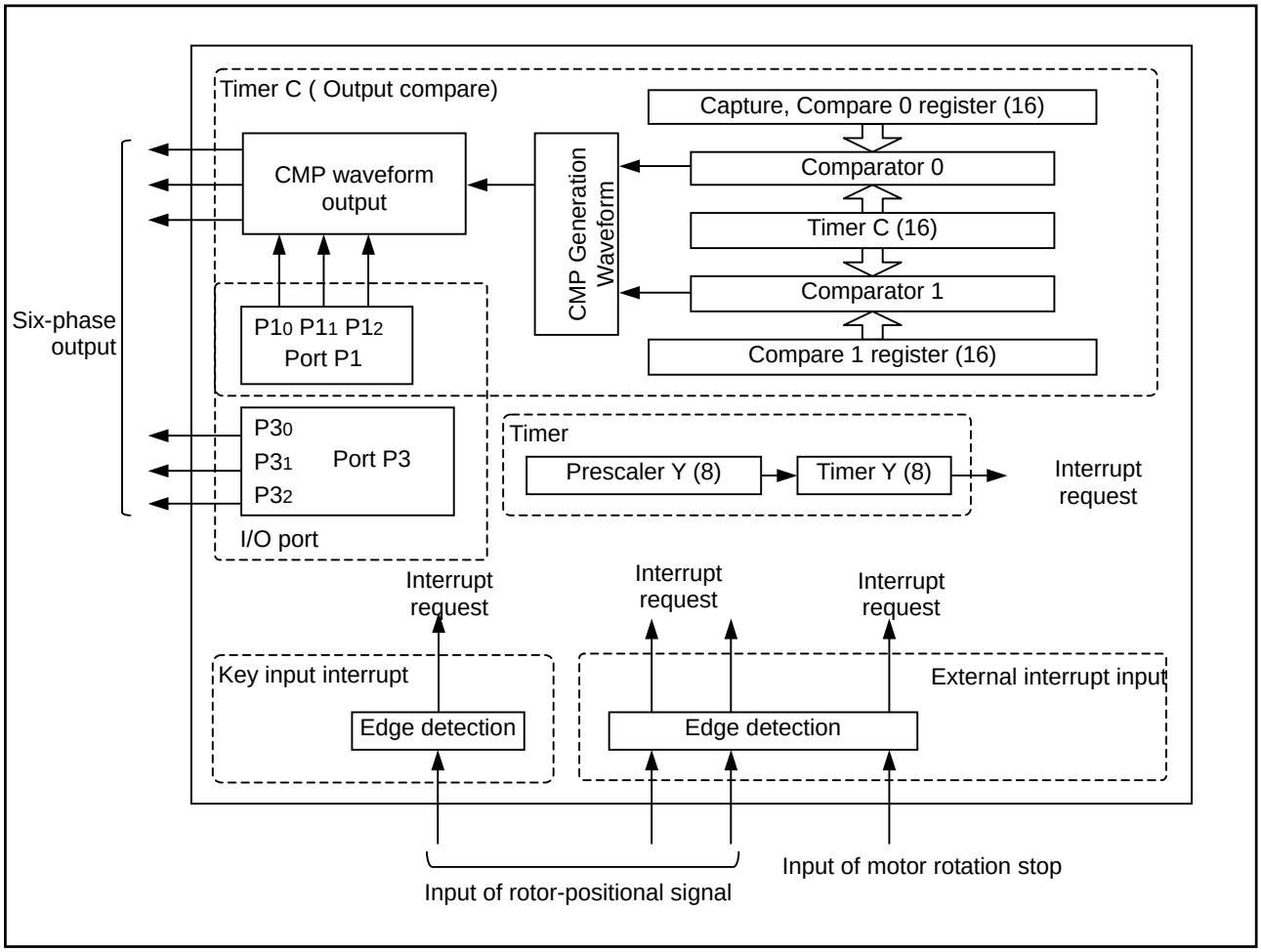
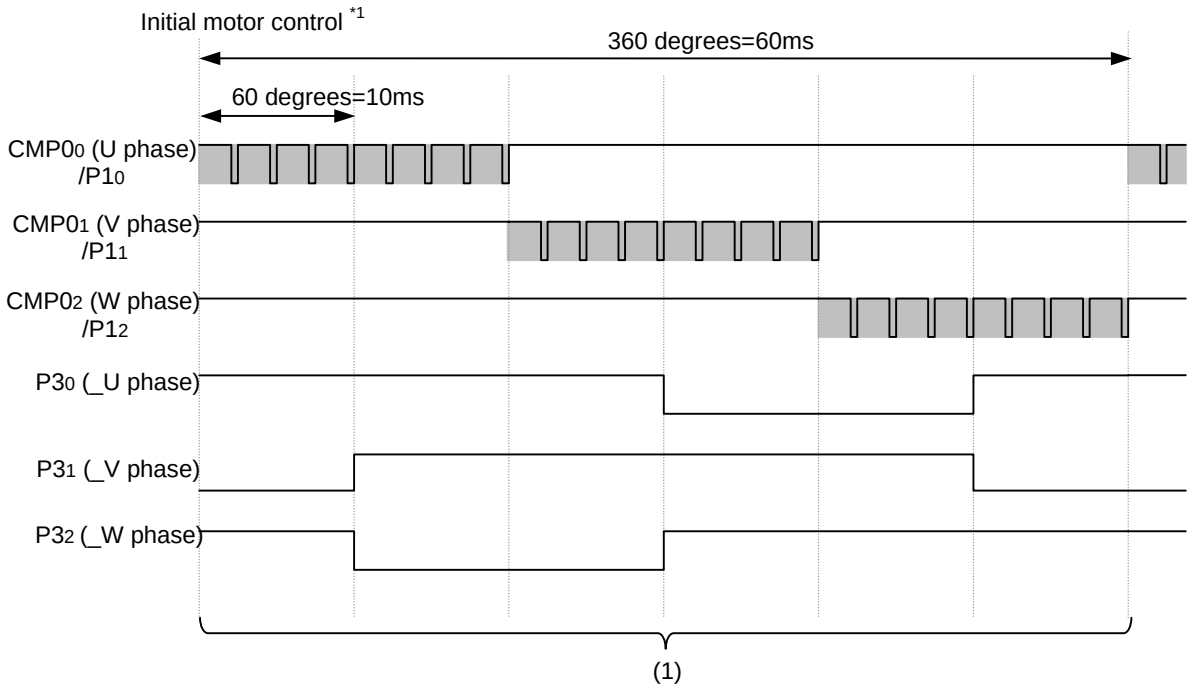


Figure 3.2 Block Diagram of the Configuration for Controlling a Brushless DC Motor

- The tasks performed by the R8C/11 Group functional blocks are outlined below.
- Timer C (output compare) : generate the waveform for chopping control when the driver transistor turns on; this is output positive three-phase data to the motor driver through $CMP0_0$ to $CMP0_2$ ports.
- External interrupt input ($\overline{INT3}$) : stops the motor in response to the driver's rotation-stop signal.
- External interrupt input ($\overline{INT0}$ to $\overline{INT1}$) : generates an interrupt request for the CPU on the rising and falling edges of the rotor positional signal.
- Key input interrupt (K13) : generates an interrupt request for the CPU on the rising and falling edges of the rotor positional signal.
- I/O port : outputs negative three-phase data to the motor driver
- Timer Y : generates an interrupt request for the CPU on a constant cycle.

3.4 Description of Operation

1. Figure 3.3 shows the principle of operation in initial motor control (until the motor has gone through its half rotation, switching of the rotating magnetic-field takes place at a constant period). Initial control of the brushless DC motor is through hardware and software processing by the R8C/11 Group as shown in figure 3.3.



	Hardware processing	Software processing
(1)	(a) Timer Y generates an interrupt request (1ms cycle)	(a) Timer Y clears an interrupt request (b) Place the sequential six-phase output data from data table in Port P1, Port P3 every 10ms

*1 The six-phase initial output pattern for initial motor control is based on the rotor-positional signal. Refer to Table 3.1 for an initial output pattern.

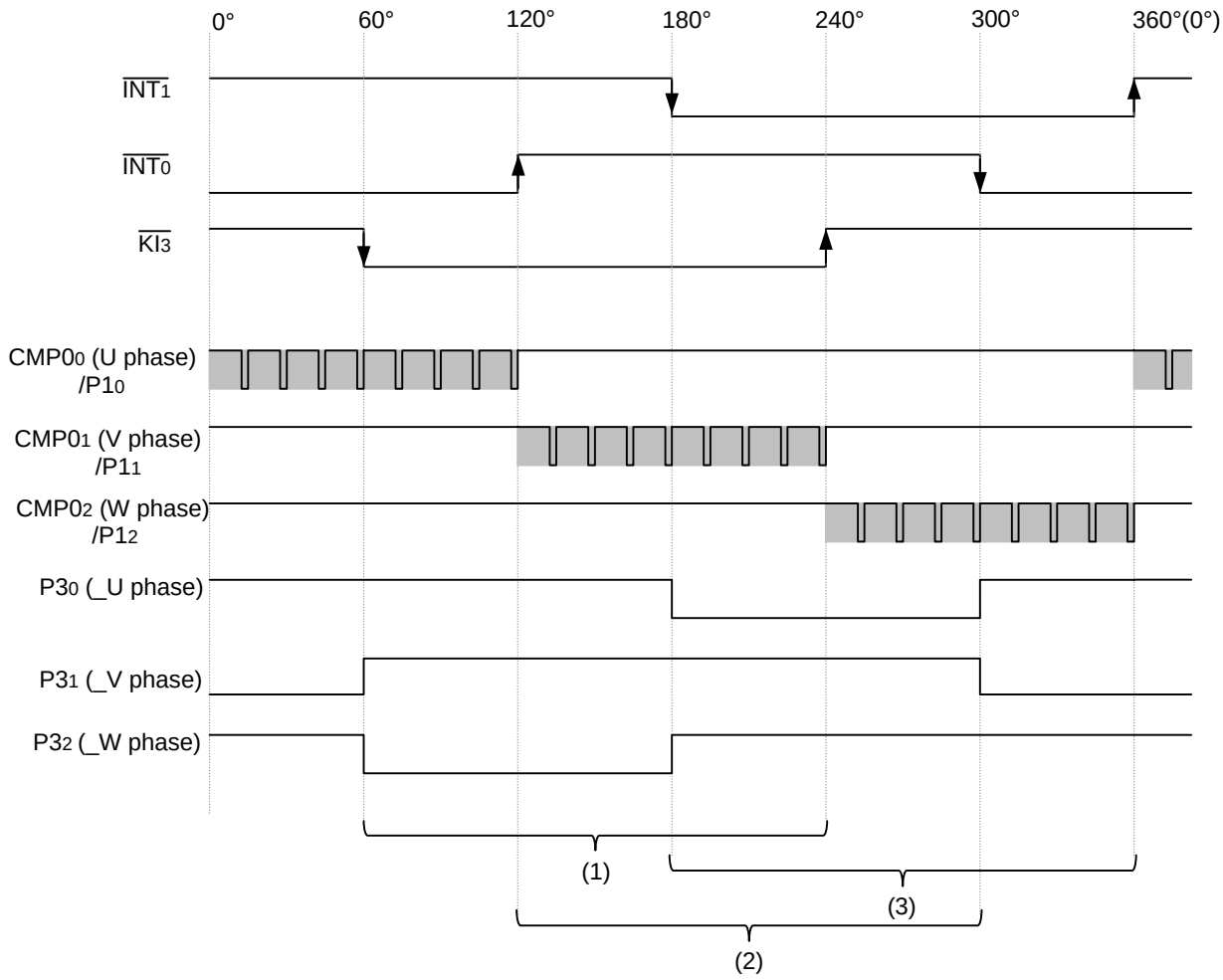
[Note] The driver is set to Low Active.

Figure 3.3 Initial Control of the Brushless DC Motor : Principle of Operation

Table 3.1 Six-phase initial output pattern for initial motor cotrol

Rotor-positional signal						
INT1	H	H	H	L	L	L
INT0	L	L	H	H	H	L
KI3	H	L	L	L	H	H
Six-phase initial output pattern						
U phase	ON	ON	OFF	OFF	OFF	OFF
V phase	OFF	OFF	ON	ON	OFF	OFF
W phase	OFF	OFF	OFF	OFF	ON	ON
_U phase	OFF	OFF	OFF	ON	ON	OFF
_V phase	ON	OFF	OFF	OFF	OFF	ON
_W phase	OFF	ON	ON	OFF	OFF	OFF

2.Figure 3.4 shows the principle of control to make the magnetic field rotate in response to the rotor-positional signal. Control of the brushless DC motor is through hardware and software processing by the R8C/11 Group, based on the detected rotor-positional signal, as is shown in figure 3.4.



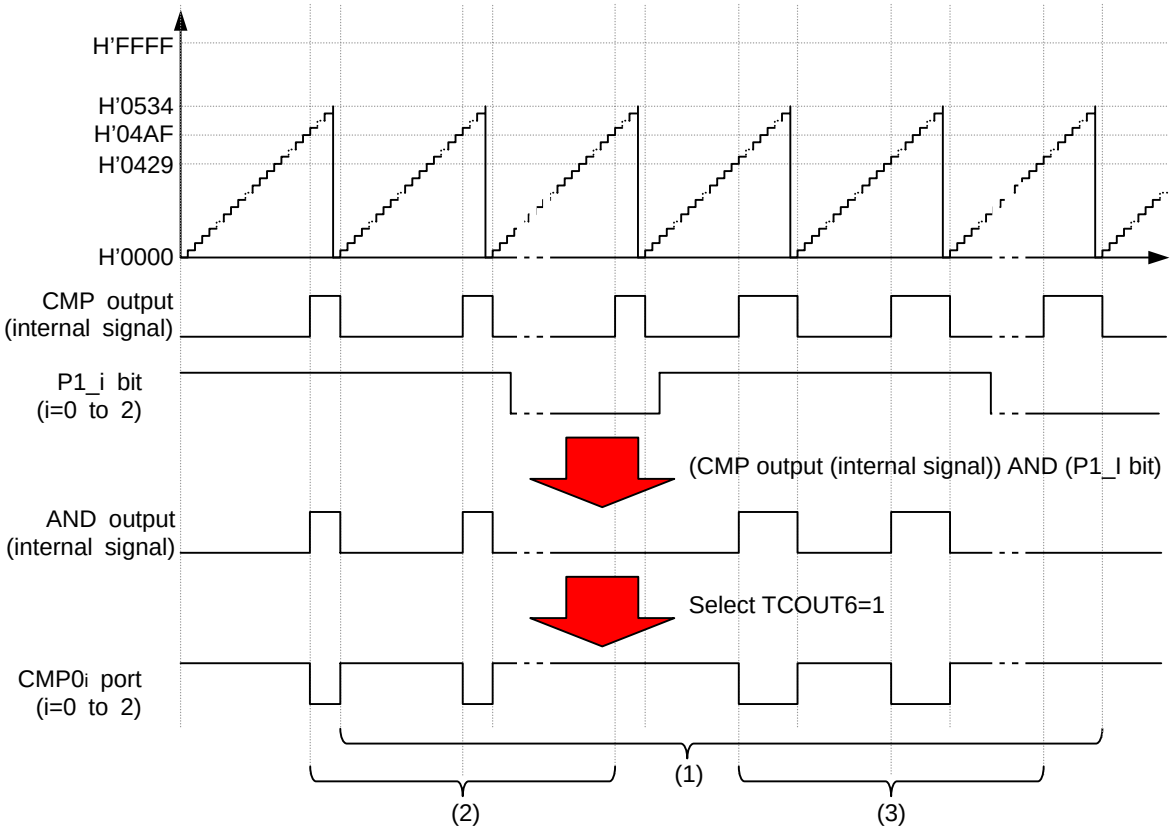
	Hardware processing	Software processing
(1)	(a) Generate a key input interrupt	(a)Verify the rotor-positional signal (b)Refer to the data table, place the corresponding six-phase output data in Port P1, Port P3. (c) Switch the KI3input polarity edge
(2)	(a) Generate INT0 interrupt	(a) Verify the rotor-positional signal (b) Refer to the data table, place the corresponding six-phase output data in Port P1, Port P3. (c)Switch the INT0 interrupt polarity edge
(3)	(a) Generate INT1 interrupt	(a) Verify the rotor-positional signal (b) Refer to the data table, place the corresponding six-phase output data in Port P1, Port P3. (c) Switch the INT1 interrupt polarity edge

*1 The output pattern on the above is a basic output pattern based on the rotor-positional signal. In this application, it is an output pattern based on 60-degree phase control. Refer to Table 4.3.

[Note]The drive is set to Low Active.

Figure 3.4 Principle of Motor Control Based on the Rotor-Positional Signal

3.In this sample task, chopping control is applied when the driver transistors on the positive-phase side are turned on. A chopping waveform generated by the output compare is output on $CMP0_0$ to $CMP0_2$. Figure 3.5 shows the principle of operation for output of the chipping waveform. Figure 3.6 shows a relationship between the internal signal and the output waveform.



	Hardware processing	Software processing
(1)	(a) Match Timer C with Compare 1 register (b) Set Timer C to "H'0000" (c) Set CMP output (internal signal) to "0" (d) Output "H" through CMP0i port	–
(2)	(a) Match Timer C with Compare 0 register (b) Set CMP output (internal signal) to "1" (c) Output "L" through CMP0i port (Duty=at 10%)	–
(3)	(a) Match Timer C with Compare 0 register (b) Set CMP output (internal signal) to "1" (c) Output "L" through CMP0i port (Duty= at 20%)	–

(i=0 to 2)

Set value
Capture, Compare 0 register : H'04AF (Duty= at 10%), H'0429 (Duty= at 20%)
Compare 1 register : H'0534

[Note]The driver is set to Low Active.

Figure 3.5 Output of the Chopping Waveform : Principle of Operation

In this application, PWM waveform is output under the setting shown in Table 3.2.

Table 3.2 Control procedure and PWM output

Control procedure	PWM cycle	Low width	Duty
Initial control of motor	66.67µs(15kHz)	6.67µs	10%
Motor control based on the rotor-positional signal		13.33µs	20%

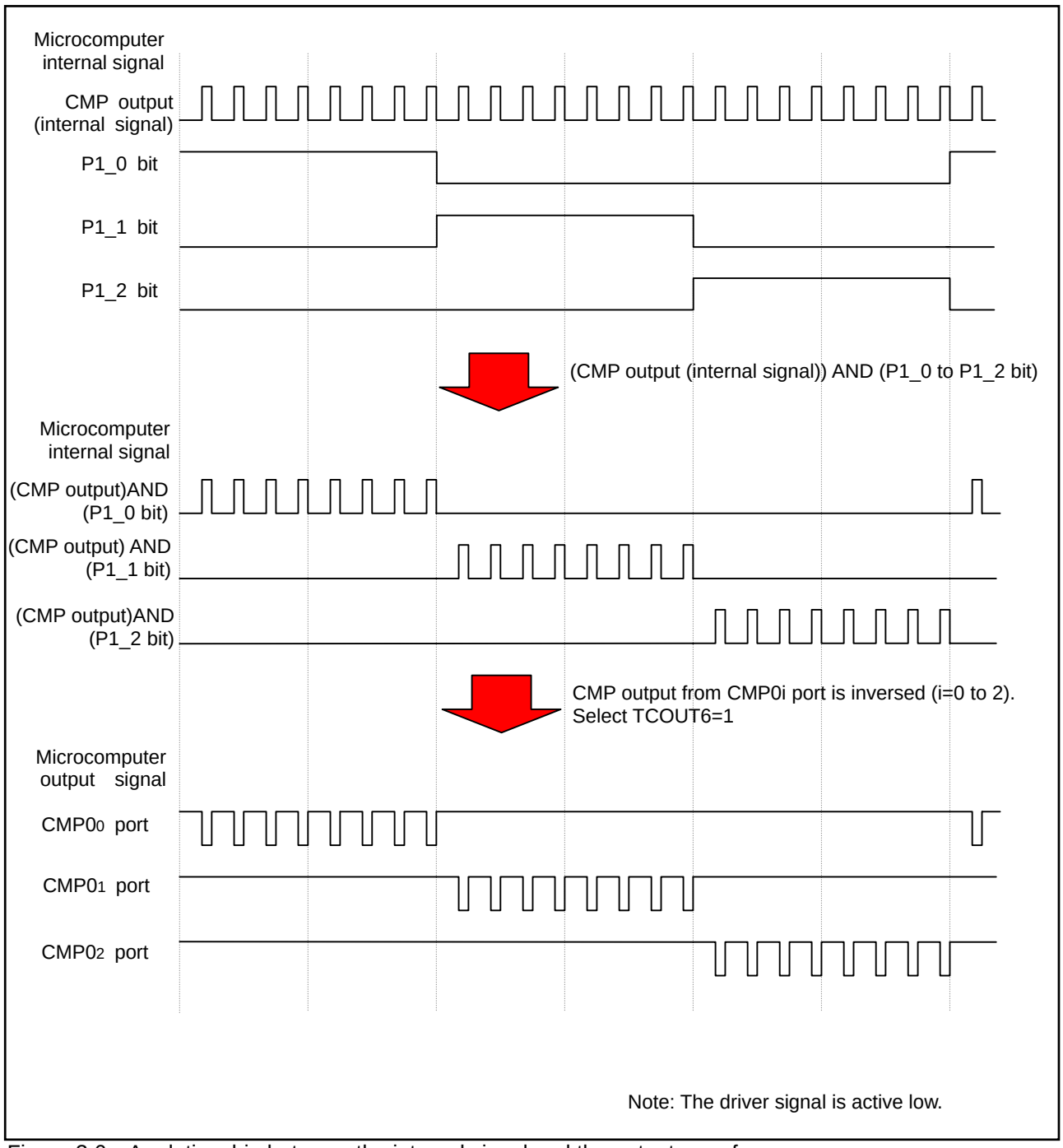


Figure 3.6 A relationship between the internal signal and the output waveform

4. Description of Software

4.1 Description of Modules

Table 4.1 describes the software used in this sample task.

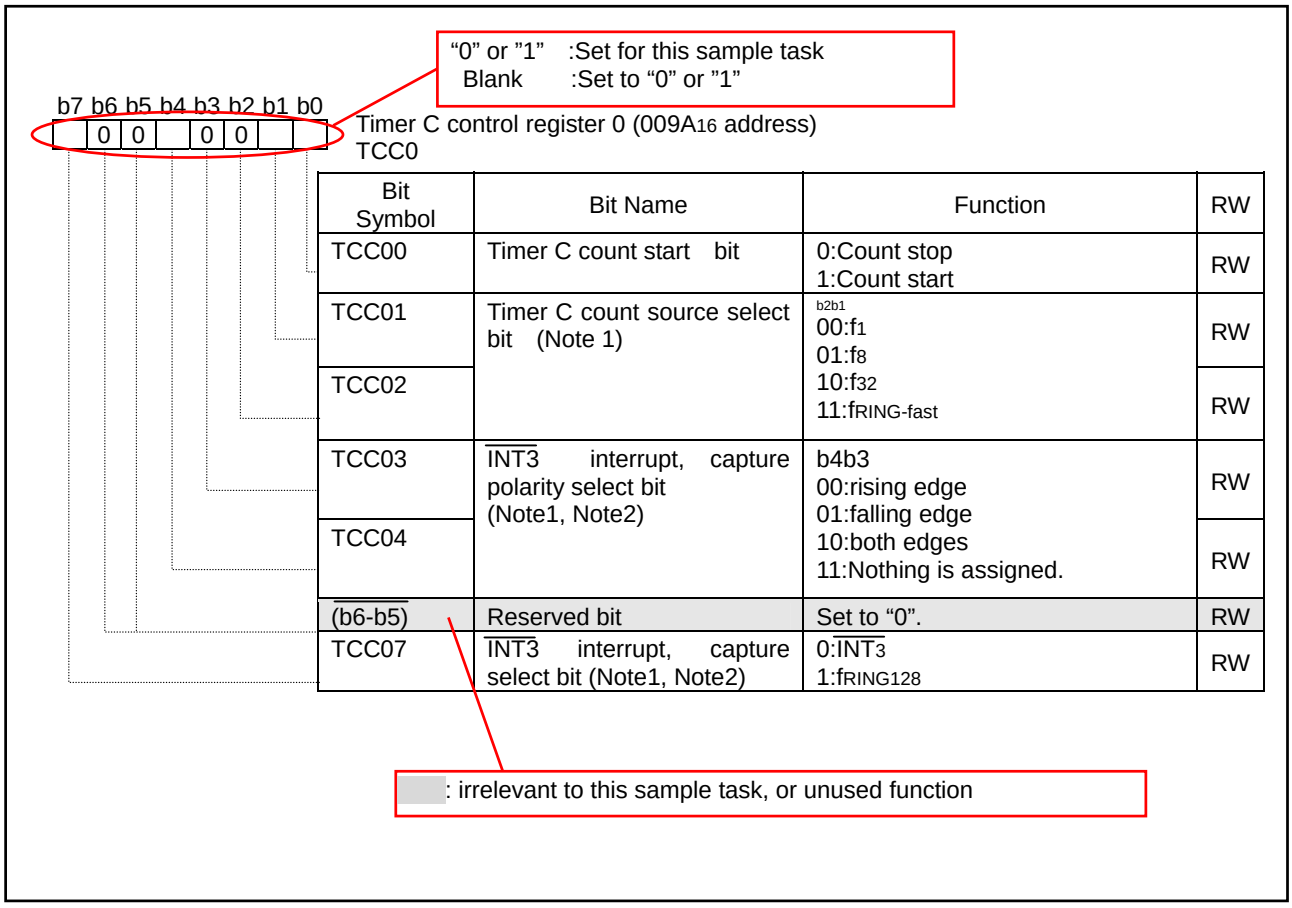
Table 4.1 Description of Modules

Module Name	Label Name
Function	
Initialization	void Init_Motor_sfr (void)
Set SFR about control.	
Main Routine	void main (void)
Perform motor control in 1ms(Timer Y) cycle.	
Motor Control Processing Routine	void Motor_Cntrl (void)
Perform initial motor control.	
Rotor-positional signal ($\overline{INT1}$) Interrupt Routine	void Sens_In0_int (void)
Switch the excited phase by rotor-positional signal output from motor to $\overline{INT1}$.	
Rotor-positional signal ($\overline{INT0}$) Interrupt Routine	void Sens_In1_int (void)
Switch the excited phase by rotor-positional signal output from motor to $\overline{INT0}$.	
Rotor-positional signal ($\overline{KI3}$) Interrupt Routine	void Sens_In2_int (void)
Switch the excited phase by rotor-positional signal output from motor to $\overline{K13}$.	
Motor-rotation stop signal ($\overline{INT3}$) Interrupt Routine	void Error_int (void)
Stop rotating the motor by motor-rotation stop signal from a brush-less DC motor driver.	
Initial-stage Excited Phase Output Routine	void _InitOutPhase_Active (u08)
Output six phases in the initial motor control.	
Excited Phase Output Routine	void _OutPhase_Active (u08)
Output six phases by rotor-positional signal in the initial motor control.	
PWM output setup Routine	void _PWM_Set (void)
Set PWM waveform output with a pre-set duty.	
Positional Signal Input Routine	u08 _Sens_Input (void)
Input rotor-positional signal.	

* u08 means unsigned char.

4.2 Description of SFR for Internal Use

The figures from 4.2 to 4.30 describes SFR for internal use in this sample task.
Figure 4.1 describes what the figures show in detail. Refer to the hardware manual for the further description of the register.



Bit Symbol	Bit Name	Function	RW
TCC00	Timer C count start bit	0:Count stop 1:Count start	RW
TCC01	Timer C count source select bit (Note 1)	b2b1 00:f_1 01:f_8 10:f_{32} 11:$f_{RING-fast}$	RW
TCC02			RW
TCC03	$\overline{INT3}$ interrupt, capture polarity select bit (Note1, Note2)	b4b3 00:rising edge 01:falling edge 10:both edges 11:Nothing is assigned.	RW
TCC04			RW
(b6-b5)	Reserved bit	Set to "0".	RW
TCC07	$\overline{INT3}$ interrupt, capture select bit (Note1, Note2)	0:$\overline{INT3}$ 1:$f_{RING128}$	RW

Figure 4.1 Description of what SFR shows

b7	b6	b5	b4	b3	b2	b1	b0
0	0	0	0	1	0	0	0

System clock control register 0 (0006$_{16}$ address)
CM0

Bit Symbol	Bit Name	Function	RW
(b1-b0)	Reserved bit	Set to"0".	RW
CM02	WAIT peripheral function clock stop bit	0:Do not stop peripheral function clock in wait mode 1:Stop peripheral function clock in wake mode (Note 6)	RW
(b3)	Reserved bit	Set to "1".	RW
(b4)	Reserved bit	Set to "0".	RW
CM05	Main clock (XIN-XOUT) stop bit (Note 2, Note 4)	0:On 1:Off (Note 3)	RW
CM06	Main clock division select bit 0 (Note 5)	0:CM16, CM17 valid 1:divide-by-8 mode	RW
(b7)	Reserved bit	Set to "0".	RW

Note1 Set the PRC0 bit of PRCR register to "1" (write enable) before writing to this register.
Note2 The CM05 bit is provided to stop the main clock when the ring oscillator mode is selected. This bit cannot be used for detection as to whether the main clock stopped or not.
(1) Set the CM06 bit to "1" (divide-by-8 mode)
(2) Set the OCD0 and OCD1 register to "00$_2$" (disabling oscillation stop detection function)
(3) Set the OCD2 bit to "1" (selecting ring oscillator)
Note3 During external clock input, only the clock oscillation buffer is turned off and clock input is accepted.
Note4 When the CM05 bit is set to "1" (main clock stop), $P4_6$ and $P4_7$ can be used as input ports.
Note5 When entering stop mode from high or middle speed mode, the CM06 bit is set to "1" (divide-by-8 mode)
Note6 During ring oscillator mode, this bit must be set to "0" (peripheral clock turned on when in wait mode)

Figure 4.2 System clock control register 0

b7	b6	b5	b4	b3	b2	b1	b0
0	1	1	1	1	0	0	0

System clock control register 1 (0007$_{16}$ address)
CM1

Bit Symbol	Bit Name	Function	RW
CM10	All clock stop control bit (Note 4)	0:Clock on 1:All clocks off (stop mode)	RW
(b1)	Reserved bit	Set to “0”.	RW
(b2)	Reserved bit	Set to “0”.	RW
CM13	Port XIN-XOUT Switch bit	0:Input portP4$_6$, P4$_7$ 1:XIN-XOUT pin	RW
CM14	Low-speed ring oscillation stop bit (Note5, 6)	0:Low-speed ring oscillator ring on 1:Low-speed ring oscillator ring off	RW
CM15	XIN-XOUT drive capability select bit (Note2)	0:LOW 1:HIGH	RW
CM16	Main clock division select bit 1 (Note3)	b7b6 00:No division mode 01:Division by 2 mode 10:Division by 4 mode 11:Division by 16 mode	RW
CM17			

Note1 Write to this register after setting the PRC0 bit of PRCR register to “1”.
Note2 When entering stop mode from high or middle speed mode, the CM15 bit is set to “1” (drive capability high).
Note3 Effective when the CM06 bit is “0” (CM16 and CM17 bits enable).
Note4 If the CM10 bit is “1” (stop mode), the internal feedback resistor becomes ineffective.
Note5 The CM14 bit can be set to “1” (low-speed ring oscillator off) if the OCD2 bit=0 (selecting main clock). When the OCD2 bit is set to “1” (selecting ring oscillator clock), the CM14 bit is set to “0” (low-speed ring oscillator on). This bit remains unchanged when “1” is written.
Note6 When using voltage detection interrupt circuit, CM14 bit is set to “0”.

Figure 4.3 System clock control register 1

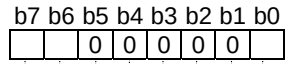
b7	b6	b5	b4	b3	b2	b1	b0
0	0	0	0		0	0	0

Oscillation stop detection register (000C16 address)
OCD

Bit symbol	Bit Name	Function	RW
OCD0	Oscillation stop detection enable bit	b1b0 00:The function is disabled 01:Avoid this setting 10:Avoid this setting 11:The function is enabled (Note4,7)	RW
OCD1			
OCD2	System clock select bit (Note6)	0:Select main clock (Note7) 1:Select ting oscillator clock (Note2)	RW
OCD3	Clock monitor bit (Note3,5)	0:Main clock on 1:Main clock off	RO
(b7-b4)	Reserved bit	Set to “0”	RW

Note1 Set the PRC0 bit in the PRCR register to “1” (write enable) before rewriting this register.
Note2 The OCD2 bit is set to “1”(selecting ring oscillator clock) automatically if a main clock oscillation stop is detected while the OCD1 to OCD0 bits are set to “112” (oscillation stop detection function enabled). If the OCD3 bit is set to “1”(main clock stop), the OCD2 bit remains unchanged when trying to write “0” (selecting main clock).
Note3 The OCD3 bit is enabled when the OCD1 to OCD0 bits are set to “112” (oscillation stop detection function enabled). Read the OCD3 bit several times with the oscillation stop detection interrupt processing program to determine the main clock state.
Note4 The OCD1 to OCD0 bits should be set to “002” (oscillation stop detection function disabled) before entering stop mode and ring oscillator (main clock stops). The OCD1 to OCD0 bits should be set to “002” when the HR01 bit in the HR0 register is set to “1” (high-speed ring oscillator selected).
Note5 The OCD3 bit remains set to “0” (main clock on) if the OCD1 to OCD0 bits are set to “002”.
Note6 The CM14 bit goes to “0”(low-speed ring oscillator on) if the OCD2 bit is set to “1” (selecting ring oscillator clock).
Note7 Refer to Figure 6.9 “switching clock source from low-speed ring oscillator to main clock” for the switching procedure when the main clock re-oscillates after detecting an oscillation stop.

Figure 4.4 Oscillation stop detection register

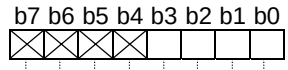


Protect register (000A16 address)
PRCR

Set Symbol	Bit Name	Function	RW
PRC0	Protect bit 0	Enable write to CM0,CM1,OCD,HR0,HR1 registers 0:Write protected 1:Write enabled	RW
PRC1	Protect bit 1	Enable write to PM0, PM1 registers 0:Write protected 1:Write enabled	RW
PRC2	Protect bit 2	Enable write to PD0 register 0:Write protected 1:Write enabled (Note1)	RW
PRC3	Protect bit 3	Enable write to VCR2, D4INT registers 0:Write protected 1:Write enabled	RW
(b5-b4)	Reserved bit	When write, should set to “0”.	RW
(b7-b6)	Reserved bit	When read, its content is “0”.	RO

Note1 The PRC2 bit is set to “0” by writing to any address after setting it to “1”. Other bits are not set to “0” by writing to any address, and must therefore be set to “0” in a program.

Figure 4.5 Protect register

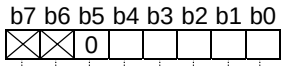


Interrupt control register
KUPIC (004D16 address)
INT1IC (005916 address)
INT3IC (005A16 address)

Bit Symbol	Bit Name	Function	RW
ILVL0	Interrupt priority level select bit	b2b1b0 000:Level 0 (interrupt disabled) 001:Level 1 010:Level 2 011:Level 3 100:Level 4 101:Level 5 110:Level 6 111:Level 7	RW
ILVL1			RW
ILVL2			RW
IR	Interrupt request bit	0:Interrupt not requested 1:Interrupt requested	RW (Note1)
~~(b7-b4)~~	Nothing is assigned. When write, set to “0”. When read, its content is indeterminate.		-

Note1 Only “0” can be written to the IR bit. (Do not write “1”).
Note2 To rewrite the interrupt control register, do so at a point that does not generate the interrupt request for that register. Refer to “19.2.6 Changing Interrupt Control Registers” in the Hardware Manual.

Figure 4.6 Interrupt control register

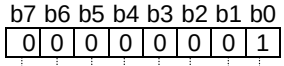


Interrupt control register
INT0IC (005D16 address)

Bit Symbol	Bit Name	Function	RW
ILVL0	Interrupt priority level select bit	b2b1b0 000:Level 0 (interrupt disabled) 001:Level 1 010:level 2 011:Level 3 100:Level 4 101:Level 5 110:Level 6 111:Level 7	RW
ILVL1			RW
ILVL2			RW
IR	Interrupt request bit	0:Interrupt not requested 1:Interrupt requested	RW (Note1)
POL	Polarity select bit	0:Selects falling edge 1:Selects rising edge (Note3)	RW
(b5)	Reserved bit	Must always be set to “0”	RW
(b7-b6)	Nothing is assigned. When write, set to “0”. When read, its content is indeterminate.		-

Note1 Only “0” can be written to the IR bit. (Do not write “1”).
Note2 To rewrite the interrupt control register, do so at appoint that does not generate the interrupt request for that register. Refer to “19.2.6 Changing Interrupt Control Registers” in the Hardware Manual.
Note3 If the INT0PL bit in the INTEN register is set to “1” (both edges), set the POL bit to “0” (selecting falling edge).
Note4 The IR bit may be set to “1”(interrupt requested) when the POL bit is rewritten. Refer to “19.2.5 Changing Interrupt source” in the Hardware Manual.

Figure 4.7 Interrupt control register



External input enable register (009616 address)
INTEN

Bit Symbol	Bit Name	Function	RW
INT0EN	INT0 input enable bit (Note1)	0:Disabled 1:Enabled	RW
INT0PL	INT0 input polarity select bit (Note2,3))	0:One edge 1:Both edges	RW
(b7-b2)	Reserved bit	Must set to “0”.	RW

Note1 This bit must be set while the INT0STG bit in the PUM register is set to “0” (one-shot trigger disabled)
Note2 When setting the INT0PL bit to “1”(selecting both edges), the POL bit in the INT0IC must be set to “0”(selecting falling edge).
Note3 The IR bit in the INT01C register may be set to “1”(interrupt requested) when the INT0PL bit is rewritten. Refer to “19.2.5 Changing Interrupts Source” in the Hardware Manual.

Figure 4.8 External input enable register

b7 b6 b5 b4 b3 b2 b1 b0
X X X X X 0 0 0

$\overline{INT0}$ input filter select register (001E16 address)
INT0F

Bit Symbol	Bit Name	Function	RW
INT0F0	$\overline{INT0}$ input filter select bit	b1b0 00 : No filter 01 : Filter with f1 sampling 10 : Filter with f8 sampling 11 : Filter with f32 sampling	RW
INT0F1			RW
(b2)	Reserved bit	Must set to "0".	RW
(b7-b3)	Nothing is assigned. When write, set to "0". If read, its content is indeterminate.		-

Figure 4.9 $\overline{INT0}$ input filter select register

b7 b6 b5 b4 b3 b2 b1 b0
0 0 0 0 0 0

Timer X mode register (008B16 address)
TXMR

Bit Symbol	Bit Name	Function	RW
TXMOD0	Operation mode select bit 0,1	b1b0 00 : Timer mode or pulse period measurement mode (Note3)	RW
TXMOD1			RW
R0EDG	$\overline{INT1}$/CNTR0 polarity switching bit (Note2,3)	0 : Rising edge 1 : Falling edge	RW
TXS	Timer X count start flag	0 : Stops counting 1 : Starts counting	RW
TXOCNT	Must set to "0" in timer mode		RW
TXMOD2	Operation mode select bit 2	0 : Other than pulse period measurement mode (Note3)	RW
TXEDG	Must set to "0" in timer mode		RW
TXUND	Must set to "0" in timer mode		RW

Note1 The IR bit in the INT1IC may be set to "1" (interrupt requested) when the R0EDG bit is rewritten. Refer to "19.2.5 Changing Interrupt Source" in the Hardware Manual.
Note2 This bit is used to select the polarity of $\overline{INT1}$ interrupt in timer mode.
Note3 When using $\overline{INT1}$ interrupts, should select timer mode.

Figure 4.10 Timer X mode register

b7	b6	b5	b4	b3	b2	b1	b0
					0		0

Timer Y, Z mode register (0080_{16} address)
TYZMR

Bit Symbol	Bit Name	Function	RW
TYMOD0	Timer Y operation mode bit	0 : Timer mode (Note1)	RW
R1EDG	$\overline{INT2}$/CNTR1 polarity switching bit (Note1,2)	0 : Rising edge 1 : Falling edge	RW
TYWC	Timer Y write control bit	0 : Write to reload register and counter simultaneously 1 : Write to reload register	RW
TYS	Timer Y count start flag	0 : Stops counting 1 : Starts counting	RW
TZMOD0	Timer Z-related bit		RW
TZMOD1			RW
TZWC			RW
TZS			RW

Note1 When using $\overline{INT2}$ interrupt, must set to timer mode.
Note2 The IR bit in the INT2IC may be "1" (interrupt requested) when R1EDG bit is rewritten. Refer to "19.2.5 Changing Interrupt Source" in the Hardware Manual.

Figure 4.11 Timer Y, Z mode register

b7	b6	b5	b4	b3	b2	b1	b0
	1		0		0		0

Key input enable register (0098_{16} address)
KIEN

Bit Symbol	Bit Name	Function	RW
KI0EN	KI0 input enable bit	0 : Disabled 1 : Enabled	RW
KI0PL	KI0 input polarity select bit	0 : Falling edge 1 : Rising edge	RW
KI1EN	KI1 input enable bit	0 : Disabled 1 : Enabled	RW
KI1PL	KI1 input polarity select bit	0 : Falling edge 1 : Rising edge	RW
KI2EN	KI2 input enable bit	0 : Disabled 1 : Enabled	RW
KI2PL	KI2 input polarity select bit	0 : Falling edge 1 : Rising edge	RW
KI3EN	KI3 input enable bit	0 : Disabled 1 : Enabled	RW
KI3PL	KI3 input polarity select bit	0 : Falling edge 1 : Rising edge	RW

Note1 The IR bit in the KUPIC may be set to "1" (interrupt requested) when the KIEN register is written. Refer to "19.2.5 Changing Interrupt Source" in the Hardware Manual.

Figure 4.12 Key input enable register

b7 b6 b5 b4 b3 b2 b1 b0
0 0 0 1

Timer count source setting register (008E16 address)
TCSS

Bit Symbol	Bit Name	Function	RW
TXCK0	Timer X count source select bit (Note1)	b1b0 00 : f1 01 : f8 10 : f32 11 : f2	RW
TXCK1			RW
TYCK0	Timer Y count source select bit (Note1)	b3b2 00 : f1 01 : f8 10 : fRING 11 : Selects input from CNTR1 pin	RW
TYCK1			RW
TZCK0	Timer Z count source select bit (Note1)	b5b4 00 : f1 01 : f8 10 : Selects Timer Y underflow 11 : f2	RW
TZCK1			RW
(b7-b6)	Reserved bit	Must set to be “0”	RW

Note1 Avoid switching a count source while a counter is in progress. Timer counter must be stopped before switching a count source.

Figure 4.13 Timer count source setting register

b7 B0

Prescaler Y register (008116 address)
PREY

Mode	Function	Setting range	RW
Timer mode	Internal count source or CNTR1 input is counted.	0016 to FF16	RW
Programmable waveform generation mode	Internal count source is counted.	0016 to FF16	RW

Figure 4.14 Prescaler Y register

b7 b0

Timer Y primary register (008316 address)
TYPR

Mode	Function	Setting range	RW
Timer mode	Underflow of Prescaler Y is counted.	0016 to FF16	RW
Programmable waveform generation mode	Underflow of Prescaler is counted. (Note1)	0016 to FF16	RW

Note1 The values of TYPR register and TYSC register are reloaded to the counter alternately for counting.

Figure 4.15 Timer Y primary register

b7	b6	b5	b4	b3	b2	b1	b0
0	0	0	1	0	0	0	

Timer C control register 0 (009A16 address)
TCC0

Bit Symbol	Bit Name	Function	RW
TCC00	Timer C count start bit	0 : Stops counting 1 : Starts counting	RW
TCC01	Timer C count source select bit (Note1)	b2b1 00 : f_1 01 : f_8 10 : f_{32} 11 : $f_{RING-fast}$	RW
TCC02			RW
TCC03	$\overline{INT3}$ interrupt and capture polarity select bit (Note1,2))	b4b3 00 : Rising edge 01 : Falling edge 10 : Both edges 11 : Avoid this setting	RW
TCC04			RW
$\overline{(b6\text{-}b5)}$	Reserved bit	Must set to “0”	RW
TCC07	$\overline{INT3}$ interrupt/capture input switching bit (Note1,2)	0 : $\overline{INT_3}$ 1 : $f_{RING128}$	RW

Note1 Change this bit when TCC00 bit is set to “0” (count stop).
Note2 The IR bit in the INT3IC may be set to “1” (interrupt requested) when the TCC03, TCC04, or TCC07 bit is rewritten. Refer to “19.2.5 Changing Interrupt Source” in the Hardware Manual.

Figure 4.16 Timer C control register 0

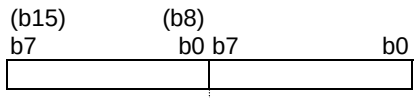
b7	b6	b5	b4	b3	b2	b1	b0
1	0	1	1	1	1	0	0

Timer C control register 1 (009B16 address)
TCC1

Bit Symbol	Bit Name	Function	RW
TCC10	$\overline{\text{INT3}}$ filter select bit (Note1)	b1b0 00 : No filter 01 : Filter with f1 sampling 10 : Filter with f8 sampling 11 : Filter with f32 sampling	RW
TCC11			RW
TCC12	Timer C counter reload select bit (Note2,3)	0 : No reload (free-run) 1 : Set TC register to "000016" at compare 1 match	WO
TCC13	Compare 0/Capture select bit	0 : Capture (input capture mode) (Note2) 1 : Compare 0 output (output compare mode)	RW
TCC14	Compare 0 output mode select mode (Note3)	b5b4 00 : CMP0 output remains unchanged even when compare 0 signal matched 01 : CMP0 output is reversed when compare 0 signal is matched. 10 : CMP0 output is set to low when compare 0 signal is matched. 11 : CMP0 output is set to high when compare 0 signal is matched.	RW
TCC15			RW
TCC16	Compare 1 output mode select bit (Note3)	b7b6 00 : CMP1 remains unchanged even when compare 1 signal is matched. 01 : CMP1output is reversed when compare 1 signal is matched. 10 : CMP1output is set to low when compare 1 signal is matched. 11 : CMP1output is set to high when compare 1 signal is matched.	RW
TCC17			RW

Note1 Input is recognized only when the same value from INT3 pin is sampled three times in succession.
Note2 The TCC13 bit in the TCC0 register should be "0" (count stop) when rewriting the TCC13 bit.
Note3 The TCC12 and TCC14 to TCC17 should be set to "0" when the TCC13 bit is "0" (input capture mode).

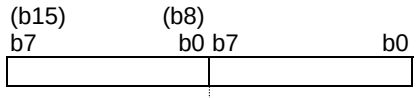
Figure 4.17 Timer C control register 1



Timer C register (0091_{16}-0090_{16} address)
TC

Function	RW
Internal count source is counted. "0000_{16}" is read out by reading TCC00 bit = 0 (stops counting). Count value is read out by reading when TCC00 bit = 1 (starts counting).	RO

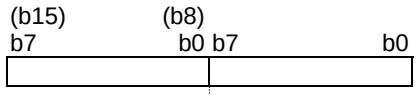
Figure 4.18 Timer C register



Capture and compare 0 register ($009D_{16}$-$009C_{16}$ address)
TM0

Mode	Function	Setting range	RW
Input capture mode	When active edge of measurement pulse is input, the value in the TC register is stored.	-	RO
Output compare mode	The value compared with Timer C is stored.	0000_{16} to $FFFF_{16}$	RW

Figure 4.19 Capture and compare 0 register



Compare 1 register ($009F_{16}$-$009E_{16}$ address)
TM1

Mode	Function	Setting range	RW
Output compare mode (Note1)	The value compared with Timer C is stored.	0000_{16} to $FFFF_{16}$	RW

Note1 Disabled when the TCC13 register is set to “0” (input capture mode).

Figure 4.20 Compare 1 register

b7	b6	b5	b4	b3	b2	b1	b0
0	1	0	0	0	1	1	1

Timer C output control register (00FF16 address)
TCOUT

Bit Symbol	Bit Name	Function	RW
TCOUT0	CMP output enable bit 0	0 : Disable CMP output from CMP00 1 : Enable CMP output from CMP00.	RW
TCOUT1	CMP output enable bit 1	0 : Disable CMP output from CMP01. 1 : Enable CMP output from CMP01.	RW
TCOUT2	CMP output enable bit 2	0 : Disable CMP output from CMP02 1 : Enable CMP output from CMP02.	RW
TCOUT3	CMP output enable bit 3	0 : Disable CMP output from CMP10. 1 : Enable CMP output form CMP10.	RW
TCOUT4	CMP output enable bit 4	0 : Disable CMP output from CMP11. 1 : Enable CMP output from CMP11.	RW
TCOUT5	CMP output enable bit 5	0 : Disable CMP output from CMP12. 1 : Enable CMP output from CMP12.	RW
TCOUT6	CMP output reverse bit 0	0 : Not reverse CMP output from CMP00 to CMP02 1 : Reverse CMP output from CMP00 to CMP02	RW
TCOUT7	CMP output reverse bit 1	0 : Not reverse CMP output from CMP10 to CMP12 1 : Reverse CMP output from CMP10 to CMP12	RW

Note1 Invalid when the TCC 13 bit in the TCC1 register is set to “0” (input capture mode).

Figure 4.21 Timer C output control register

b7 b6 b5 b4 b3 b2 b1 b0

Port P1 register (00E1$_{16}$ address)
P1

Bit Symbol	Bit Name	Function	RW
P1_0	Port P1$_0$ bit	The pin level on any I/O port which is set for input mode can be read by reading the corresponding bit in this register. The pin level on any I/O port which is set for output mode can be controlled by writing to the corresponding bit in this register. 0:"L" level 1:"H" level	RW
P1_1	Port P1$_1$ bit		RW
P1_2	Port P1$_2$ bit		RW
P1_3	Port P1$_3$ bit		RW
P1_4	Port P1$_4$ bit		RW
P1_5	Port P1$_5$ bit		RW
P1_6	Port P1$_6$ bit		RW
P1_7	Port P1$_7$ bit		RW

Figure 4.22 Port P1 register

b7 b6 b5 b4 b3 b2 b1 b0
0 0 0 0 1 1 1 1

Port P1 direction register (00E3$_{16}$ address)
PD1

Bit Symbol	Bit Name	Function	RW
PD1_0	Port P1$_0$ direction bit	0 : Input mode (Functions as an input mode) 1 : Output mode (Functions as an output mode)	RW
PD1_1	Port P1$_1$ direction bit		RW
PD1_2	Port P1$_2$ direction bit		RW
PD1_3	Port P1$_3$ direction bit		RW
PD1_4	Port P1$_4$ direction bit		RW
PD1_5	Port P1$_5$ direction bit		RW
PD1_6	Port P1$_6$ direction bit		RW
PD1_7	Port P1$_7$ direction bit		RW

Figure 4.23 Port P1 register

b7 b6 b5 b4 b3 b2 b1 b0
0 0 0

Port P3 register (00E5$_{16}$ address)
P3

Bit Symbol	Bit Name	Function	RW
P3_0	Port P3$_0$ bit	The pin level on any I/O port which is set for input mode can be read by reading the corresponding bit in this register. The pin level on any I/O port which is set for output mode can be controlled by writing to the corresponding bit in this register. 0:"L" level 1:"H" level	RW
P3_1	Port P3$_1$ bit		RW
P3_2	Port P3$_2$ bit		RW
P3_3	Port P3$_3$ bit		RW
P3_4	Port P3$_4$ bit (Note1)		RW
P3_5	Port P3$_5$ bit (Note1)		RW
P3_6	Port P3$_6$ bit (Note1)		RW
P3_7	Port P3$_7$ bit		RW

Note1 Nothing is assigned. When writing to this bit, write "1" (low level). When read, its content is "0".

Figure 4.24 Port P3 register

b7	b6	b5	b4	b3	b2	b1	b0
	0	0	0	0	1	1	1

Port P3 direction register (00E7$_{16}$ address)
PD3

Bit Symbol	Bit Name	Function	RW
PD3_0	Port P3$_0$ direction bit	0 : Input mode (Function as an input port) 1 : Output mode (Functions as an output mode)	RW
PD3_1	Port P3$_1$ direction bit		RW
PD3_2	Port P3$_2$ direction bit		RW
PD3_3	Port P3$_3$ direction bit		RW
PD3_4	Port P3$_4$ direction bit (Note1)		RW
PD3_5	Port P3$_5$ direction bit (Note1)		RW
PD3_6	Port P3$_6$ direction bit (Note1)		RW
PD3_7	Port P3$_7$ direction bit		RW

Note1 Nothing is assigned. When writing to this bit, write “0” (input mode). When reading, its content is “0”.

Figure 4.25 Port P3 direction register

b7	b6	b5	b4	b3	b2	b1	b0
			0	0	0	0	0

Port P4 register (00E8$_{16}$ address)
P4

Bit Symbol	Bit Name	Function	RW
P4_0	Port P4$_0$ bit (Note1)	The pin level on any I/O port which is set for input mode can be read by reading the corresponding bit in this register. The pin level on any I/O port which is set for output mode can be controlled by writing to the corresponding bit in this register. 0:”L” level 1:”H” level	RW
P4_1	Port P4$_1$bit (Note1)		RW
P4_2	Port P4$_2$ bit (Note1)		RW
P4_3	Port P4$_3$ bit (Note1)		RW
P4_4	Port P4$_4$ bit (Note1)		RW
P4_5	Port P4$_5$ bit		RW
P4_6	Port P4$_6$ bit		RW
P4_7	Port P4$_7$ bit		RW

Note1 Nothing is assigned. When writing to this bit, write “0” (low level). When reading, its content is “0”.

Figure 4.26 Port P4 register

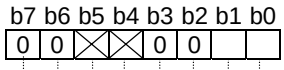
b7	b6	b5	b4	b3	b2	b1	b0
0	0	0	0	0	0	0	0

Port P4 direction register (00EA$_{16}$ address)
PD4

Bit Symbol	Bit Name	Function	RW
PD4_0	Port P4$_0$ direction bit (Note1)	0 : Input mode (Functions as an input port) 1 : Output mode (Functions as an output port)	RW
PD4_1	Port P4$_1$ direction bit (Note1)		RW
PD4_2	Port P4$_2$ direction bit (Note1)		RW
PD4_3	Port P4$_3$ direction bit (Note1)		RW
PD4_4	Port P4$_4$ direction bit (Note1)		RW
PD4_5	Port P4$_5$ direction bit		RW
PD4_6	Port P4$_6$ direction bit (Note1)		RW
PD4_7	Port P4$_7$ direction bit (Note1)		RW

Note1 Nothing is assigned. When writing to this bit, write “0” (input mode). When reading, its content is “0”.

Figure 4.27 Port P4 direction register

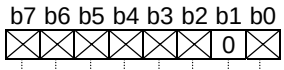


Pull-up control register 0 (00FC16 address)
PUR0

Bit Symbol	Bit Name	Function	RW
PU00	P00 to P03 pull-up (Note1)	0 : Not pulled high 1 : Pulled high	RW
PU01	P04 to P07 pull-up (Note1)		RW
PU02	P10 to P13 pull-up (Note1)		RW
PU03	P14 to P17 pull-up (Note1)		RW
(b5-b4)	Nothing is assigned. When write, set to “0”. When read, its content is indeterminate.		-
PU06	P30 to P33 pull-up (Note1)	0 : Not pulled high 1 : Pulled high	RW
PU07	P37 pull-up (Note1)		RW

Note1 The pin for which this bit is “1”(pulled high) and the direction bit is “0” (input mode) is pulled high.

Figure 4.28 Pull-up control register 0

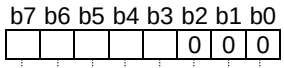


Pull-up control register 1 (00FD16 address)
PUR1

Bit Symbol	Bit Name	Function	RW
(b0)	Nothing is assigned. When write, set to “0”. When read, its content is indeterminate.		-
PU11	P45 pull-up (Note1)	0 : Not pulled high 1 : Pulled high	RW
(b7-b2)	Nothing is assigned. When write, set to “0”. When read, its content is indeterminate.		-

Note1 The P45 pin for which this bit is “1” (pulled high) and the PD4_5 bit is “0” (input mode) is pulled high.

Figure 4.29 Pull-up control register 1



Port P1 drive capacity control register (00FE16 address)
DRR

Bit Symbol	Bit Name	Function	RW
DRR0	P10 drive capacity	Set P1 N-channel output transistor drive capacity 0 : Low 1 : High	RW
DRR1	P11 drive capacity		RW
DRR2	P12 drive capacity		RW
DRR3	P13 drive capacity		RW
DRR4	P14 drive capacity		RW
DRR5	P15 drive capacity		RW
DRR6	P16 drive capacity		RW
DRR7	P17 drive capacity		RW

Figure 4.30 Port P1 drive capacity control register

4.3 Description of Data Table

In this sample task, six patterns data on the table are output to P1 and P3 at motor control. Table 4.2 shows the description of data table at initial motor control, and Table 4.3 shows the description of data table at motor control based on the rotor-positional signal.

Table 4.2 Description of Data Table (_Init_p1Active_Phase_TBL/_Init_p3Active_Phase_TBL)

Table Name	Data	Output Pattern	Size	Remarks
_Init_p1Active_Phase_TBL[0]	H'01	U=ON, V=OFF, W=OFF	1Byte	0°
_Init_p3Active_Phase_TBL[0]	H'05	_U=OFF, _V=ON, _W=OFF	1Byte	
_Init_p1Active_Phase_TBL[1]	H'01	U=ON, V=OFF, W=OFF	1Byte	60°
_Init_p3Active_Phase_TBL[1]	H'03	_U=OFF,_V=OFF, _W=ON	1Byte	
_Init_p1Active_Phase_TBL[2]	H'02	U=OFF, V=ON, W=OFF	1Byte	120°
_Init_p3Active_Phase_TBL[2]	H'03	_U=OFF, _V=OFF, _W=ON	1Byte	
_Init_p1Active_Phase_TBL[3]	H'02	U=OFF, V=ON, W=OFF	1Byte	180°
_Init_p3Active_Phase_TBL[3]	H'06	_U=ON, _V=OFF, _W=OFF	1Byte	
_Init_p1Active_Phase_TBL[4]	H'04	U=OFF, V=OFF, W=ON	1Byte	240°
_Init_p3Active_Phase_TBL[4]	H'06	_U=ON, _V=OFF, _W=OFF	1Byte	
_Init_p1Active_Phase_TBL[5]	H'04	U=OFF, V=OFF, W=ON	1Byte	300°
_Init_p3Active_Phase_TBL[5]	H'05	_U=OFF, _V=ON, _W=OFF	1Byte	

Note1 "1" is on, and "0" is off for positive phase. "0" is on , and "1" is off for negative phase.

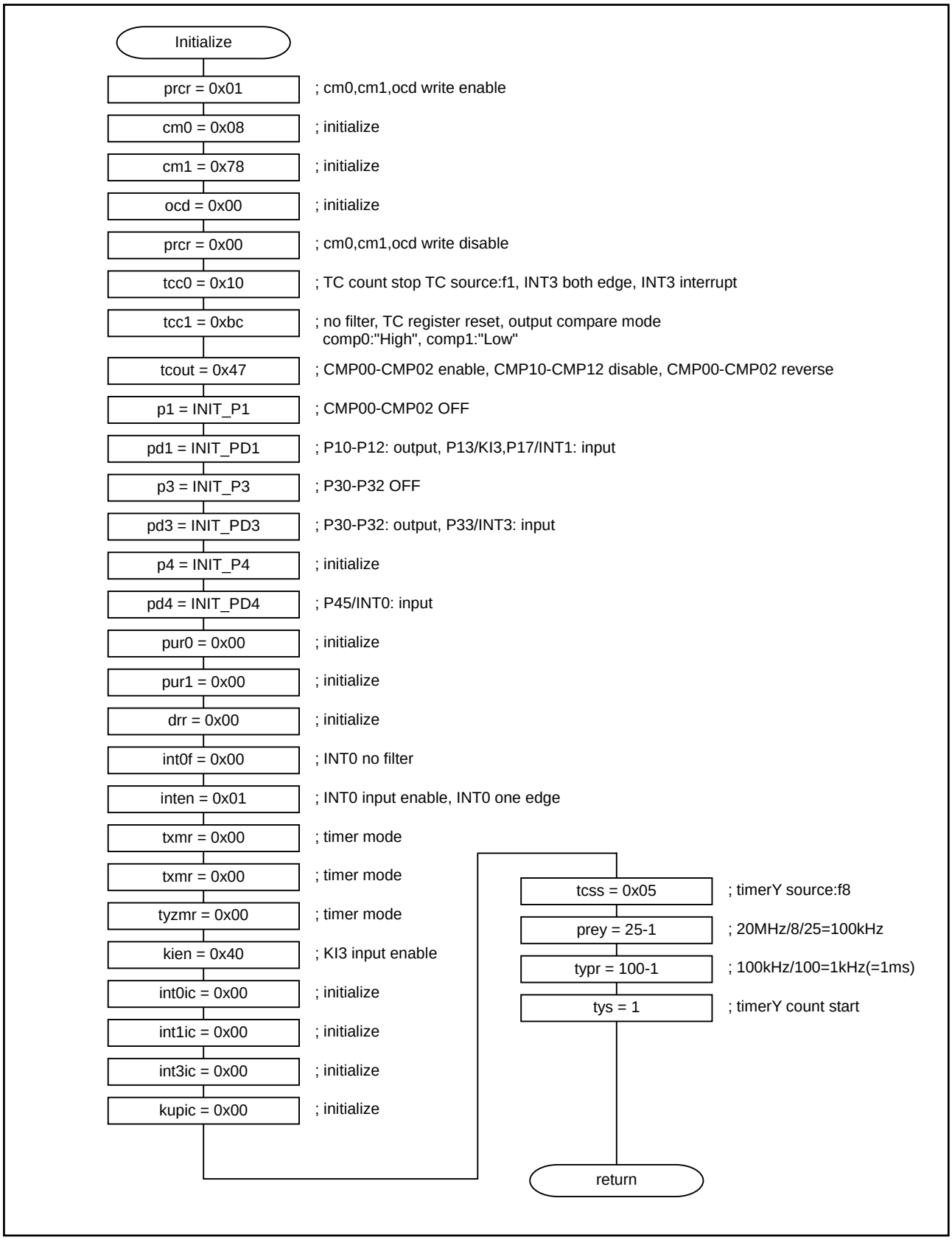
Table 4.3 Description of Data Table (_p1Active_Phase_TBL/_p3Active_Phase_TBL)

Table Name	Data	Output Pattern	Size	Remarks
_p1Active_Phase_TBL[0]	H'01	U=ON, V=OFF, W=OFF	1Byte	0°
_p3Active_Phase_TBL[0]	H'03	_U=OFF, _V=OFF, _W=ON	1Byte	
_p1Active_Phase_TBL[1]	H'02	U=OFF, V=ON, W=OFF	1Byte	60°
_p3Active_Phase_TBL[1]	H'03	_U=OFF, _V=OFF, _W=ON	1Byte	
_p1Active_Phase_TBL[2]	H'02	U=OFF, V=ON, W=OFF	1Byte	120°
_p3Active_Phase_TBL[2]	H'06	_U=ON, _V=OFF, _W=OFF	1Byte	
_p1Active_Phase_TBL[3]	H'04	U=OFF, V=OFF, W=ON	1Byte	180°
_p3Active_Phase_TBL[3]	H'06	_U=ON, _V=OFF, _W=OFF	1Byte	
_p1Active_Phase_TBL[4]	H'04	U=OFF, V=OFF, W=ON	1Byte	240°
_p3Active_Phase_TBL[4]	H'05	_U=OFF, _V=ON, _W=OFF	1Byte	
_p1Active_Phase_TBL[5]	H'01	U=ON, V=OFF, W=OFF	1Byte	300°
_p3Active_Phase_TBL[5]	H'05	_U=OFF, _V=ON, _W=OFF	1Byte	

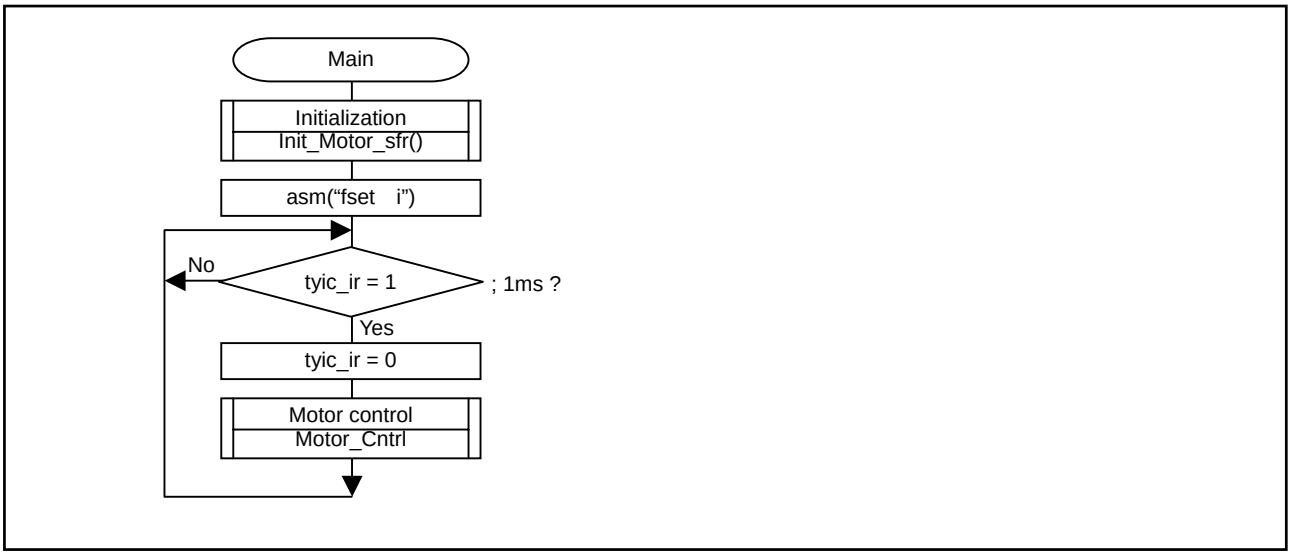
Note1 "1" is on, and "0" is off for positive phase. "0" is on , and "1" is off for negative phase.

4.4 Flowchart

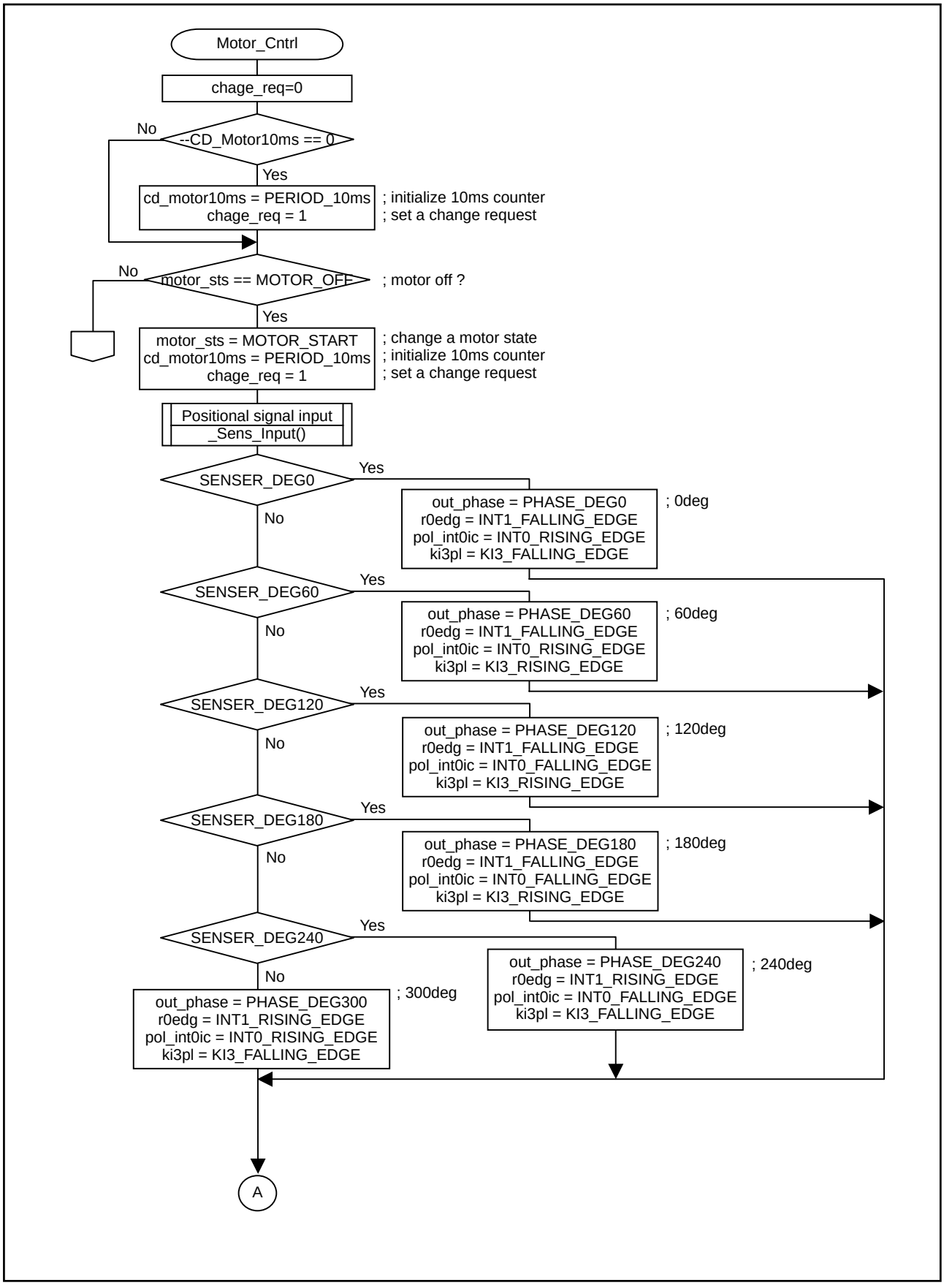
1.Register initialization routine



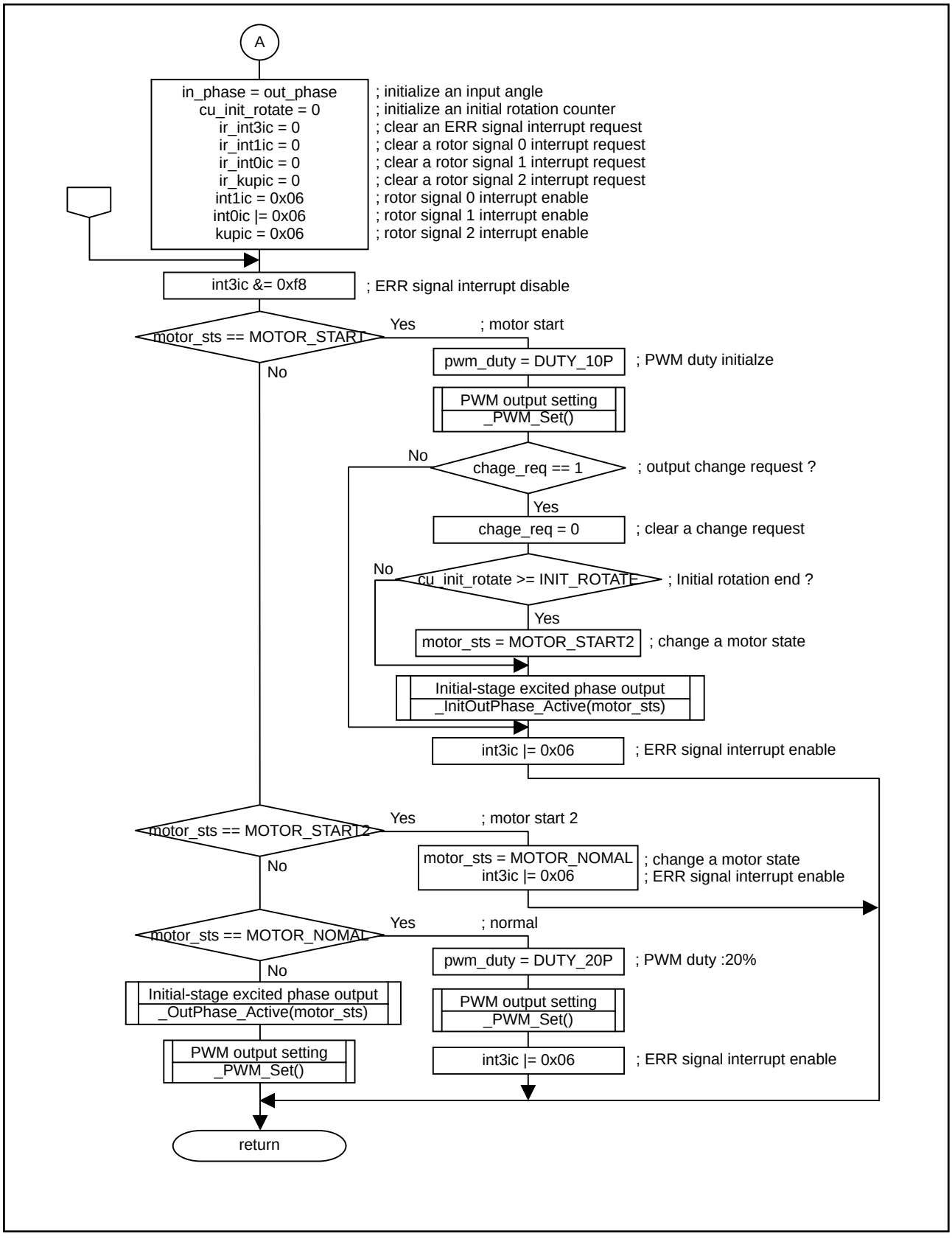
2.Main routine



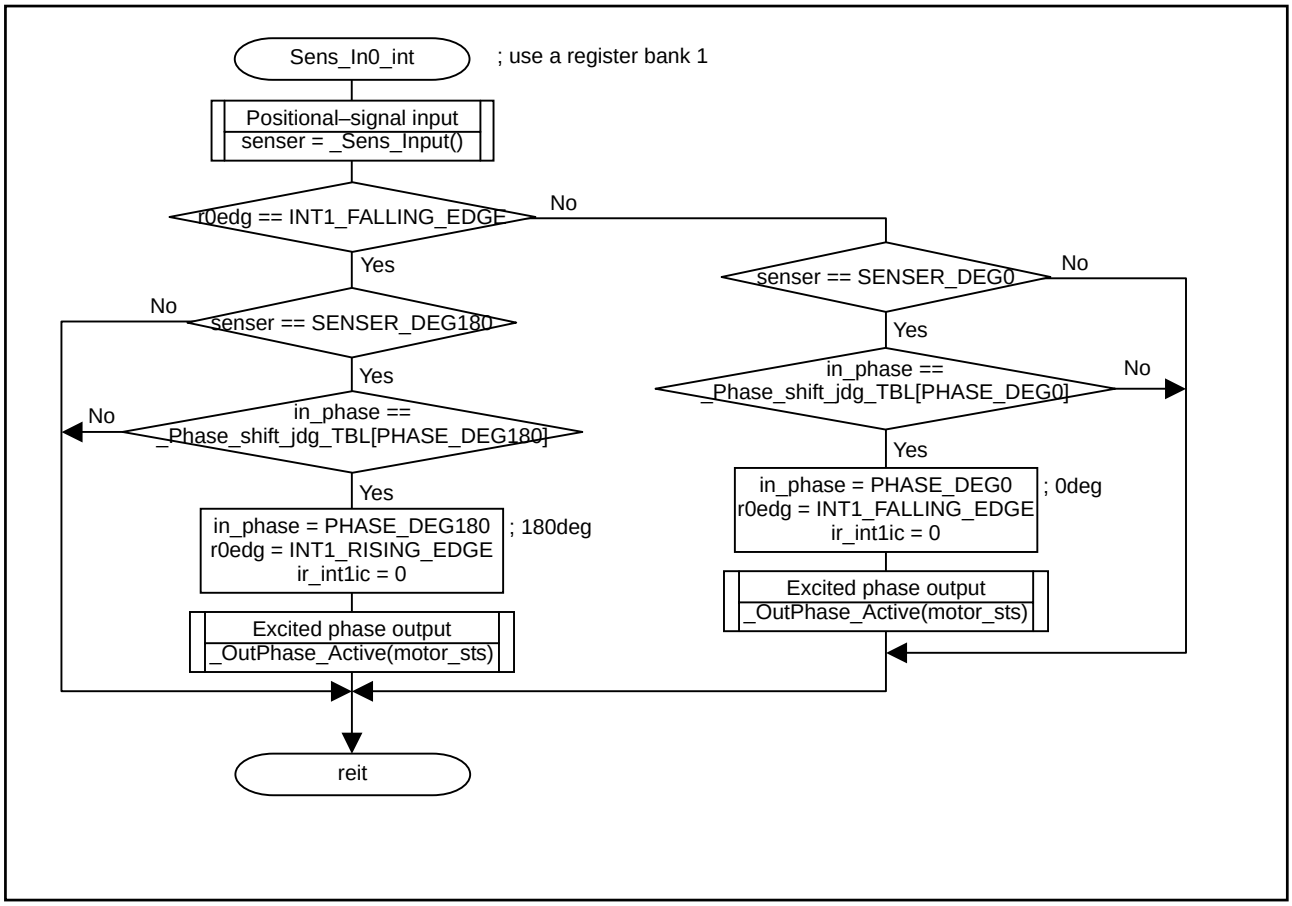
3.Motor control routine



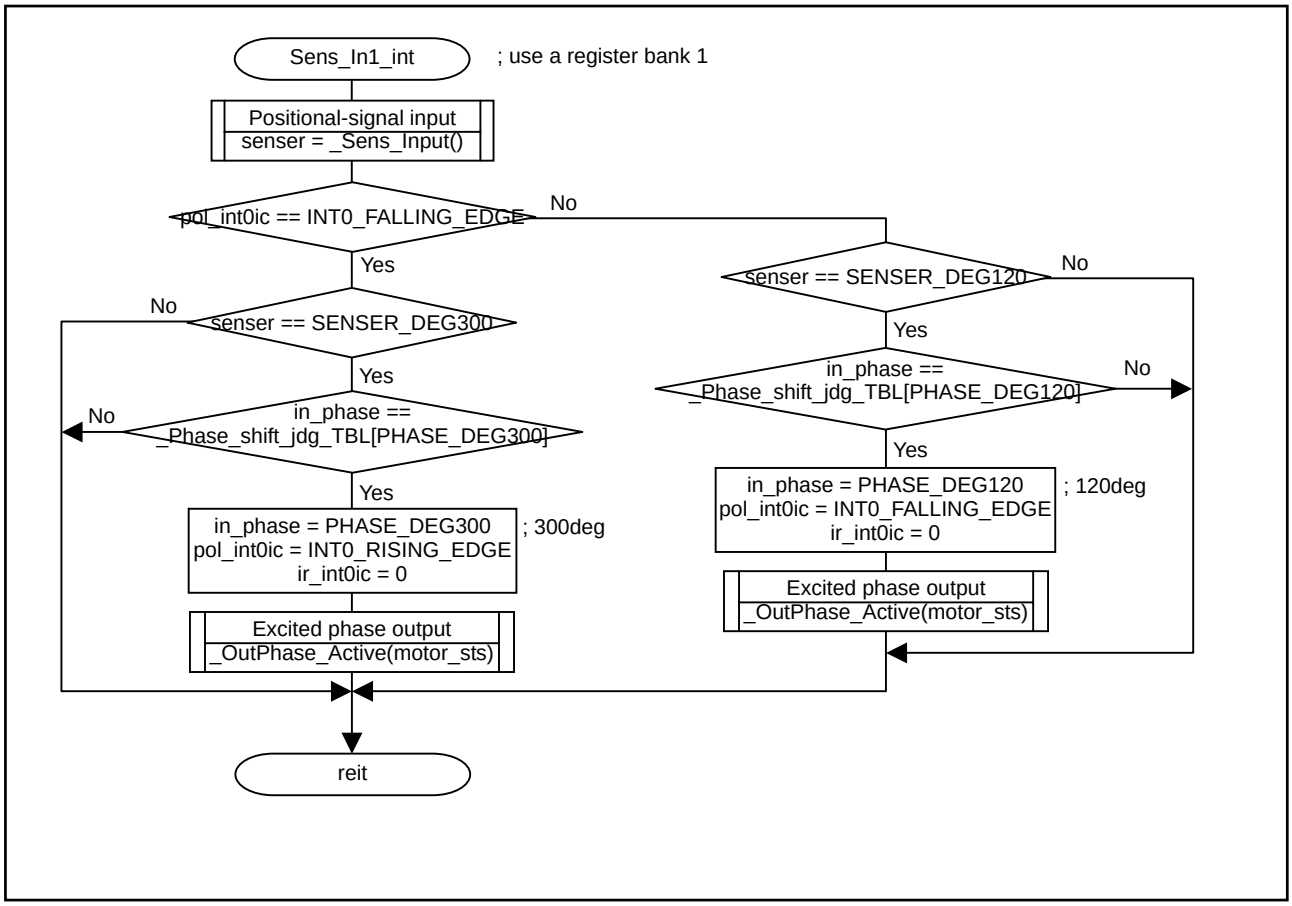
Motor control routine (continued)



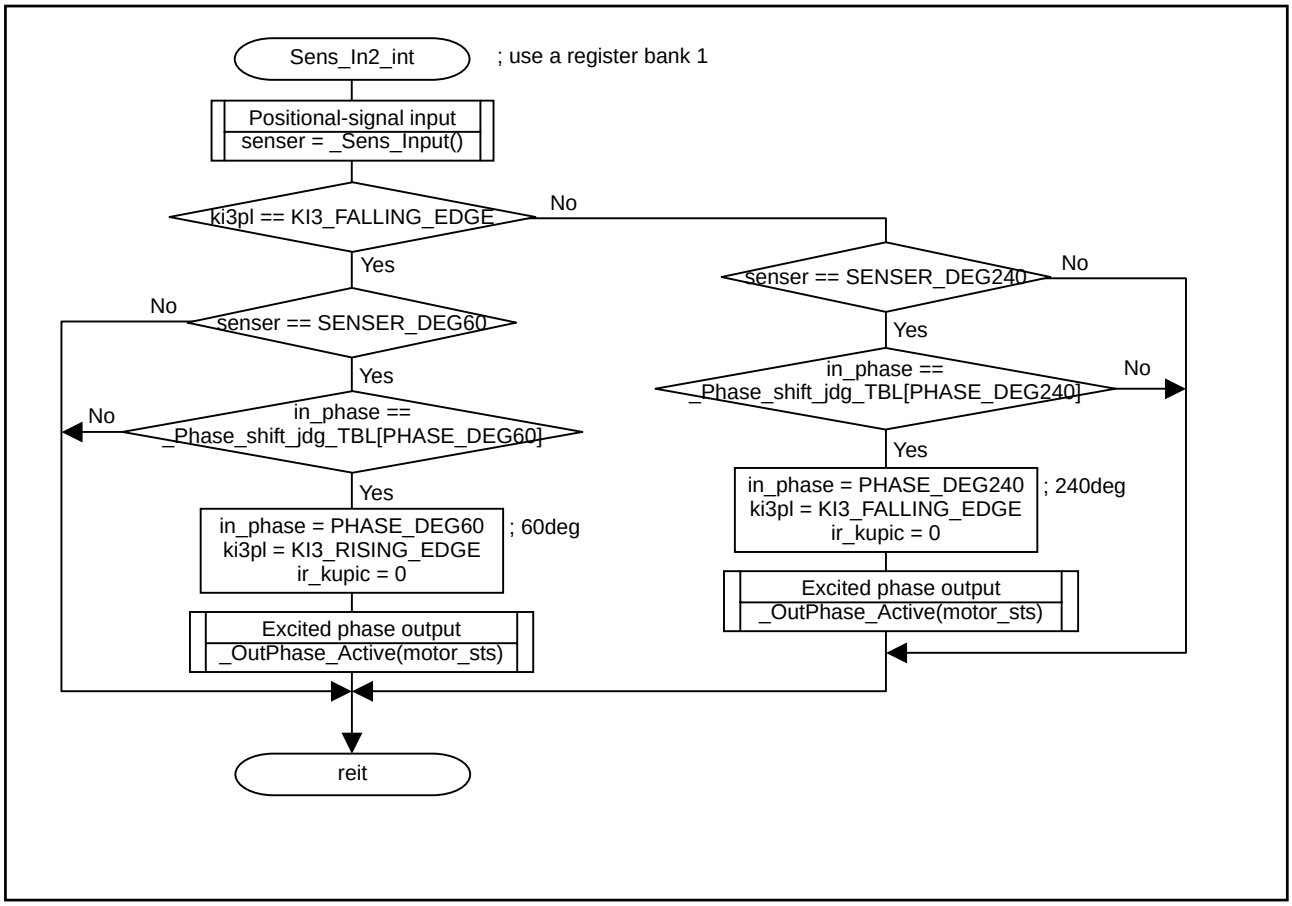
4.Rotor-positional signal (INT1) interrupt routine



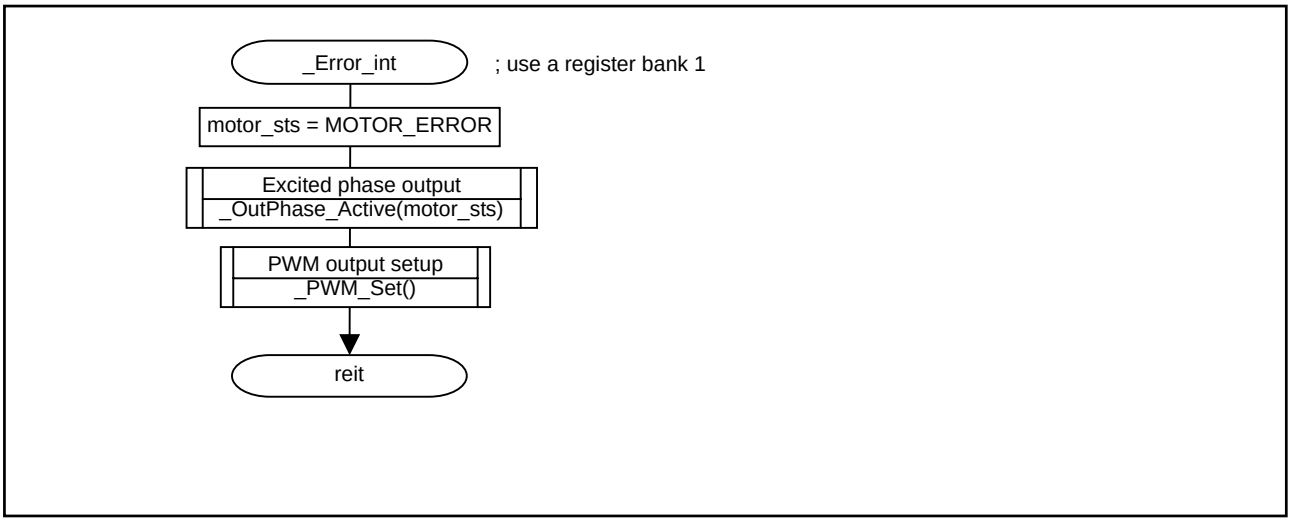
5.Rotor positional-signal (INT0) interrupt routine



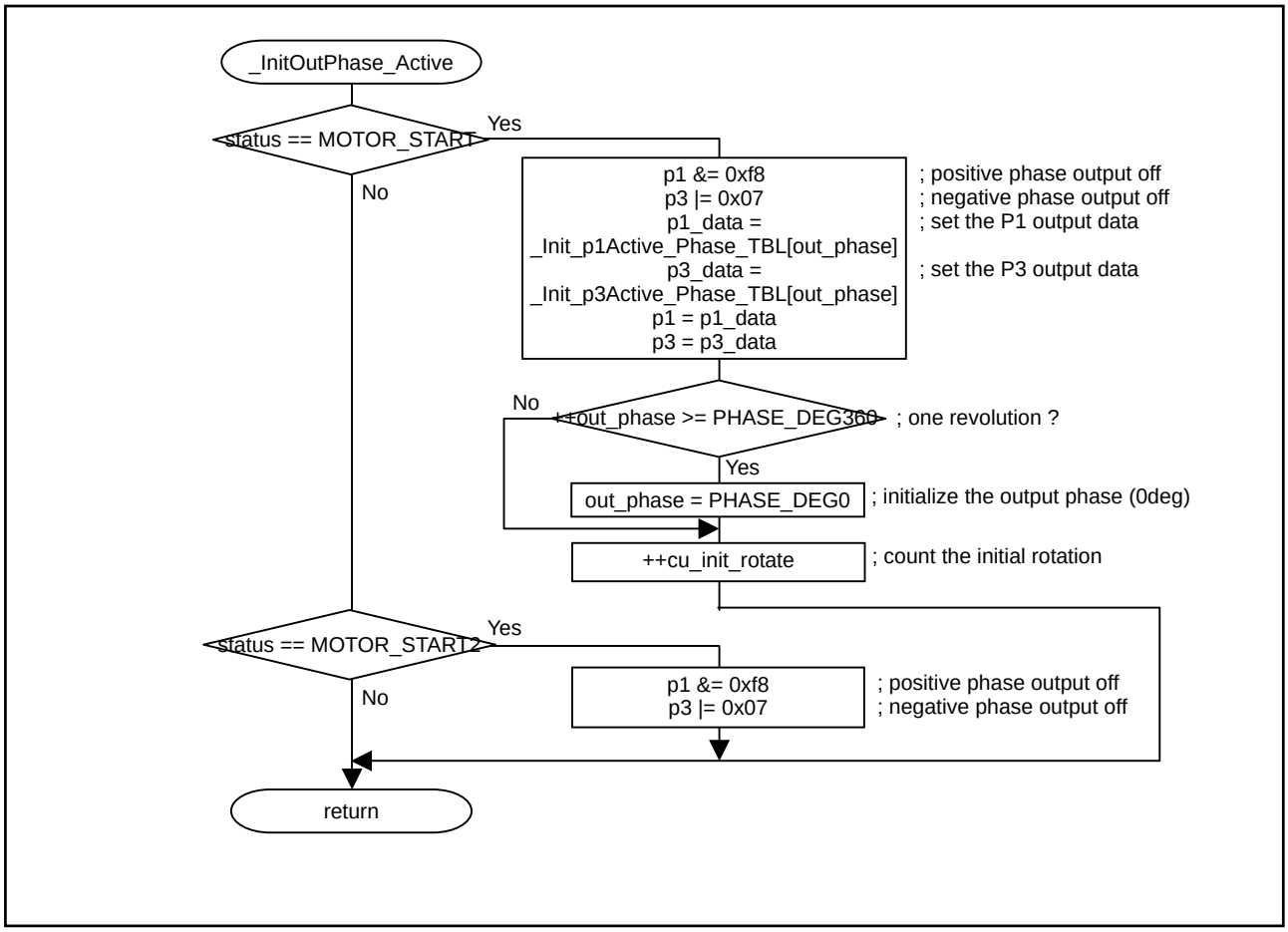
6.Rotor-positional signal (KI3) interrupt routine



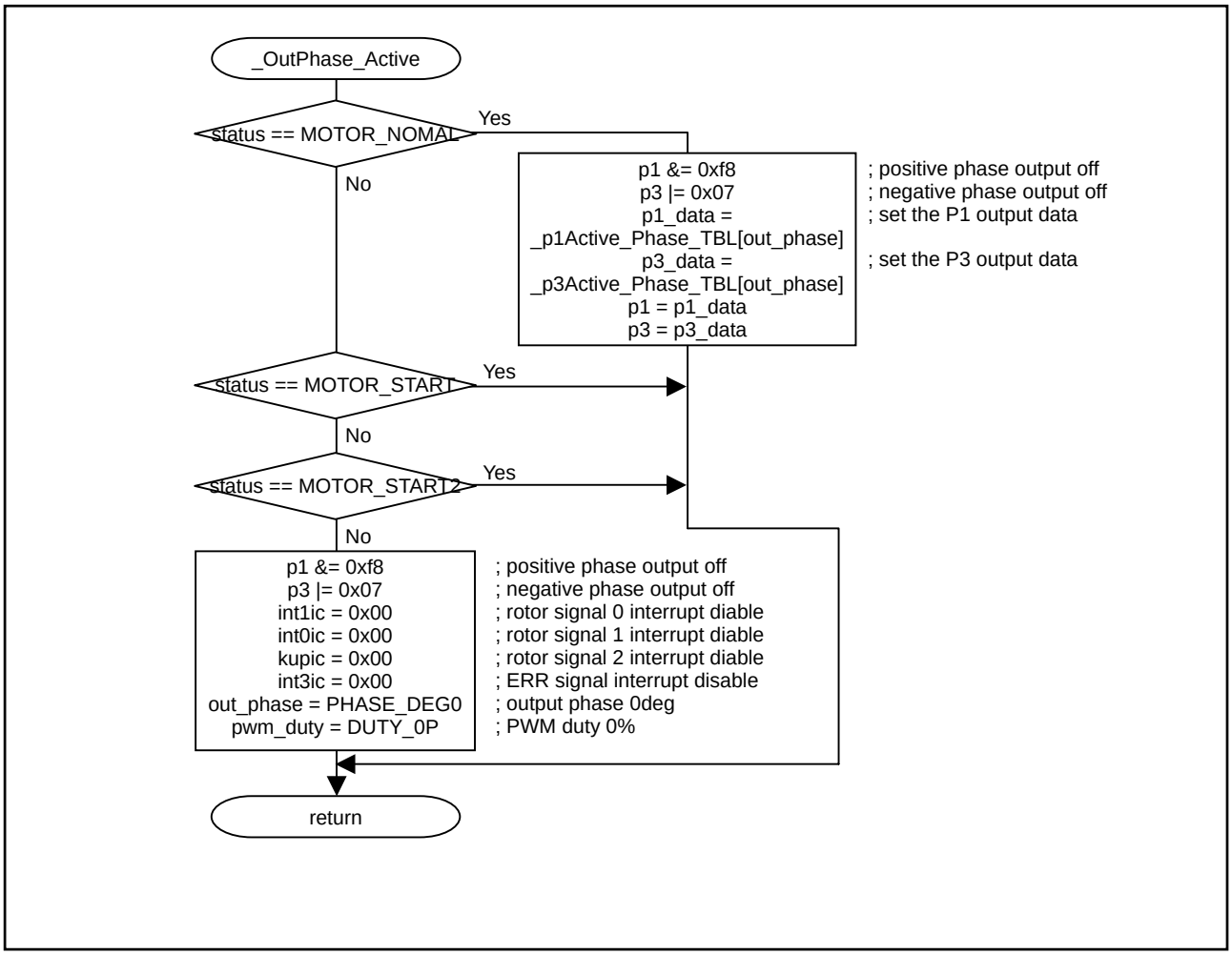
7.Motor-rotation stop signal (INT3) interrupt routine



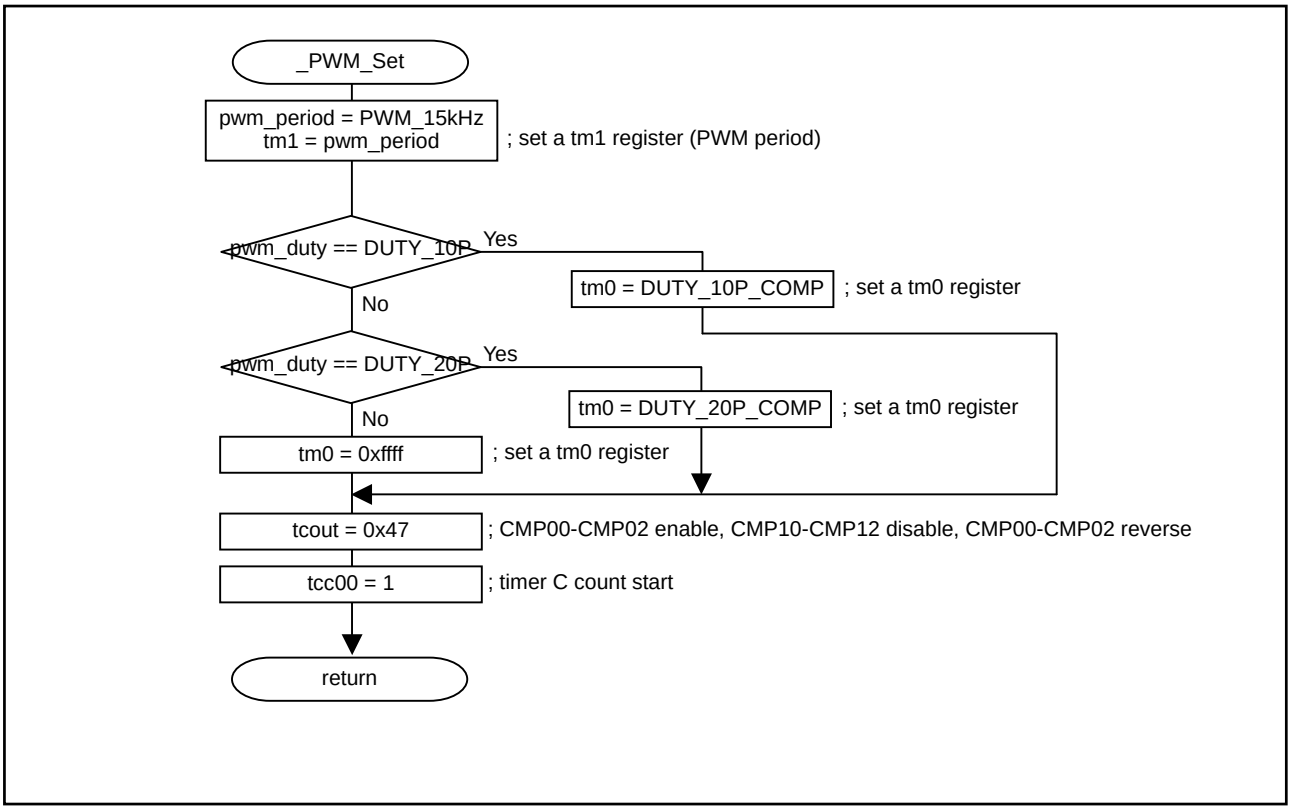
8.Initial-stage excited phase output routine



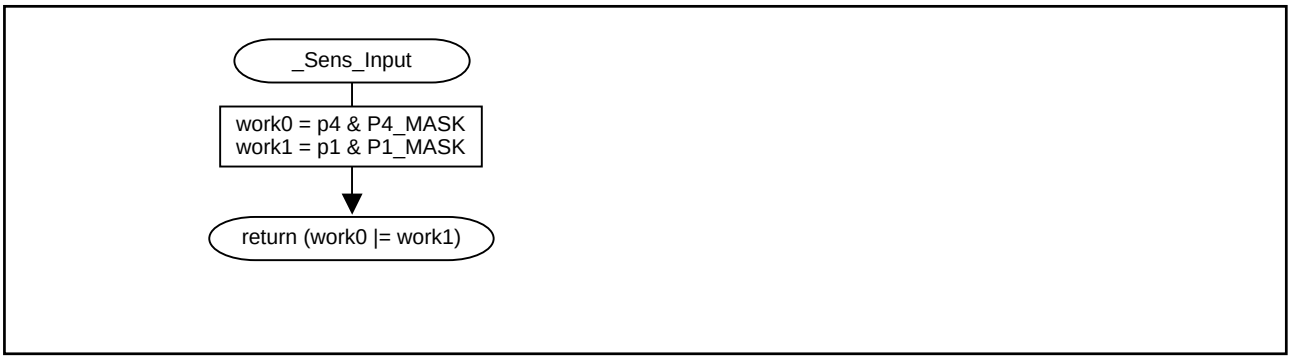
9.Excited phase output routine



10. PWM output setup output routine



11.Positional signal input routine



5. Program Listing

```
/*""FILE COMMENT""*****************************************************************
 *    System Name     : DC BrashLess Motor Control
 *    File Name       : main.c
 *    Version         : Ver 1.00
 *    Contents        : mein processing
 *    CPU             : R5F21114FP
 *    Compiler        : NC30WA V.5.20 Release 1
 *    OS              :
 *    Programmer      :
 ***********************************************************************
 * Copyright,2003 RENESAS TECHNOLOGY CORPORATION
 * AND RENESAS SOLUTIONS CORPORATION
 * ALL RIGHTS RESERVED
 ***********************************************************************
 *     History         :
 *""FILE COMMENT END""*************************************************/

#include   "sfr_r811.h"
#include   "motorlib.h"

/*""FUNC COMMENT""*****************************************************************
 *    Name            : main()
 *----------------------------------------------------------------------
 *    Parameters      : None
 *    Returns         : None
 *    Description     : main routine
 *    SubRoutine      : Init_Motor_sfr()
 *                    : Motor_Cntrl()
 *----------------------------------------------------------------------
 *    Note            :
 *----------------------------------------------------------------------
 *    History         :
 *""FUNC COMMENT END""*************************************************/

void main()
{
      Init_Motor_sfr();
      asm("fset          i");
      while (1) {
            while (!ir_tyic);                   /* 1ms ? */
            ir_tyic = 0;                        /* Yes clear timerY interrupt req. flag */
            Motor_Cntrl();                              /* motor control */
      }
}
```

```
/*""FILE COMMENT""*****************************************************************
 *   System Name     : DC BrashLess Motor Control
 *   File Name       : motorlib.h
 *   Version         : Ver 1.00
 *   Contents        : motor control processing header
 *   CPU             : R5F21114FP
 *   Compiler        : NC30WA V.5.20 Release 1
 *   OS              :
 *   Programmer      :
 ***********************************************************************
 * Copyright,2003 RENESAS TECHNOLOGY CORPORATION
 * AND RENESAS SOLUTIONS CORPORATION
 * ALL RIGHTS RESERVED
 ***********************************************************************
 *    History        :
 *""FILE COMMENT END""*************************************************************/

#define DEBUG

typedef enum {
      MOTOR_OFF,                        /* motor stop       */
      MOTOR_START,                      /* motor start      */
      MOTOR_START2,                     /* motor start 2 */
      MOTOR_NOMAL,                      /* motor normal    */
      MOTOR_ERROR                               /* motor error      */
} MOTOR_STATUS;

typedef enum {
      PHASE_DEG0,                       /*     0deg    */
      PHASE_DEG60,                      /*    60deg    */
      PHASE_DEG120,                     /* 120deg    */
      PHASE_DEG180,                     /* 180deg    */
      PHASE_DEG240,                     /* 240deg    */
      PHASE_DEG300,                     /* 300deg    */
      PHASE_DEG360                      /* 360deg    */
} ROTOR_POSITION;

#define DUTY_0P             0                   /*     0%    */
#define DUTY_1P             1                   /*     1%    */
#define DUTY_10P   10               /*    10%    */
#define DUTY_20P   20               /*    20%    */
#define DUTY_100P 100               /* 100%    */

#define PERIOD_10ms         10                  /* 10ms    */

#define INIT_ROTATE         6                   /* one revolution of electrical degree    */

#define PWM_15kHz           1333-1              /* 15kHz    */
#define DUTY_10P_COMP       1200-1              /*    10% (at 15kHz)    */
#define DUTY_20P_COMP       1066-1              /*    20% (at 15kHz)    */

#define SENSER_MASK         0xa8                /* mask data
                                                       (bit7=P17,bit5=P45,bit3=P33)=10101000b    */
#define SENSER0_IN          0x80                /* rotor signal 0 (P17/INT1)    */
#define SENSER1_IN          0x20                /* rotor signal 1 (P45/INT0)    */
#define SENSER2_IN          0x08                /* rotor signal 2 (P13/KI3)    */
#define P1_MASK             0x88                /* mask data (P17/INT1,P13/KI3)    */
#define P4_MASK             0x20                /* mask data(P45/INT0)    */
#define SENSER_DEG0         0x88                /*     0deg    */
#define SENSER_DEG60        0x80                /*    60deg    */
#define SENSER_DEG120       0xa0                /* 120deg    */
#define SENSER_DEG180       0x20                /* 180deg    */
#define SENSER_DEG240       0x28                /* 240deg    */
#define SENSER_DEG300       0x08                /* 300deg    */

#define INIT_P1             0x00                /* P10/CMP00 to P12/CMP02: OFF("Low")     */
#define INIT_PD1    0x07            /* P10/CMP00 to P12/CMP02: output mode,
```

```
                                                P13/KI3,P17/INT1: input mode    */
#define INIT_P3                 0x07                /* P30 to P32: OFF("High")    */
#define INIT_PD3    0x07                    /* P30 to P32: output mode, P33/INT3: input mode    */
#define INIT_P4                 0x00                /* initialize    */
#define INIT_PD4    0x00                    /* P45/INT0: input mode    */

#define INT0_FALLING_EDGE 0             /* INT0 Falling edge    */
#define INT0_RISING_EDGE   1            /* INT0 Rising edge    */
#define INT1_RISING_EDGE   0            /* INT1 Rising edge    */
#define INT1_FALLING_EDGE 1             /* INT1 Falling edge    */
#define KI3_FALLING_EDGE   0            /* KI3 Falling edge    */
#define KI3_RISING_EDGE                 1           /* KI3 Rising edge    */

typedef unsigned char           u08;
typedef unsigned short          u16;

void Init_Motor_sfr ( void );
void Motor_Cntrl( void );
void Sens_In0_int ( void );
void Sens_In1_int ( void );
void Sens_In2_int ( void );
void Error_int ( void );
```

```
/*""FILE COMMENT""*****************************************************************
 *   System Name     : DC BrashLess Motor Control
 *   File Name       : motorlib.c
 *   Version         : Ver 1.00
 *   Contents        : motor control processing
 *   CPU             : R5F21114FP
 *   Compiler        : NC30WA V.5.20 Release 1
 *   OS              :
 *   Programmer      :
 ***********************************************************************************
 * Copyright,2003 RENESAS TECHNOLOGY CORPORATION
 * AND RENESAS SOLUTIONS CORPORATION
 * ALL RIGHTS RESERVED
 ***********************************************************************************
 *   History         :
 *""FILE COMMENT END""*************************************************************/

#include   "sfr_r811.h"
#include   "motorlib.h"

static u08 motor_sts=MOTOR_OFF;                 /* motor state    */
static u08 out_phase;                           /* output phase    */
static u08 in_phase;                        /* input angle    */
static u08 cu_init_rotate;                  /* initial rotation counter    */
static u16 pwm_period;                          /* PWM period    */
static u08 pwm_duty;                            /* PWM duty    */

static void _InitOutPhase_Active ( MOTOR_STATUS );
static void _OutPhase_Active ( MOTOR_STATUS );
static void _PWM_Set ( void );
static u08 _Sens_Input ( void );
static const u08 _Init_p1Active_Phase_TBL[];
static const u08 _Init_p3Active_Phase_TBL[];
static const u08 _p1Active_Phase_TBL[];
static const u08 _p3Active_Phase_TBL[];
static const u08 _Phase_shift_jdg_TBL[];

/*""FUNC COMMENT""*****************************************************************
 *   Name            : Init_Motor_sfr()
 *---------------------------------------------------------------------------------
 *   Parameters      : None
 *   Returns         : None
 *   Description     : initialize SFRs for Motor control
 *   SubRoutine      : None
 *---------------------------------------------------------------------------------
 *   Note            :
 *---------------------------------------------------------------------------------
 *   History         :
 *""FUNC COMMENT END""*************************************************************/
void Init_Motor_sfr ( void )
{
   prcr = 0x01;                     /* cm0,cm1,ocd write enable */
   cm0 = 0x08;                      /* initialize    */
   cm1 = 0x78;                      /* initialize    */
   ocd = 0x00;                      /* initialize    */
   prcr = 0x00;                     /* cm0,cm1,ocd write disable */

   tcc0 = 0x10;                     /* TC count stop TC source:f1, INT3 both edge, INT3 interrupt    */
   tcc1 = 0xbc;                     /* no filter, TC register reset, output compare mode    */
                                    /* comp0:"High", comp1:"Low"    */
   tcout = 0x47;                    /* CMP00-CMP02 enable, CMP10-CMP12 disable    */
                                    /* CMP00-CMP02 reverse    */
   p1 = INIT_P1;                    /* CMP00-CMP02 OFF    */
   pd1 = INIT_PD1;   /* P10-P12: output, P13/KI3,P17/INT1: input    */
   p3 = INIT_P3;                    /* P30-P32 OFF    */
   pd3 = INIT_PD3;   /* P30-P32: output, P33/INT3: input    */
   p4 = INIT_P4;                    /* initialize    */
```

```
    pd4 = INIT_PD4;   /* P45/INT0: input   */
    pur0 = 0x00;                    /* initialize   */
    pur1 = 0x00;                    /* initialize   */
    drr = 0x00;                     /* initialize   */

    int0f = 0x00;                   /* INT0 no filter   */
    inten = 0x01;                   /* INT0 input enable, INT0 one edge   */
    txmr = 0x00;                    /* timer mode */
    tyzmr = 0x00;                   /* timer mode */
    kien = 0x40;                    /* KI3 input enable   */
    int0ic = 0x00;          /* initialize   */
    int1ic = 0x00;          /* initialize   */
    int3ic = 0x00;          /* initialize   */
    kupic = 0x00;                   /* initialize   */

    tcss = 0x05;                    /* timerY source:f8   */
    prey = 25-1;                    /* 20MHz/8/25=100kHz */
    typr = 100-1;                   /* 100kHz/100=1kHz(=1ms) */
    tys = 1;                /* timerY count start   */

}


/*""FUNC COMMENT""*****************************************************************
 *   Name               : Motor_Cntrl()
 *---------------------------------------------------------------------------
 *   Parameters         : None
 *   Returns            : None
 *   Description        : Motor control processing routine
 *   SubRoutine         : _InitOutPhase_Active()
 *                      : _OutPhase_Active()
 *                      : _PWM_Set()
 *                      : _Sens_Input()
 *---------------------------------------------------------------------------
 *   Note               : Please perform this function in a cycle of 1ms.
 *---------------------------------------------------------------------------
 *   History            :
 *""FUNC COMMENT END""*************************************************************/
void Motor_Cntrl ( void )
{
static u08 cd_motor10ms;                /* 10ms counter for motor control   */
          u08 chage_req=0;                      /* output change request flag   */

   if ( --cd_motor10ms == 0 ) {         /* 10ms count   */
      cd_motor10ms = PERIOD_10ms;               /*   Yes initialize 10ms counter   */
      chage_req = 1;                    /*   set a change request */
   }

   if ( motor_sts == MOTOR_OFF ) {      /* motor off ? { */
      motor_sts = MOTOR_START;                  /*   Yes change a motor state   */
      cd_motor10ms = PERIOD_10ms;               /*   initialize 10ms counter   */
      chage_req = 1;                    /*   set a change request   */
      switch (_Sens_Input()) {          /*   rotor position signal input */
      case SENSER_DEG0:                         /*   case SENSER_DEG0:   */
         out_phase = PHASE_DEG0;                /*    start an output phase from 0deg   */
                                                /*    Position detection start from 60deg   */
         r0edg = INT1_FALLING_EDGE;     /*    INT1 edge set   */
         pol_int0ic = INT0_RISING_EDGE; /*    INT0 edge set   */
         ki3pl = KI3_FALLING_EDGE;              /*    KI3 edge set   */
         break;
      case SENSER_DEG60:                        /*   case SENSER_DEG60:   */
         out_phase = PHASE_DEG60;               /*    start an output phase from 60deg   */
                                                /*    Position detection start from 120deg   */
         r0edg = INT1_FALLING_EDGE;     /*    INT1 edge set   */
         pol_int0ic = INT0_RISING_EDGE; /*    INT0 edge set   */
         ki3pl = KI3_RISING_EDGE;               /*    KI3 edge set   */
         break;
```

```
    case SENSER_DEG120:                     /*  case SENSER_DEG120:  */
      out_phase = PHASE_DEG120;             /*    start an output phase from 120deg  */
                                            /*    Position detection start from 180deg  */
      r0edg = INT1_FALLING_EDGE;    /*    INT1 edge set  */
      pol_int0ic = INT0_FALLING_EDGE;       /*    INT0 edge set  */
      ki3pl = KI3_RISING_EDGE;              /*    KI3 edge set  */
      break;
    case SENSER_DEG180:                     /*  case SENSER_DEG180:  */
      out_phase = PHASE_DEG180;             /*    start an output phase from 180deg  */
                                            /*    Position detection start from 240deg  */
      r0edg = INT1_FALLING_EDGE;    /*    INT1 edge set  */
      pol_int0ic = INT0_FALLING_EDGE;       /*    INT0 edge set  */
      ki3pl = KI3_RISING_EDGE;              /*    KI3 edge set  */
      break;
    case SENSER_DEG240:                     /*  case SENSER_DEG240:  */
      out_phase = PHASE_DEG240;             /*    start an output phase from 240deg  */
                                            /*    Position detection start from 300deg  */
      r0edg = INT1_RISING_EDGE;             /*    INT1 edge set  */
      pol_int0ic = INT0_FALLING_EDGE;       /*    INT0 edge set  */
      ki3pl = KI3_FALLING_EDGE;             /*    KI3 edge set  */
      break;
    default:
      out_phase = PHASE_DEG300;             /*    start an output phase from 300deg  */
                                            /*    Position detection start from 0deg  */
      r0edg = INT1_RISING_EDGE;             /*    INT1 edge set  */
      pol_int0ic = INT0_RISING_EDGE; /*    INT0 edge set  */
      ki3pl = KI3_FALLING_EDGE;             /*    KI3 edge set  */
    }
    in_phase = out_phase;               /*  initialize an input angle  */
    cu_init_rotate = 0;                         /*  initialize an initial rotation counter  */
    ir_int3ic = 0;                      /*  clear an ERR signal interrupt request  */
    ir_int1ic = 0;                      /*  clear a rotor signal 0 interrupt request  */
    ir_int0ic = 0;                      /*  clear a rotor signal 1 interrupt request  */
    ir_kupic = 0;                       /*  clear a rotor signal 2 interrupt request  */
    int1ic = 0x06;                      /*  rotor signal 0 interrupt enable  */
    int0ic |= 0x06;                     /*  rotor signal 1 interrupt enable  */
    kupic = 0x06;                       /*  rotor signal 2 interrupt enable  */
  }

  int3ic &= 0xf8;                       /* ERR signal interrupt disable  */
  switch (motor_sts) {
  case MOTOR_START:                             /* case motor start  */
    pwm_duty = DUTY_10P;                /*  PWM duty initialze  */
    _PWM_Set();                                 /*  PWM output  */
    if ( chage_req == 1 ) {             /*  output change request ?  */
      chage_req = 0;                            /*    Yes clear a change request  */
      if ( cu_init_rotate >= INIT_ROTATE )      /*    Initial rotation end ?  */
        motor_sts = MOTOR_START2;   /*     Yes change a motor state  */
      _InitOutPhase_Active(motor_sts);  /*    output at the start time  */
    }
    int3ic |= 0x06;                     /*  ERR signal interrupt enable  */
    break;
  case MOTOR_START2:                            /* case motor start 2  */
    motor_sts = MOTOR_NOMAL;                    /*  change a motor state  */
    int3ic |= 0x06;                     /*  ERR signal interrupt enable  */
    break;
  case MOTOR_NOMAL:                             /* case normal  */
    pwm_duty = DUTY_20P;                /*  PWM duty :20%  */
    _PWM_Set();                                 /*  PWM output  */
    int3ic |= 0x06;                     /*  ERR signal interrupt enable  */
    break;
  default:
    _OutPhase_Active(motor_sts);        /*  output  */
    _PWM_Set();                                 /*  PWM output  */
  }
}
```

```
/*""FUNC COMMENT""*****************************************************************
 *   Name               : Sens_In0_int()
 *-----------------------------------------------------------------------------
 *   Parameters      : None
 *   Returns          : None
 *   Description    : interrupt of Rotor position signal 0 occurs
 *   SubRoutine      : _OutPhase_Active()
 *                    : _Sens_Input()
 *-----------------------------------------------------------------------------
 *   Note               :
 *-----------------------------------------------------------------------------
 *   History          : 2003-10-08: processing time
 *                    :   (except the interrupt sequence, the bank change and return)
 *                    :                 180deg: about 18.9us / 0deg: about 18.2us
 *""FUNC COMMENT END""*************************************************************/
#pragma INTERRUPT/B Sens_In0_int
void Sens_In0_int ( void )
{
   u08      senser;

   senser = _Sens_Input();                  /* rotor position signal input   */
   if ( r0edg == INT1_FALLING_EDGE ) {  /* INT1 falling endge ?   */
      if ( senser == SENSER_DEG180 ) {   /*   Yes rotor position = 180deg ?   */
         if ( in_phase == _Phase_shift_jdg_TBL[PHASE_DEG180] ) {
                                                  /*     Yes the previous position is right ?   */
            in_phase = PHASE_DEG180;     /*       Yes updating of an input angle (180deg)   */
            r0edg = INT1_RISING_EDGE;    /*       set the INT1 rising edge   */
            ir_int1ic = 0;                        /*       clear the interrupt request   */
            _OutPhase_Active(motor_sts);    /*       output   */
         }
      }
   }else{
      if ( senser == SENSER_DEG0 ) {      /*   No   rotor position = 0deg ?   */
         if ( in_phase == _Phase_shift_jdg_TBL[PHASE_DEG0] ) {
                                                  /*     Yes the previous position is right ?   */
            in_phase = PHASE_DEG0;               /*       Yes updating of an input angle (0deg)   */
            r0edg = INT1_FALLING_EDGE;  /*       set the INT1 falling edge   */
            ir_int1ic = 0;                        /*       clear the interrupt request   */
            _OutPhase_Active(motor_sts);    /*       output   */
         }
      }
   }

}
```

```
/*""FUNC COMMENT""*****************************************************************
 *   Name               : Sens_In1_int()
 *-----------------------------------------------------------------------------
 *   Parameters       : None
 *   Returns          : None
 *   Description     : interrupt of Rotor position signal 1 occurs
 *   SubRoutine       : _OutPhase_Active()
 *                    : _Sens_Input()
 *-----------------------------------------------------------------------------
 *   Note               : use the bank register 1
 *-----------------------------------------------------------------------------
 *   History          : 2003-10-08: processing time
 *                    :    (except the interrupt sequence, the bank change and return)
 *                    :                  300deg: about 19.0us / 120deg: about 17.3us
 *""FUNC COMMENT END""*************************************************************/
#pragma INTERRUPT/B Sens_In1_int
void Sens_In1_int ( void )
{
   u08       senser;

   senser = _Sens_Input();                    /* rotor position signal input    */
   if ( pol_int0ic == INT0_FALLING_EDGE ) {
                                                  /* INT0 falling endge ? {    */
      if ( senser == SENSER_DEG300 ) {   /*    Yes rotor position = 300deg ?    */
         if ( in_phase == _Phase_shift_jdg_TBL[PHASE_DEG300] ) {
                                                  /*     Yes the previous position is right ?    */
            in_phase = PHASE_DEG300;     /*       Yes updating of an input angle (300deg)    */
            pol_int0ic = INT0_RISING_EDGE;        /*       set the INT0 rising edge    */
            ir_int0ic = 0;                        /*       clear the interrupt request    */
            _OutPhase_Active(motor_sts);    /*       output    */
         }
      }
   }else{
      if ( senser == SENSER_DEG120 ) {   /*    No    rotor position = 120deg ?    */
         if ( in_phase == _Phase_shift_jdg_TBL[PHASE_DEG120] ) {
                                                  /*     Yes the previous position is right ?    */
            in_phase = PHASE_DEG120;     /*       Yes updating of an input angle (120deg)    */
            pol_int0ic = INT0_FALLING_EDGE;/*       set the INT0 falling edge    */
            ir_int0ic = 0;                        /*       clear the interrupt request    */
            _OutPhase_Active(motor_sts);    /*       output    */
         }
      }
   }

}
```

```
/*""FUNC COMMENT""*****************************************************************
 *   Name               : Sens_In2_int()
 *---------------------------------------------------------------------------
 *   Parameters       : None
 *   Returns          : None
 *   Description      : interrupt of Rotor position signal 2 occurs
 *   SubRoutine       : _OutPhase_Active()
 *                    : _Sens_Input()
 *---------------------------------------------------------------------------
 *   Note             : use the bank register 1
 *---------------------------------------------------------------------------
 *   History          : 2003-10-08: processing time
 *                    :   (except the interrupt sequence, the bank change and return)
 *                    :                 60deg: about 19.0us / 240deg: about 17.6us
 *""FUNC COMMENT END""*************************************************************/
#pragma INTERRUPT/B Sens_In2_int
void Sens_In2_int ( void )
{
   u08       senser;

   senser = _Sens_Input();                    /* rotor position signal input    */
   if ( ki3pl == KI3_FALLING_EDGE ) {         /* key input falling endge ? {    */
      if ( senser == SENSER_DEG60 ) {         /*    Yes rotor position = 60deg ?    */
         if ( in_phase == _Phase_shift_jdg_TBL[PHASE_DEG60] ) {
                                                 /*     Yes the previous position is right ?    */
            in_phase = PHASE_DEG60;              /*       Yes updating of an input angle (60deg)    */
            ki3pl = KI3_RISING_EDGE;       /*       set the KI3 rising edge    */
            ir_kupic = 0;                        /*       clear the interrupt request    */
            _OutPhase_Active(motor_sts);   /*       output    */
         }
      }
   }else{
      if ( senser == SENSER_DEG240 ) {   /*    No    rotor position = 240deg ?    */
         if ( in_phase == _Phase_shift_jdg_TBL[PHASE_DEG240] ) {
                                                 /*     Yes the previous position is right ?    */
            in_phase = PHASE_DEG240;     /*       Yes updating of an input angle (240deg)    */
            ki3pl = KI3_FALLING_EDGE;    /*       set the KI3 falling edge    */
            ir_kupic = 0;                        /*       clear the interrupt request    */
            _OutPhase_Active(motor_sts);   /*       output    */
         }
      }
   }

}
```

```
/*""FUNC COMMENT""***************************************************************
 *   Name               : Error_int()
 *-------------------------------------------------------------------------------
 *   Parameters       : None
 *   Returns           : None
 *   Description     : interrupt of motor stop signal occurs
 *   SubRoutine       : _OutPhase_Active()
 *                      : _PWM_Set()
 *-------------------------------------------------------------------------------
 *   Note               : this interrupt is forbidden in the part which performs
 *                      : motor control by main processing.
 *-------------------------------------------------------------------------------
 *   History           : max : about 15.73us (p3 off)
 *""FUNC COMMENT END""***********************************************************/
#pragma INTERRUPT/B Error_int
void Error_int ( void )
{

   motor_sts = MOTOR_ERROR;
   _OutPhase_Active(motor_sts);                    /* output   */
   _PWM_Set();                                     /* PWM output   */

}


/*""FUNC COMMENT""***************************************************************
 *   Name               : _InitOutPhase_Active()
 *-------------------------------------------------------------------------------
 *   Parameters       : motor status
 *   Returns           : None
 *   Description     : change the output phase (at the time of a start processing)
 *   SubRoutine       : None
 *-------------------------------------------------------------------------------
 *   Note               : *1 short-circuit inhibition: about 2.3us
 *-------------------------------------------------------------------------------
 *   History           :
 *""FUNC COMMENT END""***********************************************************/
static void _InitOutPhase_Active ( MOTOR_STATUS status )
{
u08 p1_data,p3_data;

   switch (status) {
   case MOTOR_START:                               /* case MOTOR_START:   */
      p1 &= 0xf8;                                  /*   positive phase output off (note *1)   */
      p3 |= 0x07;                                  /*   negative phase output off (note *1)   */
      p1_data = _Init_p1Active_Phase_TBL[out_phase];          /*   set the P1 output data   */
      p3_data = _Init_p3Active_Phase_TBL[out_phase];          /*   set the P3 output data   */
// a cycle adjustment is required, if the CPU clock changes.
      p1 = p1_data;
      p3 = p3_data;
      if ( ++out_phase >= PHASE_DEG360 )           /*   one revolution ?   */
         out_phase = PHASE_DEG0;                   /*    Yes initialize the output phase (0deg)   */
      ++cu_init_rotate;                            /*    count the initial rotation   */
      break;
   case MOTOR_START2:                              /* case MOTOR_START2:   */
      p1 &= 0xf8;                                  /*   positive phase output off   */
      p3 |= 0x07;                                  /*   negative phase output off   */
   default:
      break;
   }
}
```

```
/*""FUNC COMMENT""*****************************************************************
 *    Name                : _OutPhase_Active()
 *-----------------------------------------------------------------------------
 *    Parameters      : motor status
 *    Returns           : None
 *    Description     : change the output phase (at the time of a normal processing)
 *    SubRoutine      : None
 *-----------------------------------------------------------------------------
 *    Note              : *1 short-circuit inhibition: about 2.3us
 *                      : *2 at the time of a start prcessing
 *                      :                : this is performed by main processing
 *                      :      at the time of a normal prcessing
 *                      :                : this is performed by main processing
 *-----------------------------------------------------------------------------
 *    History           :
 *""FUNC COMMENT END""*************************************************************/
static void _OutPhase_Active ( MOTOR_STATUS status )
{
u08 p1_data,p3_data;

   switch (status) {
   case MOTOR_NOMAL:                                /* case MOTOR_NOMAL:   */
      p1 &= 0xf8;                                   /*   positive phase output off (note *1)   */
      p3 |= 0x07;                                   /*   negative phase output off (note *1)   */
      p1_data = _p1Active_Phase_TBL[in_phase];      /*   set the P1 output data   */
      p3_data = _p3Active_Phase_TBL[in_phase];      /*   set the P3 output data   */
// a cycle adjustment is required, if the CPU clock changes.
      p1 = p1_data;
      p3 = p3_data;
      out_phase = in_phase;                 /*   change the output phase   */
      break;
   case MOTOR_START:                                /* case MOTOR_START:   */
   case MOTOR_START2:                               /* case MOTOR_START2:   */
      break;
   default:
      p1 &= 0xf8;                                   /*   positive phase output off   */
      p3 |= 0x07;                                   /*   negative phase output off   */
      int1ic = 0x00;                        /*   rotor signal 0 interrupt diable   */
      int0ic = 0x00;                        /*   rotor signal 1 interrupt diable   */
      kupic = 0x00;                         /*   rotor signal 2 interrupt diable   */
      int3ic = 0x00;                        /*   ERR signal interrupt disable   */
      out_phase = PHASE_DEG0;                       /*   output phase 0deg   */
      pwm_duty = DUTY_0P;                           /*   PWM duty 0%   */
   }
}
```

```
/*""FUNC COMMENT""*****************************************************************
 *    Name               : _PWM_Set()
 *---------------------------------------------------------------------------
 *    Parameters         : None
 *    Returns            : None
 *    Description        : set or change of PWM output
 *    SubRoutine         : None
 *---------------------------------------------------------------------------
 *    Note               : This function is performed by the main and INT3 interrupt
 *                       : processing.
 *---------------------------------------------------------------------------
 *    History            :
 *""FUNC COMMENT END""*************************************************************/
static void _PWM_Set ( void )
{
   pwm_period = PWM_15kHz;              /* PWM period    */
   tm1 = pwm_period;                            /* set a tm1 register (PWM period)    */
   switch (pwm_duty) {
   case DUTY_10P:                       /* case 10%:    */
      tm0 = DUTY_10P_COMP;              /*    set a tm0 register    */
      break;
   case DUTY_20P:                       /* case 20%:    */
      tm0 = DUTY_20P_COMP;              /*    set a tm0 register    */
      break;
   default:
      tm0 = 0xffff;                     /*    set a tm0 register    */
   }
   tcout = 0x47;                                /* CMP00-CMP02 enable, CMP10-CMP12 disable */
                                                /* CMP00-CMP02 reverse    */
   tcc00 = 1;                                   /* timer C count start    */
}


/*""FUNC COMMENT""*****************************************************************
 *    Name               : _Sens_Input()
 *---------------------------------------------------------------------------
 *    Parameters         : None
 *    Returns            : data of rotor position signal
 *    Description        : rotor position signal input
 *    SubRoutine         : None
 *---------------------------------------------------------------------------
 *    Note               :
 *---------------------------------------------------------------------------
 *    History            :
 *""FUNC COMMENT END""*************************************************************/
static u08 _Sens_Input ( void )
{
u08 work0,work1;

   work0 = p4 & P4_MASK;
   work1 = p1 & P1_MASK;
   return (work0 |= work1);
}
```

```
/**** P1 output data table (at the time of a start prcessing) **************/
static const u08 _Init_p1Active_Phase_TBL[] = {
      INIT_P1 | 0x01,                   /*     0deg:U phase,(VB phase) ON    */
      INIT_P1 | 0x01,                   /*    60deg:U phase,(WB phase) ON    */
      INIT_P1 | 0x02,                   /* 120deg:V phase,(WB phase) ON    */
      INIT_P1 | 0x02,                   /* 180deg:V phase,(UB phase) ON    */
      INIT_P1 | 0x04,                   /* 240deg:W phase,(UB phase) ON    */
      INIT_P1 | 0x04                    /* 300deg:W phase,(VB phase) ON    */
   };

/**** P3 output data table (at the time of a start prcessing) **************/
static const u08 _Init_p3Active_Phase_TBL[] = {
      INIT_P3 & 0xfd,                   /*     0deg:(U phase),VB phase ON    */
      INIT_P3 & 0xfb,                   /*    60deg:(U phase),WB phase ON    */
      INIT_P3 & 0xfb,                   /* 120deg:(V phase),WB phase ON    */
      INIT_P3 & 0xfe,                   /* 180deg:(V phase),UB phase ON    */
      INIT_P3 & 0xfe,                   /* 240deg:(W phase),UB phase ON    */
      INIT_P3 & 0xfd                    /* 300deg:(W phase),VB phase ON    */
   };

/**** P1 output data table (at the time of a normal prcessing) *************/
static const u08 _p1Active_Phase_TBL[] = {
      INIT_P1 | 0x01,                   /*    60deg:U phase,(WB phase) ON    */
      INIT_P1 | 0x02,                   /* 120deg:V phase,(WB phase) ON    */
      INIT_P1 | 0x02,                   /* 180deg:V phase,(UB phase) ON    */
      INIT_P1 | 0x04,                   /* 240deg:W phase,(UB phase) ON    */
      INIT_P1 | 0x04,                   /* 300deg:W phase,(VB phase) ON    */
      INIT_P1 | 0x01                    /*     0deg:U phase,(VB phase) ON    */
   };

/**** P3 output data table (at the time of a normal prcessing) *************/
static const u08 _p3Active_Phase_TBL[] = {
      INIT_P3 & 0xfb,                   /*    60deg:(U phase),WB phase ON    */
      INIT_P3 & 0xfb,                   /* 120deg:(V phase),WB phase ON    */
      INIT_P3 & 0xfe,                   /* 180deg:(V phase),UB phase ON    */
      INIT_P3 & 0xfe,                   /* 240deg:(W phase),UB phase ON    */
      INIT_P3 & 0xfd,                   /* 300deg:(W phase),VB phase ON    */
      INIT_P3 & 0xfd                    /*     0deg:(U phase),VB phase ON    */
   };

/**** previous position judge table ***************************************/
static const u08 _Phase_shift_jdg_TBL[] = {
      PHASE_DEG300,                             /*     0deg    */
      PHASE_DEG0,                               /*    60deg    */
      PHASE_DEG60,                              /* 120deg    */
      PHASE_DEG120,                             /* 180deg    */
      PHASE_DEG180,                             /* 240deg    */
      PHASE_DEG240,                             /* 300deg    */
      0xff                              /* void     */
   };
```

6.0 Reference

Hardware Manual

R8C/11 Group Hardware Manual

(Acquire the most current version from Renesas Technology website)

6.0 Web-site and contact for support

Renesas Web-site

http://www.renesas.com

For more information on Renesas technical support about M16C family products

Mail to : support_apl@renesas.com

REVISION HISTORY	R8C/11 Group Application Note Control of a Brushless DC Motor

Rev.	Date	Description	
		Page	Summary
1.00	Nov.18, 2003	-	First edition issued

Keep safety first in your circuit designs!

1. Renesas Technology Corporation puts the maximum effort into making semiconductor products better and more reliable, but there is always the possibility that trouble may occur with them. Trouble with semiconductors may lead to personal injury, fire or property damage.
   Remember to give due consideration to safety when making your circuit designs, with appropriate measures such as (i) placement of substitutive, auxiliary circuits, (ii) use of nonflammable material or (iii) prevention against any malfunction or mishap.

Notes regarding these materials

1. These materials are intended as a reference to assist our customers in the selection of the Renesas Technology Corporation product best suited to the customer's application; they do not convey any license under any intellectual property rights, or any other rights, belonging to Renesas Technology Corporation or a third party.
2. Renesas Technology Corporation assumes no responsibility for any damage, or infringement of any third-party's rights, originating in the use of any product data, diagrams, charts, programs, algorithms, or circuit application examples contained in these materials.
3. All information contained in these materials, including product data, diagrams, charts, programs and algorithms represents information on products at the time of publication of these materials, and are subject to change by Renesas Technology Corporation without notice due to product improvements or other reasons. It is therefore recommended that customers contact Renesas Technology Corporation or an authorized Renesas Technology Corporation product distributor for the latest product information before purchasing a product listed herein.
   The information described here may contain technical inaccuracies or typographical errors.
   Renesas Technology Corporation assumes no responsibility for any damage, liability, or other loss rising from these inaccuracies or errors.
   Please also pay attention to information published by Renesas Technology Corporation by various means, including the Renesas Technology Corporation Semiconductor home page (http://www.renesas.com).
4. When using any or all of the information contained in these materials, including product data, diagrams, charts, programs, and algorithms, please be sure to evaluate all information as a total system before making a final decision on the applicability of the information and products. Renesas Technology Corporation assumes no responsibility for any damage, liability or other loss resulting from the information contained herein.
5. Renesas Technology Corporation semiconductors are not designed or manufactured for use in a device or system that is used under circumstances in which human life is potentially at stake. Please contact Renesas Technology Corporation or an authorized Renesas Technology Corporation product distributor when considering the use of a product contained herein for any specific purposes, such as apparatus or systems for transportation, vehicular, medical, aerospace, nuclear, or undersea repeater use.
6. The prior written approval of Renesas Technology Corporation is necessary to reprint or reproduce in whole or in part these materials.
7. If these products or technologies are subject to the Japanese export control restrictions, they must be exported under a license from the Japanese government and cannot be imported into a country other than the approved destination.
   Any diversion or reexport contrary to the export control laws and regulations of Japan and/or the country of destination is prohibited.
8. Please contact Renesas Technology Corporation for further details on these materials or the products contained therein.