V850E2/ML4

Complementary PWM Output Function

R01AN1224EJ0100
Rev.1.00
Jun. 22, 2012

Abstract

This document describes how to set up the complementary PWM function and also gives an outline of the operation and describes the procedures for using a sample program for the V850E2/ML4.

The features of the [function or operation] are described below.

- 6-phase complementary PWM output with dead-time, using culling timer
- Controlling brushless motor by 120degree (3-phase) electrical currency.
- Feedback motor control by PID (portion, integral, and differential element) control.
  (in case of motor control, do not use differential element)
- Changing the angular velocity of the motor, by A/D conversion of the VR-switch.

Products

V850E2/ML4

Integrated development environments

CubeSuite+, GHS MULTI V5.1.7D, and IAR for V850 Kickstart V3.80.

When using this application note with other Renesas MCUs, careful evaluation is recommended after making modifications to comply with the alternate MCU.

Contents

1. Specifications

The sample program has following features.

- 6-phase complementary PWM output with dead-time, using culling timer
- Controlling brushless motor by 120degree (3-phase) electrical currency.
- Feedback motor control by PID (portion, integral, and differential element) control.
  (in case of motor control, do not use differential element)
- Changing motor velocity, by A/D conversion of the signal from VR-switch.

The functions of timer channels are as follows.

Timer channel	Function
TAUA0 channel 0	Master channel, interrupts every 25usec.
TAUA0 channel 1	Culling timer, slave, interrupts every 50usec (career interrupt). In career interrupt, the system controls the motor.
TAUA0 channel 2, 4, 6	Slave, up and down counting mode, triggered by up and down output of the master channel. Outputs high-side of U, V, W of PWM. The duty-ratio is depended on the value of TAUA0CDRn register.
TAUA0 channel 3, 5, 7	Slave, one-counting mode, triggered by dead-time output. Outputs low-side of U, V, W of PWM. The dead-time is depended on the value of TAUA0CDRn register.
TAUA0 channel 8	Interrupts every 1msec.
TAUA0 channel 9	Free-running counter, interrupts every 1sec.

Table 1.1 lists the Peripheral Functions and their Applications and Figure 1.1 shows the Usage Example.

Table 1.1 Peripheral Functions and their Applications

Peripheral Function	Application
Ports(P4_3, P4_4)	Connected to LEDs, and light on or off LEDs.
INTP5	Interrupts by a signal of U-phase hole sensor.
INTP6	Interrupts by a signal of V-phase hole sensor.
INTP7	Interrupts by a signal of W-phase hole sensor.
A/D converter	A/D converts the analog signal from VR switch via pin ANI08

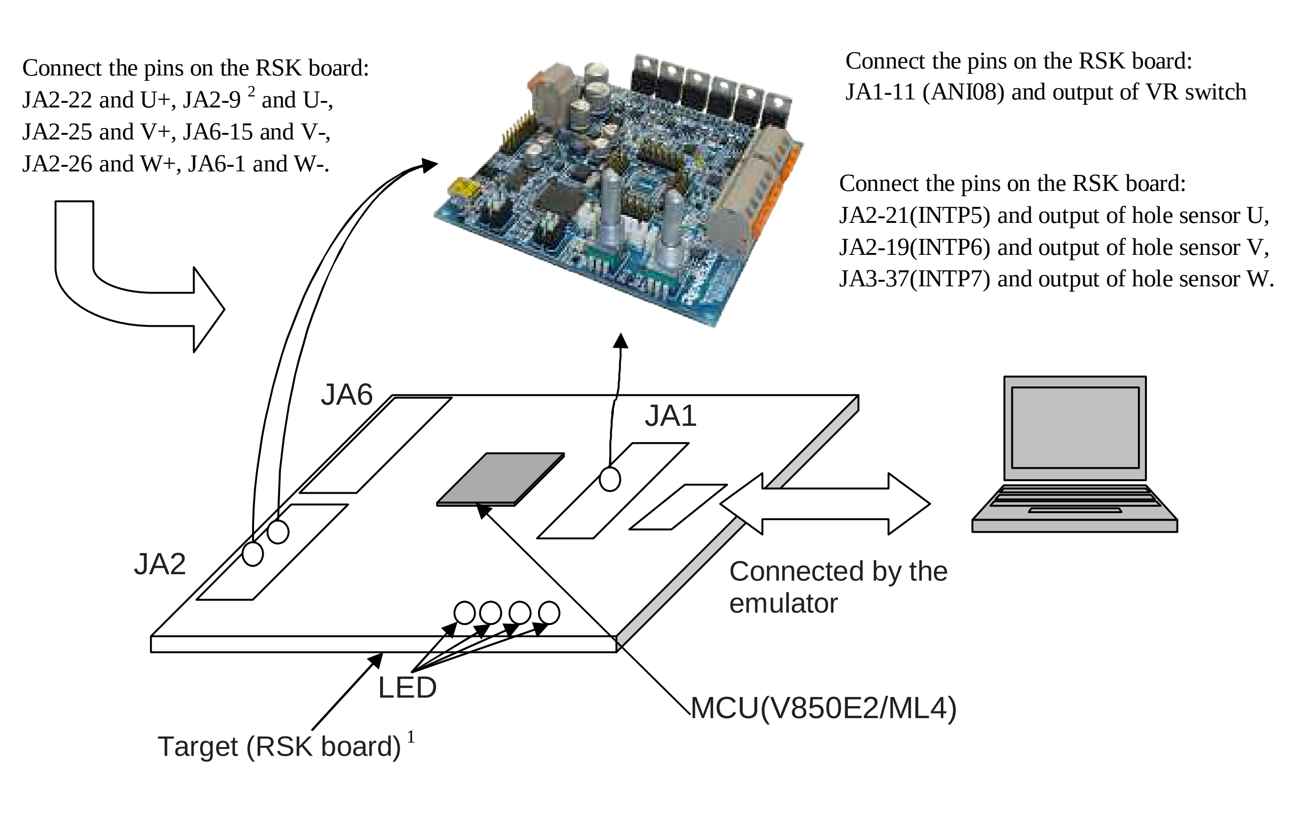


Figure 1.1 Usage Example

[1] Mass production of RSK board will start in August, 2012.

[2] RSK JA2-9 is not connected by default; you can connect by 0-ohm resistor.

2. Operation Confirmation Conditions

The sample code accompanying this application note has been run and confirmed under the conditions below.

Table 2.1 Operation Confirmation Conditions

Item	Contents
MCU used	V850E2/ML4
Operating frequency	200MHz (PLL multiplies the oscillator input frequency (fx: 10MHz) by 20.)
Operating voltage	3.3V
Integrated development environment	CubeSuite+ V1.00
	GHS MULTI V5.1.7D
	IAR for V850 Kickstart V3.80.1
C compiler	CX V1.20(CubeSuite+), optimization: default
	C-V850E 5.1.7 RELEASE(GHS MULTI) , optimization: default
	IAR C/C++ Compiler for V850 3.80.1 [Kickstart] (3.80.1.30078), optimization: default
Operating mode	Normal operation mode
Sample code version	V1.00
Board used	RSK board
Device used	E1 emulator or MINICUBE, stable power source(KENWOOD product)
Tool used	none

3. Hardware

3.1 Hardware Configuration

Figure 3.1 shows a hardware configuration.

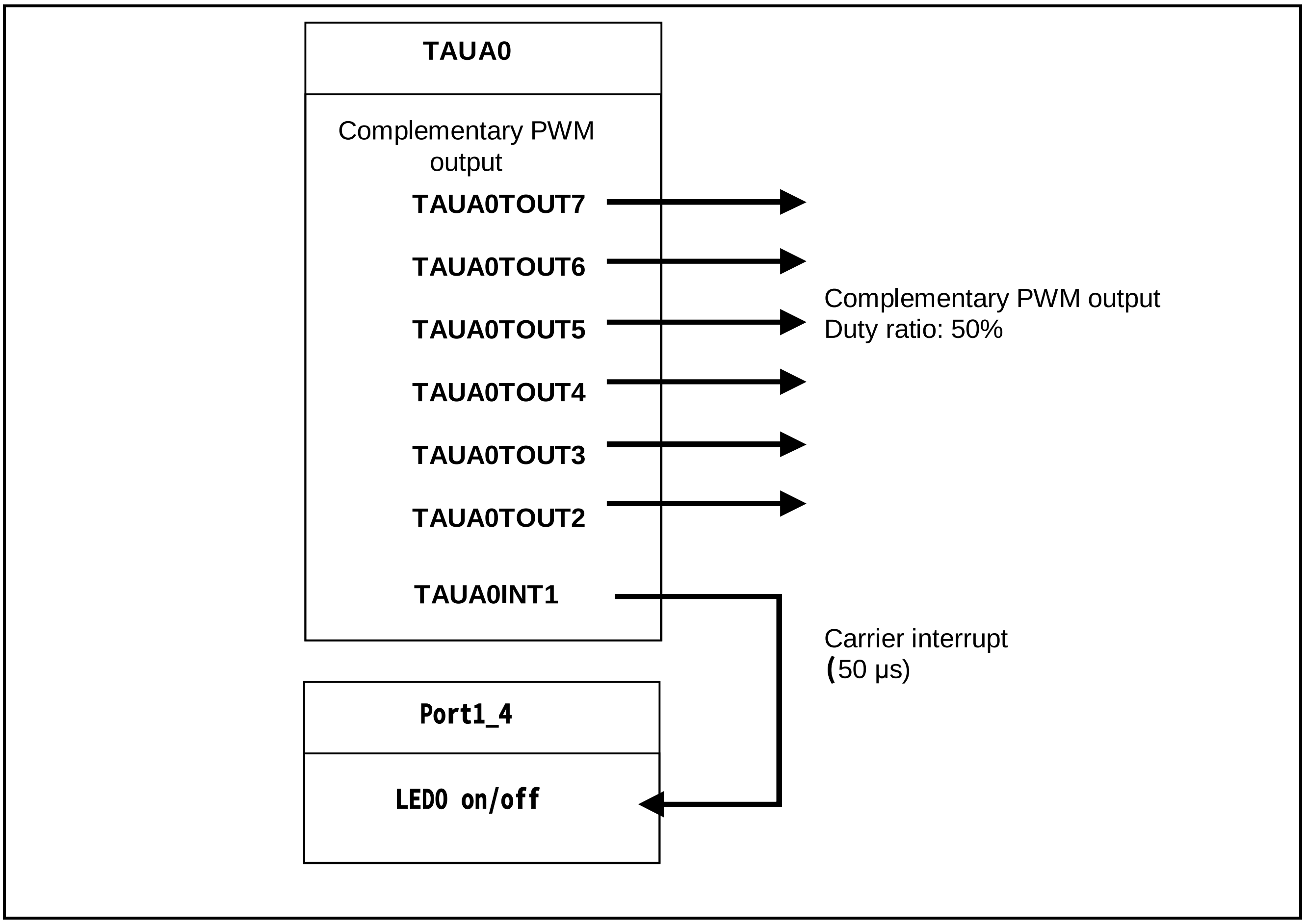


Figure 3.1 Hardware configuration

3.2 Pin(s) Used

Table 3.1 lists the Pins Used and Its Function.

Table 3.1 Pins Used and Its Functions

Pin Name	I/O	Function
PORT P4_3	output	Port mode, output, LED2
PORT P4_4	output	Port mode, output, LED3
P1_2/D18/TA0_I2/TE0_TI1/INTP7/**TA0_O2**	output	Complementary PWM output (high-side of U phase)
P1_3/D19/TA0_I3/INTP8/**TA0_O3**/PPON	output	Complementary PWM output (low-side of U phase)
P1_4/D20/TA0_I4/TE0_AI/INTP9/**TA0_O4**	output	Complementary PWM output (high-side of V phase)
P1_5/D21/TA0_I5/INTP10/**TA0_O5**	output	Complementary PWM output (low-side of V phase)
P1_6/D22/TA0_I6/TE0_BI/INTP11/**TA0_O6**	output	Complementary PWM output (high-side of W phase)
P1_7/D23/TA0_I7/INTP12/**TA0_O7**	output	Complementary PWM output (low-side of W phase)
P1_0/D16/TA0_I0/TE0_TI0/**INTP5**/TA0_O0	input	Hole sensor U interrupt.
P1_1/D17/TA0_I1/OCI/**INTP6**/TA0_O1	input	Hole sensor V interrupt.
P4_0/A16/**INTP7/**TA1_I8/TA1_O8/CSI0F_CS2	input	Hole sensor W interrupt.
P8_2/**ANI08**	input	Signal from VR switch

4. Software

4.1 Operation Overview

The following figure shows overview of software operation. Function main() calls each initializing function and waits interrupt.

Figure 4.1 shows the Sequence Diagram (Initialization).

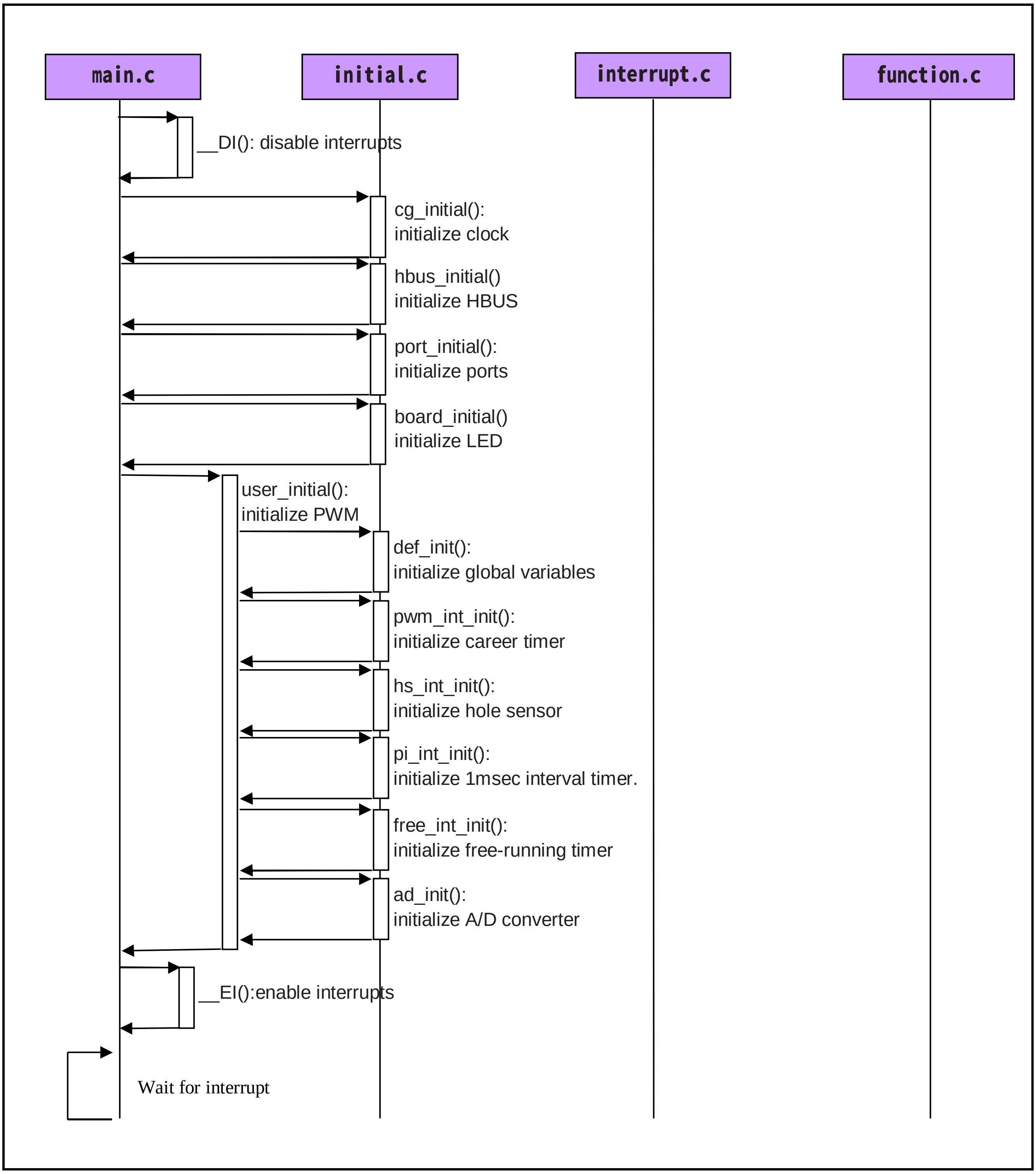


Figure 4.1 Sequence Diagram (Initialization)

Figure 4.2 shows Sequence Diagram (interrupt operation).

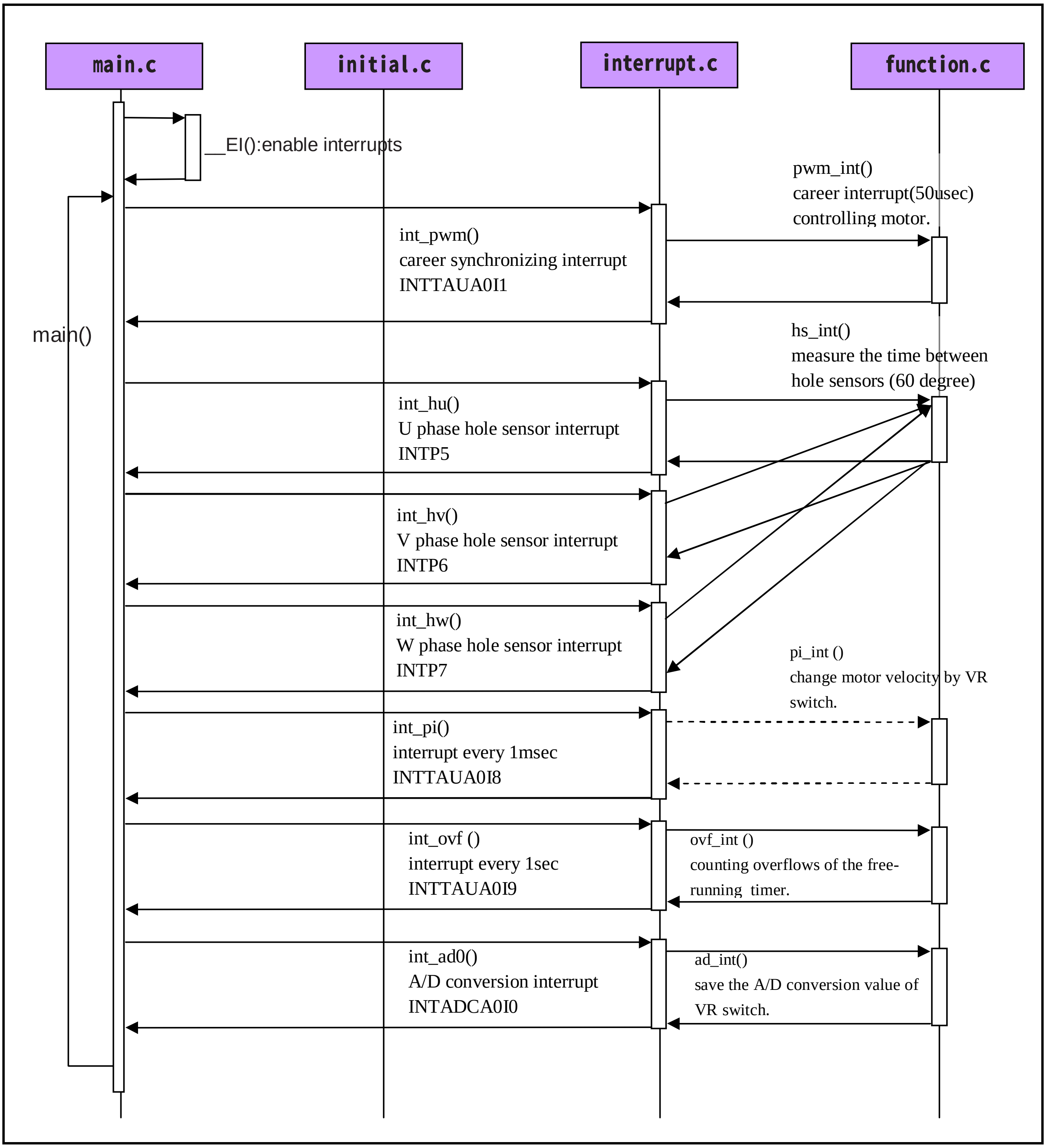


Figure4.2 Sequence Diagram (interrupt operation)

4.2 Required Memory Size

Table 4.1 lists the Required Memory Size. (CubeSuite+, optimization=default)

Table 4.1 Required Memory Size

Memory Used	Size	Remarks
ROM	9056	Shown as ROM area size in map file
RAM	4152	Shown as RAM area size in map file
Maximum user stack usage	8	CubeSuite+ stack estimation tool calculated.
Maximum interrupt stack usage	588	The same as above.

Note: • The required memory size varies depending on the C compiler version and compiler options.

4.3 File Composition

Table 4.2 lists the File(s) Used in the Sample Code. Files not generated by the integrated development environment should not be listed in this table.

Table 4.2 File(s) Used in the Sample Code

File Name	Outline	Remarks
crtE.s	Initialize hardware	Only in the project for CubeSuite+
startup.s		Only in the project for GHS MULTI
V850E2ML4.dir	Linker directive file	Only in the project for CubeSuite+
V850E2_ML4 ADC.ld		Only in the project for GHS MULTI
vector.s	Vector table	Only in the project for GHS MULTI
adc.h	Declare variables and functions.	
df4022_800.h	Declare register macros for V850E2/ML4	Only in the project for GHS MULTI
main.c	Main routine	
initial.c	Initialize software	
function.c	Control the motor	
sin_table180.c	Sine table	
interrupt.c	Interrupt routines	
portconfig.c	Port configurasion.	

4.4 Option-Setting Memory

This sample does not specify any option-bytes. Specify them if necessary.

4.5 Constant(s)

Table 4.3 lists the Constant(s) Used in the Sample Code.

Table 4.3 Constant(s) Used in the Sample Code

Constant Name	Setting Value	Contents
onst unsigned short u2_sintbl_U30[]	{ 0, 2, 3, 5, 7, 9…}	Sine table from 0 to 29 degree.
const unsigned short u2_sintbl_U60[]	{50, 52, 53, 54, 56…}	Sine table from 30 to 89 degree.

4.6 Variable(s)

Table 4.4 lists the Global Variable(s).

Table 4.4 Global Variable(s)

Type	Variable Name	Contents	Function Used
unsigned char	flg_tm9_ovf;	Overflow flag of TAUA0 channel 9	pwm_int() ovf_int() def_init();
unsigned short	u2_pwm_tm9_cnt;	The counted value of TAUA0 channel 9, as of PWM career interrupt.	pwm_int()
unsigned char	u1_hs_st	The status of hole sensor as of PWM career interrupt.	pwm_int()
unsigned short	u2_swing_pwm	The maximum swing of PWM output	pwm_int() pi_int() def_init()
unsigned short	u2_hs_tm9_cnt	The counted value of TAUA0 channel 9, as of hole sensor interrupt (now).	pwm_int() hs_int()
unsigned short	u2_hs_tm9_cnt_old	The counted value of TAUA0 channel 9, as of last hole sensor interrupt.	hs_int() def_init()
unsigned short	u2_hs_tm9_cnt_dif	Difference of counted values of TAUA0 channel 9, as of now and the last hole sensor interrupt.	hs_int() pi_int()
unsigned short	u2_vr_st	The reduction of maximum swing of PWM output , changed by VR switch.	pi_int() ad_int() def_init()
unsigned short	u2_order_rpm	Commanded value of angular velocity (rpm)	pi_int() def_init()
unsigned short	u2_rpm	angular velocity (rpm)	pi_int()
unsigned short	u2_ang_us	angular velocity (angle/usec)*2^16 (to avoid floating point operation)	def_init() pwm_int() pi_int()
unsigned long	u4_temp_cal_pi	variable for PI control operation	pi_int()

4.7 Function(s)

Table 4.5 lists the Function(s).

Table 4.5 Function(s)

Function Name	Outline
void main(void)	Calls necessary initialization functions before entering an infinite loop.
void port_initial(void)	Set up ports and their mode.
void PortConfigulation0(void)	Set up port group 0
void PortConfigulation1(void)	Set up port group 1
void PortConfigulation2(void)	Set up port group 2
void PortConfigulation3(void)	Set up port group 3
void PortConfigulation4(void)	Set up port group 4
void PortConfigulation5(void)	Set up port group 5
void PortConfigulation6(void)	Set up port group 6
void PortConfigulation7(void)	Set up port group 7
void PortConfigulation8(void)	Set up port group 8
void cg_initial(void)	Initialize clock
void hbus_initial(void)	Initialize the AHB bus
void board_initial(void)	Initialize the LEDs
void user_initial(void)	initialize PWM
void def_init(void)	Initialize global variables
void pwm_int_init(void)	Initialize career timer
void hs_int_init(void)	initialize hole sensor
void pi_int_init(void)	Initialize 1msec interval timer.
void free_int_init(void)	Initialize free-running timer
void ad_init(void)	Initialize A/D converter
interrupt void int_pwm(void)	career synchronizing interrupt INTTAUA0I1 (every 50us)
interrupt void int_hu(void)	U phase hole sensor interrupt INTP5
interrupt void int_hv(void)	V phase hole sensor interrupt INTP6
interrupt void int_hw(void)	W phase hole sensor interrupt INTP7
interrupt void int_pi(void)	interrupt every 1msec INTTAUA0I8
interrupt void int_ovf(void)	interrupt every 1sec INTTAUA0I9
interrupt int_ad0(void)	A/D conversion interrupt INTADCA0I0
void pwm_int(void)	Career interrupt (50usec) controlling motor.
void hs_int(void)	measure the time between hole sensors (60 degree)
void ovf_int(void)	Counting overflows of the free-running timer.
void pi_int(void)	Change motor velocity by VR switch.
void ad_int(void)	Save the A/D conversion value of VR switch.

4.8 Function Specification(s)

The following tables list the sample code function specifications.

main()		
Outline	Main routine	
Header	—	
Declaration	void main(void)	
Description	Calls initializing functions, enters infinite loop and waits PWM interrupt.	
Arguments	none	-
Return Value	none	

port_initial		
Outline	Sets up ports and their mode.	
Header	pwm.h	
Declaration	void port_initial (void)	
Description	Sets up ports for LED, timer output, hole sensor input, and A/D converter input.	
Arguments	none	-
Return Value	none	

PortConfigulation0() ～PortConfigulation8()		
Outline	Set up each port group.	
Header	pwm.h	
Declaration	void PortConfigulation0(void)…void PortConfigulation8(void)	
Description	Called by main(), and set up port group.	
Arguments	none	-
Return Value	none	

cg_initial()		
Outline	Initialize clock	
Header	pwm.h	
Declaration	void cg_initial(void)	
Description	Initialize clock.	
Arguments	none	-
Return Value	none	

hbus_initial()		
Outline	Initialize H-bus	
Header	pwm.h	
Declaration	void hbus_initial(void)	
Description	Initializes AHB-bus	
Arguments	none	-
Return Value	none	

board_initial()

Outline	Initialize board
Header	adc.h
Declaration	void board_initial(void)
Description	Initialize LED on the board.
Arguments	— —
Return Value	—

[Function Name]

Outline	Initialize users' memory.
Header	pwm.h
Declaration	void ram_initial(void)
Description	Initialize LED on the board.
Arguments	— —
Return Value	—

user_initial()

Outline	Initialize PWM.
Header	pwm.h
Declaration	void adc_initial(void)
Description	Set up interrupt and initialize PWM.
Arguments	— —
Return Value	—

def_init()

Outline	Initialize variables for controlling the motor.
Header	—
Declaration	void def_init(void)
Description	Initialize variables for controlling the motor.
Arguments	— —
Return Value	—

pwm_int_init()

Outline	Initialize timer
Header	—
Declaration	void pwm_int_init(void)
Description	Set up timer channels in TAUA0, for oscillate PWM.
Arguments	— —
Return Value	—

hs_int_init()

Outline	Initialize hole sensor
Header	pwm.h
Declaration	void hs_int_init(void)
Description	Unmask hole sensor interrupts (INTP5, INTP6, INTP7).
Arguments	— —
Return Value	—

pi_int_init()

Outline	Initialize 1msec interval timer.
Header	pwm.h
Declaration	void pi_int_init(void)
Description	Set up TAUA0 channel 8 as 1msec interval timer.
Arguments	— —
Return Value	—

free_int_init()

Outline	Initialize free-running timer
Header	pwm.h
Declaration	free_int_init(void)
Description	Set up TAUA0 channel 9 as 1sec free-running timer.
Arguments	— —
Return Value	—

ad_init()

Outline	Initialize A/D converter
Header	pwm.h
Declaration	void ad_init(void)
Description	Initialize A/D converter and, start conversion.
Arguments	— —
Return Value	—

int_pwm()

Outline	Career synchronizing interrupt (INTTAUA0I1, every 50usec)
Header	—
Declaration	__interrupt void int_pwm(void)
Description	Calls pwm_int().
Arguments	— —
Return Value	—

int_hu()

Outline	U phase hole sensor interrupt (INTP5)
Header	—
Declaration	__interrupt void int_hu(void)
Description	Calls hs_int().
Arguments	— —
Return Value	—

int_hv()

Outline	VU phase hole sensor interrupt (INTP6)
Header	—
Declaration	__interrupt void int_hv(void)
Description	Calls hs_int().
Arguments	— —
Return Value	—

int_hw()

Outline	W phase hole sensor interrupt (INTP5)
Header	—
Declaration	__interrupt void int_hw(void)
Description	Calls hs_int().
Arguments	— —
Return Value	—

int_pi()

Outline	interrupt every 1msec (INTTAUA0I8)
Header	—
Declaration	__interrupt void int_pi(void)
Description	Calls pi_int().
Arguments	— —
Return Value	—

int_ovf()

Outline	interrupt every 1sec (INTTAUA0I9)
Header	—
Declaration	__interrupt void int_ovf (void)
Description	Calls ovf_int().
Arguments	— —
Return Value	—

int_ad0()

Outline	A/D conversion interrupt(INTADCA0I0)
Header	—
Declaration	__interrupt void int_ad0(void)
Description	Calls ad_int().
Arguments	— —
Return Value	—

pwm_int()

Outline	Career interrupt (50usec) controlling motor.
Header	pwm.h
Declaration	void pwm_int(void)
Description	Estimate the electrical degree from the time between hole sensor and career interrupts, and change electrical currency by changed PWM setting.
Arguments	— —
Return Value	—

hs_int ()

Outline	measure the time between hole sensors (60 degree)
Header	pwm.h
Declaration	void hs_int(void)
Description	In the hole sensor interrupt, measure the time from the last interrupt to now.
Arguments	— —
Return Value	—

ovf_int ()

Outline	Operate interrupt of free-running timer.
Header	pwm.h
Declaration	void ovf_int(void)
Description	Count overflows of the free-running timer.
Arguments	— —
Return Value	—

pi_int ()

Outline	Operate interrupt of 1msec interval timer.
Header	pwm.h
Declaration	void pi_int(void)
Description	Calculate the ordered velocity (u2_order_rpm) from A/D conversion of the VR switch.
Arguments	— —
Return Value	—

ad_int()

Outline	Operate interrupt of A/D conversion.
Header	pwm.h
Declaration	void ad_int(void)
Description	Save the A/D conversion value of VR switch.
Arguments	— —
Return Value	—

4.9 Flowchart(s)

4.9.1 Main Processing

Figure 4.1 shows the Main Processing.

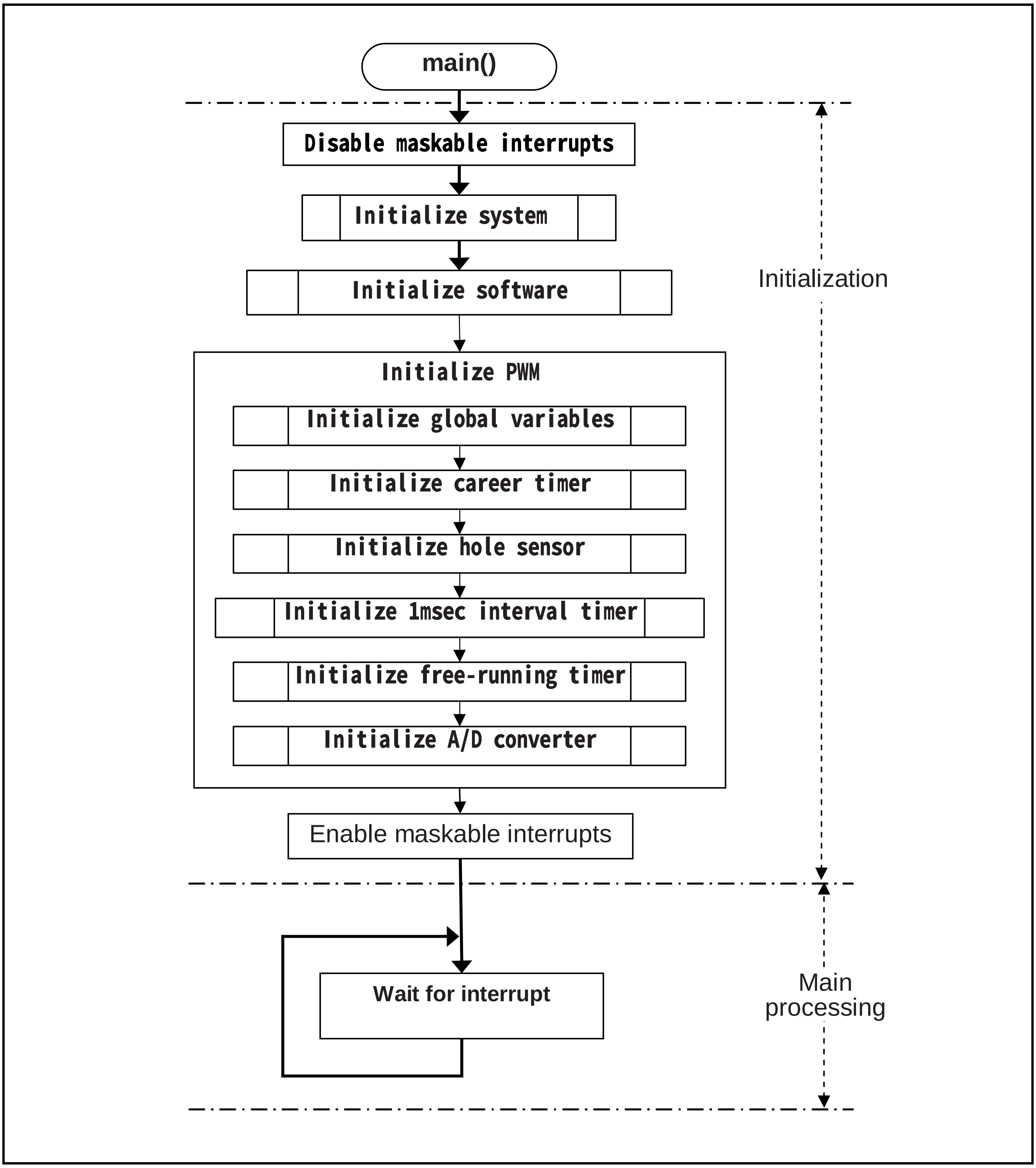


Figure 4.1 Main Processing

4.9.2 Inittialize global variables

Figure 4.2 shows the Initializing global variables.

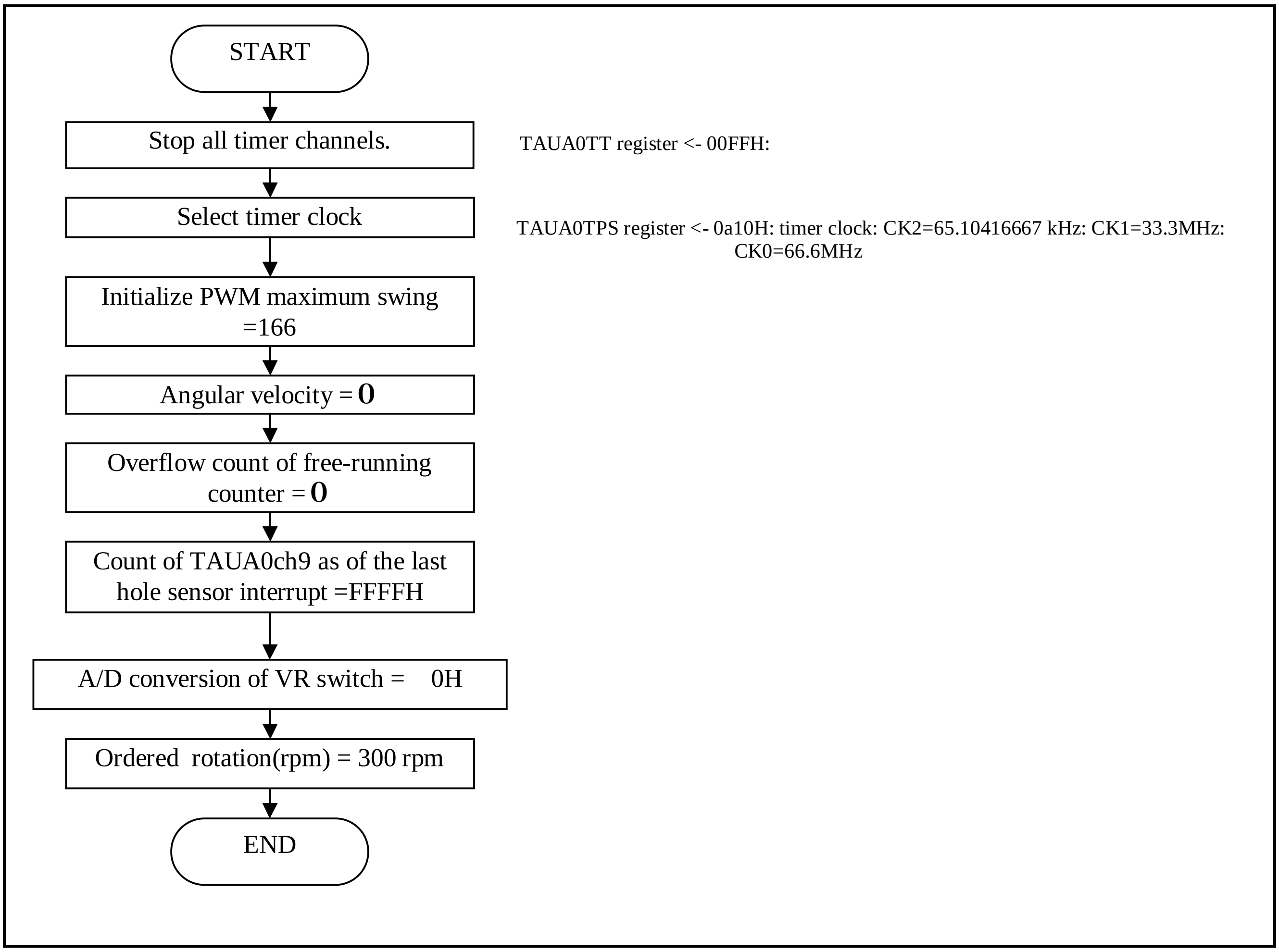


Figure 4.2 Initializing global variables

4.9.3 Initialize PWM timer

Figure 4.3 shows the initializing PWM timer.

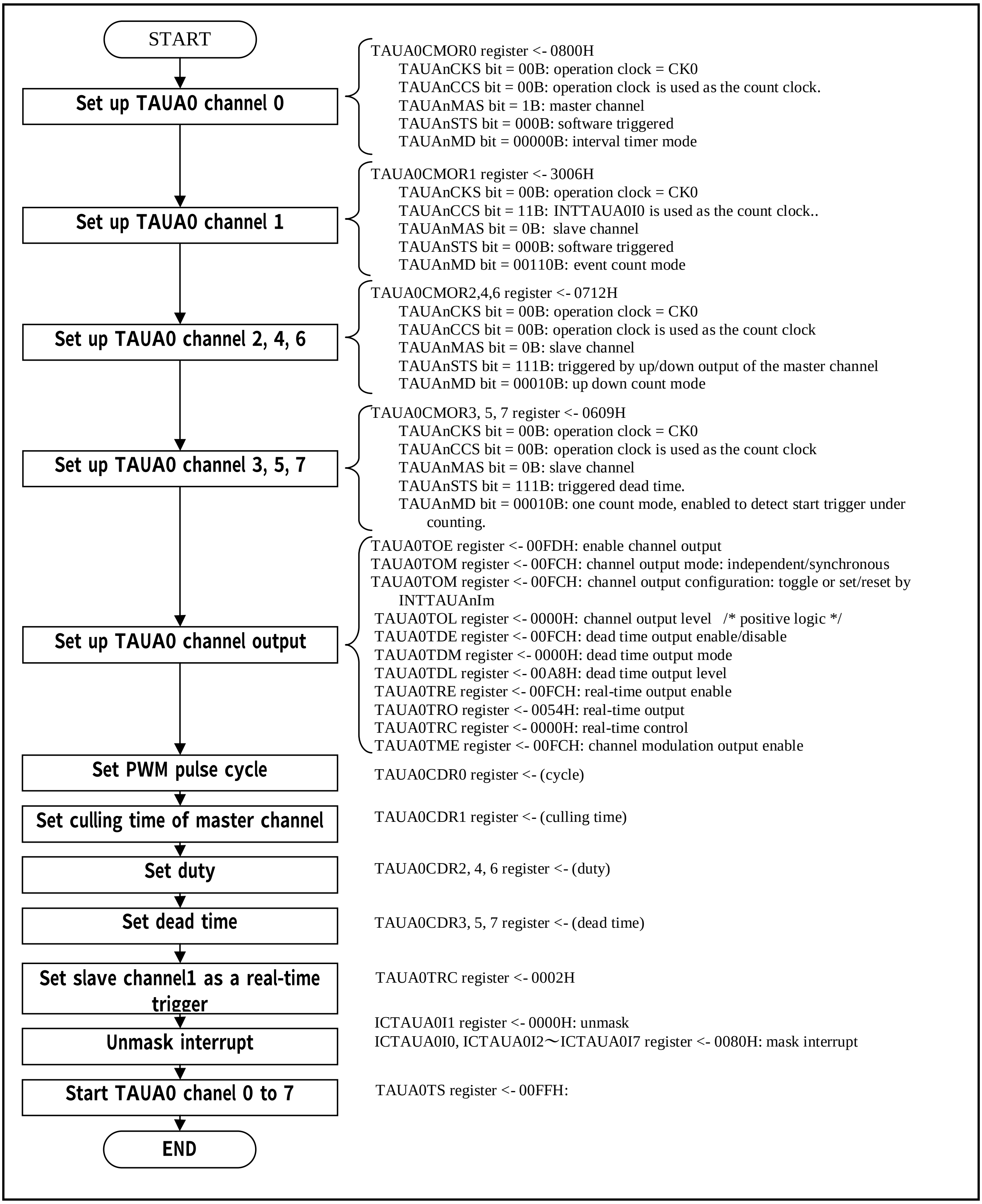


Figure 4.3 initializing PWM timer

4.9.4 Initialize 1msec interval timer

Figure 4.4shows the initializing 1msec interval timer.

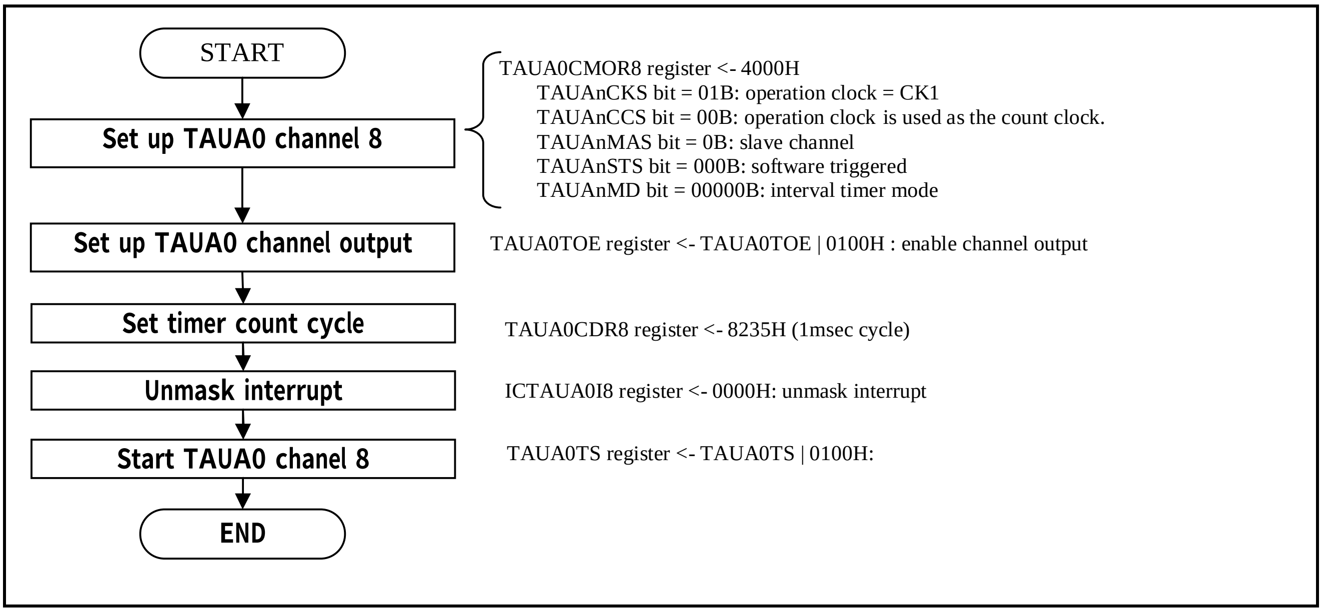


Figure 4.4 initializing 1msec interval timer

4.9.5 Initialize 1sec free-running timer

Figure 4.5 shows the initializing 1sec free-running timer.

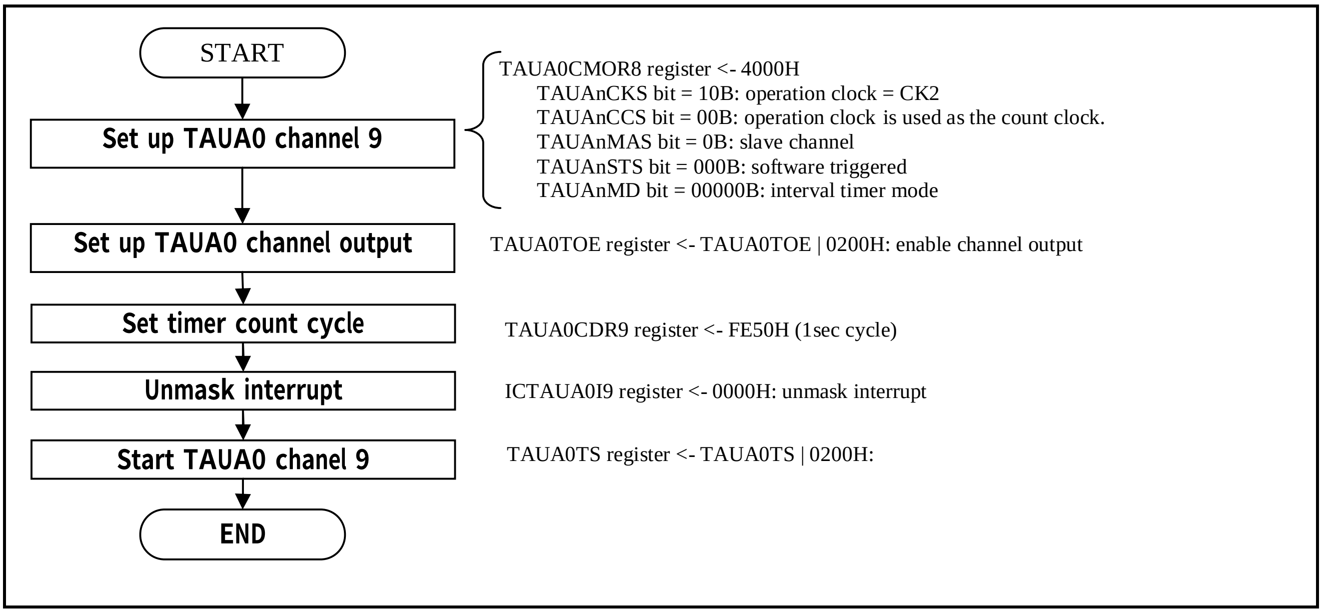
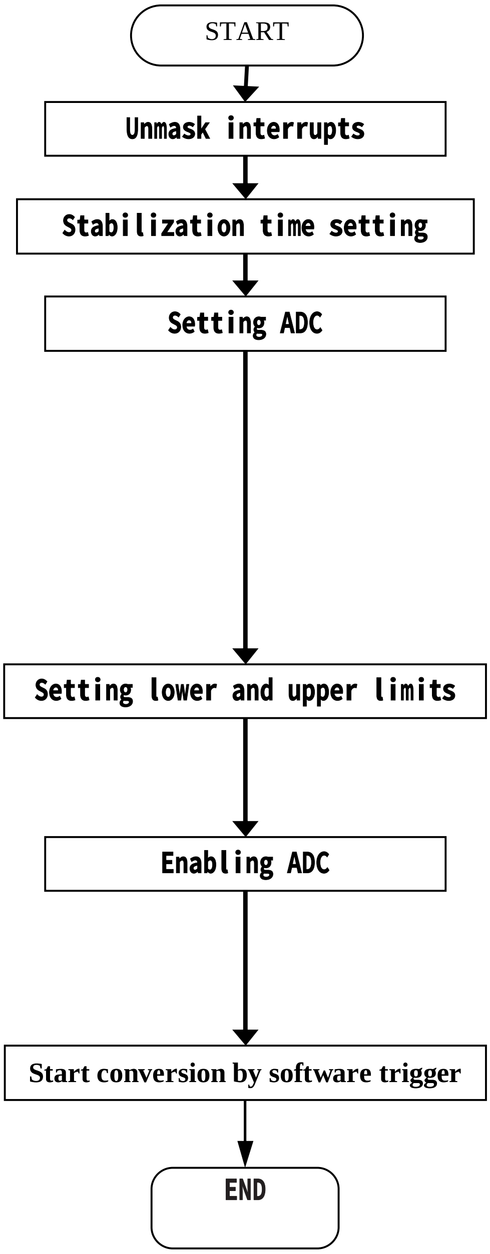


Figure 4.5 initializing 1sec free-running timer

4.9.6 Initialize A/D converter

Figure 4.6 shows the initializing A/D converter, to A/D convert the signal from VR switch.



ICADCA0I0 register<- 000FH: unmask conversion end interrupt

ADCAnCNT register <- FFH: stabilization time after power on.

ADCA0CTL1 register <- 00100301H
ADCAnMD1 bit = 0 : software trigger
ADCAnMD0 bit = 1 : continuous conversion mode
ADCAnCTYP bit = 0 : 12-bit resolution
ADCAnTiETS bit = 0011:ADCAn clock = PCLK/5

ADCA0CG0 register <- 00000100H: converts CGi(ANI8) input signal

ADCA0IOC0 register <- 80000100H
: interrupt (ADCATINT8) occurs when conversion is terminated.

ADCA0TSEL0 register <- 0000H: disable extra trigger

ADCAnCTL2 register <- 0000H
ADCAnRCKm bit=0 : the conversion result limit comparison: disabled

ADCAnLL register <- 0000H : lower limit
ADCAnUL register <- FFC0H : upper limit

ADCA0CTL0 register <- 1f80H
ADCAnOEM4 bit =1 : ADCAnLCR overwrite interrupt : disabled
ADCAnOEM bit =1 : ADCAnDBiCR overwrite interrupt : disabled
ADCAnOEM0 bit =1 : ADCAnCmCR overwrite interrupt : disabled
ADCAnCE bit =1 : enable A/D converter
ADCAnSCTi bit=00 : the number of repetitions of CGi =1

ADCA0TRG0 register <- 01H
ADCAnSTTi bit =1 : software triggered

Back to main(), enters infinite loop, and waits A/D conversion interrupt.

Figure 4.6 initializing A/Dconverter

4.9.7 Interrupt: Controlling motor in career interrupt(every 50msec)

Function pwm_int() modulate PWM in every career interrupt.

Figure 4.7 shows the flowchart of Controlling motor in career interrupt(every 50msec)

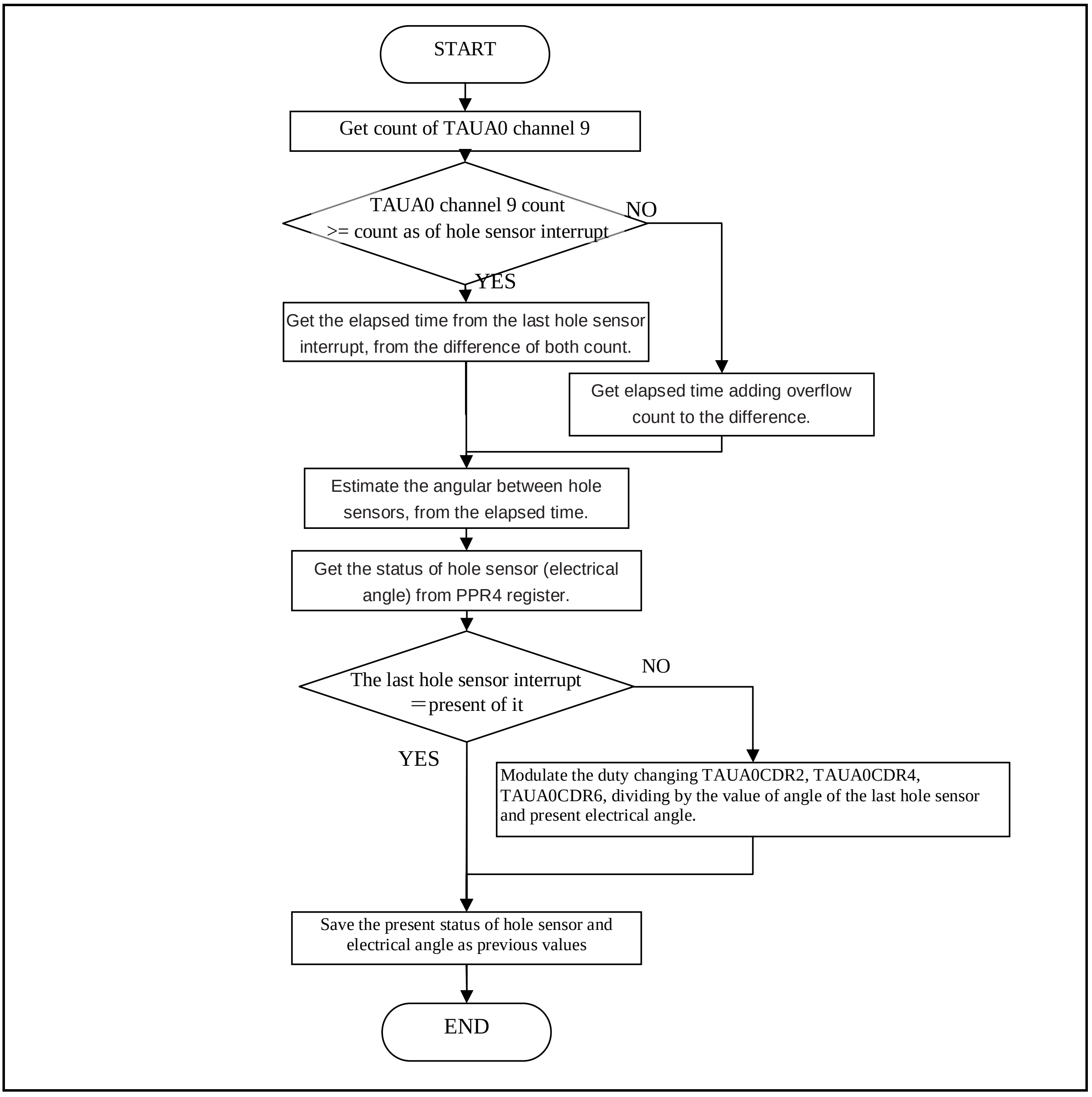


Figure 4.7 Controlling motor in career interrupt(every 50msec)

4.9.8 Interrupt: operation in hole sensor interrupt (every 60 degree in electrical angle)

Function pwm_hs() measures the time in every hole sensor interrupt, to adjust angular velocity.

Figure 4.8 shows the flowchart of operation in hole sensor interrupt.

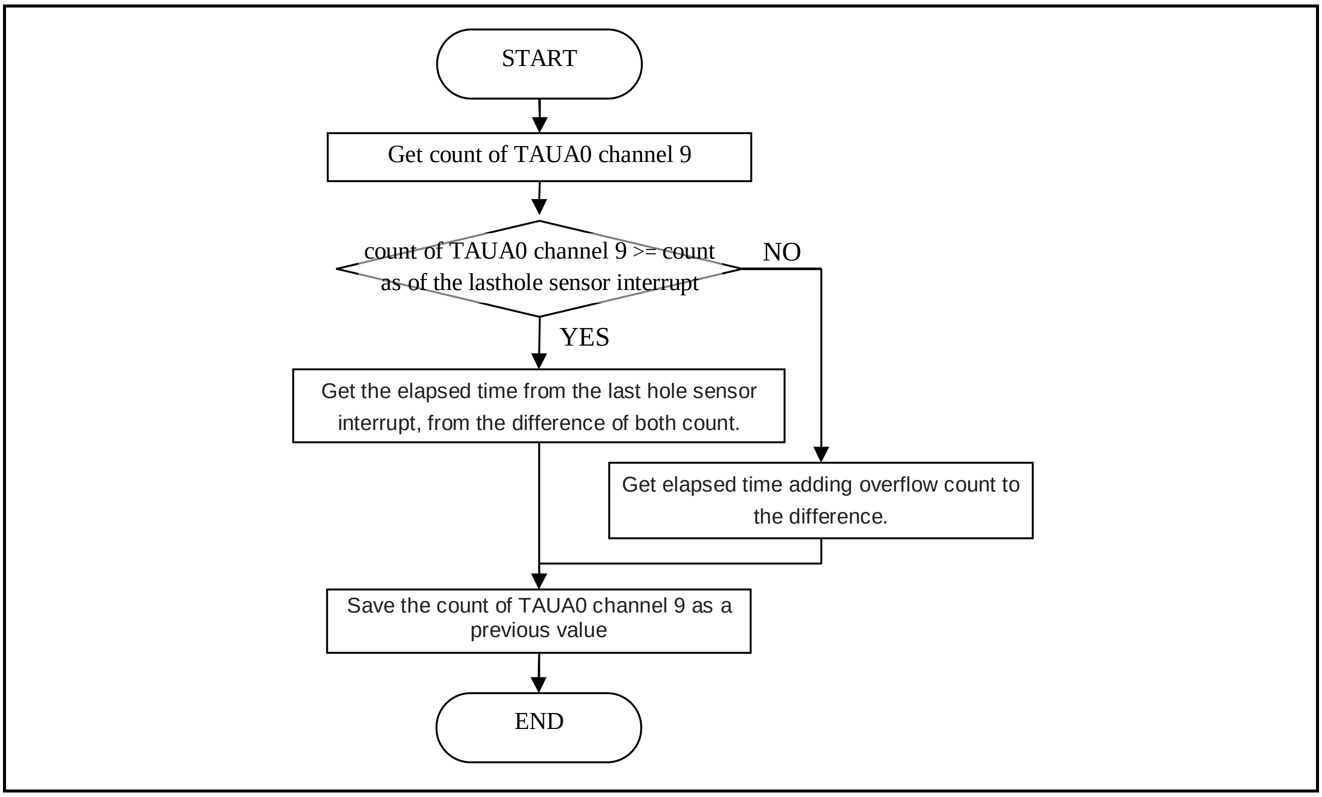


Figure 4.8 operation in hole sensor interrupt

4.9.9 Interrupt: operation in free-running counter interrupt (every 1sec)

Function ovf_int() increments the overflow counter .

Figure 4.9 shows the flowchart of operation in free-running counter interrupt.

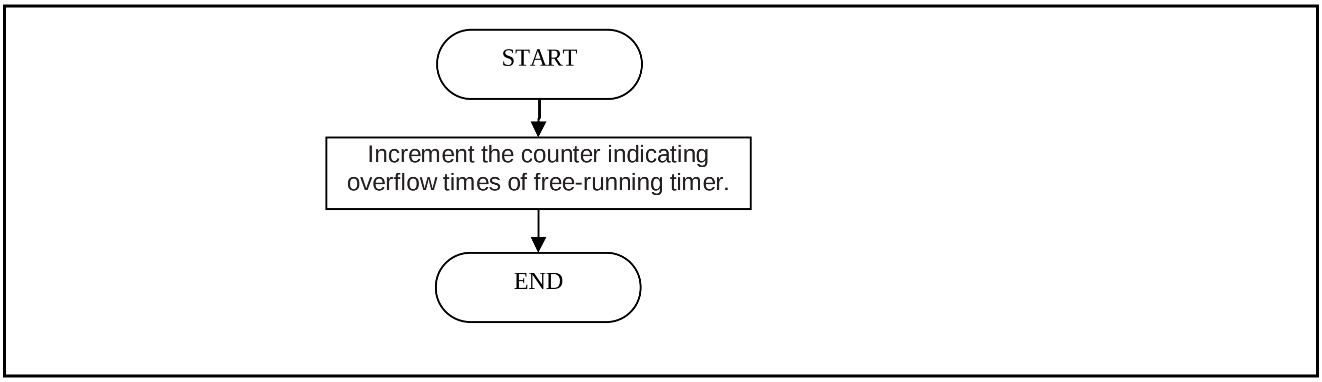


Figure 4.9 operation in free-running counter interrupt

4.9.10 Interrupt: operation in interval timer interrupt (every 1msec)

Function pi_int() calculates a swing angle from the ordered angular velocity (300-2000rpm) derived from VR switch.

Figure 4.10shows the flowchart of operation in interval timer interrupt.

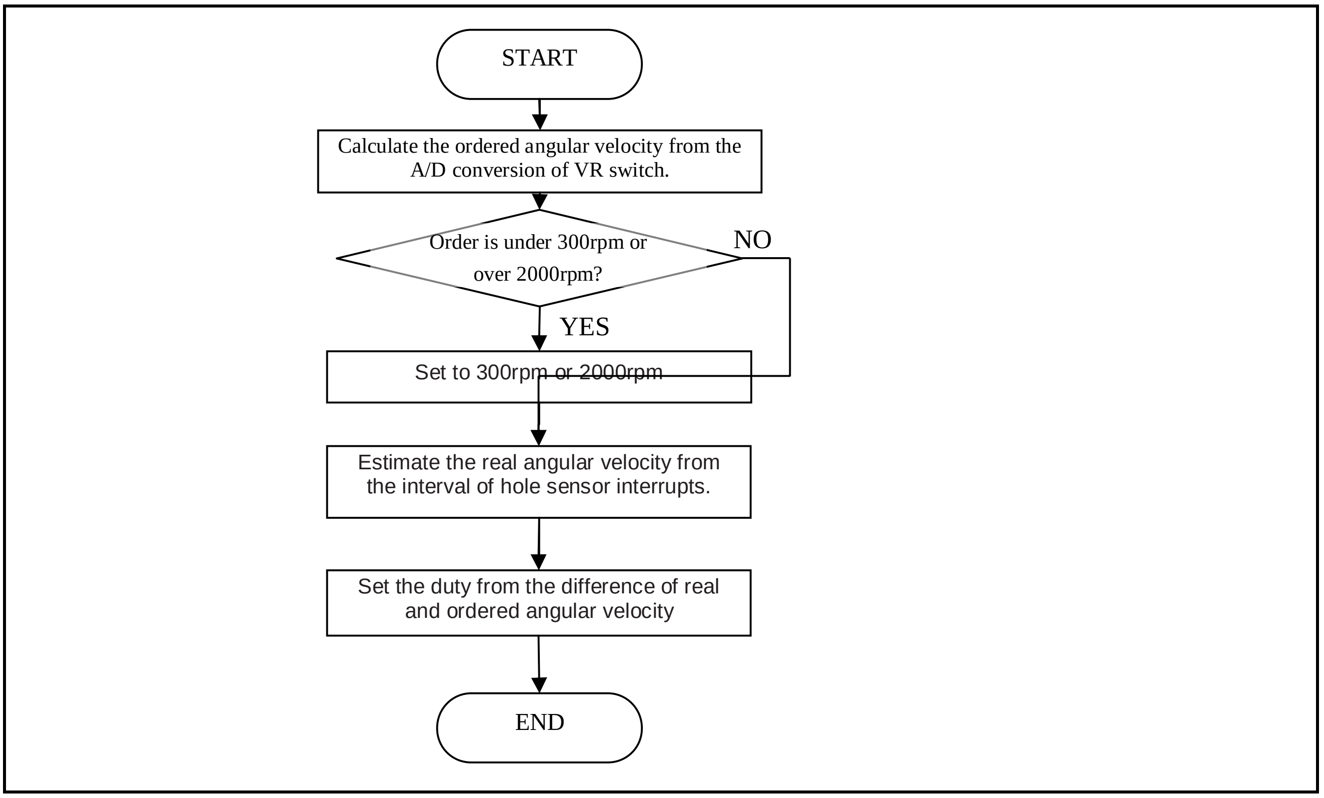


Figure 4.10 operation in interval timer interrupt

5. Sample Code

Sample code can be downloaded from the Renesas Electronics website.

6. Reference Documents

User's Manual: Hardware
V850E2/ML4 User's Manual: Hardware (R01UH0262EJ)
The latest version can be downloaded from the Renesas Electronics website.

Technical Update/Technical News
The latest information can be downloaded from the Renesas Electronics website.

Website and Support

Renesas Electronics website
http://www.renesas.com

Inquiries
http://www.renesas.com/contact/

REVISION HISTORY	V850E2/ML4 Application Note Complementary PWM Output Function

Rev.	**Date**	**Description**	
		Page	**Summary**
1.00	**Jun. 22, 2012**	—	First edition issued

General Precautions in the Handling of MPU/MCU Products

The following usage notes are applicable to all MPU/MCU products from Renesas. For detailed usage notes on the products covered by this manual, refer to the relevant sections of the manual. If the descriptions under General Precautions in the Handling of MPU/MCU Products and in the body of the manual differ from each other, the description in the body of the manual takes precedence.

1. Handling of Unused Pins
   Handle unused pins in accord with the directions given under Handling of Unused Pins in the manual.
   — The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible. Unused pins should be handled as described under Handling of Unused Pins in the manual.
2. Processing at Power-on
   The state of the product is undefined at the moment when power is supplied.
   — The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the moment when power is supplied.
   In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the moment when power is supplied until the reset process is completed.
   In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the moment when power is supplied until the power reaches the level at which resetting has been specified.
3. Prohibition of Access to Reserved Addresses
   Access to reserved addresses is prohibited.
   — The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed.
4. Clock Signals
   After applying a reset, only release the reset line after the operating clock signal has become stable. When switching the clock signal during program execution, wait until the target clock signal has stabilized.
   — When the clock signal is generated with an external resonator (or from an external oscillator) during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Moreover, when switching to a clock signal produced with an external resonator (or by an external oscillator) while program execution is in progress, wait until the target clock signal is stable.
5. Differences between Products
   Before changing from one product to another, i.e. to one with a different type number, confirm that the change will not lead to problems.
   The characteristics of MPU/MCU in the same group but having different type numbers may differ because of the differences in internal memory capacity and layout pattern. When changing to products of different type numbers, implement a system-evaluation test for each of the products.

Notice

1. [illegible]
2. [illegible]
3. [illegible]
4. [illegible]
5. [illegible]
6. [illegible]
7. [illegible]
8. [illegible]
9. [illegible]
10. [illegible]
11. [illegible]
12. [illegible]

(Note 1) [illegible]

(Note 2) [illegible]

SALES OFFICES **Renesas Electronics Corporation** http://www.renesas.com

[illegible]

[illegible]