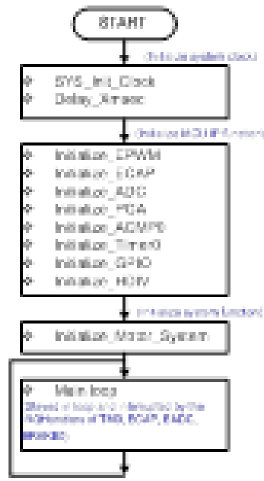
Introduction to Basic Motor Control System by NM1120

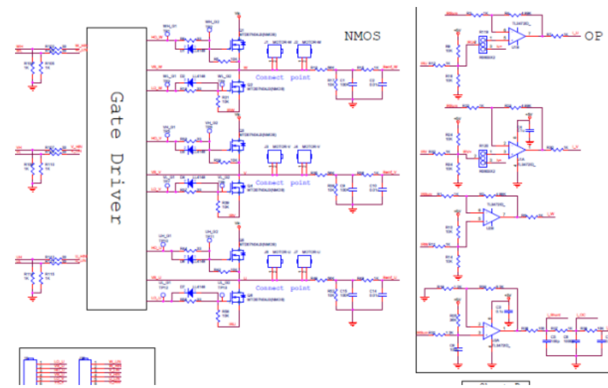
2019/May

Demo Purpose

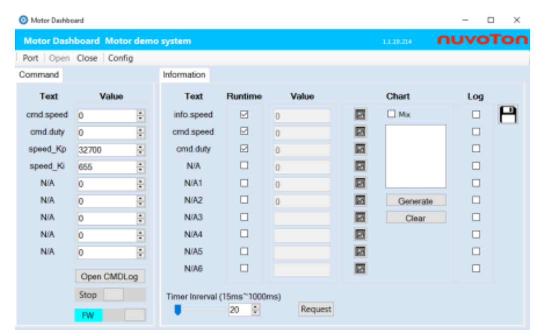
- **Easily & Rapidly Getting Start for Motor Application**
  - Provide demo system for refer-design

- **Motor Drive System**
  - Elementary S/W flow
  - Refer H/W circuit

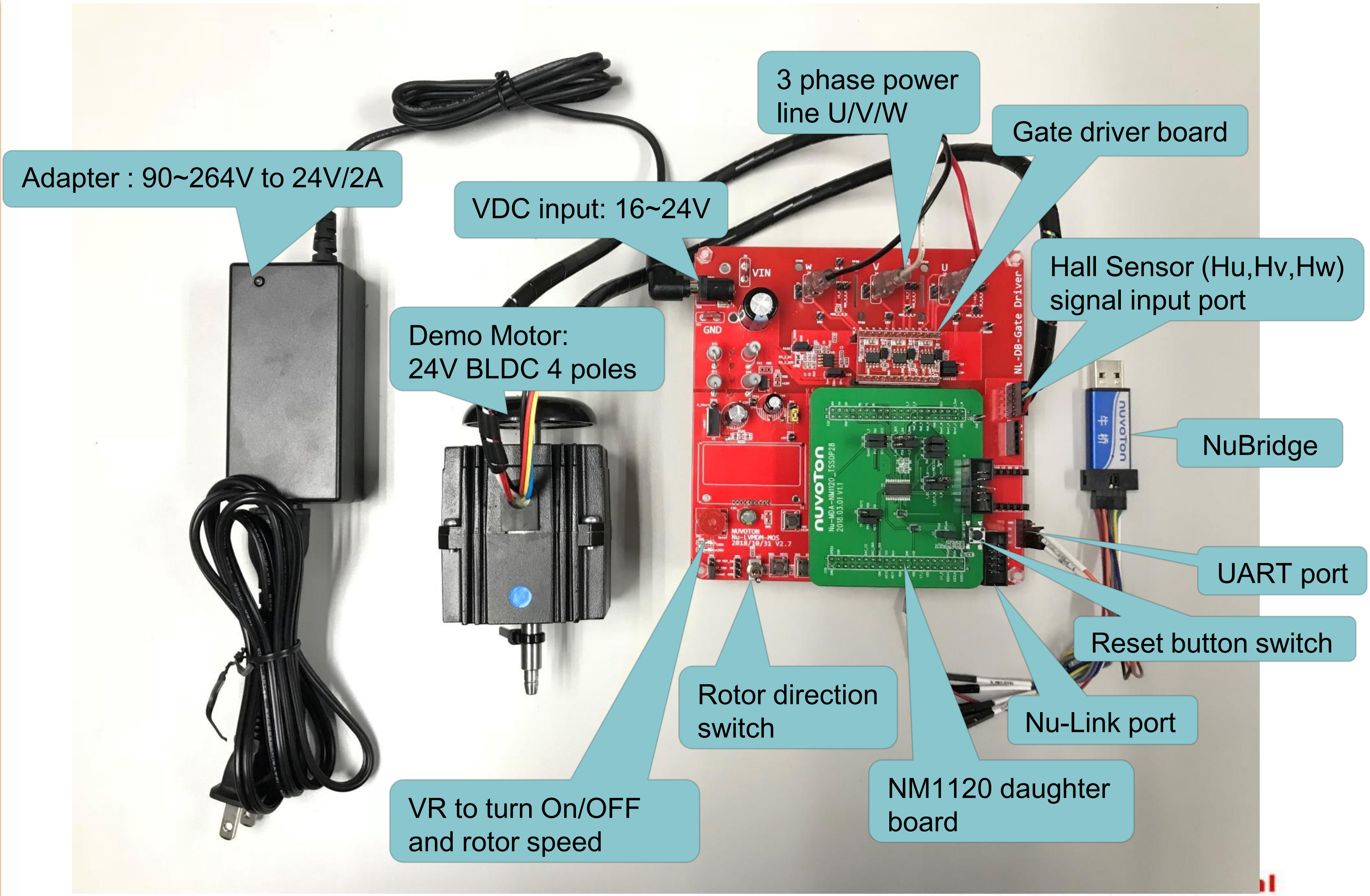




- **Friendly GUI Interface Linking PC and System Board through UART(COM port)**
  - User can easily demo motor behavior



Demo System



3 phase power line U/V/W
Gate driver board
Adapter : 90~264V to 24V/2A
VDC input: 16~24V
Hall Sensor (Hu,Hv,Hw) signal input port
Demo Motor: 24V BLDC 4 poles
NuBridge
UART port
Reset button switch
Rotor direction switch
Nu-Link port
NM1120 daughter board
VR to turn On/OFF and rotor speed

Overview

- Demo how to use NM1120 to implement a basic motor drive system with
  - Hall + FOC + 2-shunt R
  - Hall + FOC + 1-shunt R
  - Hall + Six step square wave control
    - Duty open loop
    - Speed close loop
- Customer may refer this demo system to develop own specific system.

Demo Function

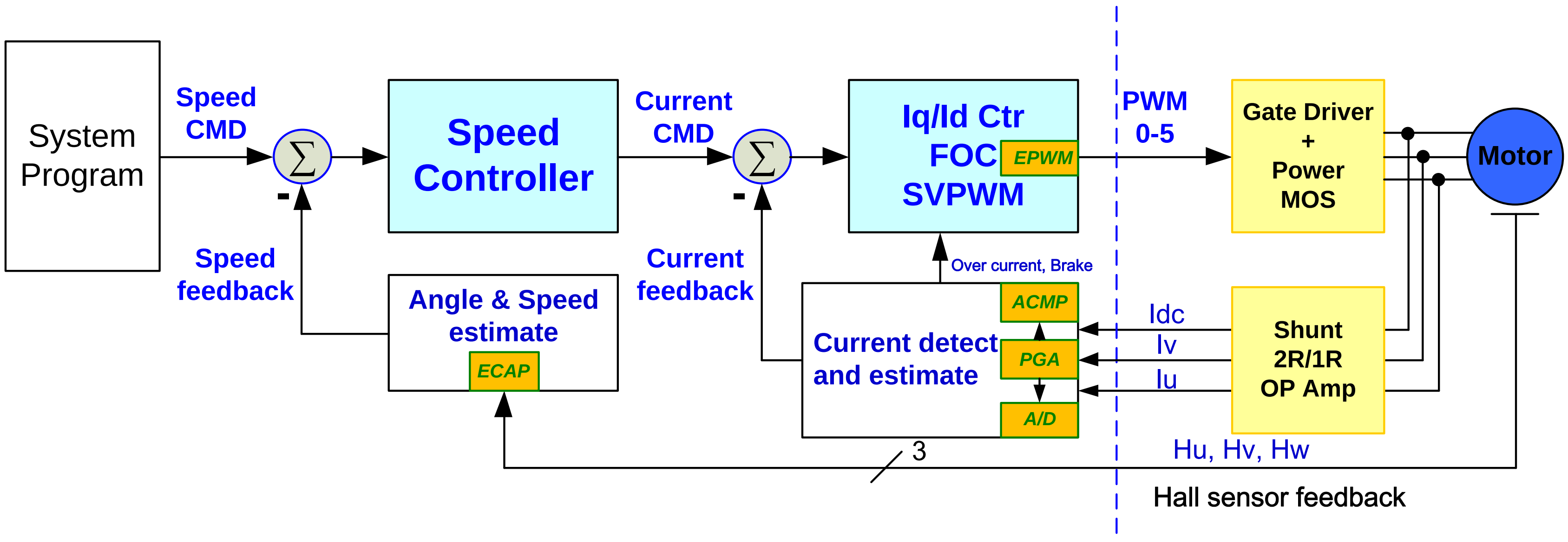
- ✓ **Demo NM1120 MCU basic setting**
  - Initialize MCU
  - Timer
  - Interrupt
  - ECAP (Enhance Capture Timer/Counter)
  - Complementary PWM0~5 setting
  - 12-bit 2-S/H ADC
  - PGA (Programmable Gain Amplifier)
  - ACMP (Analog Comparator)
- ✓ **Basic PI control** (Proportional and Integral control) for speed and current control
- ✓ **Basic FOC + SVPWM (Library)**

Demo Function

- ✓ **6-Step square wave control**
- ✓ **Basic motor speed estimation by ECAP**
- ✓ **System control mode**
  - VSP control mode : External VR to turn on/off the motor and adjust speed
  - UART control mode : Set command from PC to MCU through UART protocol.
- ✓ **Control mode selection :**
  - VSP control mode : Rotate VR to make VSP initial voltage < 2.5V before power on .
  - UART control mode : Rotate VR to make VSP initial voltage > 2.5V before power on .

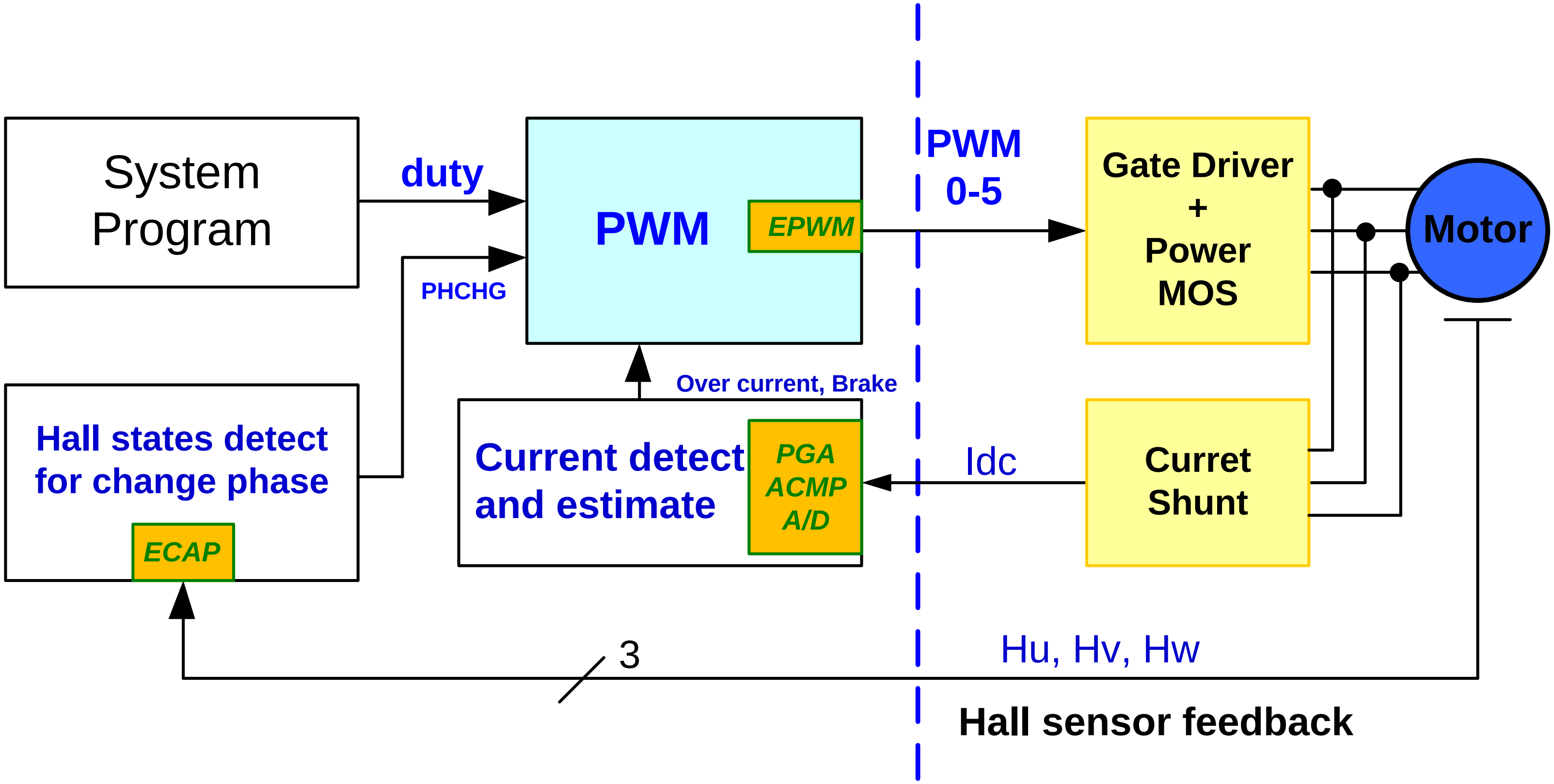
Speed Control by FOC

- Demo System : Hall + FOC
- Control mode : 2R / 1R (R : current shunt resistance)



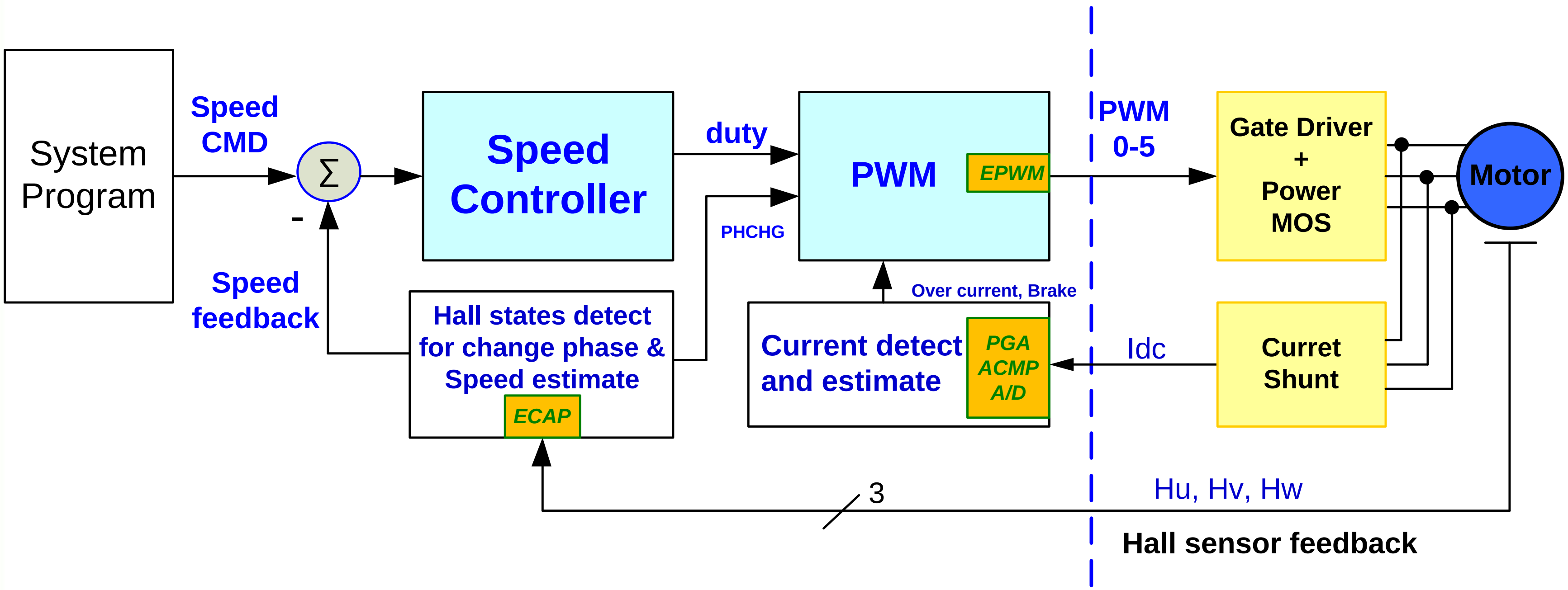
PWM Duty Control by 6-Step

- Demo System : Hall + Six Step square wave control
- Control mode : Duty open loop



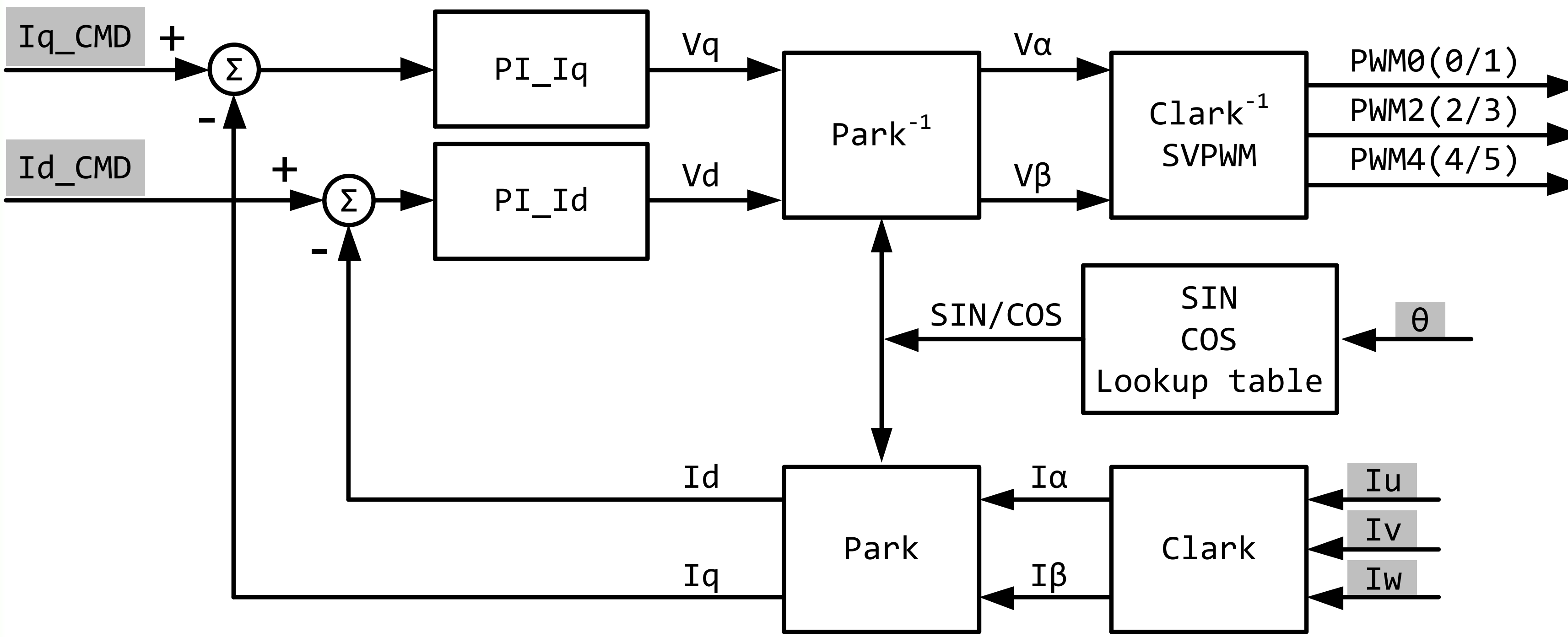
Speed Control by 6-Step

- Demo System : Hall + Six Step square wave control
- Control mode : Speed close loop

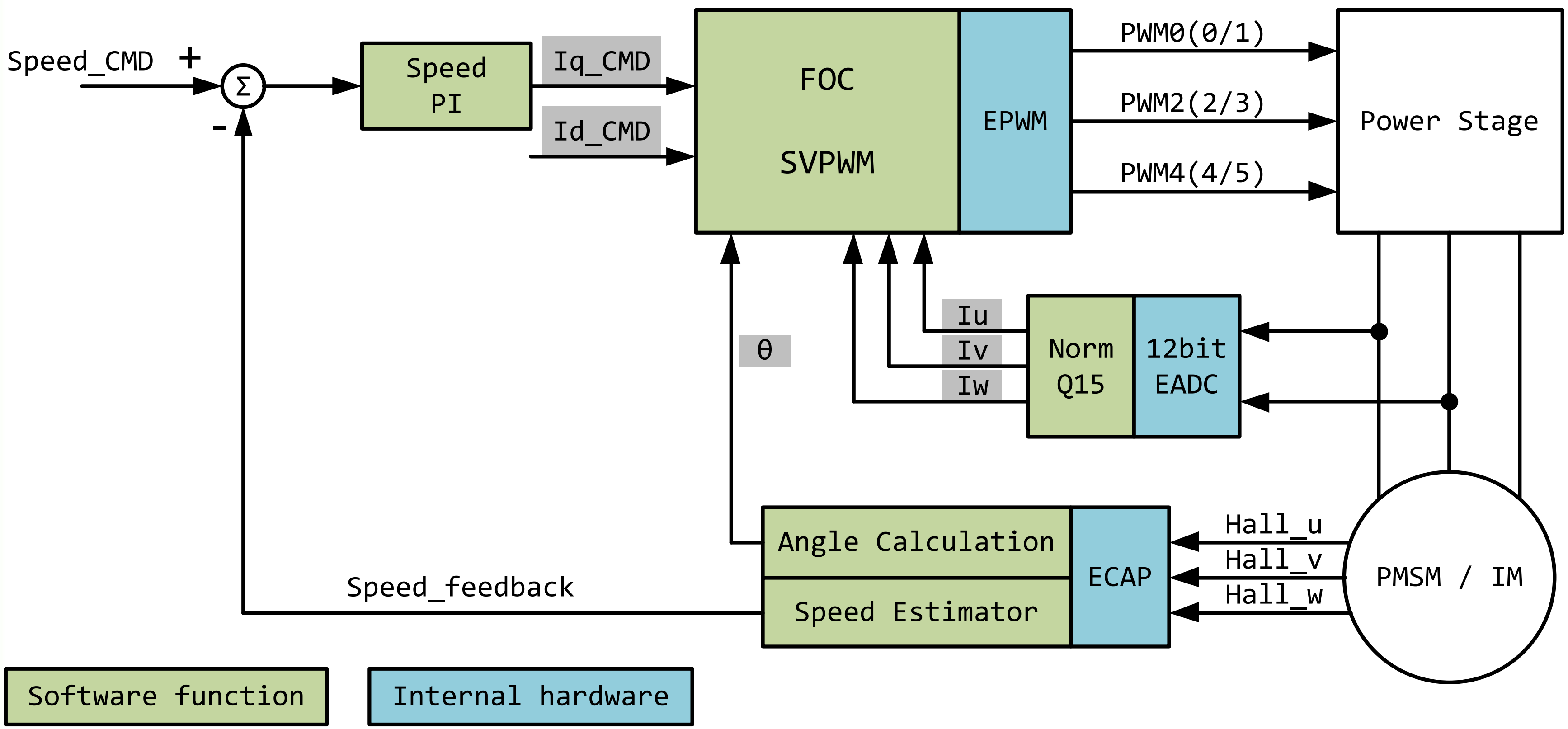


FOC by Software Library

- FOC: Field Oriented Control, 磁場導向控制
- Vector Control: 向量控制, 矢量控制
- a-b-c stationary frame
- α-β stationary frame
- d-q rotating frame



FOC control block

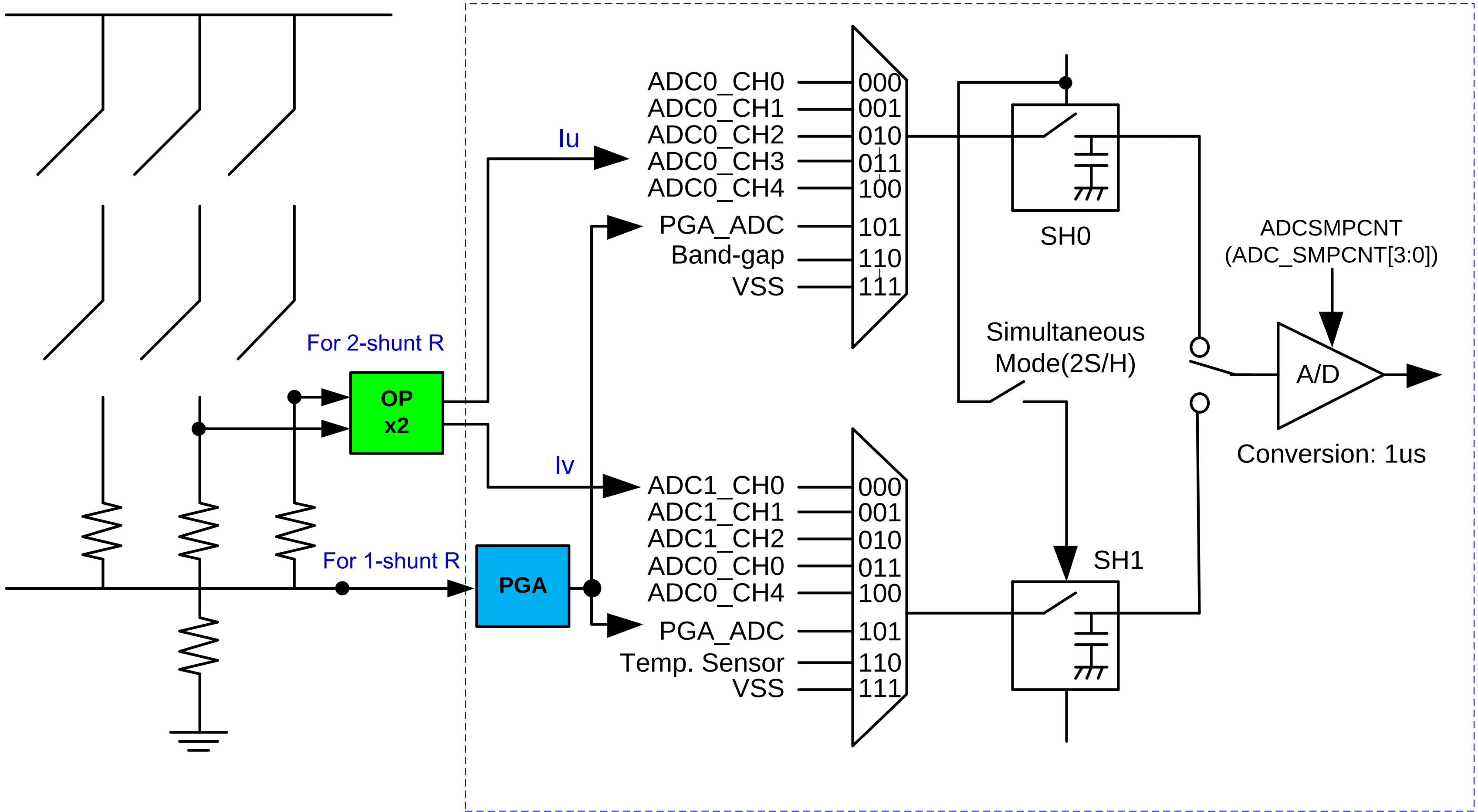


FOC Function Call

- void **Iuvw_to_Idq**(AMotor* Motor, int32 sin, int32 cos)
  - Do Clark Transfer: Transfer Iu/v/w to Ialfa/beta
  - Do Park Transfer: Transfer Ialfa/beta to Id/q
- void **Vdq_to_SVPWM**(AMotor* Motor, EPWM_T* epwm, int32 pwm_full_scale, int32 pwm_max_duty, int32 sin, int32 cos)
  - Do Inv_Park Transfer: Transfer Vd/q to Valfa/beta
  - Do Modified Inv_Clark Transfer: Transfer Valfa/Vbeta to Vrefx/y/z
  - Do SVPWM calculation to produce Duty0/2/4
  - Update the duty to EPWM->CMPDAT[0/2/4]

Phase Current Detection by ADC

- 2-shunt R : Iu / Iv from external two OP
- 1-shunt R : Iu / Iv / Iw through internal PGA



Six Step square wave control

- Sequentially output PWM according with Hall-State.
- Do phase change by setting EPWM Phase Change Register (EPWM_PHCHG).

Forward

STAGE FW	HALL STATES	HW PB2	HV PB1	HU PB0	Exciting phase	Gate signal					
						U_H	U_L	V_H	V_L	W_H	W_L
1	5	1	0	1	u→w	PWM	0	0	0	0	1
2	1	0	0	1	v→w	0	0	PWM	0	0	1
3	3	0	1	1	v→u	0	1	PWM	0	0	0
4	2	0	1	0	w→u	0	1	0	0	PWM	0
5	6	1	1	0	w→v	0	0	0	1	PWM	0
6	4	1	0	0	u→v	PWM	0	0	1	0	0

Reverse

STAGE RW	HALL STATES	HW PB2	HV PB1	HU PB0	Exciting phase	Gate signal					
						U_H	U_L	V_H	V_L	W_H	W_L
1	3	0	1	1	u→v	PWM	0	0	1	0	0
2	1	0	0	1	w→v	0	0	0	1	PWM	0
3	5	1	0	1	w→u	0	1	0	0	PWM	0
4	4	1	0	0	v→u	0	1	PWM	0	0	0
5	6	1	1	0	v→w	0	0	PWM	0	0	1
6	2	0	1	0	u→w	PWM	0	0	0	0	1

Six Step square wave Function Call

- void SIX_STEP_CHANGE_PHASE_FW(void)
  - Do phase change function for rotor forward
  - Using EPWM Phase Change Register (EPWM_PHCHG) to change phase
- void SIX_STEP_CHANGE_PHASE_RW(void)
  - Do phase change function for rotor reverse
  - Using EPWM Phase Change Register (EPWM_PHCHG) to change phase

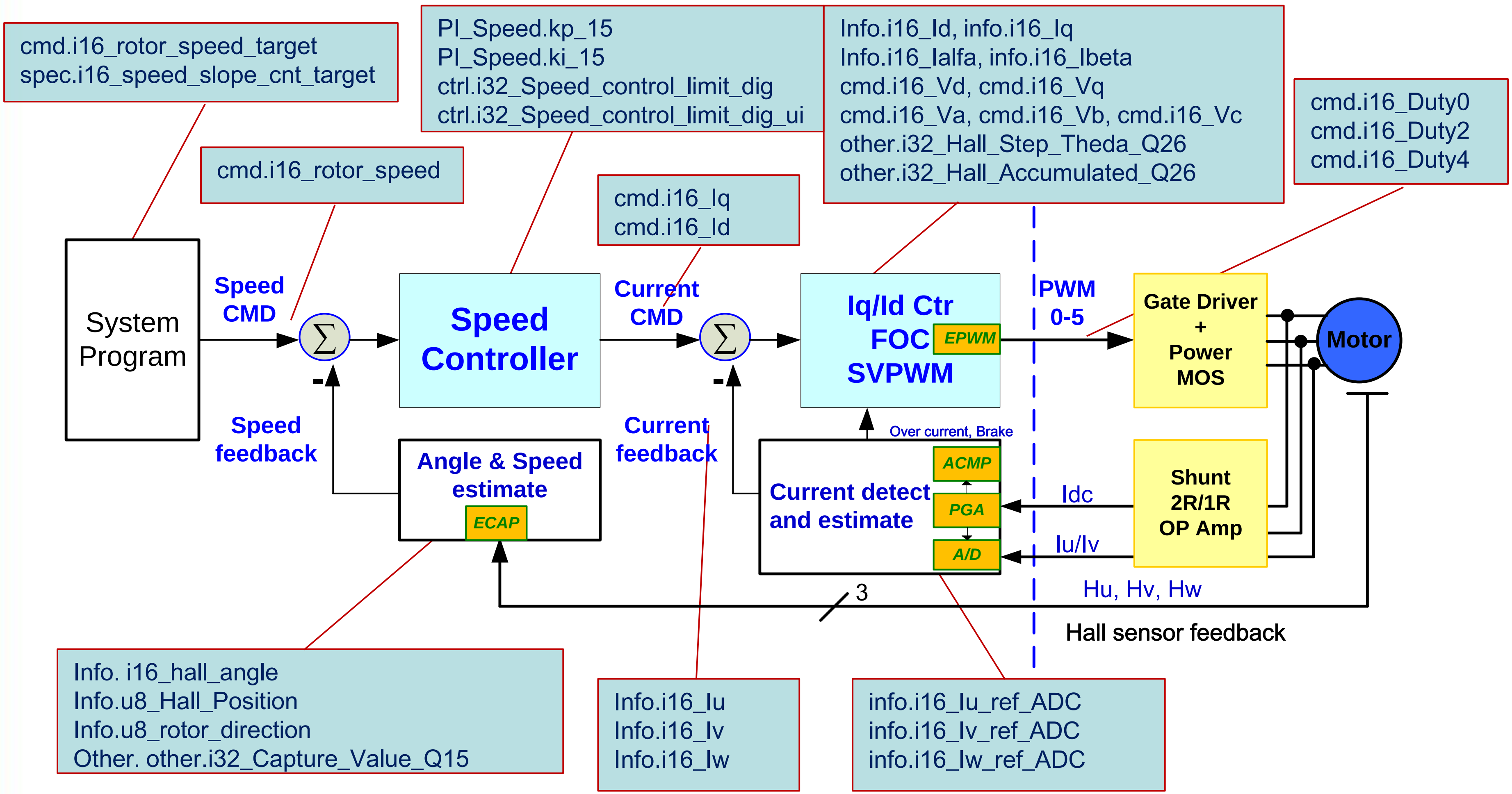
Motor Variable Structure

```
typedef struct tag_Motor
 {
     AMotorSpec          spec;    馬達系統基本規格, ex: PWM 頻率
     AMotorInfo          info;    即時的data, ex: Iu, Iv, Iw, rotor_speed
     AMotorCommand       cmd;     馬達驅動的命令, ex:轉子速度命令rpm
     AMotorController    ctrl;    控制器的最大限制值
     AMotorSym           sym;     系統資訊, ex: 過流錯誤回報資料
     AMotorOther         other;   一些系統程式執行時需要的變數或flag
 } AMotor;
```

PI Controller Variable Structure

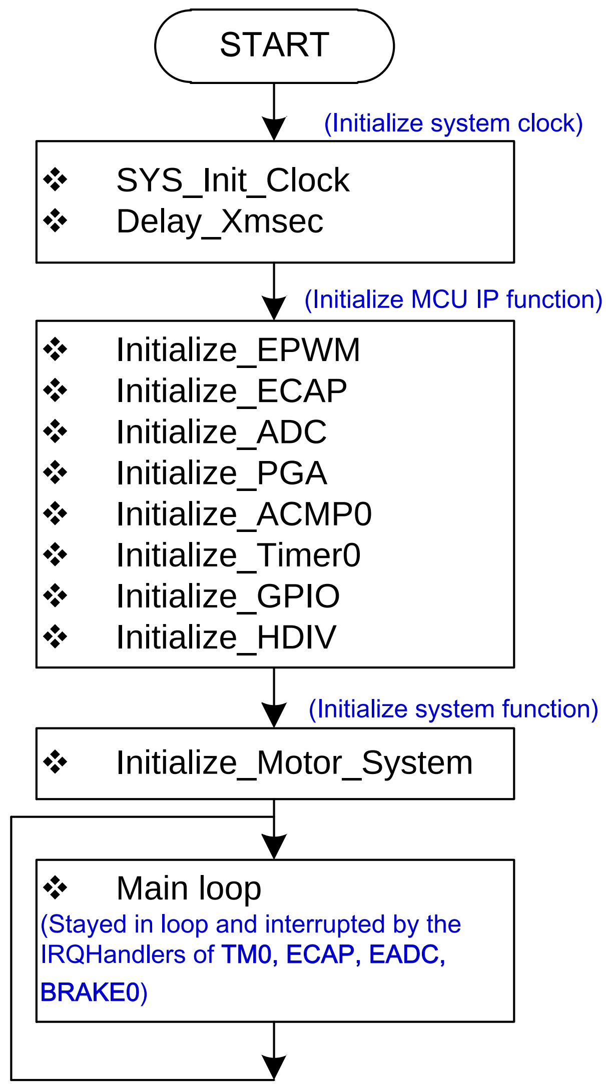
```
typedef struct
 {
        INT32   e1;              Error = cmd – info, Q15 format
        INT32   ui_31;           Accumulated integrator, Q31 format
        INT32   kp_15,ki_15;     Parameters of Kp & Ki, Q31 format
 }PICs;
```

Key Variables



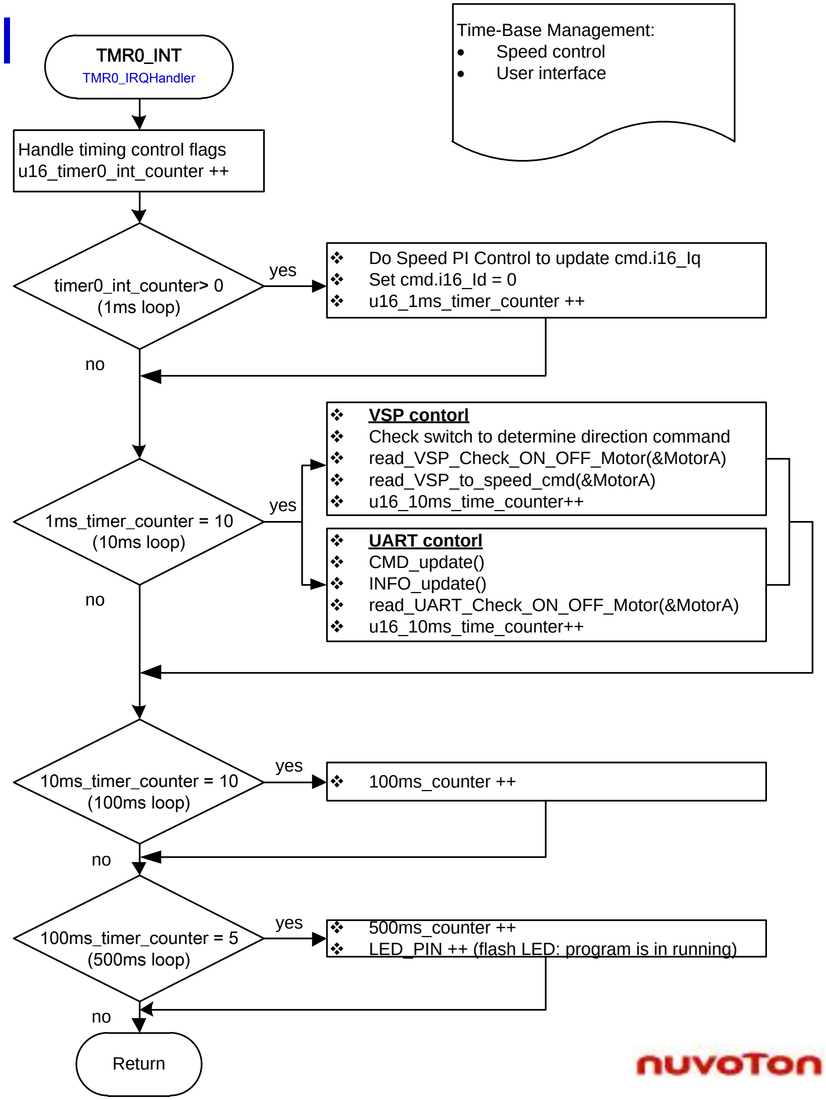
Flow Chart of main.c

▶ Main program to initialize MCU and system



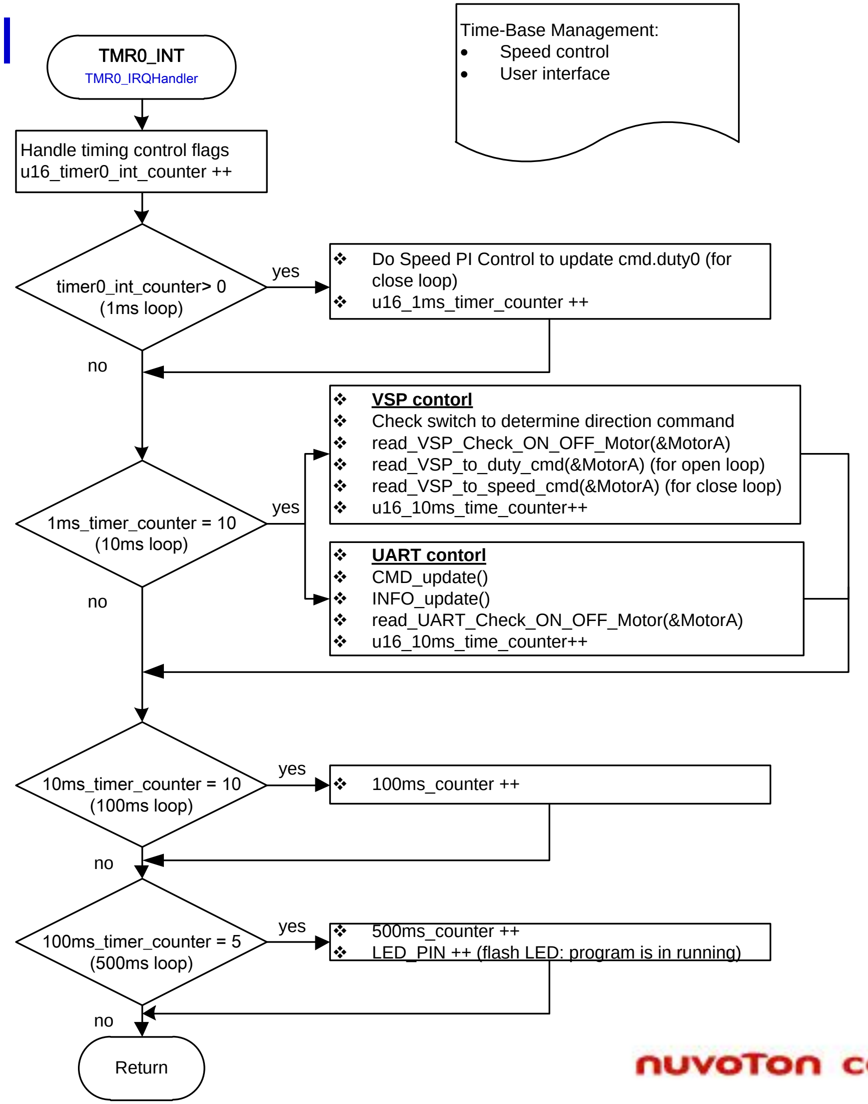
Flow Chart of TMR0_INT (for FOC)

- Timer0 INT for system timer handles the user interface and speed control



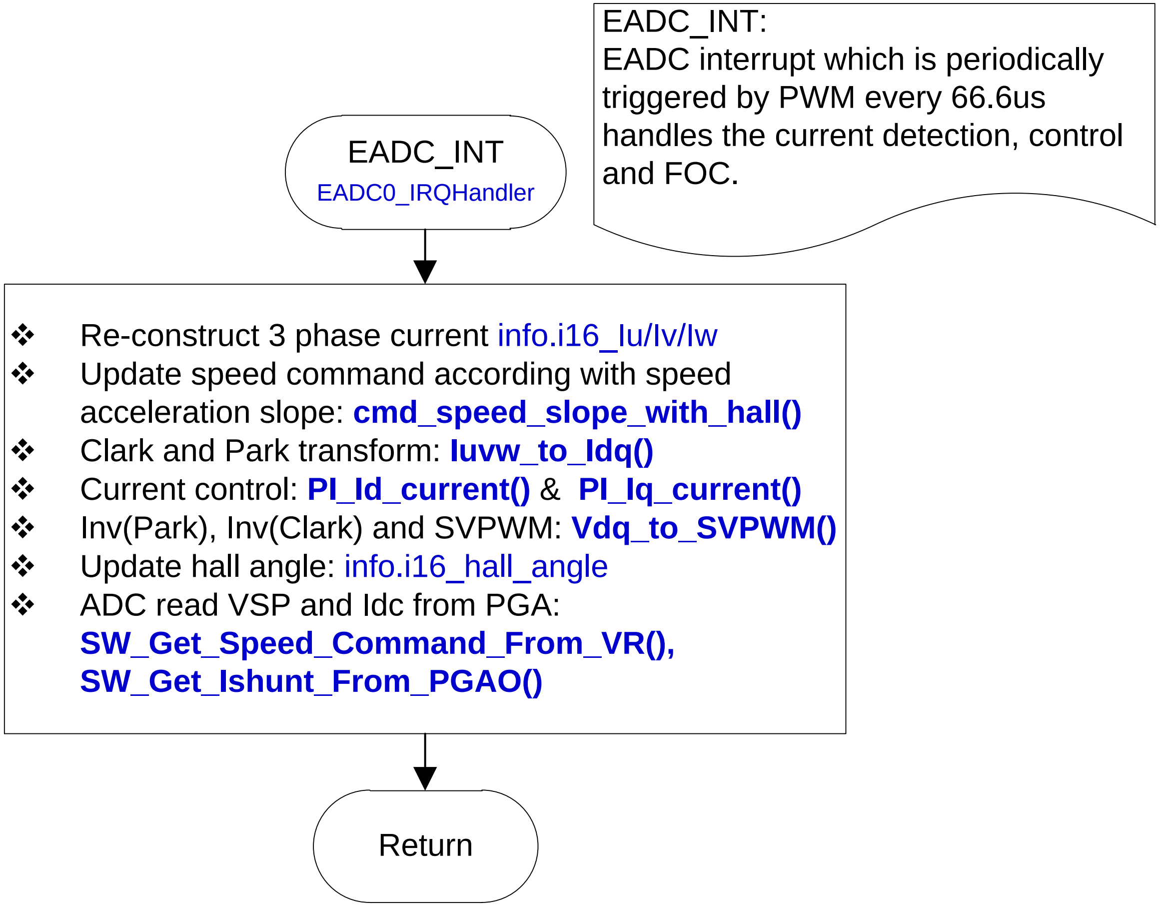
Flow Chart of TMR0_INT (for Six step)

- Timer0 INT for system timer handles the user interface and speed control



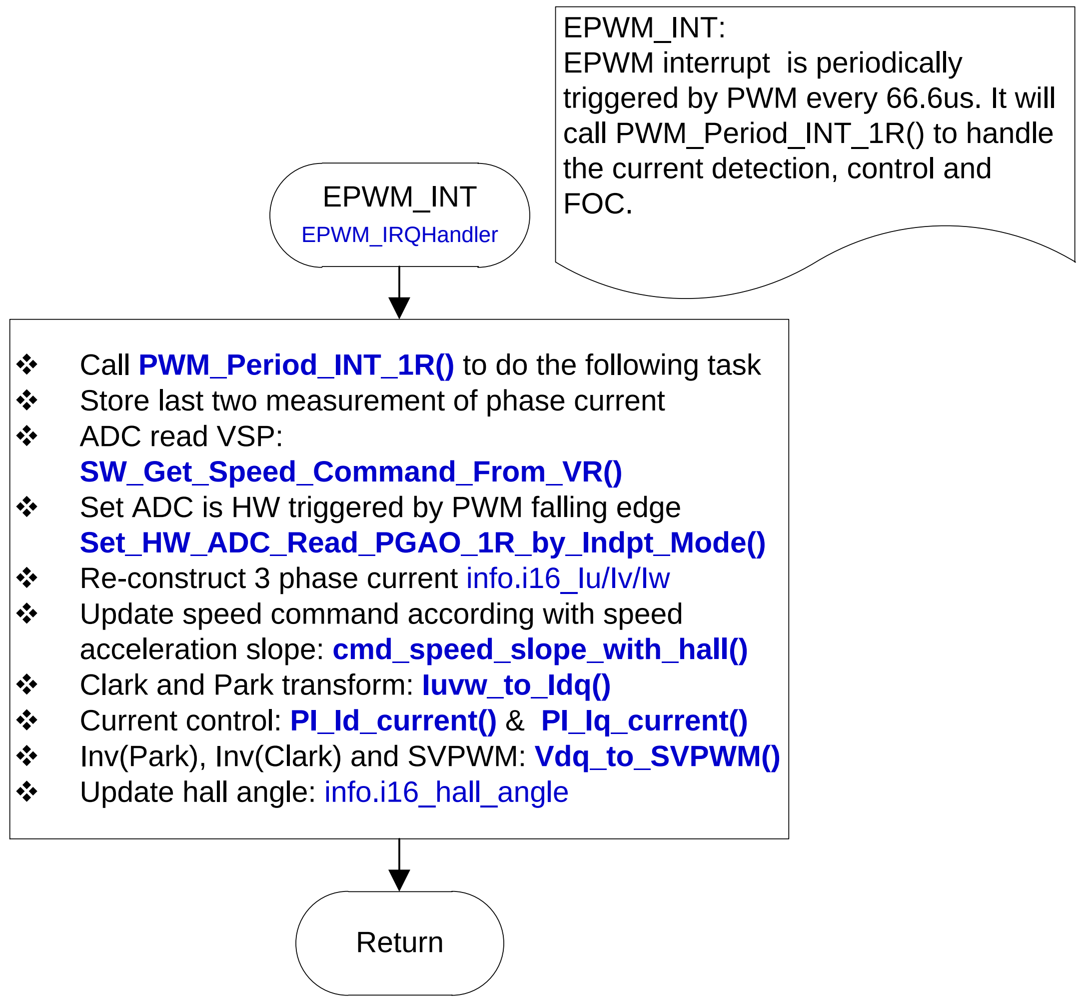
Flow Chart of EADC_INT (for 2R)

► EADC INT: for current control and FOC



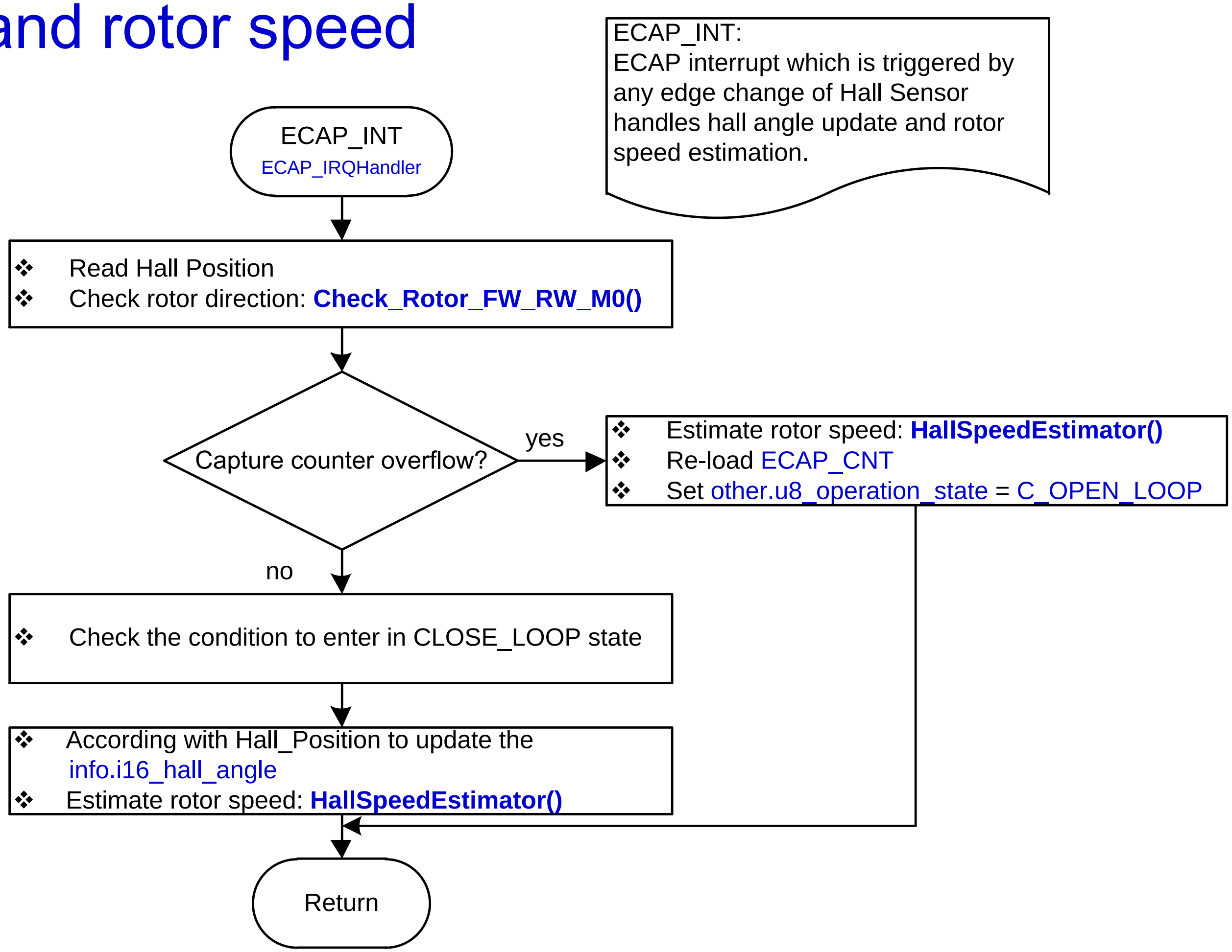
Flow Chart of EPWM_INT (for 1R)

- EPWM INT: for current control and FOC (1R)



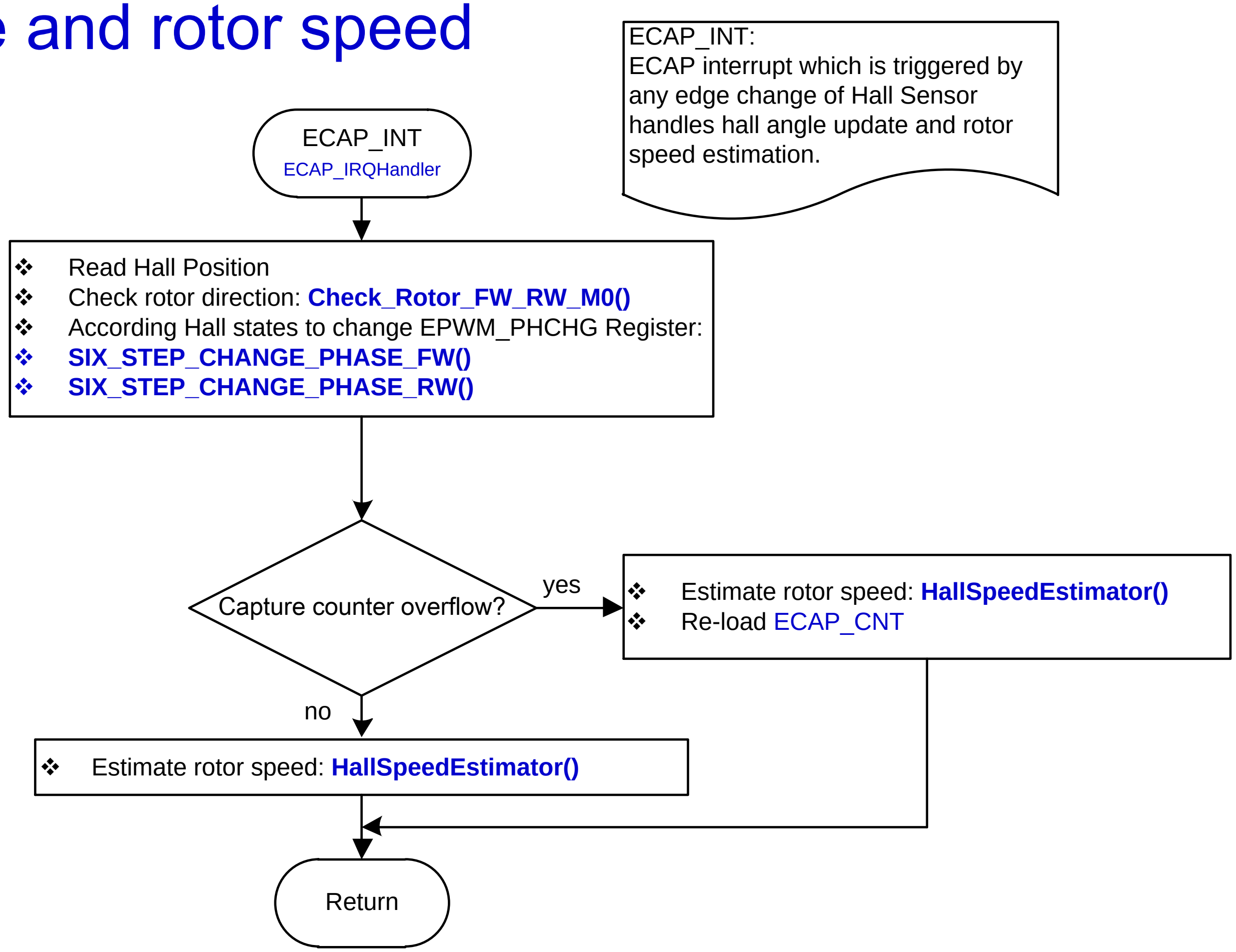
Flow Chart of ECAP_INT(for FOC)

- ECAP INT for capturing hall sensor signals to update the hall angle and rotor speed



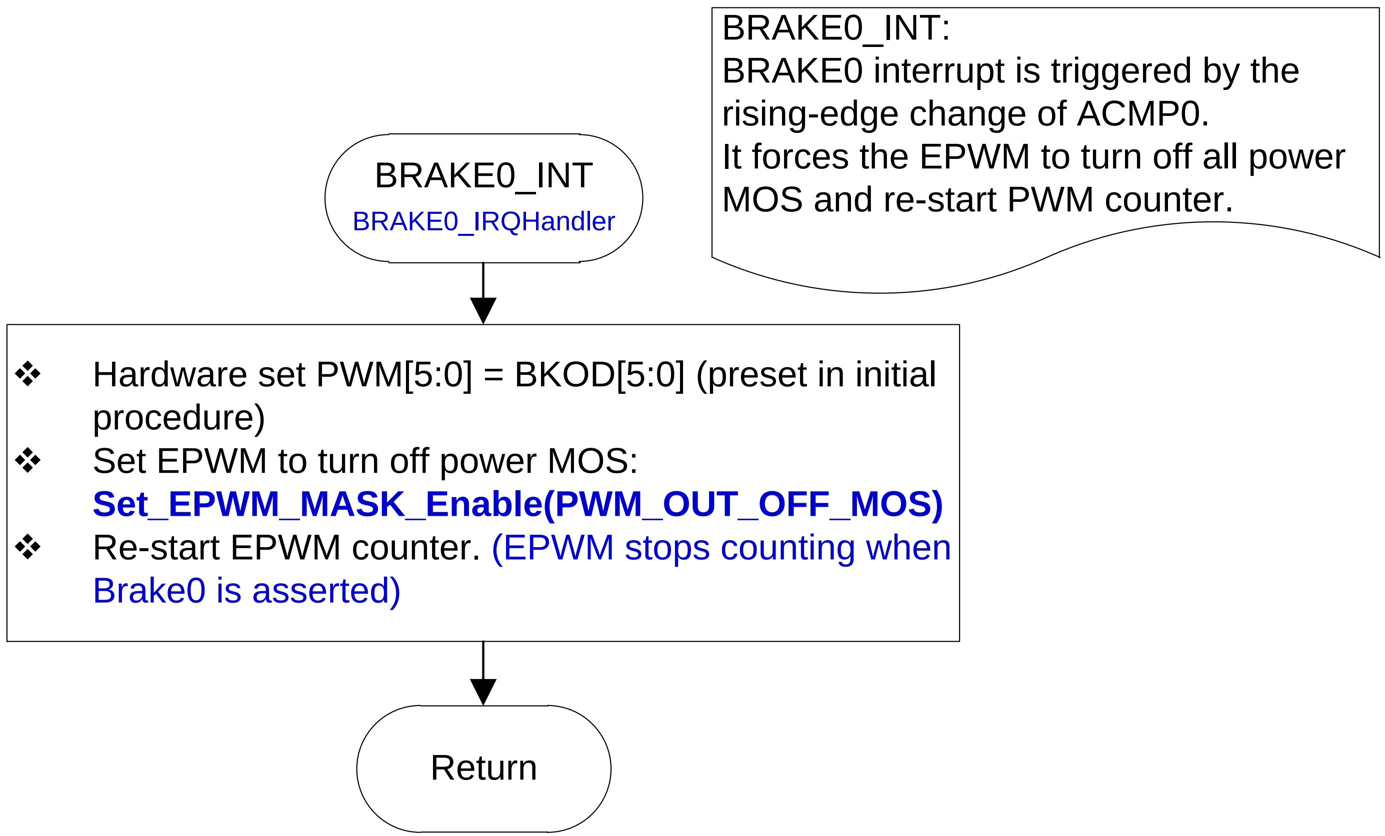
Flow Chart of ECAP_INT(for Six step)

- ECAP INT for capturing hall sensor signals to update the hall angle and rotor speed



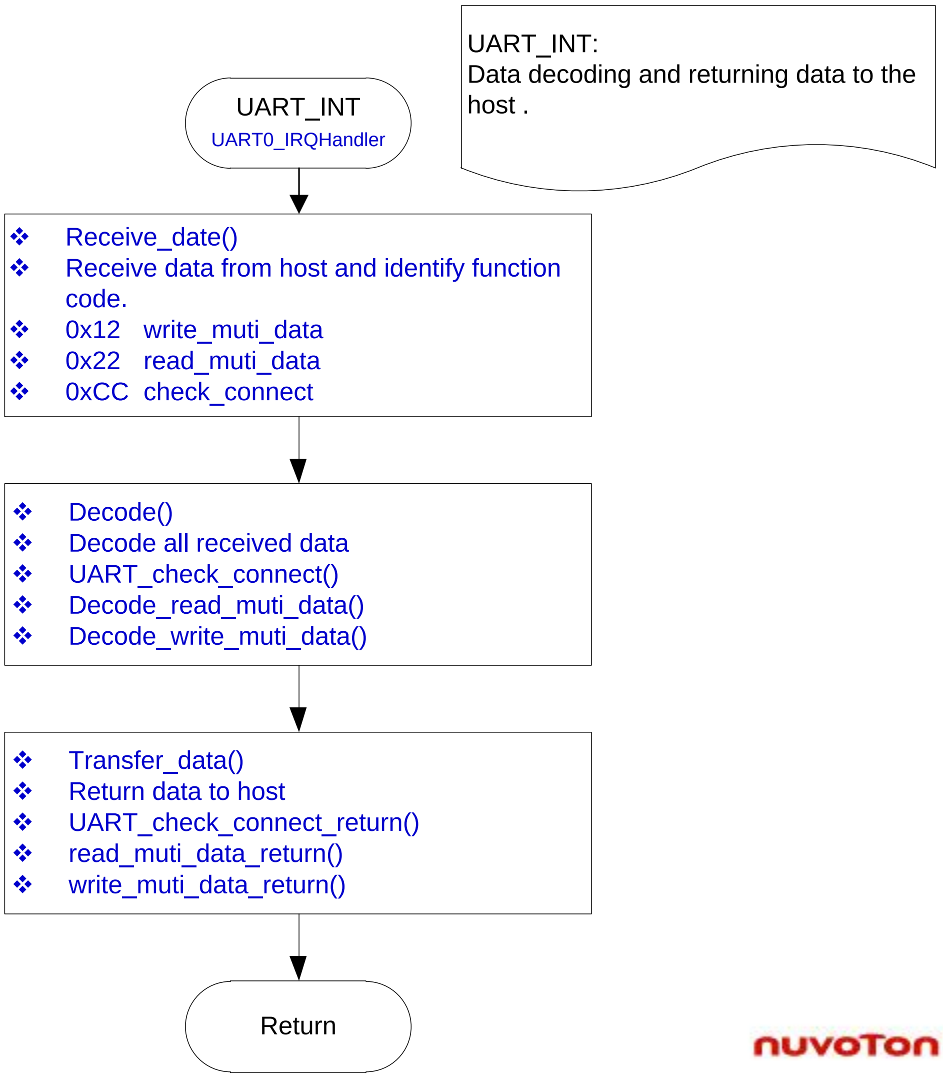
Flow Chart of BRAKE0_INT

- BRAKE0 INT to STOP PWM and turn off MOS when over current

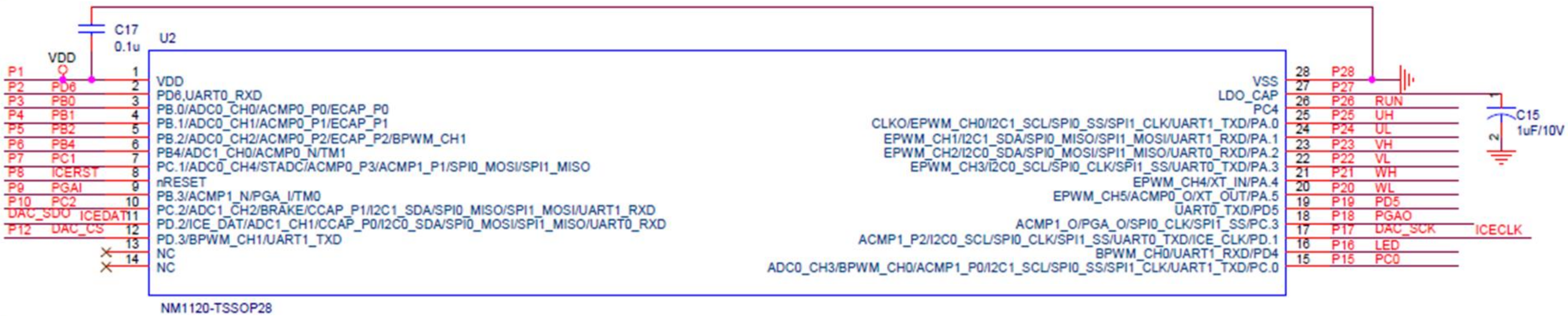


Flow Chart of UART_INT

- UART INT for data decoding and returning data to the host

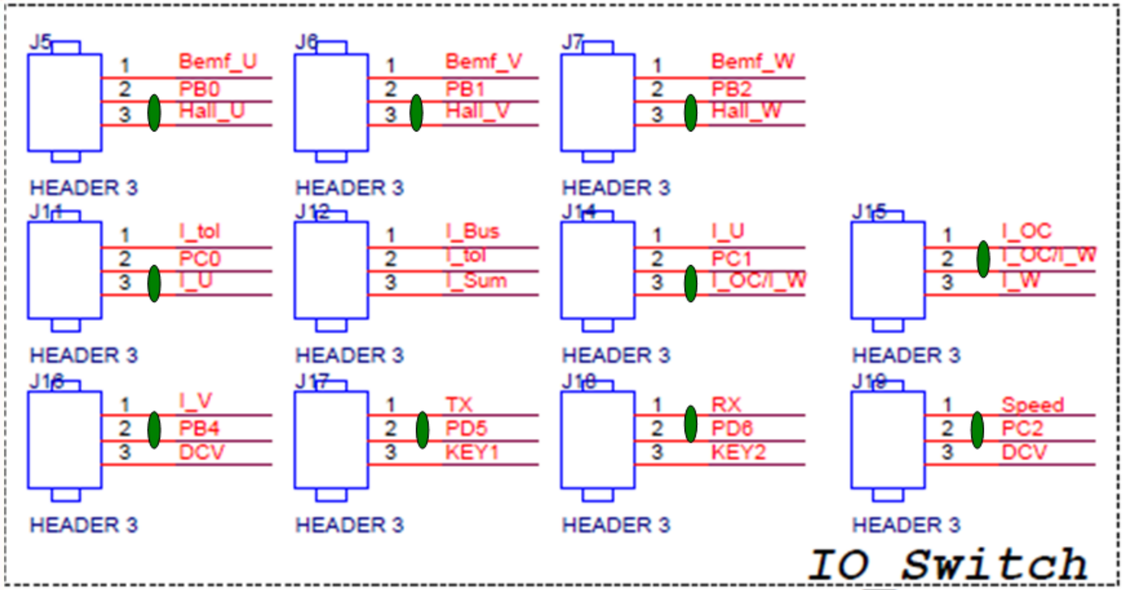


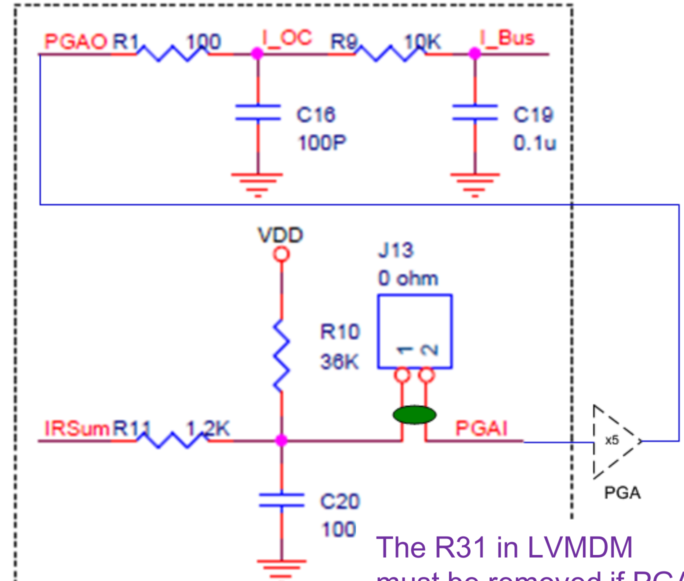
Pin Configure vs Circuit (I)



- ✓ **PWM output (6 pin) by EPWM**
  - (PWM5~PWM0) = (PA5~PA0)
- ✓ **Direction command from RUN switch (1 pin) by GPIO**
  - PC4: read the state of RUN switch
- ✓ **Light LED (1 pin) by GPIO**
  - PD4: control LED on/off
- ✓ **UART port (2 pin) by USCI-UART0**
  - UART0_RXD = PD6
  - UART0_TXD = PD5
- ✓ **Nu-Link ICE port by SWD**
  - ICE_DAT = PD2
  - ICE_CLK = PD1
- ✓ **(Optional) DAC module by SPI0**
  - SPI0_MOSI = PD2
  - SPI0_CLK = PD1
  - CS = PD3 (GPIO mode)

Pin Configure vs Circuit (II)





The R31 in LVMDM must be removed if PGA is used

- ✓ **Hall sensor (3 pins) by ECAP**
  - (Hw, Hv, Hu) = (PB2, PB1, PB0) = (ECAP2, ECAP1, ECAP0)
- ✓ **Phase current feedback (2 pins) by ADC**
  - Iu: ADC0_CH3 = PC0
  - Iv: ADC1_CH0 = PB4
- ✓ **Total current amplify(2 pin) by PGA**
  - IRSum = PGA_I = PB3
  - Set PGA gain = 5
  - PGA_O = PC3 to I_OC through RC filter
- ✓ **Over current detect (1 pin) by ACMP0**
  - I_OC = ACMP0_P3 = PC1
- ✓ **Speed command from VR (1 pin) by ADC**
  - Speed = ADC1_CH2 = PC2

Pin Configure in Demo System (I)

- ✓ **Hall sensor (3 pins) by ECAP**
  - (Hw, Hv, Hu) = (PB2, PB1, PB0) = (ECAP2, ECAP1, ECAP0)
- ✓ **Phase current feedback (2 pins) by ADC (for 2-shunt R)**
  - Iu: ADC0_CH3 = PC0
  - Iv: ADC1_CH0 = PB4
- ✓ **Total current amplify (2 pin) by PGA**
  - IRSum = PGA_I = PB3
  - Set PGA gain = 5
  - PGA_O = PC3 →RC filter → I_OC = ACMP0_P3
  - Internally PGA_O to ADC for phase current measurement (1-shunt R)
- ✓ **Over current detect (1 pin) by ACMP0**
  - I_OC = ACMP0_P3 = PC1
- ✓ **Speed command from VR (1 pin) by ADC**
  - Speed = ADC1_CH2 = PC2

Pin Configure in Demo System (II)

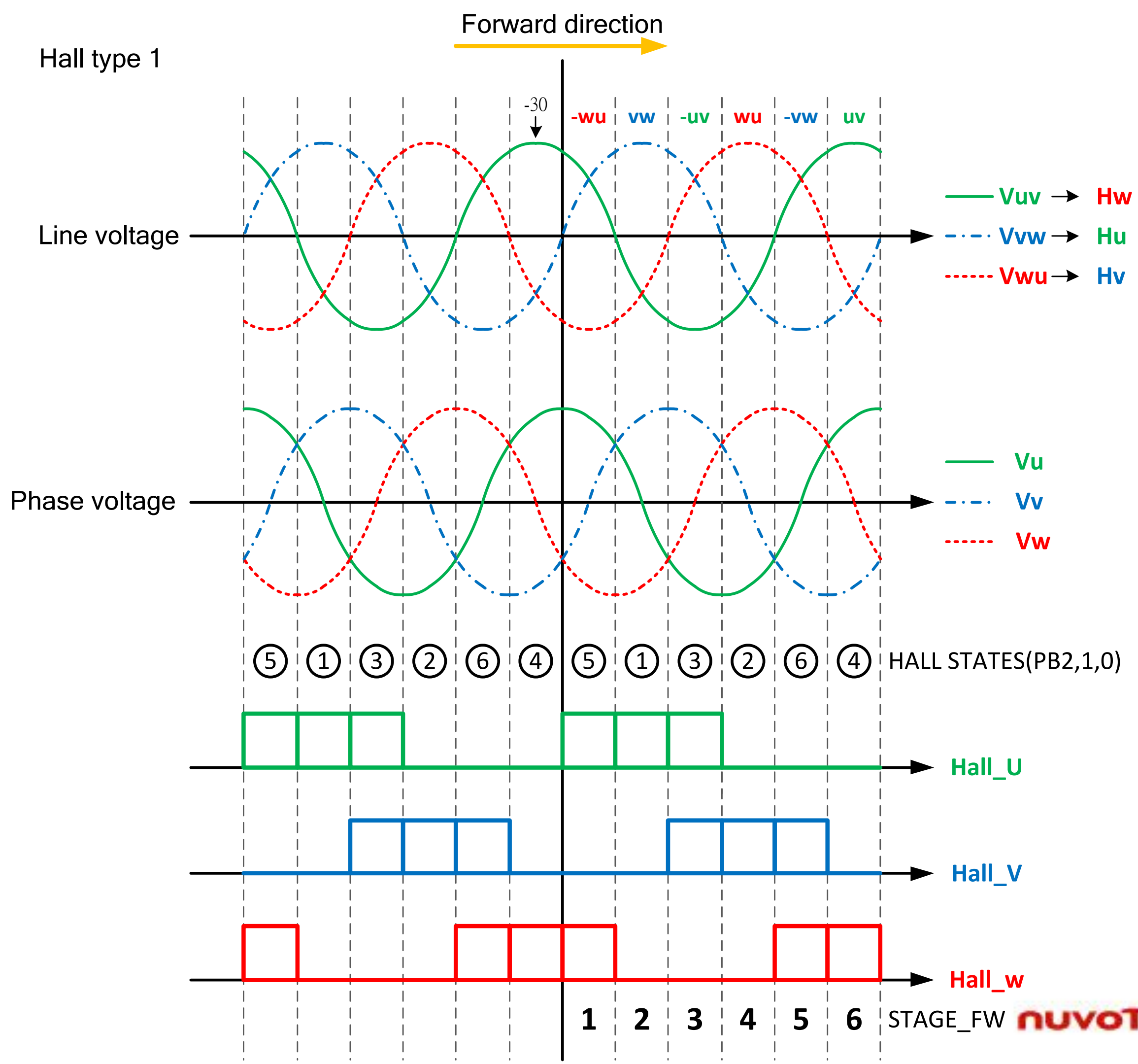
- ✓ **PWM output (6 pin) by EPWM**
  - (PWM5~PWM0) = (PA5~PA0)
- ✓ **Direction command from RUN switch (1 pin) by GPIO**
  - PC4: read the state of RUN switch
- ✓ **Light LED (1 pin) by GPIO**
  - PD4: control LED on/off
- ✓ **UART port (2 pin) by USCI-UART0**
  - UART0_RXD = PD6
  - UART0_TXD = PD5
- ✓ **Nu-Link ICE port by SWD**
  - ICE_DAT = PD2
  - ICE_CLK = PD1

Pin Configure in Demo System (III)

- ✓ (Optional) DAC module by SPI0
  - SPI0_MOSI = PD2
  - SPI0_CLK = PD1
  - CS = PD3 (GPIO mode)

Hall signal correction

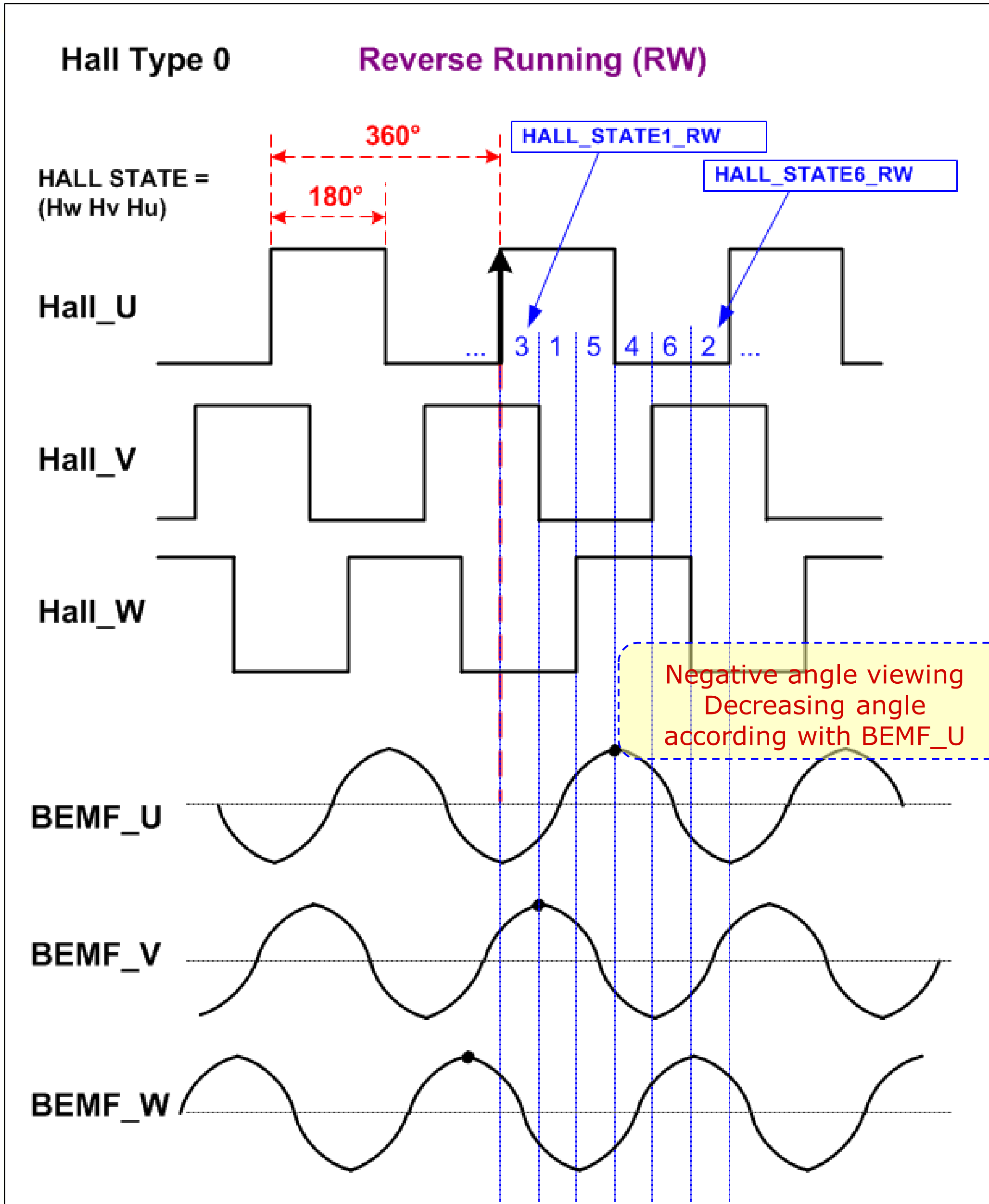
- The correspondence between BEMF & hall signal as bellow
- User can correct with line or phase voltage (example: Type 1)



Hall signal correction

- User can select hall type 0 or type 1 in the example code
- Hall type 0 mean the falling edge of hall signal align phase voltage peak
- Hall type 1 mean the rising edge of hall signal align phase voltage peak

Type0 : Hall and BEMF for RW



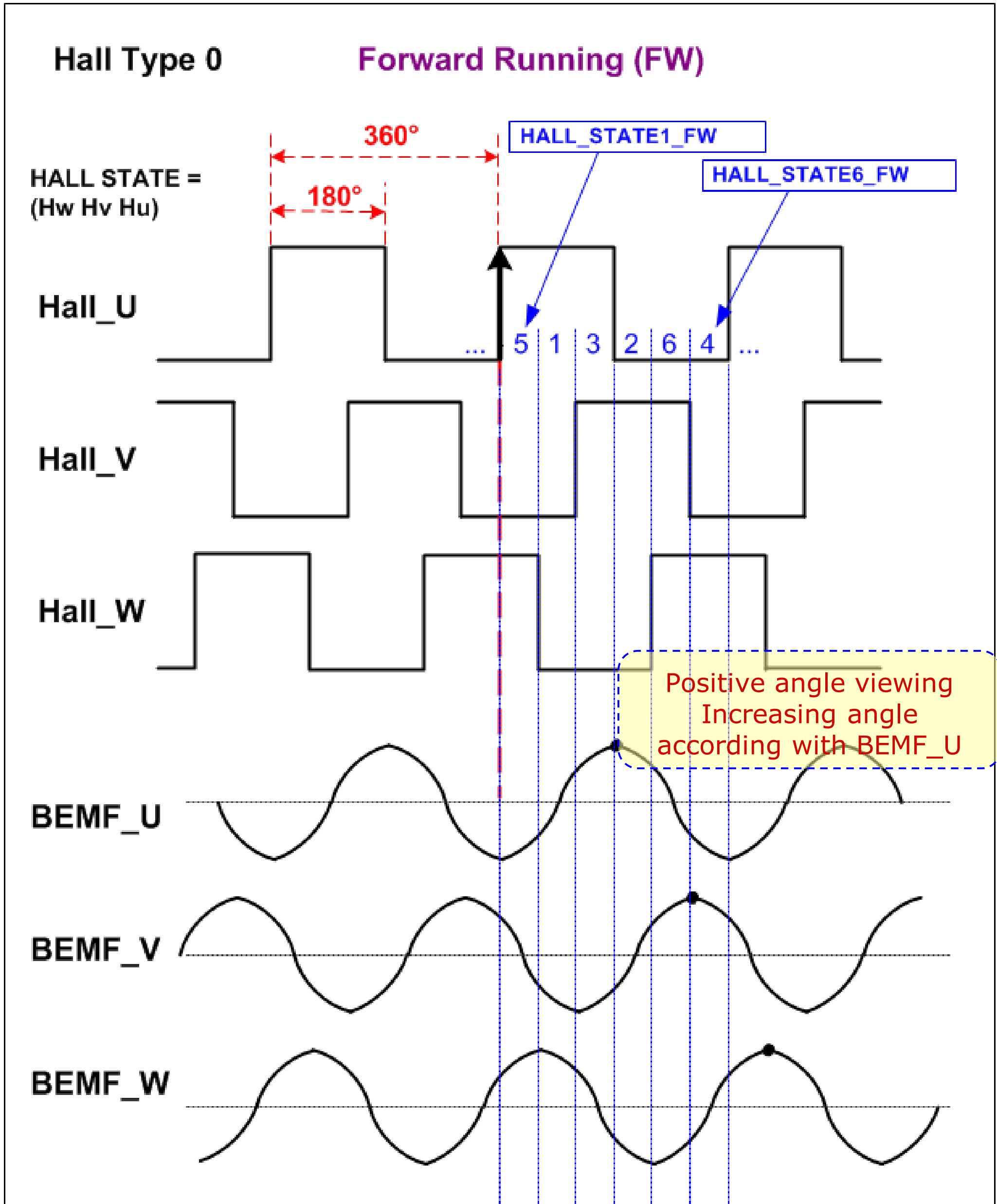
Define the 1st hall state right after the rising edge at IC0 as "HALL_STATE1_RW"

Hall Sensor (W,V,U) =
Input Capture (IC2, IC1, IC0) =
I/O Port (PB2, PB1, PB0) =
3, 1, 5, 4, 6, 2…for CCW

```
#define HALL_STATE1_RW  3
#define HALL_STATE2_RW  1
#define HALL_STATE3_RW  5
#define HALL_STATE4_RW  4
#define HALL_STATE5_RW  6
#define HALL_STATE6_RW  2
```

```
#define HALL_STATE1_R_ANGLE
1024*(-90+0)/360

#define HALL_STATE2_R_ANGLE
1024*(-150+0)/360

#define HALL_STATE3_R_ANGLE
1024*(-210+0)/360

#define HALL_STATE4_R_ANGLE
1024*(-270+0)/360

#define HALL_STATE5_R_ANGLE
1024*(-330+0)/360

#define HALL_STATE6_R_ANGLE
1024*(-30+0)/360
```

Type0 : Hall and BEMF for FW



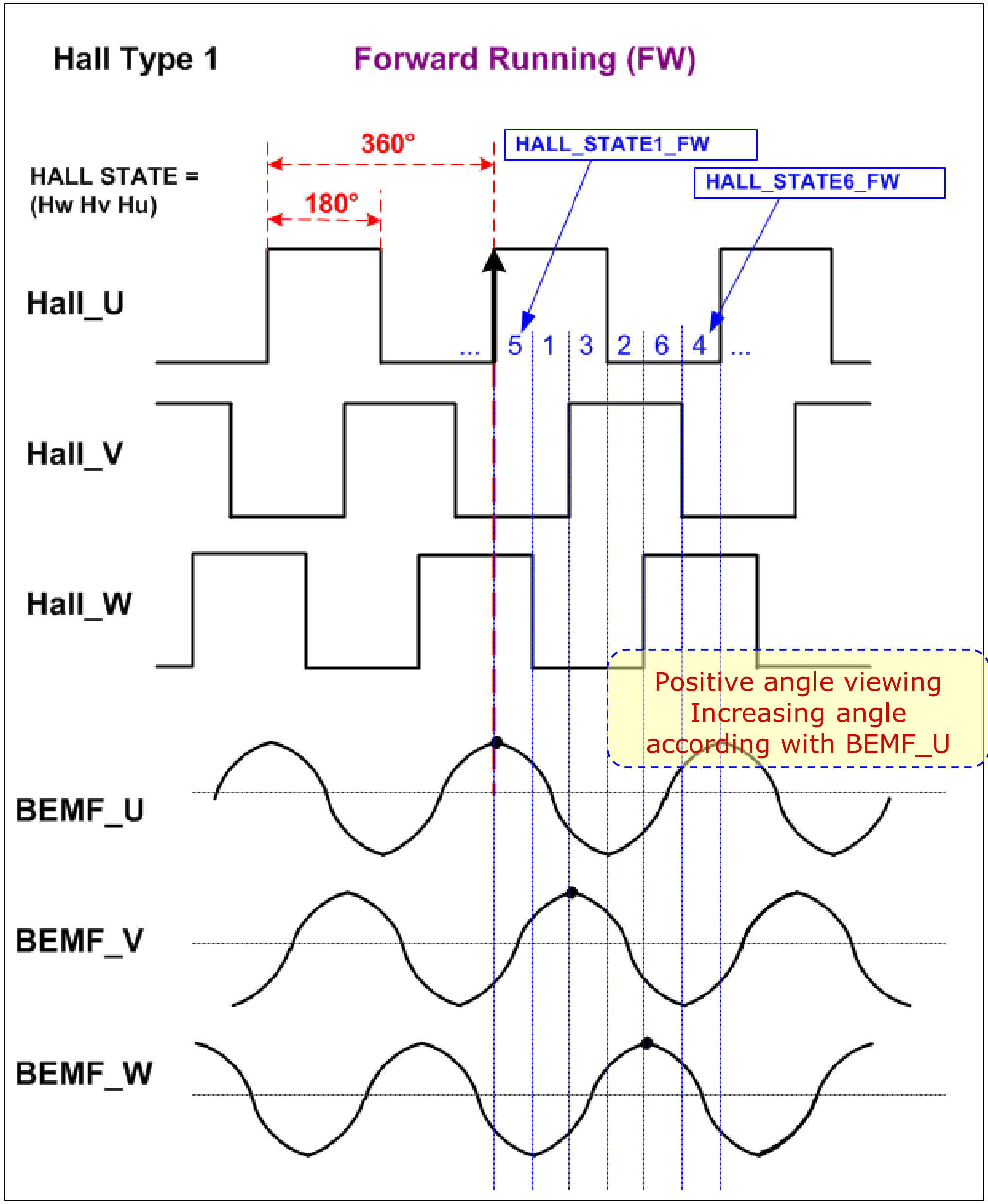
Define the 1st hall state right after the rising edge at IC0 as "HALL_STATE1_FW"

Hall Sensor (W,V,U) =
Input Capture (IC2, IC1, IC0) =
I/O Port (PB2, PB1, PB0) =
5, 1, 3, 2, 6, 4…for FW

```
#define HALL_STATE1_FW  5
#define HALL_STATE2_FW  1
#define HALL_STATE3_FW  3
#define HALL_STATE4_FW  2
#define HALL_STATE5_FW  6
#define HALL_STATE6_FW  4
```

```
#define HALL_STATE1_F_ANGLE
1024*(270+0)/360 + SHIFT_180

#define HALL_STATE2_F_ANGLE
1024*(330+0)/360 + SHIFT_180

#define HALL_STATE3_F_ANGLE
1024*(30+0)/360  + SHIFT_180

#define HALL_STATE4_F_ANGLE
1024*(90+0)/360 + SHIFT_180

#define HALL_STATE5_F_ANGLE
1024*(150+0)/360 + SHIFT_180

#define HALL_STATE6_F_ANGLE
1024*(210+0)/360 + SHIFT_180
```

Type1 : Hall and BEMF for FW



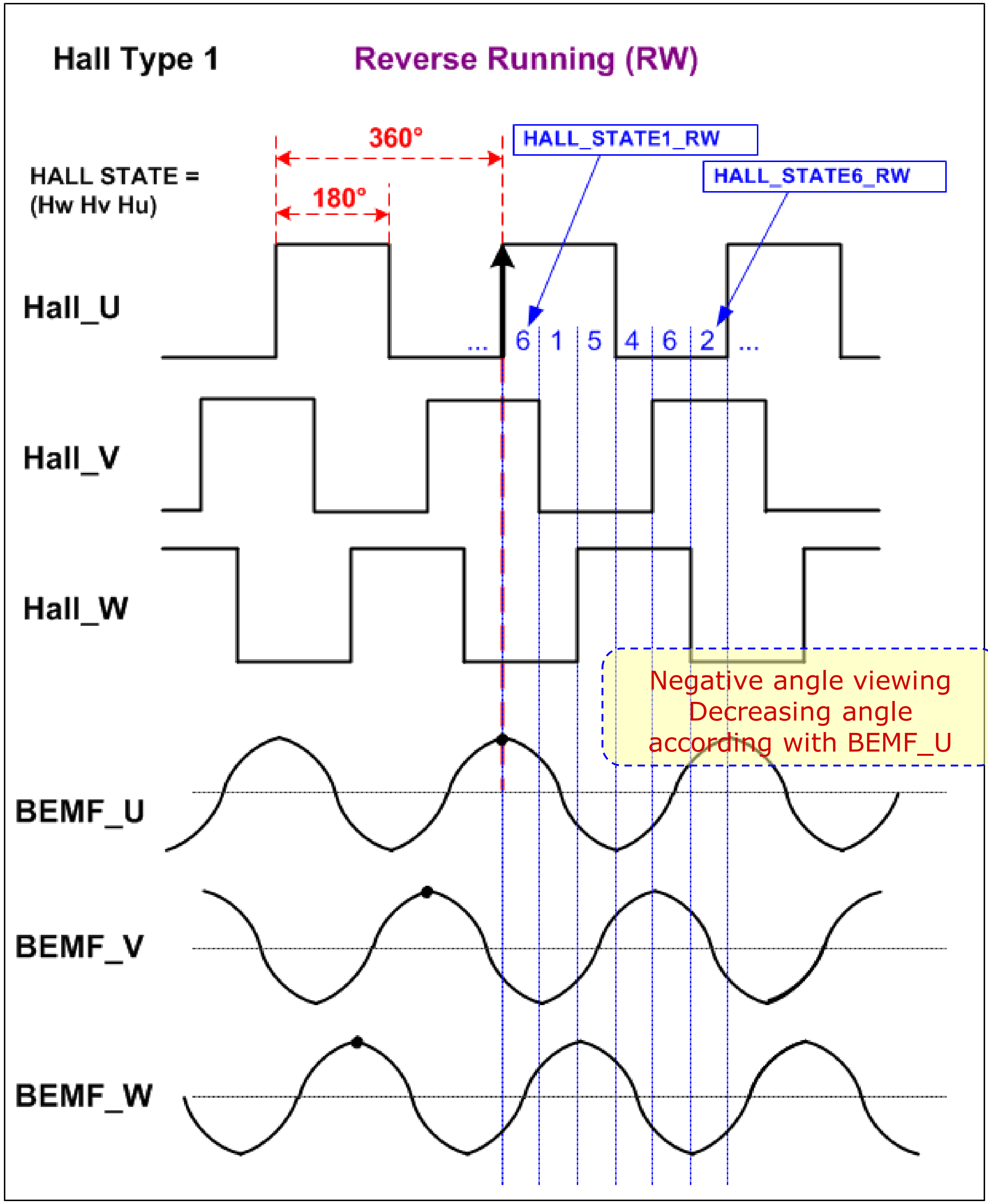
Define the 1st hall state right after the rising edge at IC0 as "HALL_STATE1_FW"

Hall Sensor (W,V,U) =
Input Capture (IC2, IC1, IC0) =
I/O Port (PB2, PB1, PB0) =
5, 1, 3, 2, 6, 4…for FW

```
#define HALL_STATE1_FW  5
#define HALL_STATE2_FW  1
#define HALL_STATE3_FW  3
#define HALL_STATE4_FW  2
#define HALL_STATE5_FW  6
#define HALL_STATE6_FW  4
```

```
#define HALL_STATE1_F_ANGLE
1024*(90+0)/360 + SHIFT_180

#define HALL_STATE2_F_ANGLE
1024*(150+0)/360 + SHIFT_180

#define HALL_STATE3_F_ANGLE
1024*(210+0)/360  + SHIFT_180

#define HALL_STATE4_F_ANGLE
1024*(270+0)/360 + SHIFT_180

#define HALL_STATE5_F_ANGLE
1024*(330+0)/360 + SHIFT_180

#define HALL_STATE6_F_ANGLE
1024*(30+0)/360 + SHIFT_180
```

Type1 : Hall and BEMF for RW



Define the 1st hall state right after the rising edge at IC0 as "HALL_STATE1_RW"

Hall Sensor (W,V,U) =
Input Capture (IC2, IC1, IC0) =
I/O Port (PB2, PB1, PB0) =
3, 1, 5, 4, 6, 2…for CCW

```
#define HALL_STATE1_RW  3
#define HALL_STATE2_RW  1
#define HALL_STATE3_RW  5
#define HALL_STATE4_RW  4
#define HALL_STATE5_RW  6
#define HALL_STATE6_RW  2
```

```
#define HALL_STATE1_R_ANGLE
1024*(-270+0)/360

#define HALL_STATE2_R_ANGLE
1024*(-330+0)/360

#define HALL_STATE3_R_ANGLE
1024*(-30+0)/360

#define HALL_STATE4_R_ANGLE
1024*(-90+0)/360

#define HALL_STATE5_R_ANGLE
1024*(-150+0)/360

#define HALL_STATE6_R_ANGLE
1024*(-210+0)/360
```

END