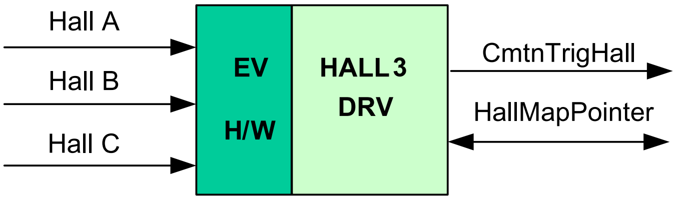
HALL3 DRV | *Hall effect interface driver for sensored BLDC control*

Description

This module produces a commutation trigger for a 3-ph BLDC motor, based on hall signals received on GPIO pins 24, 25, and 26. Edges detected are validated or debounced, to eliminate false edges often occurring from motor oscillations. The software attempts all (6) possible commutation states to initiate motor movement. Once the motor starts moving, commutation occurs on each debounced edge from received hall signals.



Availability

This 16-bit module is available in one interface format:

1) The C interface version

Module Properties

Type: Target Dependent, Application Independent

Target Devices: 28x Fixed Point or PiccoloB

C Version File Names: f2803xhall3.h (for x2803x)

IQmath library files for C: N/A

C Interface

Object Definition

The structure of HALL3 object is defined by following structure definition

```
typedef struct { Uint16 CmtnTrigHall;       // Output: Commutation trigger for Mod6cnt input
                 Uint16 CapCounter;        // Variable: Running cnt of detected edges on CAP/GPIO
                 Uint16 DebounceCount;    // Variable: Counter/debounce delay current value
                 Uint16 DebounceAmount;   // Parameter: Counter delay amount to
                                             //   validate/debounce GPIO readings
                 Uint16 HallGpio;             // Variable: Most recent logic level on CAP/GPIO
                 Uint16 HallGpioBuffer;      // Variable: Buffer of last logic level on CAP/GPIO while
                                          //       being debounced
                Uint16 HallGpioAccepted;  // Variable: Debounced logic level on CAP/GPIO
                Uint16 EdgeDebounced;   // Variable: Trigger from Debounce macro to Hall_Drv,
                                         //     if = 0x7FFF edge is debounced
                Uint16 HallMap[6];  // Variable: CAP/GPIO logic levels for HallMapPointer = 0-5
                Uint16 CapFlag;     // Variable: CAP flags, indicating which CAP/GPIO detected
                                    // the edge
                Uint16 StallCount;  // Variable: If motor stalls, this counter overflows triggers
                                    // commutation to start rotation. Rotation is defined as
                                    // an edge detection of a hall signal.
               Uint16 HallMapPointer; // Input/Output: During the map created, this variable points
                                         // to the current commutation state.  After map creation, it
                                         // points to the next commutation state.
               int16 Revolutions;  // Parameter: Running counter, with a revolution defined as 1-
                                   //  cycle of the 6 hall stateson
       } HALL3;

typedef HALL3 *HALL3_handle;
```

Item	Name	Description	Format	Range(Hex)
Inputs	CAP/GPIO	CAP/GPIO inputs (H/W)	N/A	0-3.3 v
	HallMapPointer	As an input, it is defined by MOD6_CNT	Q0	0 - 5
Outputs	CmtnTrigHall	Commutation trigger for Mod6cnt input	Q0	0 or 7FFF
	HallMapPointer	During hall map creation, this variable points to the current commutation state. After map creation, it points to the next commutation state.	Q0	0 - 5
HALL3 parameter	DebounceAmount	Counter delay amount to validate/debounce GPIO readings	Q0	0000-7FFF
	Revolutions	Running counter, with a revolution defined as 1-cycle of the 6 hall states	Q0	8000-7FFF
Internal	CapCounter	Running count of detected edges on CAP/GPIO	Q0	0000-7FFF
	DebounceCount	Counter/debounce delay current value	Q0	0000-7FFF
	HallGpio	Most recent logic level on CAP/GPIO	Q0	0000-0007
	HallGpioBuffer	Buffer of last logic level on CAP/GPIO while being debounced	Q0	0000-0007
	HallGpioAccepted	Debounced logic level on CAP/GPIO	Q0	0000-0007
	EdgeDebounced	Trigger from Debounce macro to Hall_Drv, if = 0x7FFF edge is debounced	Q0	0 or 7FFF
	HallMap[6]	CAP/GPIO logic levels for HallMapPointer = 0-5	Q0	0000-0007
	CapFlag	CAP flags, indicating which CAP/GPIO detected the edge	Q0	0000-0007
	StallCount	If motor stalls, this counter overflow triggers commutation to start rotation. Rotation is defined as an edge detection of a hall signal.	Q0	0000-FFFF

Special Constants and Data types

HALL3

The module definition is created as a data type. This makes it convenient to instance an interface to the HALL3 driver. To create multiple instances of the module simply declare variables of type HALL3.

HALL3_handle

User defined Data type of pointer to HALL3 module

HALL3_DEFAULTS

Structure symbolic constant to initialize HALL3 module. This provides the initial values to the terminal variables as well as method pointers.

Methods

HALL3_INIT_MACRO (HALL3 *)
HALL3_READ_MACRO (HALL3 *)

This default definition of the object implements two methods – the initialization and the runtime compute macro for HALL3. This is implemented by means of a macro pointer, and the initializer sets this to HALL3_INIT_MACRO and HALL3_READ_MACRO macros for x280x. The argument to this macro is the address of the HALL3 object.

Module Usage

Instantiation

The following example instances one HALL3 object
HALL3 hall;

Initialization

To Instance pre-initialized objects
HALL3 hall = HALL3_DEFAULTS;

Invoking the computation macro

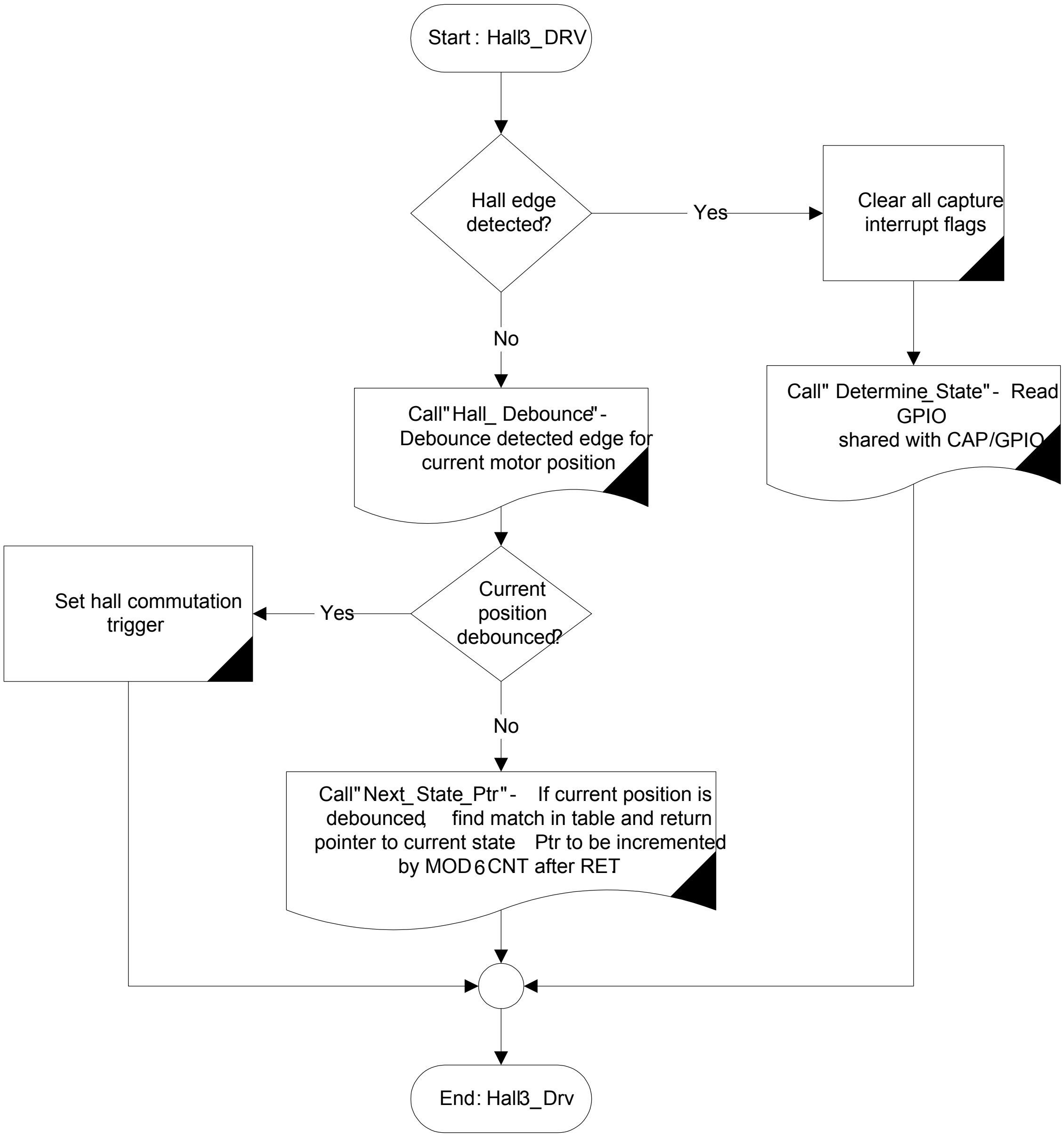
HALL3_INIT_MACRO (hall);
HALL3_READ_MACRO (hall);

Example

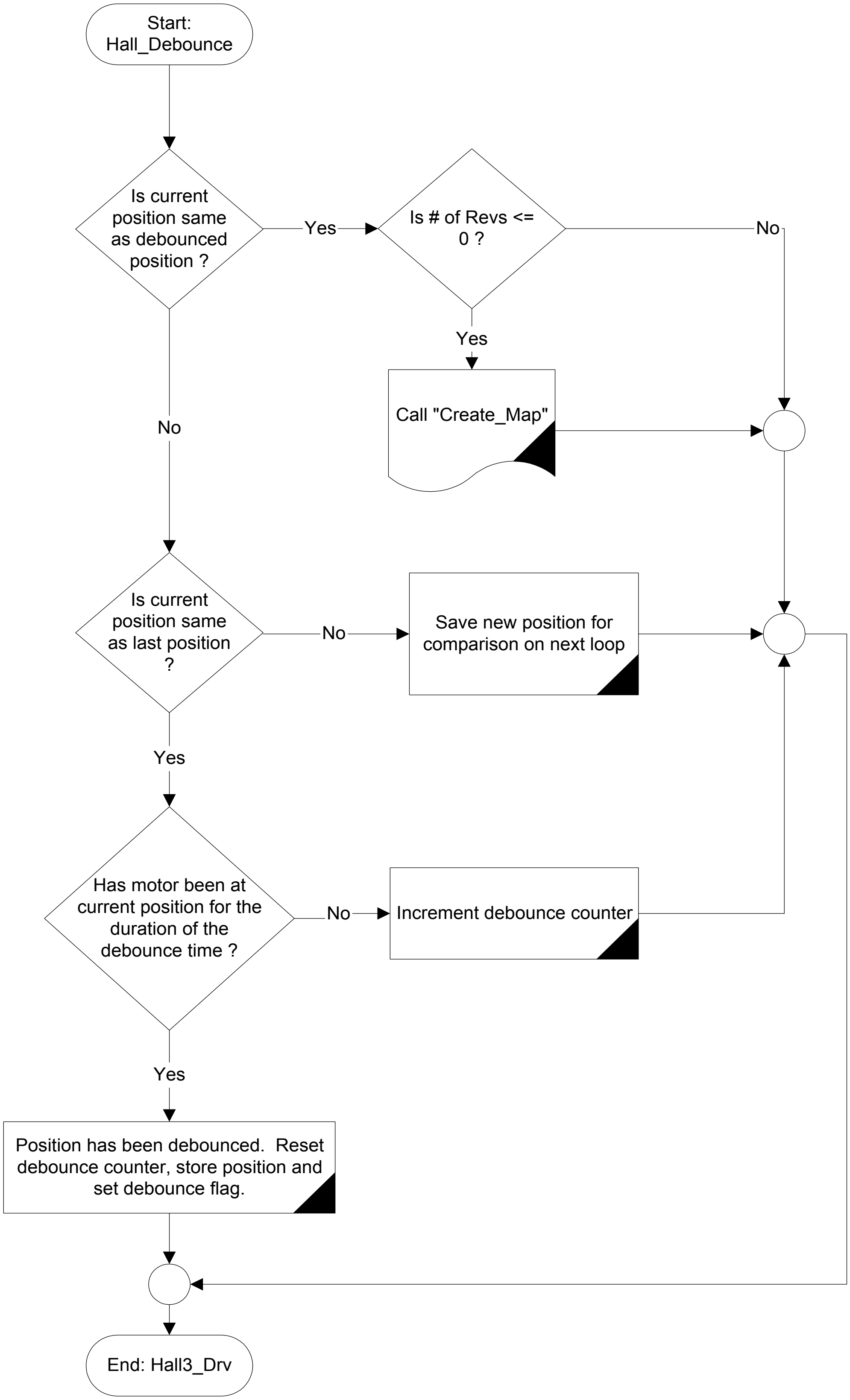
The following pseudo code provides the information about the module usage.

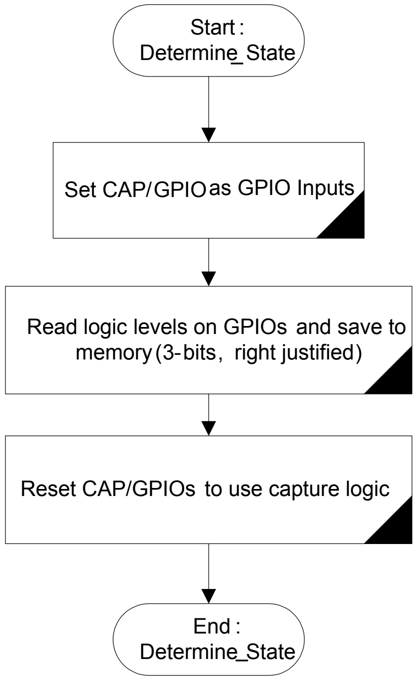
```
main()
{

        HALL3_INIT_MACRO (hall);                // Call init macro for hall3

}

void interrupt periodic_interrupt_isr()
{

       HALL3_READ_MACRO (hall);     // Call the hall3 read macro


}
```

Software Flowcharts



Start : Hall3_DRV
Hall edge detected?
Yes
Clear all capture interrupt flags
No
Call" Hall_ Debounce" - Debounce detected edge for current motor position
Call" Determine_State" - Read GPIO shared with CAP/GPIO
Set hall commutation trigger
Yes
Current position debounced?
No
Call" Next_State_Ptr" - If current position is debounced, find match in table and return pointer to current state Ptr to be incremented by MOD6CNT after RET
End: Hall3_Drv


Start:
Hall_Debounce
Is current position same as debounced position ?
Yes
Is # of Revs <= 0 ?
No
Yes
Call "Create_Map"
No
Is current position same as last position ?
No
Save new position for comparison on next loop
Yes
Has motor been at current position for the duration of the debounce time ?
No
Increment debounce counter
Yes
Position has been debounced. Reset debounce counter, store position and set debounce flag.
End: Hall3_Drv


Start : Determine_State
Set CAP/GPIO as GPIO Inputs
Read logic levels on GPIOs and save to memory (3-bits, right justified)
Reset CAP/GPIOs to use capture logic
End : Determine_State