LSI3144A

Encoder / Linear Scale Counter Card

Software Manual (V2.2)

健昇科技股份有限公司
JS AUTOMATION CORP.

新北市汐止區中興路 100 號 6 樓
6F., No.100, Zhongxing Rd.,
Xizhi Dist., New Taipei City, Taiwan
TEL：+886-2-2647-6936
FAX：+886-2-2647-6940
http://www.automation.com.tw
http://www.automation-js.com/
E-mail：control.cards@automation.com.tw

Correction record

Version	Record
1.0	for driver V1.0 up
V1.0->V1.1	for driver V2.2 up
	add: **LSI3144A_counter_direction_read**
	LSI3144A_direction_compare_value_set
	LSI3144A_direction_compare_value_read
	LSI3144A_direction_FIFO_set
	LSI3144A_direction_FIFO_read
	LSI3144A_compare_CMP_OUT_set
	LSI3144A_compare_CMP_OUT_read
V1.1->V1.2	Modify the order of the contents (flow chart)
V1.2->V2.0	disable the software key function with return value always true
V2.0->V2.1	for DLL v3.2 up Add new functions
	add: **LSI3144A_position_trigger_set**
	LSI3144A_position_trigger_read
	LSI3144A_PWM_motion_config_set
	LSI3144A_PWM_motion_config_read
	LSI3144A_PWM_motion_control_set
	LSI3144A_PWM_motion_control_read
V2.1->V2.2	For DLL v3.3 up Add new function
	add: **LSI3144A_Latch_FIFO_initial( )**
	LSI3144A_Latch_FIFO_control_set( )
	LSI3144A_Latch_FIFO_control_read( )
	LSI3144A_Latch_FIFO_data_read( )
	LSI3144A_Latch_FIFO_status_read( )

Contents

1. How to install the software of LSI3144A 6
   1.1 Install the PCI driver 6
2. Where to find the file you need 7
3. About the LSI3144A software 8
   3.1 What you need to get started 8
   3.2 Software programming choices 8
4. LSI3144A Language support 9
   4.1 Building applications with the LSI3144A software library 9
   4.2 LSI3144A Windows Libraries 9
5. Basic concepts 10
   5.1 Quadrature Encoder 10
   5.2 Homing (counter clear mode) 12
   5.3 External trigger to latch counter 14
6. Basic concepts of counter compare function 15
   6.1 Counter compare mode 15
   6.2 Trigger output width 18
   6.3 Segment mask off and external gate function 18
7. Function format and language difference 20
   7.1 Function format 20
   7.2 Variable data types 21
   7.3 Programming language considerations 22
8. Flow chart of application implementation 24
9. Software overview and dll function 32
   9.1 Compatibility of LSI3144A and LSI3144 32
   9.2 Initialization and close 32
      LSI3144A_initial 32
      LSI3144A_close 32
      LSI3144A_info 33
   9.3 Input / Output function 34
      LSI3144A_input_polarity_set 35
      LSI3144A_input_polarity_read 36
      LSI3144A_input_debounce_set 37
      LSI3144A_input_debounce_read 38
      LSI3144A_input_read 40
      LSI3144A_output_polarity_set 40
      LSI3144A_output_polarity_read 41
      LSI3144A_output_set 41
      LSI3144A_output_read 42
   9.4 Homing (to clear counter) and clear input and clear output function 43

LSI3144A_HOMING_mode_set 44
LSI3144A_HOMING_mode_read 47
LSI3144A_HOMING_flag_read 48
LSI3144A_HOMING_software 48
LSI3144A_CLR_OUT_mode_set 49
LSI3144A_CLR_OUT_mode_read 50
LSI3144A_CLR_oneshot_duration_set 51
9.5 Basic counter function 52
LSI3144A_CI_mode_set 53
LSI3144A_CI_mode_read 54
LSI3144A_counter_control_set 55
LSI3144A_counter_control_read 55
LSI3144A_counter_set 56
LSI3144A_counter_read 56
9.6 Counter latch / load mode 57
LSI3144A_counter_preset 58
LSI3144A_counter_preset_read 58
LSI3144A_latch_mode_set 59
LSI3144A_latch_mode_read 60
LSI3144A_latch_control_set 60
LSI3144A_latch_control_read 61
LSI3144A_latch_flag_read 61
LSI3144A_latched_value_read 62
9.7 Basic compare function 63
LSI3144A_compare_mode_set 64
LSI3144A_compare_mode_read 65
LSI3144A_compare_value_set 65
LSI3144A_compare_value_read 66
LSI3144A_direction_compare_value_set 66
LSI3144A_direction_compare_value_read 67
LSI3144A_CMP_mode_set 67
LSI3144A_CMP_mode_read 68
LSI3144A_CMP_oneshot_duration_set 68
LSI3144A_CMP_oneshot_duration_read 69
LSI3144A_compare_CMP_OUT_set 70
LSI3144A_compare_CMP_OUT_read 71
LSI3144A_compare_control_set 72
LSI3144A_compare_control_read 72
9.8 Auto increment compare mode 73
LSI3144A_compare_increment_set 74
LSI3144A_compare_increment_read 74

9.9 FIFO compare mode 75
LSI3144A_FIFO_clear 77
LSI3144A_FIFO_set 77
LSI3144A_FIFO_read 78
LSI3144A_direction_FIFO_set 78
LSI3144A_direction_FIFO_read 79
LSI3144A_FIFO_threshold_set 80
LSI3144A_FIFO_threshold_read 80
LSI3144A_FIFO_unused_read 81
LSI3144A_FIFO_status_read 81
LSI3144A_position_trigger_set 85
LSI3144A_position_trigger_read 85
9.10 Compare output mask function 86
LSI3144A_CMP_mask_source_set 87
LSI3144A_CMP_mask_source_read 87
LSI3144A_CMP_segment_set 88
LSI3144A_CMP_segment_read 88
LSI3144A_mask_off_set 89
LSI3144A_mask_off_read 89
LSI3144A_segment_control_set 90
LSI3144A_segment_control_read 90
9.11 Miscellaneous function 91
LSI3144A_com_trigger_control 91
LSI3144A_counter_direction_read 91
9.12 PWM function 92
LSI3144A_PWM_set 93
LSI3144A_PWM_read 93
LSI3144A_PWM_control_set 94
LSI3144A_PWM_control_read 94
9.13 Vector speed to PWM function 95
LSI3144A_PWM_motion_config_set 96
LSI3144A_PWM_motion_config_read 97
LSI3144A_PWM_motion_control_set 98
LSI3144A_PWM_motion_control_read 98
9.14 Timer function 99
LSI3144A_timer_set 99
LSI3144A_timer_read 100
LSI3144A_timer_start 100
LSI3144A_timer_stop 100
9.15 Interrupt function 101
LSI3144A_IRQ_mask_set 102

LSI3144A_IRQ_mask_read 103
LSI3144A_IRQ_process_link 103
LSI3144A_IRQ_status_read 104
LSI3144A_IRQ_enable 105
LSI3144A_IRQ_disable 105
9.16 Latch FIFO 106
LSI3144A_Latch_FIFO_initial 107
LSI3144A_Latch_FIFO_control_set 107
LSI3144A_Latch_FIFO_control_read 107
LSI3144A_Latch_FIFO_status_read 108
LSI3144A_Latch_FIFO_data_read 108
9.17 Security function 109
LSI3144A_password_set 109
LSI3144A_password_change 110
LSI3144A_password_clear 110
LSI3144A_security_unlock 110
LSI3144A_security_status_read 111
10. Dll list 112
11. LSI3144A Error code summary 116
11.1 LSI3144A Error code table 116

1. How to install the software of LSI3144A

1.1 Install the PCI driver

The PCI card is a plug and play card, once you add a new card on the window system will detect while it is booting. Please follow the following steps to install your new card.

In WinXP/7 system you should: (take Win XP as example)

1. Make sure the power is off
2. Plug in the interface card
3. Power on
4. A hardware install wizard will appear and tell you it finds a new PCI card
5. Do not response to the wizard, just Install the file (..\LSI3144A\Software\WinXP_7\ or if you download from website please execute the file LSI3144A_Install.exe to get the file)
6. After installation, power off
7. Power on, it's ready to use

For a more detail descriptions, please visit our user club http://automation.com.tw and register as a member then download the file "Installation" which will take you go through step by step.

2. Where to find the file you need

WinXP/7 and up

The directory will be located at

.. \ JS Automation \LSI3144A\API\ (header files and lib files for VB,VC,BCB,C#)

.. \ JS Automation \LSI3144A\Driver\ (backup copy of LSI3144A drivers)

.. \ JS Automation \LSI3144A\exe\ (demo program and source code)

The system driver is located at ..\system32\Drivers and the DLL is located at ..\system.

For your easy startup, the demo program with source code demonstrates the card functions and help file.

3. About the LSI3144A software

LSI3144A software includes a set of dynamic link library (DLL) and system driver that you can utilize to control the interface card's functions.

Your LSI3144A software package includes setup driver, tutorial and test program that help you how to setup and run appropriately, as well as an executable file which you can use to test each of the LSI3144A functions within Windows' operation system environment.

3.1 What you need to get started

To set up and use your LSI3144A software, you need the following:

- LSI3144A software
- LSI3144A hardware

  Main board

  Wiring board (Option)

3.2 Software programming choices

You have several options to choose from when you are programming LSI3144A software. You can use Borland C/C++, Microsoft Visual C/C++, Microsoft Visual Basic, or any other Windows-based compiler that can call into Windows dynamic link libraries (DLLs) for use with the LSI3144A software.

4. LSI3144A Language support

The LSI3144A software library is a DLL used with WinXP/7 and up. You can use these DLL with any Windows integrating development environment that can call Windows DLLs.

4.1 Building applications with the LSI3144A software library

The LSI3144A function reference topic contains general information about building LSI3144A applications, describes the nature of the LSI3144A files used in building LSI3144A applications, and explains the basics of making applications using the following tools:

Applications tools

- Microsoft Visual C/C++
- Borland C/C++
- Microsoft Visual C#
- Microsoft Visual Basic
- Microsoft VB.net

If you are not using one of the tools listed, consult your development tool reference manual for details on creating applications that call DLLs.

4.2 LSI3144A Windows Libraries

The LSI3144A for Windows function library is a DLL called LSI3144A.dll. Since a DLL is used, LSI3144A functions are not linked into the executable files of applications. Only the information about the LSI3144A functions in the LSI3144A import libraries is stored in the executable files.

Import libraries contain information about their DLL-exported functions. They indicate the presence and location of the DLL routines. Depending on the development tools you are using, you can make your compiler and linker aware of the DLL functions through import libraries or through function declarations.

Refer to **Table 1** to determine to which files you need to link and which to include in your development to use the LSI3144A functions in LSI3144A.dll.

Header Files and Import Libraries for Different Development Environments		
Language	**Header File**	**Import Library**
Microsoft Visual C/C++	LSI3144A.h	LSI3144AVC.lib
Borland C/C++	LSI3144A.h	LSI3144ABC.lib
Microsoft Visual C#	LSI3144A.cs	
Microsoft Visual Basic	LSI3144A.bas	
Microsoft VB.net	LSI3144A.vb	

Table 1

5. Basic concepts

5.1 Quadrature Encoder

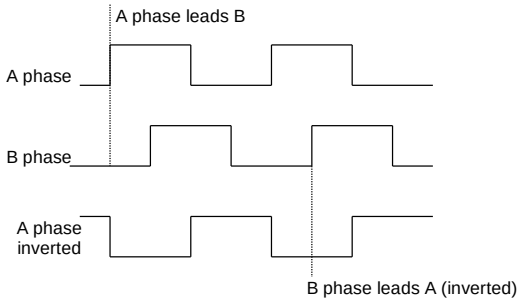
Input debounce time

If the counter input signal comes from the noisy environment, the input needs to filter out the unwanted signals and keep the meaningful signals to go through to counter. A programmable debounce digital filter put in the way of input signal to drop out the unwanted signal is a good choice.

Users can use the default debounce time constant or change depending on the signal speed and environment noise. A noisy environment normally needs large time constant to drop out the unwanted signal and high pulse rate limits the time constant you can choose. At default, the debounce function will drop the pulse duration less than 1us (debounce frequency 1M). You can choose one from 512K, 1M, 2M, 4M, 8M, 16M to meet your requirement.

Input polarity

For the maximum flexibility, the polarity function will change the input signal to meet the requirements of the following function blocks. Say A phase leads B phase in your external signal input, you can invert the A phase to change to B phase leads A phase without actually change the wiring.

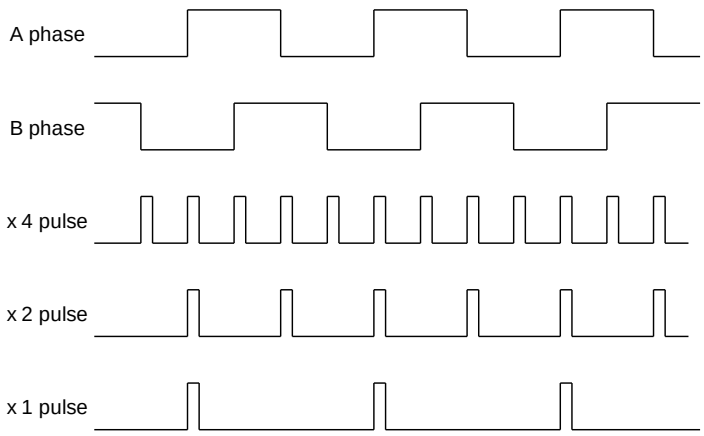


From left diagram, you can see A phase change polarity is equivalent to change A phase from lead state to lag state.

Signal input type

In LSI3144A card, there are 3 major signal types can be count.

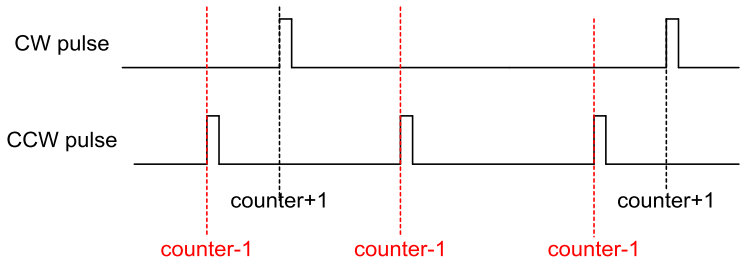
Quadrature input type



The left diagram shown that A phase leads B phase, if we take A leads B as up count and the counting pulse of up count will depends on the multiple rate.

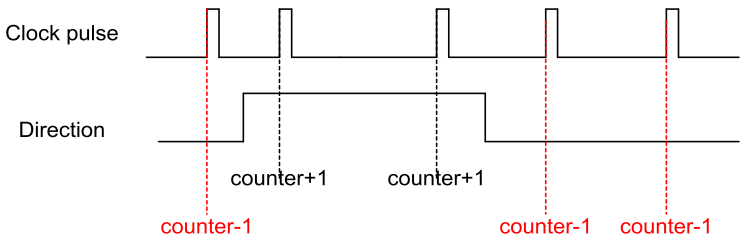
On the other hand, if B phase leads A phase, the counter will be down count.

CW and CCW input type (Dual pulse mode)



The left diagram shown that CW and CCW pulses. Any CW pulse input will increase counter by 1 and any CCW pulse input will decrease counter by 1.

Clock and direction input type (Single pulse mode)



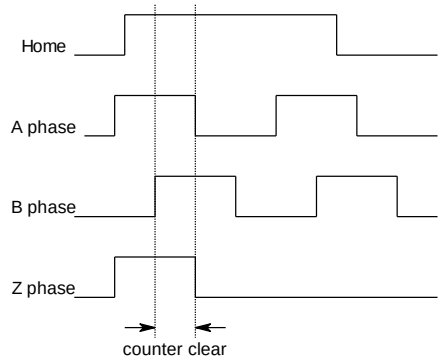
The left diagram shown that Clock and Direction pulses. Any Clock pulse input will increase counter by 1 while the Direction signal is make and any Clock pulse input will decrease counter by 1 while Direction signal is break.

5.2 Homing (counter clear mode)

Normal counters use external asynchronous reset to clear counter but the quadrature counter generally provides more versatile functions to fit the need of different applications. In most quadrature counter applications the counter clear function also called as counter “HOMING”.

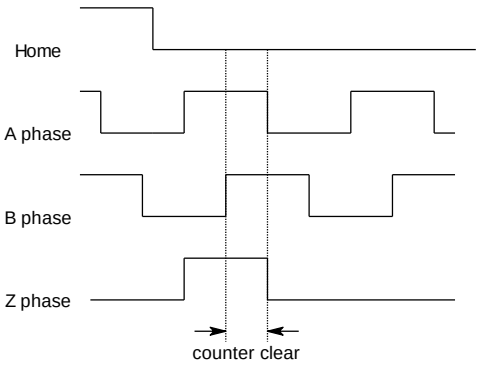
There are several modes to do homing:

1. counter clear at A,B,Z and Home active



The counter will be cleared while A,B Z and Home are all “make”.

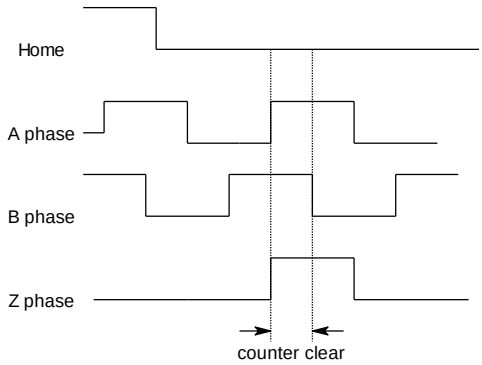
2. counter clear at first A,B,Z active after HOME turn to inactive and up count



The counter will be cleared while the following conditions meet:

1. HOME switch turns into “BREAK” state.
2. first A,B Z are all “MAKE” and counter up count (suppose A phase leads B phase).

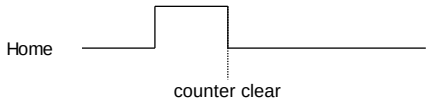
3. counter clear at first A,B,Z active after HOME turn to inactive and down count



The counter will be cleared while the following conditions meet:

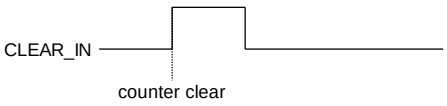
1. the HOME switch turns into “BREAK” state.
2.first A,B Z are all “MAKE” and counter down count (suppose B phase leads A phase).

4. counter clear at tailing edge of HOME



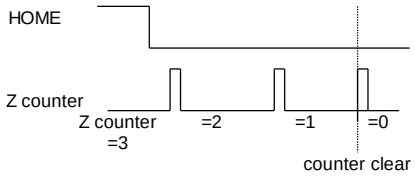
The counter will be cleared at the tailing edge of the HOME switch.

5. counter clear at rising edge of CLEAR_IN



The counter will be cleared at the rising edge of the CLEAR_IN signal input.

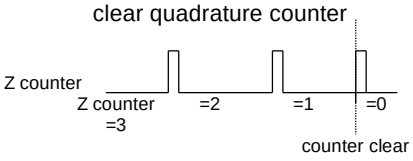
6. Trailing edge of HOME starts Z phase counter and count down to "0" clear quadrature counter



The counter will be cleared the following conditions meet:

1. the HOME switch turns into “BREAK” state.
2. Since the HOME tailing edge, Z phase counter countdown to “0”

7. Z phase counter count down to "0" clear quadrature counter



The counter will be cleared while Z phase counter countdown to “0”.

CLR_OUT as servo error counter clear input

The homing function can co-operate with the CLR_OUT to clear external device while HOMING condition meets. Normally, the motion control uses a servo motor to drive the motion component. To do HOMING is to give the power on system a known coordinate at known condition, but on the other hand, if you use a pulse type servo drive, there are some remained pulses on the condition of HOMING meets. If you want to clear the servo drive to a zero error counter state to ensure the accuracy of HOMING, CLR_OUT is a good choice.

5.3 External trigger to latch counter

The counter counts input pulses on the fly, if you do not want to accurately record single or multiple axes data on a special occasion; use the hardware trigger to latch. A good example of the application is the coordinate machine. It has 3 or more linear scale to count the position while the touch probe triggers to latch the coordinate it measures.

6. Basic concepts of counter compare function

The most powerful function of LSI3144A card is the high speed comparison function. You can use this function to trigger external devices such as CCD camera to catch vision data.

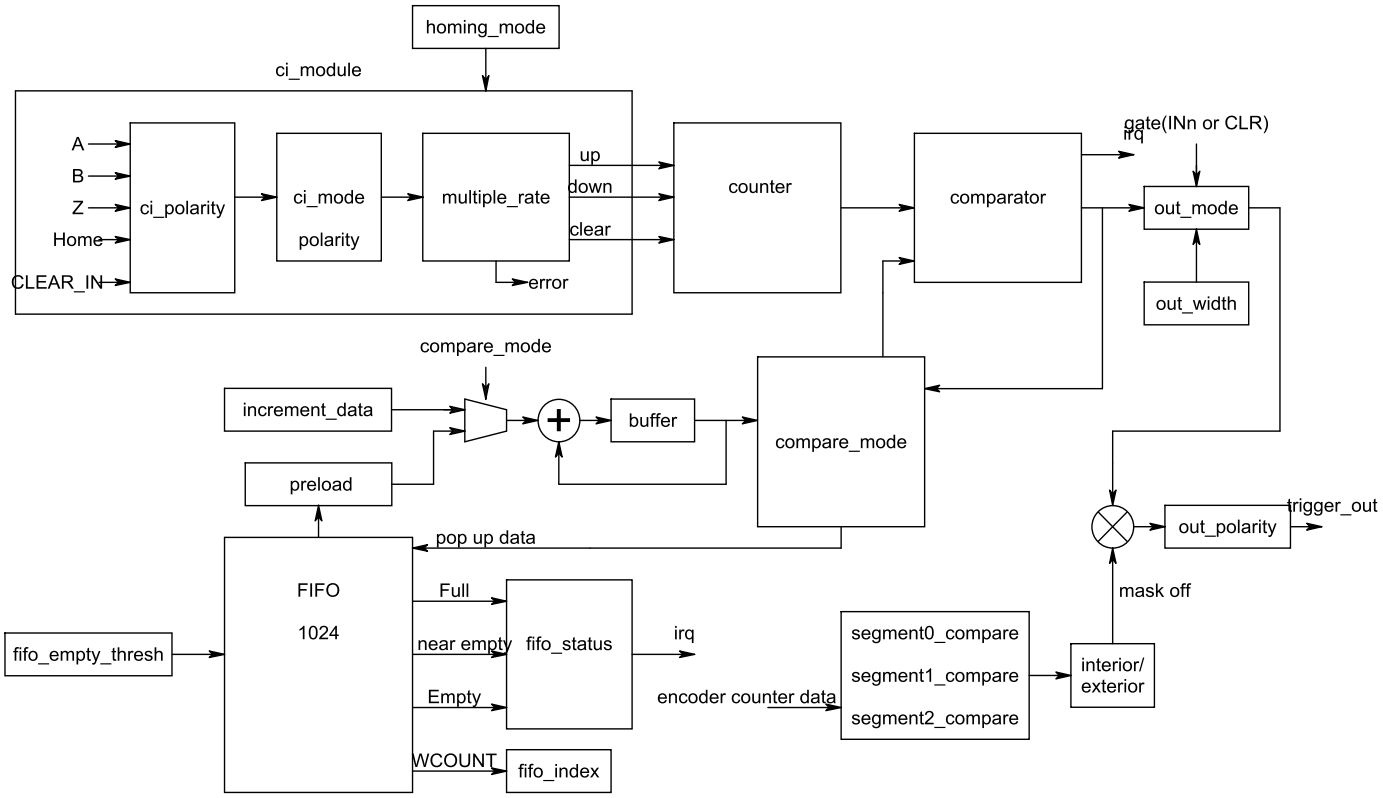


fig 6.1 counter function block diagram

From the above diagram, while the comparator compares equal, it will generate a trigger output and maintain the pulse at out_width duration. External gate function can block the trigger output while it is at “break” state.

6.1 Counter compare mode

The comparator can work in one of the 3 modes, single compare, auto-increment compare and FIFO compare mode.

Single compare mode (One time compare mode)

The desired compare value is pre-loaded, if the quadrature counter value and the compare value meet the compare condition (i.e. data equal and motion direction as expect), generate output trigger.

Auto increment mode

If the compare value (compare data) not only store at the preset register (compare value register) but also other subsequent data is define by an incremental value of current compare value. After one compared condition met, the preset register (compare value register) will be loaded a data which is the sum of current compare value and the incremental value to proceed the next compare.

new compare value= current compare value + auto increment value

(NOTE: the incremental value can be a minus value, this means a decrement of current compare value)

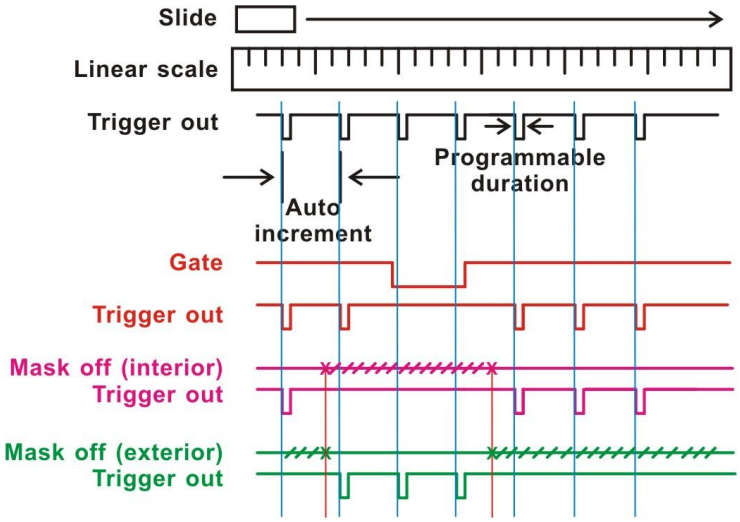


fig 6.2 Auto increment compare and compare output mask function

The above diagram shows trigger output on fixed increment position and the output pulse width is controlled by programmable duration.

FIFO compare mode

If the compare value (compare data) not only store at the preset register (compare value register) but also other subsequent data stored at a position FIFO (first in first out memory), after one compared condition met, the position FIFO will supply the preset register (compare value register) a new pop up data to proceed the next compare.

compare value= current compare value + value pop up from position FIFO; at relative mode or

compare value= position FIFO preset value; at absolute mode.

The compare function will continue until the position FIFO is empty.

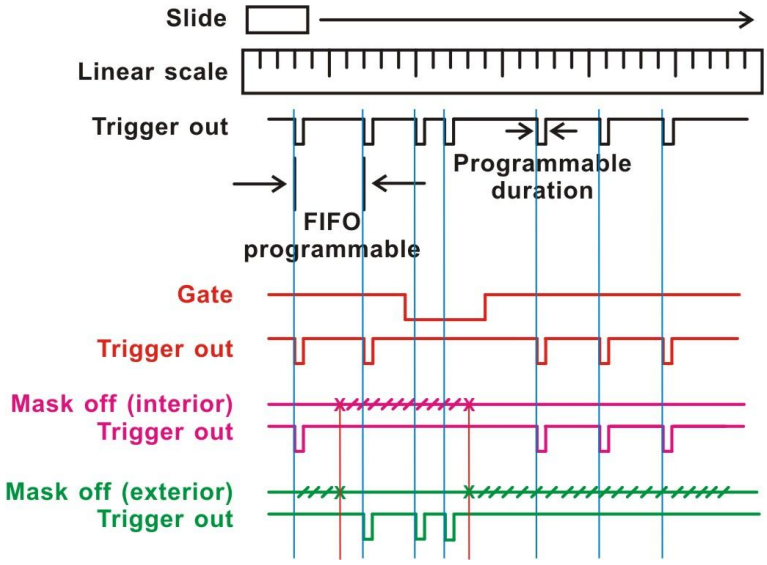


fig. 6.3 FIFO compare and compare output mask function

The FIFO mode can also work with PWM FIFO, the PWM FIFO stores the PWM duty. On the occurrence of position FIFO compare condition meet; it generate the PWM pulse on CLR_OUT pin.

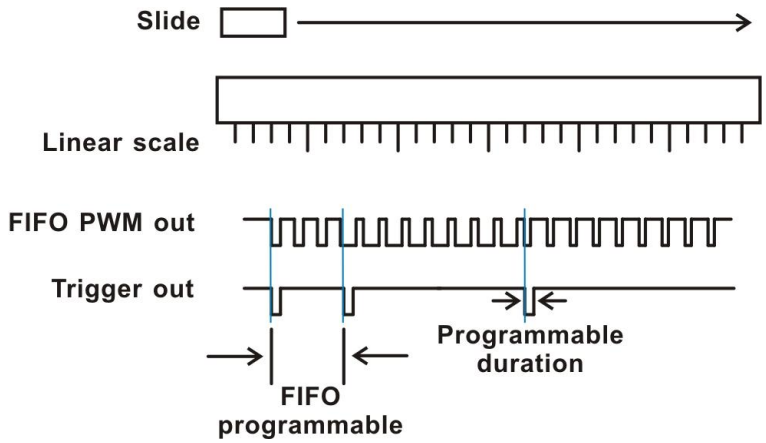


fig 6.4 FIFO compare and PWM FIFO function

6.2 Trigger output width

It is apparently that you will use the CMP output to trigger some device to start some tasks. Not every device is so fast to recognize the compare out pulse. A compare out pulse width (or duration) timer will extend the pulse to your need. LSI3144A card provide the compare equal pulse duration on a 1us base and 16 bit data length.

6.3 Segment mask off and external gate function

The segment mask off function is only meaningful for FIFO mode and auto increment mode. The external gate control (INn, or CLR) can override to disable the trigger output by external signal.

Let us begin to explain the segment mask off function from fig 6.5 segment mask off and external gate shown as follows. At the middle top, the counter is counting on the fly once your configuration is done. The A, B ,Z phase input signals determines the counter value and direction.

The counter value is sent to comparator at which another comparison source is selected from FIFO or Auto increment mode. If the two coming values are met, the comparator will generate a trigger to proceed with the auto increment state machine or pop up data from FIFO. But the trigger will going out as CMP output signal or not depends on the other control signals.

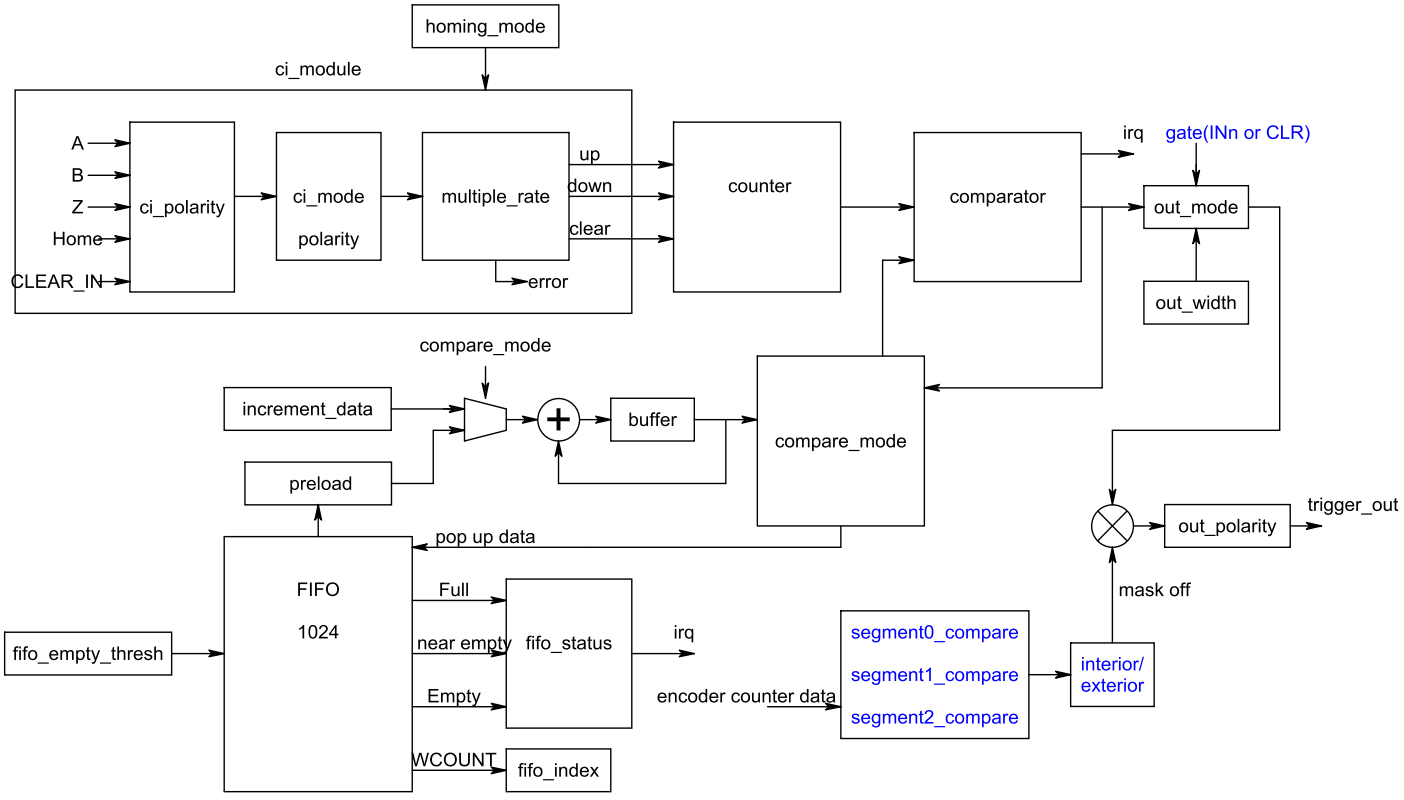


fig 6.5 segment mask off and external gate

At the right most, the CMP_OUT is the final output trigger, it is controlled by compare segment and interior/exterior mask off and external gate. The external gate signal comes from INn (X_IN0, Y_IN1, Z_IN2, A_IN3) or CLR (X_CLR, Y_CLR, Z_CLR, A_CLR) its polarity can be programmed as your physical hardware to gate the trigger signal to CMP output pin.

Refer to fig. 6.2 and fig 6.3, you can see the hardware mask and segment mask off (interior or exterior) example.

There are total 3 segments to configure. You can set the segment at a specific coordinate, say segment0 from1,000 ~ 10,000, then enable segment0. If you set mask off to interior, the compare equal pulses at interior of segment0 will be masked off and only the compare equal pulses of segment exterior can pass the compare out trigger. If you set mask off at exterior, only the compare equal pulses inside the segment0 can generate compare out trigger. The segment1 and segment2 also have the same function as segment0 does. If you disable the segment function, no segment mask off function will be of the disabled segment.

7. Function format and language difference

7.1 Function format

Every LSI3144A function is consist of the following format:

Status = function_name (parameter 1, parameter 2, … parameter n)

Each function returns a value in the Status global variable that indicates the success or failure of the function. A returned Status equal to zero that indicates the function executed successfully. A non-zero status indicates failure that the function did not execute successfully because of an error, or executed with an error.

Note: Status is a 32-bit unsigned integer.

The first parameter to almost every LSI3144A function is the parameter CardID which is located the driver of LSI3144A board you want to use those given operation. The CardID is assigned by rotary switch. You can utilize multiple devices with different card ID within one application; to do so, simply pass the appropriate CardID to each function.

Note: CardID is set by rotary switch (0x0-0xF)

7.2 Variable data types

Every function description has a parameter table that lists the data types for each parameter. The following sections describe the notation used in those parameter tables and throughout the manual for variable data types.

Primary Type Names					
Name	**Description**	**Range**	**C/C++**	**Visual BASIC**	**Pascal (Borland Delphi)**
u8	8-bit ASCII character	0 to 255	char	Not supported by BASIC. For functions that require character arrays, use string types instead.	Byte
i16	16-bit signed integer	-32,768 to 32,767	short	Integer (for example: deviceNum%)	SmallInt
u16	16-bit unsigned integer	0 to 65,535	unsigned short for 32-bit compilers	Not supported by BASIC. For functions that require unsigned integers, use the signed integer type instead. See the i16 description.	Word
i32	32-bit signed integer	-2,147,483,648 to 2,147,483,647	long	Long (for example: count&)	LongInt
u32	32-bit unsigned integer	0 to 4,294,967,295	unsigned long	Not supported by BASIC. For functions that require unsigned long integers, use the signed long integer type instead. See the i32 description.	Cardinal (in 32-bit operating systems). Refer to the i32 description.
f32	32-bit single-precision floating-point value	-3.402823E+38 to 3.402823E+38	float	Single (for example: num!)	Single
f64	64-bit double-precision floating-point value	-1.797683144862315E+308 to 1.797683144862315E+308	double	Double (for example: voltage Number)	Double

Table 2

7.3 Programming language considerations

Apart from the data type differences, there are a few language-dependent considerations you need to be aware of when you use the LSI3144A API. Read the following sections that apply to your programming language.

Note: Be sure to include the declaration functions of LSI3144A prototypes by including the appropriate LSI3144A header file in your source code. Refer to Chapter 4.1 Building applications with the LSI3144A software library for the header file appropriate to your compiler.

7.3.1 C/C++

For C or C++ programmers, parameters listed as Input/Output parameters or Output parameters are pass-by-reference parameters, which means a pointer points to the destination variable should be passed into the function. For example, the Read Port function has the following format:

```
Status = LSI3144A_input_read (u8 CardID, u8 axis, u8 point, u8 *state);
```

where CardID and axis and point are input parameters, and state is an output parameter.
Consider the following example:

```
u8 CardID, axis , point ;
u8 state,
u32 Status;
Status = LSI3144A_input_read( CardID, axis, point, &state);
```

7.3.2 Visual basic

The file LSI3144A.bas contains definitions for constants required for obtaining LSI Card information and declared functions and variable as global variables. You should use these constants symbols in the LSI3144A.bas, do not use the numerical values.

In Visual Basic, you can add the entire LSI3144A.bas file into your project. Then you can use any of the constants defined in this file and call these constants in any module of your program. To add the LSI3144A.bas file for your project in Visual Basic 4.0, go to the File menu and select the Add File... option. Select LSI3144A.bas, which is browsed in the LSI3144A \ API directory. Then, select Open to add the file to the project.

To add the LSI3144A.bas file to your project in Visual Basic 5.0 and 6.0, go to the Project menu and select Add Module. Click on the Existing tab page. Select LSI3144A.bas, which is in the LSI3144A \API directory. Then, select Open to add the file to the project.

7.3.3 Borland C++ builder

To use Borland C++ builder as development tool, you should generate a .lib file from the .dll file by implib.exe.

implib LSI3144ABC.lib LSI3144A.dll

Then add the LSI3144ABC.lib to your project and add

#include “LSI3144A.h” to main program.

Now you may use the dll functions in your program. For example, the Read Input function has the following format:

```
Status = LSI3144A_input_read (u8 CardID, u8 axis, u8 point, u8 *state);
```

where CardID and axis, point are input parameters, and state is an output parameter. Consider the following example:

```
u8 CardID, axis, point;
u8 state;
u32 Status;
Status = LSI3144A_read_input_status ( CardID, axis, point, &state);
```

8. Flow chart of application implementation

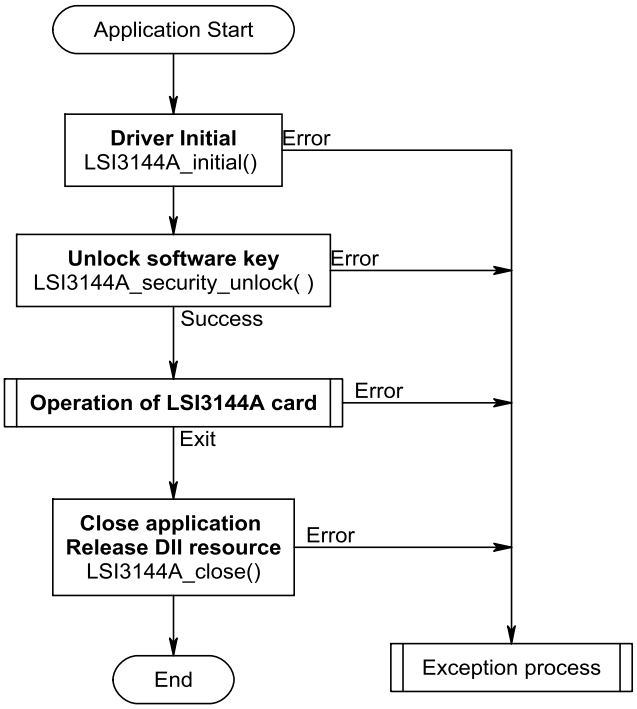


fig. 8-1 main application flow

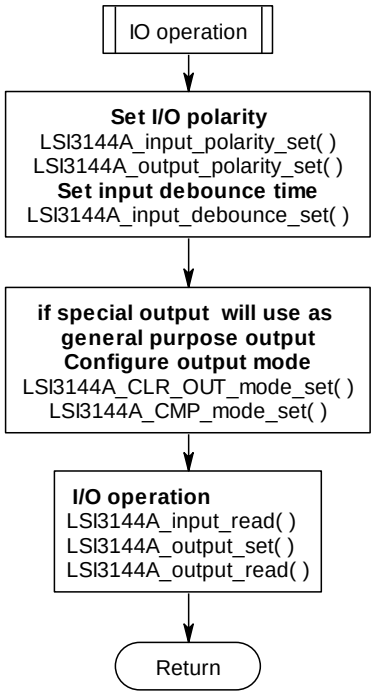


fig. 8-2 I/O operation flow

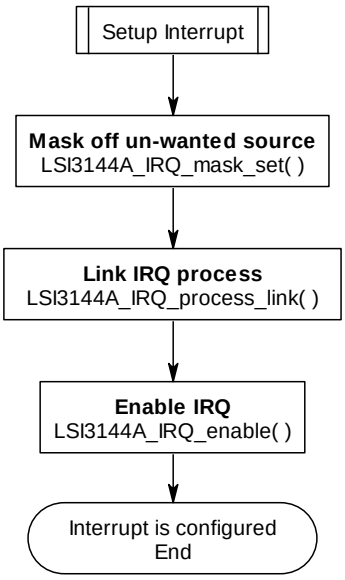


fig. 8-3 Interrupt set-up flow

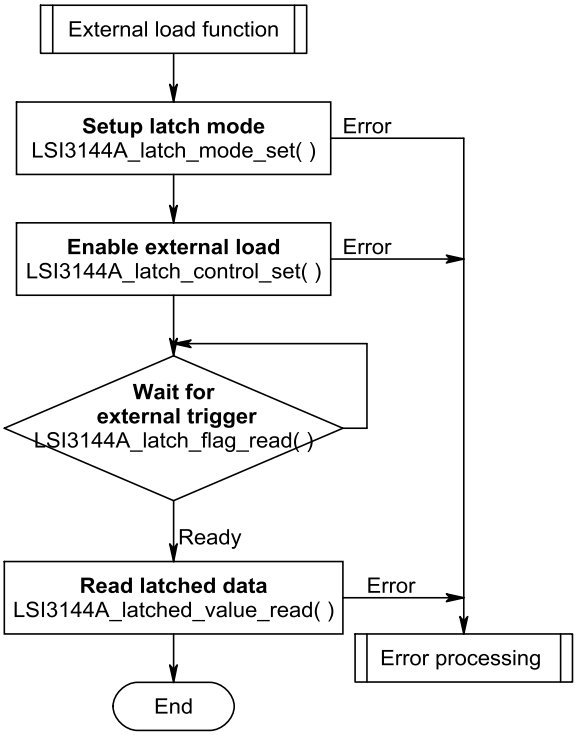


fig. 8-4 External load function flow

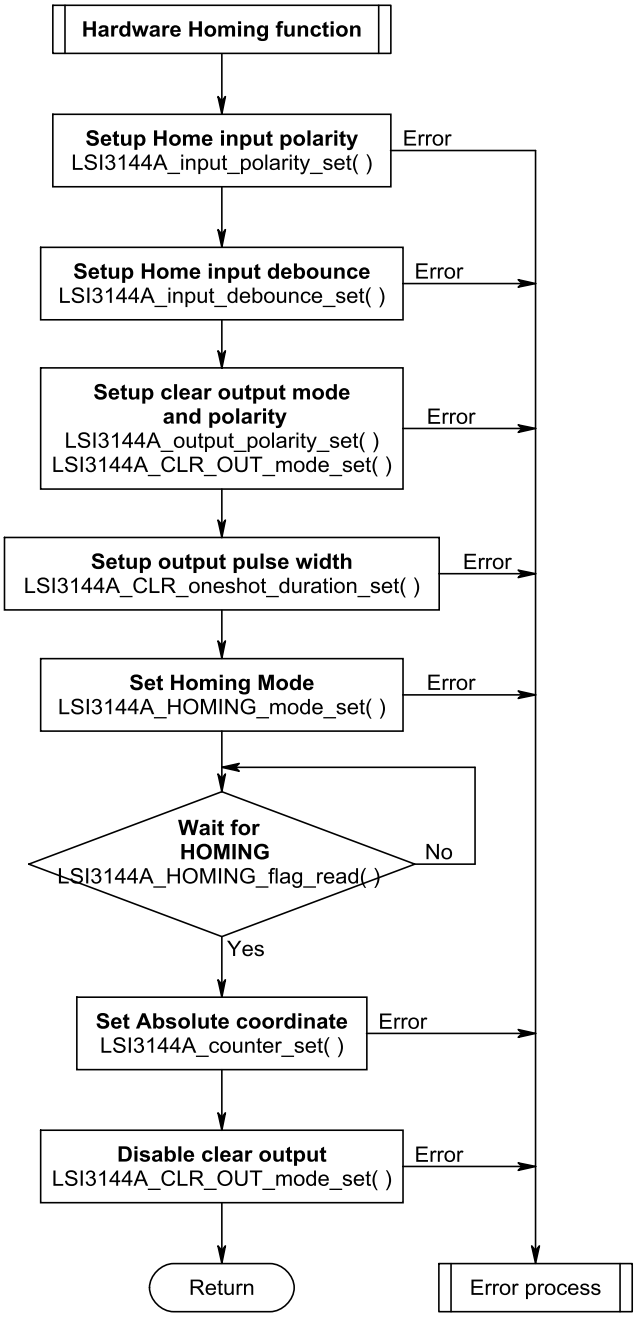


fig. 8-5 Hardware homing function flow

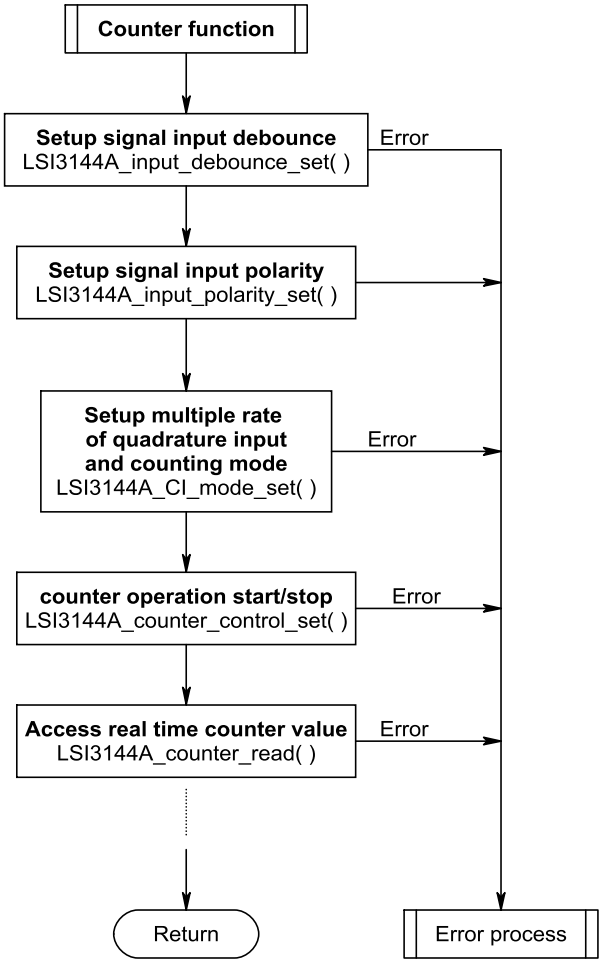


fig. 8-6 counter function flow

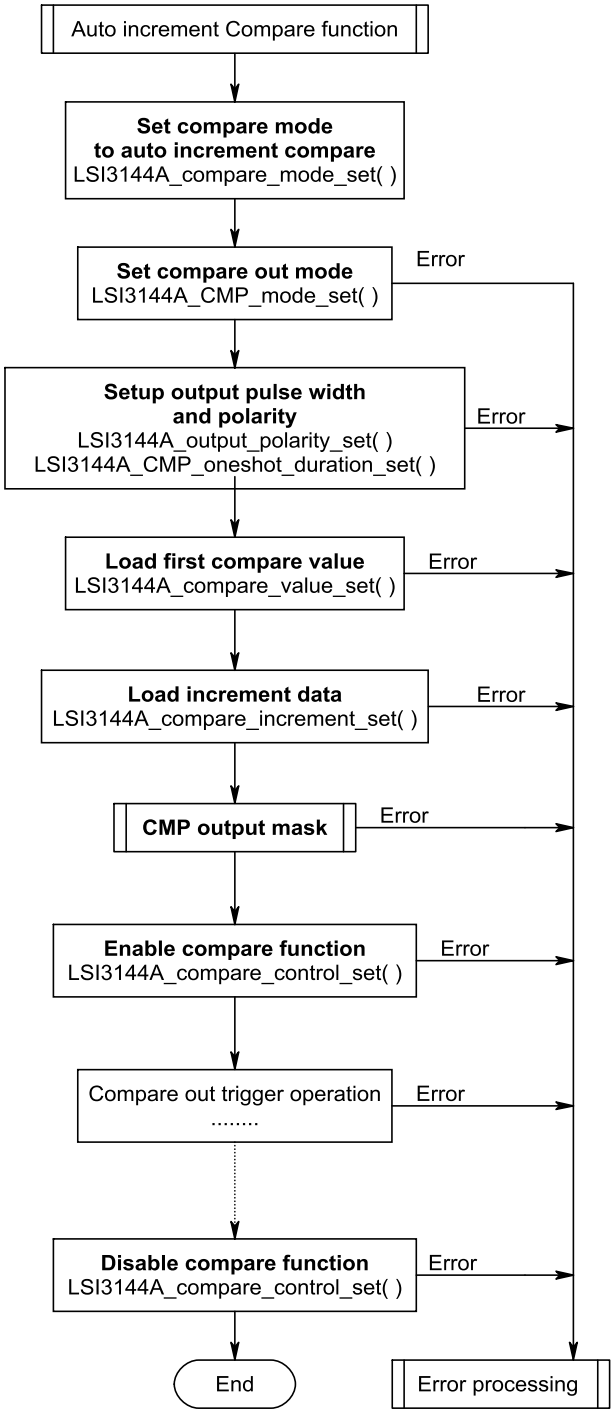


fig. 8-7 Auto-increment compare function flow

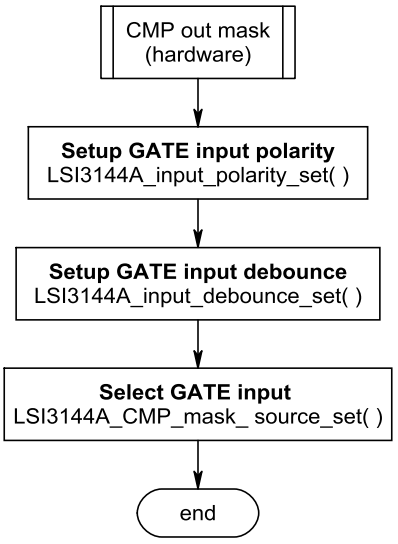


fig. 8-8 CMP out hardware mask function flow

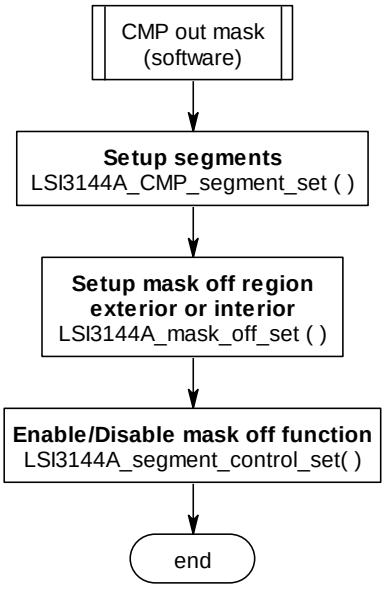


fig. 8-9 CMP out hardware mask function flow

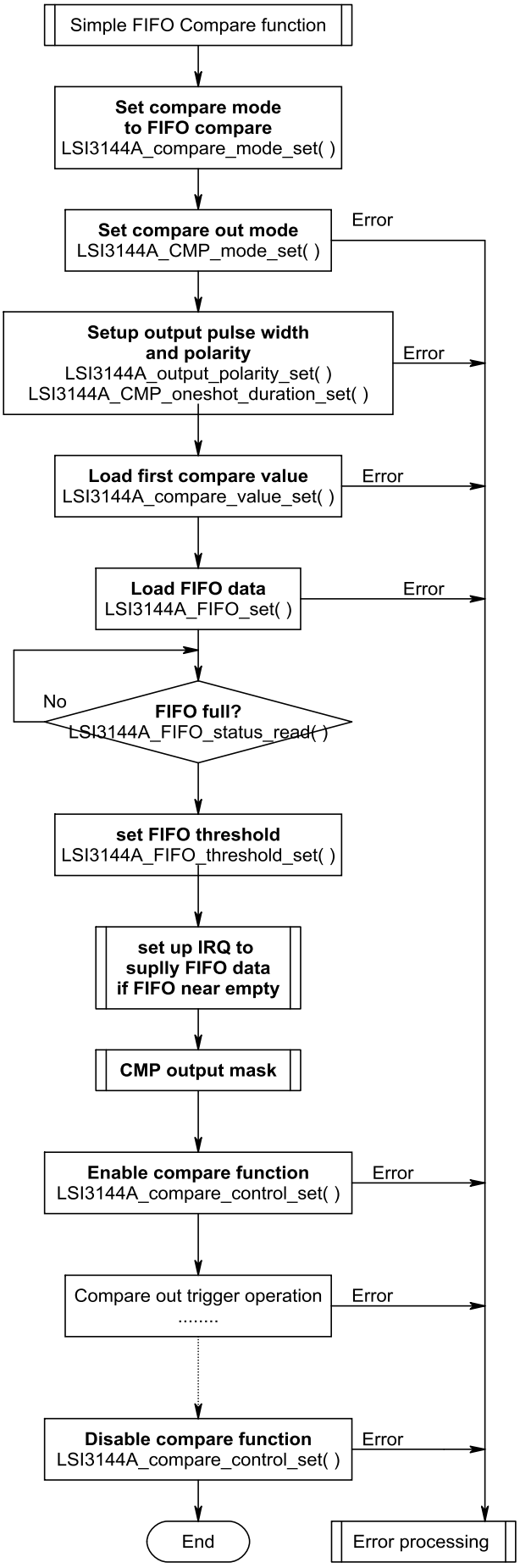


fig. 8-10 Simple FIFO compare function flow

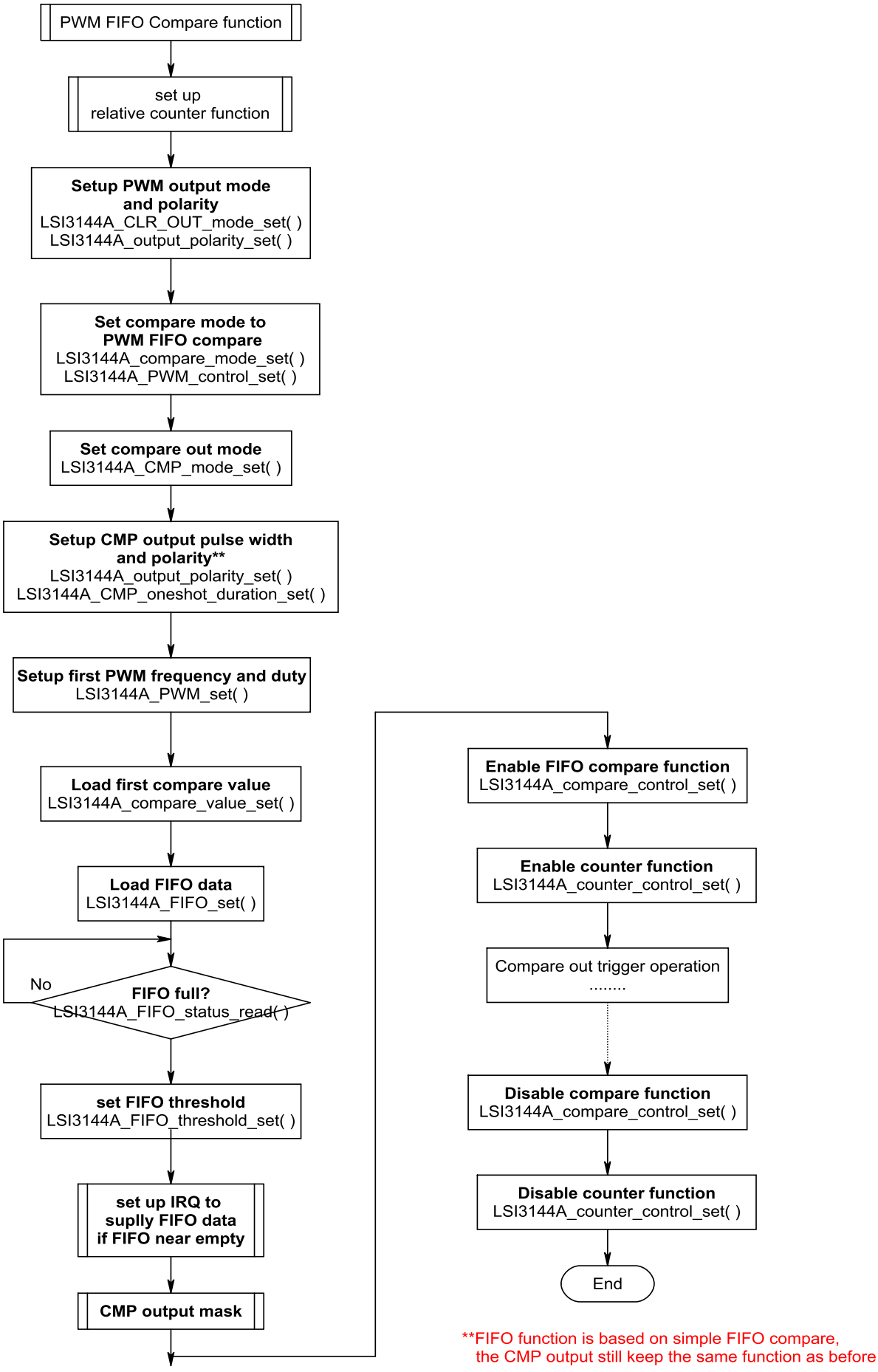


fig. 8-11 PWM FIFO compare function flow

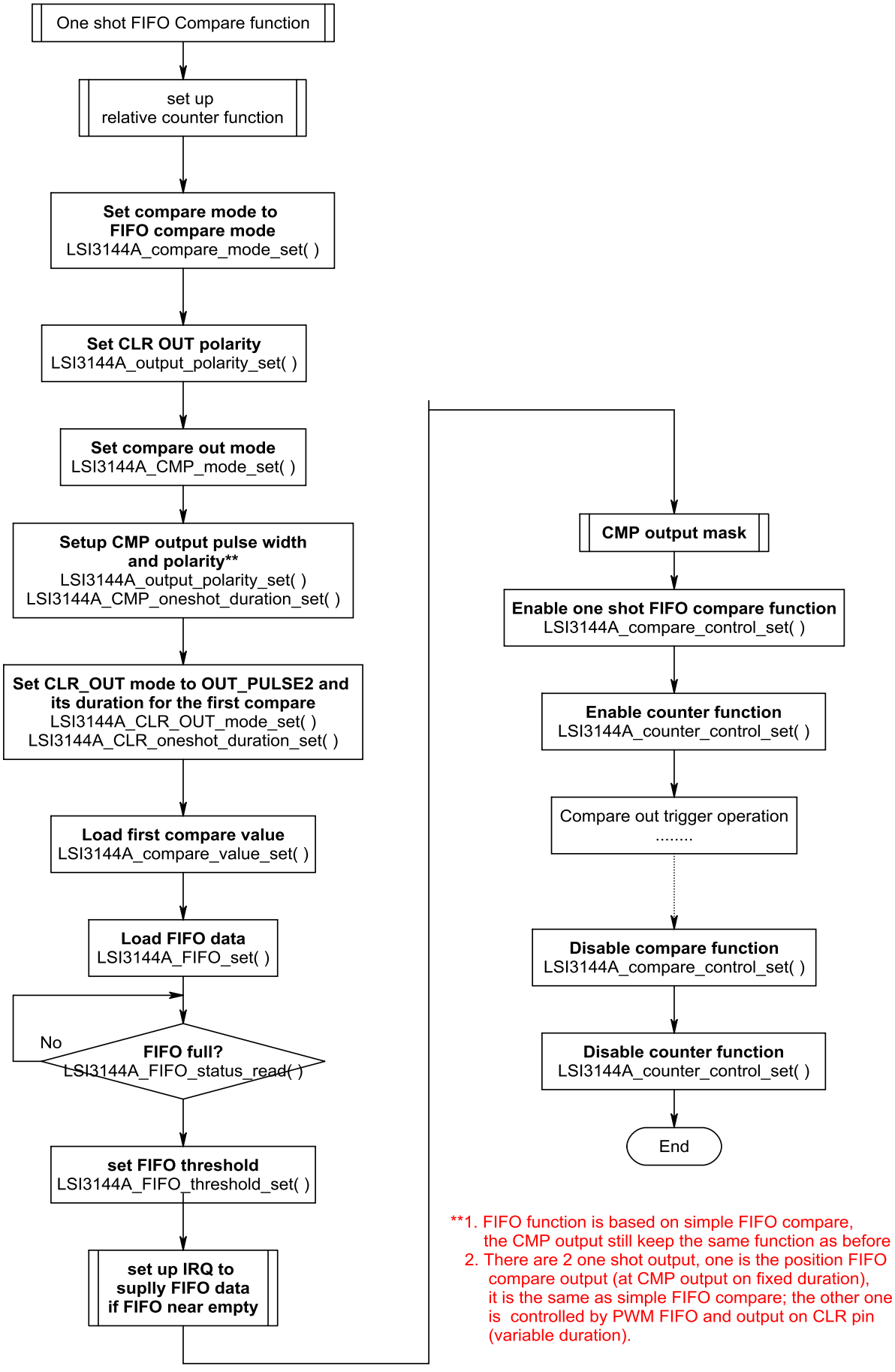


fig. 8-12 One shot FIFO compare function flow

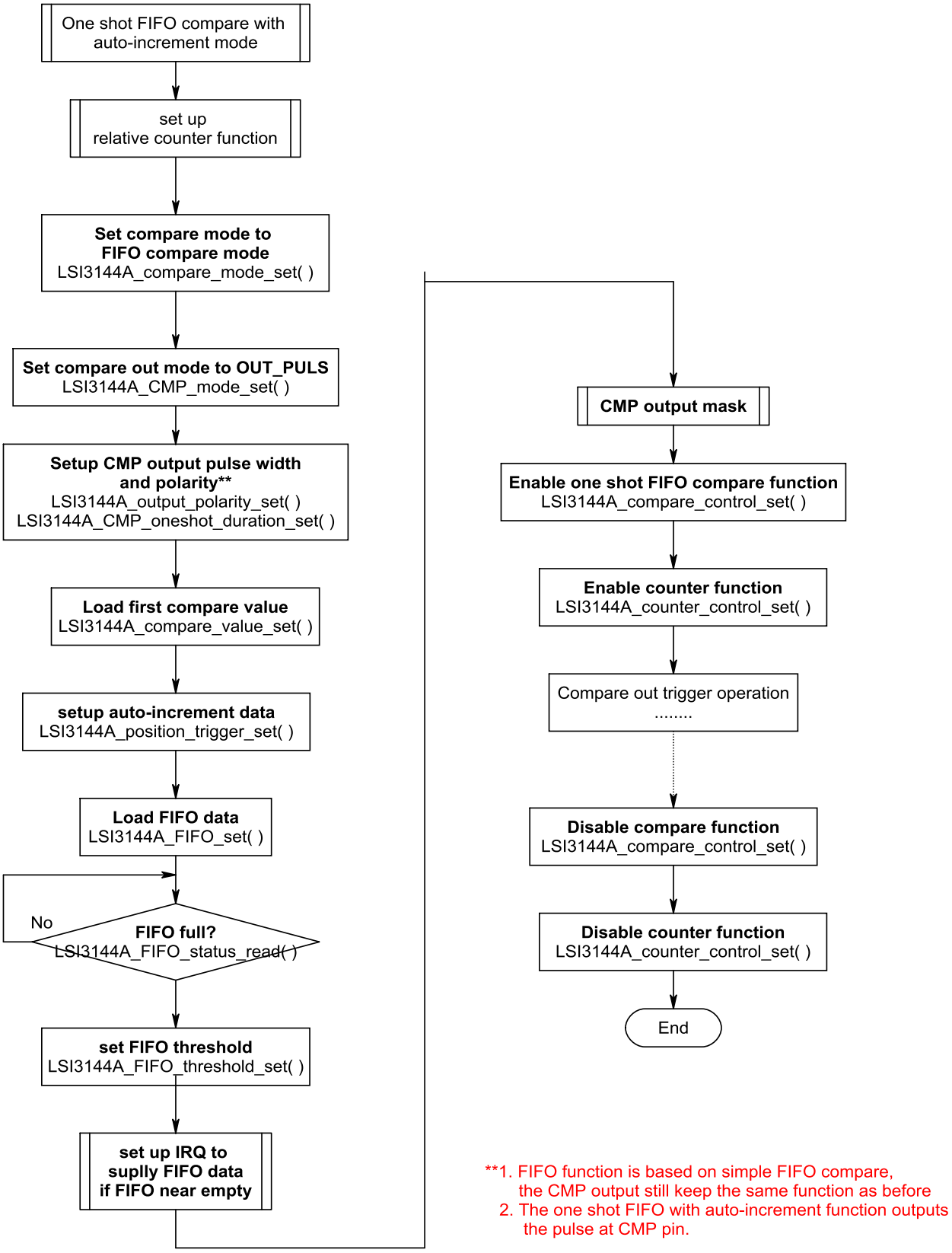


fig. 8-13 One shot FIFO compare with auto-increment function flow

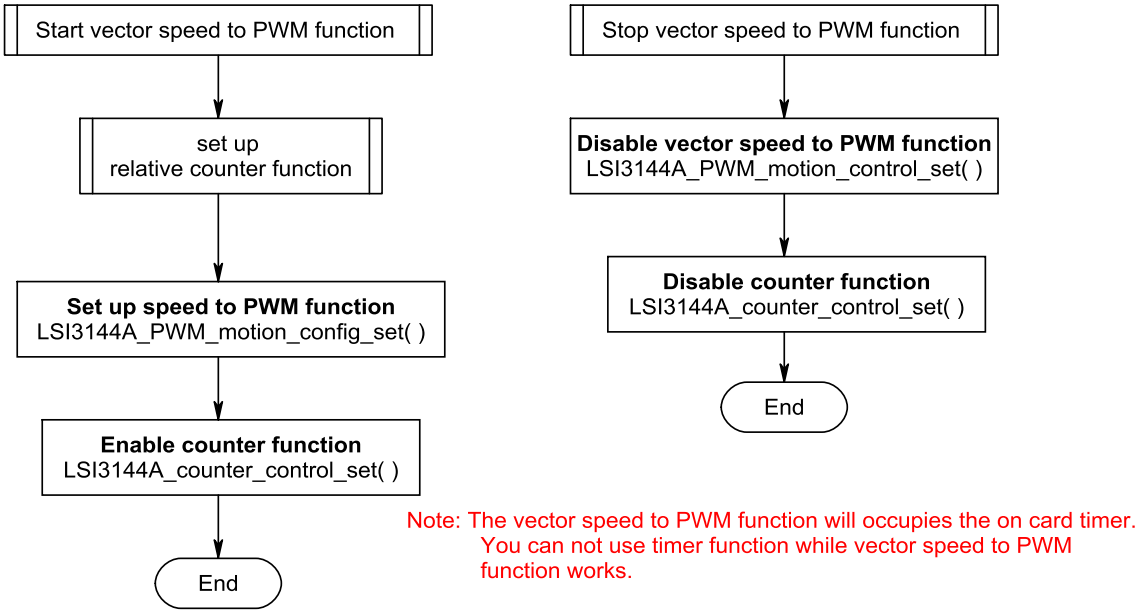


fig. 8-14 Vector speed to PWM function flow

9. Software overview and dll function

9.1 Compatibility of LSI3144A and LSI3144

The hardware of LSI3144A is super set function of LSI3144. If you want to keep the software compatible with the old version, you can install the new driver software and dll (new LSI3144.dll)to work with new card LSI3144A but the function convention remained as LSI3144. This means you do not modify the original software source code. (refer the LSI3144 software manual for the function it provides).

We recommend using new drivers in your new project or if you want to revise the project source code. This means use LSI3144A.h and LSI3144A.lib and LSI3144A.dll. and call functions in the new convention.

9.2 Initialization and close

You need to initialize system resource each time you run your application,

LSI3144A_initial( ) will do.

Once you want to close your application, call

LSI3144A_close( ) to release all the resource.

If you want to know the physical address assigned by OS, use

LSI3144A_info( ) to get the address.

- **LSI3144A_initial**

Format : **u32 status =LSI3144A_initial(void)**

Purpose: Initial the LSI3144A resource when start the Windows applications.

- **LSI3144A_close**

Format : **u32 Status = LSI3144A_close(void);**

Purpose: The LSI3144A_close () function is corresponded with LSI3144A_initial( ) function to make LSI3144A card windows application program completely ended and memory fully be released.

- **LSI3144A_info**

Format : **u32 status =LSI3144A_info(u8 CardID, u8 *CardType, u16 *address)**

Purpose: Read the physical I/O address assigned by O.S..

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW

Output:

Name	Type	Description
CardType	u8	0:LSI3144 1:LSI3144A
address	u16	physical I/O address assigned by OS

9.3 Input / Output function

For the easy use of digital input / output or the signal and control input / output, the logic polarity configure as you need will release the complexity of your application.

Note:

1. Each axis has its own A, B, Z phase, Home, LAH (latch input), CLR (clear input) and CMP (compare output), CLR_OUT (clear output) output.
2. General input are assigned as follows:
   IN0 maps to X axis, IN1 maps to Y axis, IN2 maps to Z axis, IN3 maps to A axis.

Use

LSI3144A_input_polarity_set( ) to set input logic polarity of A, B, Z phase, Home, LAH(latch input), CLR(clear input) and INn (general input).

LSI3144A_input_polarity_read( ) to read back the polarity setting.

The digital inputs (digital input, HOME, LAH and CLR signal inputs) may be contaminated by environment noise, filtering out the unwanted signal will be beneficial to the system stability. The LSI3144A card provides digital debounce function, you can choose the debounce time to optimize the function you need. To setup the input debounce filter,

LSI3144A_input_debounce_set( ) to set up the debounce filter required,

LSI3144A_input_debounce_read( ) to read back the debounce setting.

To read the input data by:

LSI3144A_input_read( )

At the output block, we also can configure the polarity by:

LSI3144A_output_polarity_set( ) and read back by:

LSI3144A_output_polarity_read( )

To output data using:

LSI3144A_output_set( ), output can also read back for verification by:

LSI3144A_output_read( )

- **LSI3144A_input_polarity_set**

Format : u32 status = LSI3144A_input_polarity_set(u8 CardID, u8 axis, u8 point, u8 polarity)

Purpose: To set input point polarity.

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axis	u8	0: X	1: Y
		2: Z	3: A
point	u8	0: INn (general) input point 1: HOME input point 2: LAH (latch) input point 3: CLR (clear)input point 4: A_PHASE input point 5: B_PHASE input point 6: Z_PHASE input point	
polarity	u8	0: negative polarity, input low active (default) 1: positive polarity, input high active	

- **LSI3144A_input_polarity_read**

Format : **u32 status = LSI3144A_input_polarity_read (u8 CardID, u8 axis, u8 point, u8 *polarity)**

Purpose: To read back polarity of input point.

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axis	u8	0: X	1: Y
		2: Z	3: A
point	u8	0: INn (general) input point 1: HOME input point 2: LAH (latch) input point 3: CLR (clear)input point 4: A_PHASE input point 5: B_PHASE input point 6: Z_PHASE input point	

Output:

Name	Type	Description
polarity	u8	0: negative polarity, input low active (default) 1: positive polarity, input high active

- **LSI3144A_input_debounce_set**

Format : **u32 status=LSI3144A_input_debounce_set(u8 CardID,u8 axis,u8 index,u8 data)**

Purpose: To set debounce time

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axis	u8	0:X	1:Y
		2:Z	3:A
index	u8	0: Digital (IN0~IN3) 1: HOME 2: LAH (latch) input point 3: CLR (clear) input point 4: encoder A,B,C(Z) input	
data	u8	if index=0; digital in 0: no debounce 1: 100 Hz 2: 200 Hz 3: 1K Hz (default) 4: 10K Hz if index=1,2,3; LATCH and CLR、HOME 0: no debounce 1: 100 Hz 2: 200 Hz 3: 1K Hz 4: 10K Hz 5: 50K Hz 6: 100K Hz 7: 500K Hz 8: 512K Hz 9: 1M Hz 10: 2M Hz (default) if index=4; A,B,Z input 0: filter out duration less than 1.95us signal, counter bandwidth less than 512K.	

data	u8	1: filter out duration less than 1us signal (default), counter bandwidth less than 1M. 2: filter out duration less than 0.5us signal, counter bandwidth less than 2M. 3: filter out duration less than 0.25us signal, counter bandwidth less than 4M. 4: filter out duration less than 0.125us signal, counter bandwidth less than 8M. 5: filter out duration less than 0.1us signal, counter bandwidth less than 10M. 6: filter out duration less than 0.0625us signal, counter bandwidth less than 16M.

- **LSI3144A_input_debounce_read**

Format : u32 status=LSI3144A_input_debounce_read (u8 CardID,u8 axis,u8 index, u8 *data)

Purpose: To read back debounce setting

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axis	u8	0:X	1:Y
		2:Z	3:A
index	u8	0: Digital (IN0~IN3) 1: HOME 2: LAH (latch) input point 3: CLR (clear) input point 4: encoder A,B,C(Z) input	

Output:

Name	Type	Description
data	u8	if index=0; digital in 0: no debounce 1: 100 Hz 2: 200 Hz 3: 1K Hz (default) 4: 10K Hz

data	u8	if index=1,2,3; LATCH and CLR、HOME 0: no debounce 1: 100 Hz 2: 200 Hz 3: 1K Hz 4: 10K Hz 5: 50K Hz 6: 100K Hz 7: 500K Hz 8: 512K Hz 9: 1M Hz 10: 2M Hz (default) if index=4; encoder A,B,Z input 0: filter out duration less than 1.95us signal, counter bandwidth less than 512K. 1: filter out duration less than 1us signal (default), counter bandwidth less than 1M. 2: filter out duration less than 0.5us signal, counter bandwidth less than 2M. 3: filter out duration less than 0.25us signal, counter bandwidth less than 4M. 4: filter out duration less than 0.125us signal, counter bandwidth less than 8M. 5: filter out duration less than 0.1us signal, counter bandwidth less than 10M. 6: filter out duration less than 0.0625us signal, counter bandwidth less than 16M.

- **LSI3144A_input_read**

Format : **u32 status = LSI3144A_input_read (u8 CardID, u8 axis, u8 point, u8 *state)**

Purpose: To read input status.

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axis	u8	0: X	1: Y
		2: Z	3: A
point	u8	0: INn (general) input point 1: HOME input point 2: LAH (latch) input point 3: CLR (clear)input point 4: A_PHASE input point 5: B_PHASE input point 6: Z_PHASE input point 7: Z_PHASE trigger toggled flag	

Output:

Name	Type	Description
state	u8	0: inactive 1: active

- **LSI3144A_output_polarity_set**

Format : **u32 status = LSI3144A_output_polarity_set (u8 CardID, u8 axis, u8 point, u8 polarity)**

Purpose: To set output polarity.

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axis	u8	0: X	1: Y
		2: Z	3: A
point	u8	0: choose CMP point 1: choose CLR_OUT point	
polarity	u8	0: output NO. type (default) 1: output NC. type	

- **LSI3144A_output_polarity_read**

Format : **u32 status = LSI3144A_output_polarity_read (u8 CardID, u8 axis, u8 point, u8 *polarity)**

Purpose: To read back polarity of output point.

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axis	u8	0: X	1: Y
		2: Z	3: A
point	u8	0: choose CMP point 1: choose CLR_OUT point	

Output:

Name	Type	Description
polarity	u8	0: output NO. type (default) 1: output NC. type

- **LSI3144A_output_set**

Format : **u32 status = LSI3144A_output_set (u8 CardID, u8 axis, u8 point, u8 on_off)**

Purpose: To write output point.

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axis	u8	0: X	1: Y
		2: Z	3: A
point	u8	0: choose CMP point 1: choose CLR_OUT point	
on_off	u8	0: inactive 1: active	

- **LSI3144A_output_read**

Format : **u32 status = LSI3144A_output_read (u8 CardID, u8 axis, u8 point, u8 *state)**

Purpose: To read back status of output point.

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axis	u8	0: X	1: Y
		2: Z	3: A
point	u8	0: choose CMP point 1: choose CLR_OUT point	

Output:

Name	Type	Description
state	u8	0: inactive 1: active

9.4 Homing (to clear counter) and clear input and clear output function

At the beginning of an application, the position of encoder / linear scale needs a reference point of coordinate, use

LSI3144A_HOMING_mode_set( ) to clear counter while the special hardware condition meet.

LSI3144A_HOMING_mode_read( ) can be used to verify the mode you set.

To check if hardware homing occurred, use

LSI3144A_HOMING_flag_read( ) to read homing flag to check.

If anytime you want to clear counter,

LSI3144A_HOMING_software( ) will do.

The CLR input can use as gate of CMP output or as asynchronous clear input, if work in asynchronous clear mode, after configure the polarity (***LSI3144A_input_polarity_***set) and debounce time (***LSI3144A_input_debounce_***set); just connect the signal to the input pin the it will function as hardware asynchronous input. If CLR input work as gate of CMP output, please refer 9.10 Compare output mask function.

The clear output CLR_OUT pin can be configured as

1. general digital output,
2. pulsed output on the homing condition meet
3. PWM output controlled by PWM function (PWM FIFO or single PWM mode).

You can configure the function as you need by:

LSI3144A_CLR_OUT_mode_set( ) can set the mode.

You can read back to verify the setting by:

LSI3144A_CLR_OUT_mode_read( )

The output pulse duration of all axes CLR output can be set by

LSI3144A_CLR_oneshot_duration_set( )

- **LSI3144A_HOMING_mode_set**

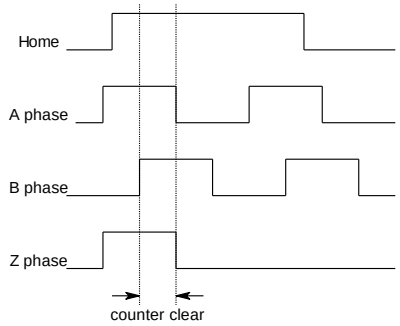
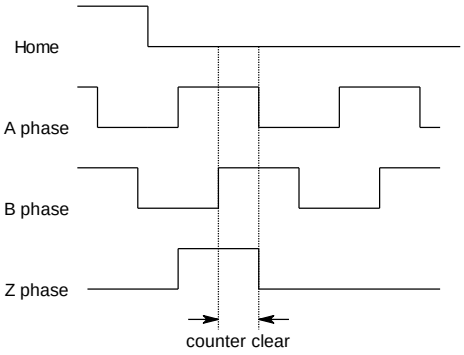
Format : u32 status = LSI3144A_HOMING_mode_set (u8 CardID,u8 axis,
u8 homing_mode, u16 z_count, u8 single_cont)

Purpose: To set homing mode of high speed counter.

Parameters:

Input:

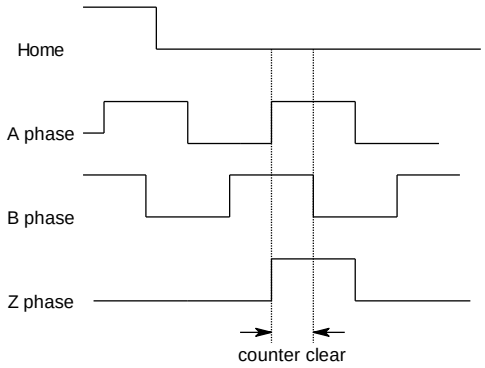
Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axis	u8	0:X	1:Y
		2:Z	3:A

homing_mode	u8	0: NORMAL (default) NORMAL is used for counting or compare. While the homing mode completes, the homing mode will reset to NORMAL. 1: HOME_ABZ Clear counter while A,B,Z and HOME signals are MAKE simultaneously. 1. counter clear at A,B,Z and Home active  2: HOME_ABZ_UP Clear counter at first A,B,Z are MAKE after HOME signal turned to BREAK and counter up count. 2. counter clear at first A,B,Z active after HOME turn to inactive and up count 

3: HOME_ABZ_DOWN

Clear counter at first A,B,Z are MAKE after HOME signal turned to BREAK and counter down count.

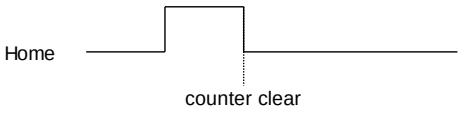
3. counter clear at first A,B,Z active
after HOME turn to inactive and down count



4: HOME_

Clear counter at the tailing edge of HOME input.

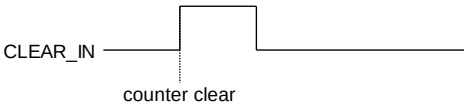
4. counter clear at tailing edge of HOME



5: H_CLEAR_IN

Clear counter while CLEAR_IN input active transition.

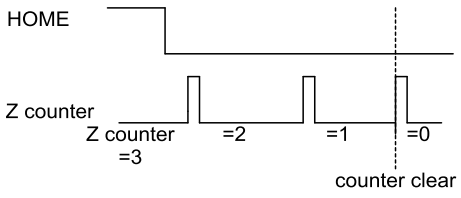
5. counter clear at rising edge of CLEAR_IN

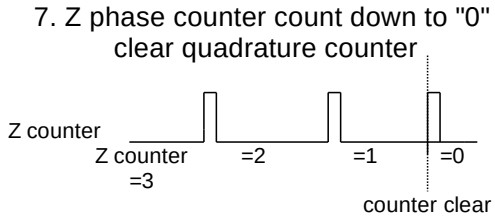


6: HOME_ZN

Clear counter while HOME active to enactive and Z phase counts z-count pulses.

6. Trailing edge of HOME starts Z phase counter
and count down to "0" clear quadrature counter



		7:H_ZN Clear counter while Z phase counts z-count pulses. 
z_count	u16	Z phase count pulses at HOME_ZN and H_ZN homing mode.
single_cont	u8	0: SINGLE, once counter clears, homing mode reset to NORMAL. 1: CONT, continuous mode, always doing homing function.

- **LSI3144A_HOMING_mode_read**

Format : u32 status = LSI3144A_HOMING_mode_read (u8 CardID,u8 axis, u8 *homing_mode, u16 *z_count, u8 *single_cont)

Purpose: To read back high speed counter homing mode setting.

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axis	u8	0: X	1: Y
		2: Z	3: A

Output:

Name	Type	Description
homing_mode	u8	Refer explanation of LSI3144A_HOMING_mode_set
z_count	u16	
single_cont	u8	

- **LSI3144A_HOMING_flag_read**

Format : **u32 status = LSI3144A_HOMING_flag_read (u8 CardID, u8 axis, u8 *flag)**

Purpose: Read hardware homing flag.

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axis	u8	0: X	1: Y
		2: Z	3: A

Output:

Name	Type	Description
flag	u8	0: no operation. 1: hardware homing happened, after reading, the flag will reset to 0

Note:

The homing flag only can read once each time hardware homing happens; because the function will clear the flag after it been read.

Hardware homing can also lead an immediate interrupt. To run the interrupt program, interrupt related functions must set before the interrupt generated.

- **LSI3144A_HOMING_software**

Format : **u32 status = LSI3144A_HOMING_software (u8 CardID, u8 axis)**

Purpose: To clear counter.

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axis	u8	0: X	1: Y
		2: Z	3: A

- **LSI3144A_CLR_OUT_mode_set**

Format : **u32 status = LSI3144A_CLR_OUT_mode_set(u8 CardID,u8 axis, u8 mode)**

Purpose: To set the clear out

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axis	u8	0: X	1: Y
		2: Z	3: A
mode	u8	0: NORMAL_OUT CLEAR_OUT pin as general digital output (can be controlled by ***LSI3144A_output_set***) 1: OUT_PULSE CLEAR_OUT pin output pulse as homing condition meet. (***LSI3144A_CLR_oneshot_duration_set***) 2:OUT_TOGGLE CLEAR_OUT pin toggles output as counter cross zero . (***LSI3144A_PWM_set*** or ***LSI3144A_FIFO_set***), it also as PWM output while in PWM_FIFO mode 3: OUT_PULSE2 CLEAR_OUT pin as one shot output while PWM FIFO work in one shot mode	

Note: Refer PWM FIFO function at p.82 PWM FIFO compare mode

- **LSI3144A_CLR_OUT_mode_read**

Format : **u32 status = LSI3144A_CLR_OUT_mode_read (u8 CardID, u8 axis, u8 *mode)**

Purpose: Read back the clear out mode

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axis	u8	0: X	1: Y
		2: Z	3: A

Output:

Name	Type	Description
mode	u8	0: NORMAL_OUT CLEAR_OUT pin as general digital output (can be controlled by ***LSI3144A_output_set***) 1: OUT_PULSE CLEAR_OUT pin output pulse as homing condition meet. (***LSI3144A_CLR_oneshot_duration_set***) 2:OUT_TOGGLE CLEAR_OUT pin toggles output as counter cross zero . (***LSI3144A_PWM_set*** or ***LSI3144A_FIFO_set***), it also as PWM output while in PWM_FIFO mode 3: OUT_PULSE2 CLEAR_OUT pin as one shot output while PWM FIFO work in one shot mode

- **LSI3144A_CLR_oneshot_duration_set**

Format : u32 status =LSI3144A_CLR_oneshot_duration_set (u8 CardID, u8 axis , u16 value)

Purpose: To set the one shot duration time for all CLR_OUT (clear output).

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axis	u8	0: X	1: Y
		2: Z	3: A
value	u16	Duration time = value * 1us 1<= value <= 65535 set 0 equal to 65536.	

Note: CLR_OUT initial state can be defined by ***LSI3144A***_input_polarity_set or ***LSI3144A_output_set***.

9.5 Basic counter function

The major function of quadrature encoder / linear scale counter card is to count the input pulse train accurately. The input signal is recommended to set adequate debounce filter to filter out the unwanted signal (***LSI3144A_input_debounce_***set) and setup the signal polarity (***LSI3144A_input_polarity_***set). Then, set counter input mode (multiple rate, dual pulse or single pulse input…) as you need by:

LSI3144A_CI_mode_set( ) and read back to verify by

LSI3144A_CI_mode_read( )

The counter operation (basic counter function) can be enabled / disabled by:

LSI3144A_counter_control_set( ) and read back by:

LSI3144A_counter_control_read( )

If you want to set counter value, for example, after homing the counter value =0 but you want the homing point has the coordinate value at 1000, then you can set the counter by:

LSI3144A_counter_set( ) to load the counter.

To read the counter value (get counter data on the fly) at any time, use

LSI3144A_counter_read( )

- **LSI3144A_CI_mode_set**

Format : u32 status = LSI3144A_CI_mode_set (u8 CardID,u8 axis, u8 in_mode, u8 multiple_rate)

Purpose: To set high speed counter input mode.

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axis	u8	0:X	1:Y
		2:Z	3:A
in_mode	u8	0: A,B phases input quadrature up count mode(if A lead B).(default) 1: A,B phases input quadrature down count mode(if A lead B). 2: A input is CLOCK,B input is DIRECTION, up count mode. 3: A input is CLOCK,B input is DIRECTION, down count mode. 4: A input is UP CLOCK,B input is DOWN CLOCK, dual clock mode. 5: A input is DOWN CLOCK,B input is UP CLOCK, dual clock mode.	
multiple_rate	u8	Only valid for quadrature mode, in other mode, this parameter is ignored. 0: MULTIPLE_4 (default) A,B phase input multiple rate is 4 1: MULTIPLE_2 A,B phase input multiple rate is 2 2: MULTIPLE_1 A,B phase input multiple rate is 1	

- **LSI3144A_CI_mode_read**

Format : u32 status = LSI3144A_CI_mode_read (u8 CardID,u8 axis, u8 *in_mode, u8 *multiple_rate)

Purpose: To read back the counter input mode.

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axis	u8	0:X	1:Y
		2:Z	3:A

Output:

Name	Type	Description
in_mode	u8	0: A,B phases input quadrature up count mode(if A lead B).(default) 1: A,B phases input quadrature down count mode(if A lead B). 2: A input is CLOCK,B input is DIRECTION, up count mode. 3: A input is CLOCK,B input is DIRECTION, down count mode. 4: A input is UP CLOCK,B input is DOWN CLOCK, dual clock mode. 5: A input is DOWN CLOCK,B input is UP CLOCK, dual clock mode.
multiple_rate	u8	Only valid for quadrature mode, in other mode, this parameter is ignored. 0: MULTIPLE_4 (default) A,B phase input multiple rate is 4 1: MULTIPLE_2 A,B phase input multiple rate is 2 2: MULTIPLE_1 A,B phase input multiple rate is 1

- **LSI3144A_counter_control_set**

Format : **u32 status = LSI3144A_counter_control_set (u8 CardID, u8 axis, u8 mode)**

Purpose: To set up counter operation mode.

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY switch	
axis	u8	0:X	1:Y
		2:Z	3:A
mode	u8	0: STOP_COUNTER counter stops. 1: NORMAL_COUNTER operation in counter mode (including HOMING). (default)	

- **LSI3144A_counter_control_read**

Format : **u32 status = LSI3144A_counter_control_read (u8 CardID, u8 axis, u8 *mode)**

Purpose: To read back counter operation mode.

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY switch	
axis	u8	0:X	1:Y
		2:Z	3:A

Output:

Name	Type	Description
mode	u8	0: STOP_COUNTER counter stopped. 1: NORMAL_COUNTER operation in counter mode (including HOMING). (default)

- **LSI3144A_counter_set**

Format : **u32 status = LSI3144A_counter_set (u8 CardID, u8 axis, i32 value)**

Purpose: To load value into counter.

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axis	u8	0: X	1: Y
		2: Z	3: A
value	i32	32bit counter value to be load. (-2,147,483,648 ~ 2,147,483,647)	

Note:

Use ***LSI3144A_counter_set*** to set current position after the hardware homing means you define the coordinate origin (at certain point).

- **LSI3144A_counter_read**

Format : **u32 status = LSI3144A_counter_read(u8 CardID, u8 axis, i32 *value)**

Purpose: To read LSI3144A card counter. The 32bit counter on the fly value will return.

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axis	u8	0: X	1: Y
		2: Z	3: A

Output:

Name	Type	Description
value	i32	32bit counter value. (-2,147,483,648 ~ 2,147,483,647)

9.6 Counter latch / load mode

Counter can be external triggered to latch the current counter value or external triggered to load preset value. If your application needs to load counter on external trigger, you must preset a value at the counter buffer then the external trigger comes in to load the buffer value to the counter. Use

LSI3144A_counter_preset( ) to set up the counter buffer data.

LSI3144A_counter_preset_read( ) to read back preset data.

To setup the working mode as latch or load by:

LSI3144A_latch_mode_set( ) to set up as latch mode or external load mode. After set up these functions, once the hardware external trigger (nLAH) input active, immediately the preset value will transfer to counter by hardware mechanism under load mode or the counter value latched on the fly under latch mode.

The latch mode can be read back by:

LSI3144A_latch_mode_read( )

After setup working mode, you can enable or disable the latch/load function by:.

LSI3144A_latch_control_set( ) and read back to verify by:

LSI3144A_latch_control_read( )

Owing the external trigger to latch signal is asynchronous, you need to verify if the trigger has happened or not by:

LSI3144A_latch_flag_read( ) and after the confirmation, the latched data read by:

LSI3144A_latched_value_read( )

If you want to latch all axes with one trigger input, ***LSI3144A_com_trigger_control*** (ref. p91) will do.

- **LSI3144A_counter_preset**

Format : **u32 status = LSI3144A_counter_preset (u8 CardID, u8 axis, i32 preset_value)**

Purpose: To set value into preset buffer for counter external load.

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axis	u8	0: X	1: Y
		2: Z	3: A
preset_value	i32	32bit value to be preset. (-2,147,483,648 ~ 2,147,483,647)	

- **LSI3144A_counter_preset_read**

Format : **u32 status = LSI3144A_counter_preset_read (u8 CardID, u8 axis, i32 *preset_value)**

Purpose: To read preset buffer data.

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axis	u8	0: X	1: Y
		2: Z	3: A

Output:

Name	Type	Description
preset_value	i32	32bit value to be preset. (-2,147,483,648 ~ 2,147,483,647)

- **LSI3144A_latch_mode_set**

Format : **u32 status = LSI3144A_latch_mode_set(u8 CardID, u8 axis, u8 mode)**

Purpose: To assign the hardware external trigger input function.

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axis	u8	0: X	1: Y
		2: Z	3: A
mode	u8	0 : continuous external trigger to latch counter on the fly. (default) 1: one-shot external trigger to latch counter on the fly. Once triggered the further trigger will be disabled. New trigger should be enable by ***LSI3144A_latch_control( )***. 2: continuous external trigger load (counter) mode. 3: one-shot external trigger load mode. Once triggered the further trigger will be disabled. New trigger should be enable by ***LSI3144A_latch_control( )***.	

Note:

If you will load the counter from external trigger

1. load preset counter first (***LSI3144A_counter_***preset)
2. select load mode (***LSI3144A_latch_mode_set***)
3. enable load function (***LSI3144A_latch_control_set***)
4. hardware now is waiting the external trigger, once the hardware external trigger (nLAH) input occurs at its active edge, immediately the preset value will transfer to counter by hardware mechanism, simultaneously interrupt request may occur.

If you want to latch counter data on the fly (say, for touch probe)

1. select latch mode (***LSI3144A_latch_mode_set***)
2. enable latch function (***LSI3144A_latch_control_set***)
3. hardware now is waiting the external trigger, once the hardware external trigger (nLAH) input occurs at its active edge, immediately the counter value will be latched by hardware mechanism, simultaneously interrupt request may occur.

- **LSI3144A_latch_mode_read**

Format : **u32 status = LSI3144A_latch_mode_read(u8 CardID, u8 axis, u8 *mode)**

Purpose: To read back the hardware external trigger input mode.

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axis	u8	0: X	1: Y
		2: Z	3: A

Output:

Name	Type	Description
mode	u8	0 : continuous external trigger latch (counter) mode. (default) 1: one-shot external trigger latch mode. Once triggered the further trigger will be disabled. New trigger should be enable by ***LSI3144A_latch_control( )***. 2: continuous external trigger load (counter) mode. 3: one-shot external trigger load mode. Once triggered the further trigger will be disabled. New trigger should be enable by ***LSI3144A_latch_control( )***.

- **LSI3144A_latch_control_set**

Format : **u32 status = LSI3144A_latch_control_set(u8 CardID, u8 axis, u8 control)**

Purpose: To enable/disable latch (load) function, which is designated to nLAH (latch) input.

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axis	u8	0: X	1: Y
		2: Z	3: A
control	u8	0:disable latch/load function. 1:enable latch/load function. (Default value)	

- **LSI3144A_latch_control_read**

Format : **u32 status = LSI3144A_latch_control_read(u8 CardID, u8 axis, u8 *control)**

Purpose: To read back setting of the latch/load control mode.

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axis	u8	0: X	1: Y
		2: Z	3: A

Output:

Name	Type	Description
control	u8	0:disable latch/load function. 1:enable latch/load function. (Default value)

- **LSI3144A_latch_flag_read**

Format : **u32 status = LSI3144A_latch_flag_read (u8 CardID, u8 axis, u8 *flag)**

Purpose: To read latch flag, which triggered by nLAH input.

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axis	u8	0: X	1: Y
		2: Z	3: A

Output:

Name	Type	Description
flag	u8	0: no operation. 1: occurrence of latch/load trigger event.

Note: To read the latch flag will reset latch flag to no operation state.

- **LSI3144A_latched_value_read**

Format : **u32 status = LSI3144A_latched_value_read (u8 CardID, u8 axis, i32 *value)**

Purpose: To read LSI3144A counter latched value.

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axis	u8	0: X	1: Y
		2: Z	3: A

Output:

Name	Type	Description
value	i32	32bit counter latched value. (-2,147,483,648 ~ 2,147,483,647)

9.7 Basic compare function

Compare the counter to a preset value is a useful but special function. In application that needs to trigger external devices on the fly at specific point, the compare function is a good solution.

There are 3 compare modes provided by LSI3144A card:

- one time compare mode

the compare output (physical output at CMP output) will be triggered while the preset compare data equals the counter on the fly data. This is the basic compare function.

- auto increment compare mode

the compare output (physical output at CMP output) will be triggered while the preset compare data equals the counter on the fly data. The next compare data will be loaded based on the preset increment value and current compare data.

- FIFO compare mode (refer. 9.9 FIFO compare mode)

Some application not only need the position preset to compare but also need to take care of the motion direction, say, position 1000 but only on up count is valid compare condition. The LSI3144A on X, Z axes also provides the parameter to identify the motion direction.

To setup the compare mode:

LSI3144A_compare_mode_set( ) is used to define the operation mode.

Use

LSI3144A_compare_mode_read( ) to read back setting.

No matter what compare mode you want to use, the first compare data must setup.

To setup the first compare data by using:

LSI3144A_compare_value_set( ) and read back to verify by:

LSI3144A_compare_value_read( )

If you need to identify the compare data with direction information, use:

LSI3144A_direction_compare_value_set( ) and read back to verify by:

LSI3144A_direction_compare_value_read( )

The purpose of compare function is to generate a trigger signal for the specific usage. LSI3144A provides 4-output mode to meet your application.

- NO_OUT

Use CMP as general digital output, compare function does not give trigger.

- OUT_PULSE

CMP output pulse and the pulse width is programmable.

- OUT_LEVEL

CMP output is preset at high or low level then on the occurrence of compare equal, the CMP changes state. (Note: Once the state changed, it will not response to next coming compare equal signal)

- OUT_TOGGLE

CMP output is preset at high or low level then on each time the occurrence of compare equal the CMP toggles state.

Now configure the compare output (CMP output) mode as you need by:

LSI3144A_CMP_mode_set( ) and read back to verify by:

LSI3144A_CMP_mode_read( )

Setting the CMP output duration by:

LSI3144A_CMP_oneshot_duration_set( ) and read back to verify by:

LSI3144A_CMP_oneshot_duration_read( )

A new approach to use the combo function to setup mode and output pulse width:

LSI3144A_compare_CMP_OUT_set( ) and read back by:

LSI3144A_compare_CMP_OUT_read( )

Of course, you must enable the counter function first then enable/disable the compare function by:

LSI3144A_compare_control_set( ) and read back to verify by:

LSI3144A_compare_control_read( )

- **LSI3144A_compare_mode_set**

Format : **u32 status = LSI3144A_compare_mode_set(u8 CardID, u8 axis, u8 mode)**

Purpose: To set the compare mode

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axis	u8	0:X	1:Y
		2:Z	3:A
mode	u8	0: Single compare mode(default) 1: FIFO compare mode 2: Auto increment compare mode	

- **LSI3144A_compare_mode_read**

Format : **u32 status = LSI3144A_compare_mode_read (u8 CardID, u8 axis, u8 *mode)**

Purpose: Read back the compare mode

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axis	u8	0:X	1:Y
		2:Z	3:A

Output:

Name	Type	Description
mode	u8	0: Single compare mode(default) 1: FIFO compare mode 2: Auto increment compare mode

- **LSI3144A_compare_value_set**

Format : **u32 status = LSI3144A_compare_value_set (u8 CardID, u8 axis, i32 value)**

Purpose: To set the compare value. (This is no direction attribute, only compare the position)

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axis	u8	0: X	1: Y
		2: Z	3: A
value	i32	32 bit value (-2,147,483,648 ~ 2,147,483,647) to be compared with counter.	

- **LSI3144A_compare_value_read**

Format : u32 status = LSI3144A_compare_value_read (u8 CardID, u8 axis, i32 *value)

Purpose: To read the compare value.

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axis	u8	0: X	1: Y
		2: Z	3: A

Output:

Name	Type	Description
value	i32	32 bit value (-2,147,483,648 ~ 2,147,483,647) to be compared with counter.

- **LSI3144A_direction_compare_value_set**

Format : u32 status = LSI3144A_direction_compare_value_set (u8 CardID, u8 axis, i32 value, u8 direction)

Purpose: To set the first compare value and its direction for FIFO and Auto increment compare mode or setup the compare value and direction for single compare mode.

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axis	u8	0: X	1: not available
		2: Z	3: not available
value	i32	32 bit value (-2,147,483,648 ~ 2,147,483,647) to be compared with counter.	
direction	u8	0: up count 1: down count	

- **LSI3144A_direction_compare_value_read**

Format : **u32 status = LSI3144A_direction_compare_value_read (u8 CardID, u8 axis, i32 *value, u8 *direction)**

Purpose: To read the first compare value and its direction.

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axis	u8	0: X	1: not available
		2: Z	3: not available

Output:

Name	Type	Description
value	i32	32 bit value (-2,147,483,648 ~ 2,147,483,647) to be compared with counter.
direction	u8	0: up count 1: down count

- **LSI3144A_CMP_mode_set**

Format : **u32 status = LSI3144A_CMP_mode_set (u8 CardID, u8 axis, u8 mode)**

Purpose: To set the CMP (compare out) mode.

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axis	u8	0: X	1: Y
		2: Z	3: A
mode	u8	0:NO_OUT (CMP as purely digital output) (default) 1:OUT_PULSE (compare condition meet, CMP output pulse according ***LSI3144A_CMP_oneshot_duration_set( )***.) 2:OUT_LEVEL (compare condition meet, CMP output edge transition.) 3:OUT_TOGGLE (compare condition meet, CMP output toggles)	

Note: CMP output initial state can be defined by ***LSI3144A_output_set*** or ***LSI3144A_output_polarity_set*.**

- **LSI3144A_CMP_mode_read**

Format : **u32 status = LSI3144A_CMP_mode_read (u8 CardID, u8 axis, u8 *mode)**

Purpose: To readback the CMP (compare out) mode.

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axis	u8	0: X	1: Y
		2: Z	3: A

Output:

Name	Type	Description
mode	u8	0:NO_OUT (CMP as purely digital output) (default) 1:OUT_PULSE (compare condition meet, CMP output pulse according ***LSI3144A_CMP_oneshot_duration_set( ).***) 2:OUT_LEVEL (compare condition meet, CMP output edge transition.) 3:OUT_TOGGLE (compare condition meet, CMP output toggles)

- **LSI3144A_CMP_oneshot_duration_set**

Format : **u32 status = LSI3144A_CMP_oneshot_duration_set (u8 CardID,u8 axis, i32Value)**

Purpose: To set the one shot duration time for compare out.

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axis	u8	0: X	1: Y
		2: Z	3: A
Value	i32	Duration time = (value+1) * 1us 0<= value <= 16777215	

- **LSI3144A_CMP_oneshot_duration_read**

Format : **u32 status = LSI3144A_CMP_oneshot_duration_read (u8 CardID,u8 axis, i32*Value)**

Purpose: To read back the one shot duration time for compare out.

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axis	u8	0: X	1: Y
		2: Z	3: A

Output:

Name	Type	Description
Value	i32	Duration time = (value+1) * 1us 0<= value <= 16777215

- **LSI3144A_compare_CMP_OUT_set**

Format : u32 status = LSI3144A_compare_CMP_OUT_set (u8 CardID ,u8 axis , u8 polarity , u8 out_mode , u32 out_width)

Purpose: To set the CMP_OUT parameter.

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axis	u8	0:X	1: Y
		2:Z	3: Z
polarity	u8	0: normal 1: invert	
out_mode	u8	0:NO_OUT (CMP as purely digital output) (default) 1:OUT_PULSE (compare condition meet, CMP output pulse according to out_width set.) 2:OUT_LEVEL (compare condition meet, CMP output edge transition.) 3:OUT_TOGGLE (compare condition meet, CMP output toggles)	
out_width	u32	Output duration of OUT_PULSE mode. Duration = 1us * (out_width +1); 0 <= out_width <= 16777215 If in other output mode, this parameter is trivial.	

Note: ***LSI3144A_compare_CMP_OUT_set*** is a composite command, ***LSI3144A_CMP_oneshot_duration_set*** is a single function command, you can use ether one to setup the one shot pulse.

- **LSI3144A_compare_CMP_OUT_read**

Format : u32 status = LSI3144A_compare_CMP_OUT_read (u8 CardID , u8 axis , u8 *polarity , u8 *out_mode , u32 *out_width)

Purpose: To read back the CMP_OUT parameter.

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axis	u8	0:X	1: Y
		2:Z	3: Z

Output:

Name	Type	Description
polarity	u8	0: normal 1: invert
out_mode	u8	0:NO_OUT (CMP as purely digital output) (default) 1:OUT_PULSE (compare condition meet, CMP output pulse according ***LSI3144A_CMP_oneshot_duration_set( )***.) 2:OUT_LEVEL (compare condition meet, CMP output edge transition.) 3:OUT_TOGGLE (compare condition meet, CMP output toggles)
out_width	u32	Output duration of OUT_PULSE mode. Duration = 1us * (out_width +1) 0 <= out_width <= 16777215 If in other output mode, this parameter is trivial.

- **LSI3144A_compare_control_set**

Format : u32 status = LSI3144A_compare_control_set(u8 CardID, u8 axis, u8 compare_control)

Purpose: To enable/disable compare function**.**

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axis	u8	0: X	1: Y
		2: Z	3: A
compare_control	u8	0:disable compare function. 1:enable single / auto increment compare function 2: enable FIFO compare function (Simple FIFO or PWM FIFO mode) 3: enable one shot FIFO compare function	

- **LSI3144A_compare_control_read**

Format : u32 status = LSI3144A_compare_control_read(u8 CardID, u8 axis, u8 *compare_control)

Purpose: To read back setting of the compare control.

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axis	u8	0: X	1: Y
		2: Z	3: A

Output:

Name	Type	Description
compare_control	u8	0:disable compare function. 1:enable single / auto increment compare function 2: enable FIFO compare function 3: enable one shot FIFO compare function

9.8 Auto increment compare mode

The auto increment compare mode gives you fixed incremental compare data. You can generate trigger output at fixed interval of distance.

To use this mode, you must go through the procedures as the one time compare mode and more there is one parameter you must configure, the increment value, setup by:

LSI3144A_compare_increment_set( ) and read back for verification by:

LSI3144A_compare_increment_read( )

If you will control start or stop compare function, please refer 9.7 Basic compare function (***LSI3144A_compare_control_set***) and flow chart 'Auto increment Compare function' on p.26.

Note: The first compare value (position) can be specified the only by ***LSI3144A_compare_value_set( )*** or position and direction by ***LSI3144A_direction_compare_value_set( )*** depends on your application.

Note: The compare equal trigger output was defined by ***LSI3144A_CMP_oneshot_duration_set*** or ***LSI3144A_compare_CMP_OUT_set.***

- **LSI3144A_compare_increment_set**

Format : **u32 status = LSI3144A_compare_increment_set (u8 CardID, u8 axis, i32 value)**

Purpose: To load the increment value.

At compare out mode 2 (auto increment mode), next compare value will be loaded after compare out trigger.

Next compare value = current compare value + increment value

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axis	u8	0: X	1: Y
		2: Z	3: A
value	i32	32 bit value (-2,147,483,648 ~ 2,147,483,647) to be incremented with counter.	

Note: The available auto increment value is physically influenced by the pulse in-coming speed, we recommend the minimum incremental value at 6 on the max. in-coming speed(16M).

- **LSI3144A_compare_increment_read**

Format : **u32 status = LSI3144A_compare_increment_read (u8 CardID, u8 axis, i32 *value)**

Purpose: To read the incremental data.

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axis	u8	0: X	1: Y
		2: Z	3: A

Output:

Name	Type	Description
value	i32	32 bit value (-2,147,483,648 ~ 2,147,483,647) to be incremented with counter.

9.9 FIFO compare mode

If your application is not increase at regular distance, using position FIFO to program the absolute position is the right solution.

There are 4 FIFO compare modes provide by LSI3144A card.

- Simple FIFO compare mode: (refer p.75 FIFO operation and simple FIFO compare mode)
  Use position FIFO only to compare position with FIFO data to generate trigger with pulse width.
- PWM FIFO compare mode: (refer p.82 PWM FIFO compare mode)
  Use position FIFO associate with PWM FIFO to compare position with FIFO data to generate PWM output with PWM FIFO data.
- One shot FIFO compare mode: (refer p.83 One shot FIFO compare mode)
  Use position FIFO associate with PWM FIFO to compare position with FIFO data to generate one shot output with PWM FIFO data.
- One shot FIFO compare with auto-increment mode (refer p. 84 One shot FIFO compare with auto-increment mode)
  Use position FIFO to trigger one shot pulse width update (with PWMFIFO data) and between the position FIFO segment, the one shot re-triggered by auto-incremented position data.

FIFO operation and simple FIFO compare mode

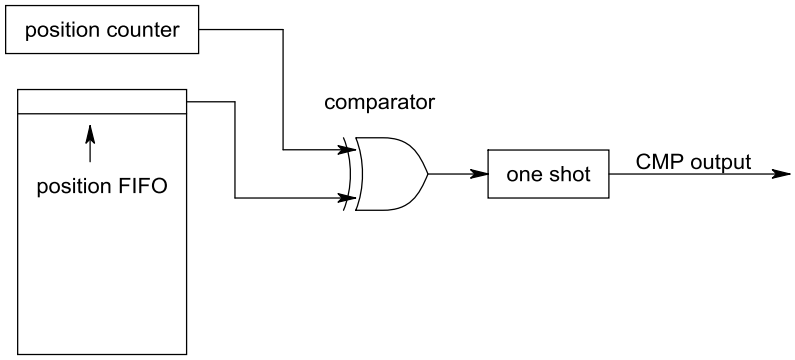


fig. 9.9-1 Function block of simple FIFO compare mode

Normally, we speak of FIFO means the position FIFO, it holds the position data to be compared with the counter.

There are several functions to manipulate on the position FIFO,

LSI3144A_FIFO_clear( ) to clear the whole FIFO data.

LSI3144A_FIFO_set( ) to fill the FIFO data (position and PWM), if you want to use FIFO with direction tag to control the valid position and direction of incoming pulse train:

LSI3144A_direction_FIFO_set( )

LSI3144A_FIFO_read( ) (no direction tag) or

LSI3144A_direction_FIFO_read( ) (with direction tag) to pop up data from top of position FIFO but the data will lost.

The application may pop up data very fast, you need to set a level of FIFO consuming to signal for FIFO re-filling.

LSI3144A_FIFO_threshold_set( ) to set the threshold to signal for requesting data filling.

LSI3144A_FIFO_threshold_read( ) to read back for verification.

Of course, if you want to check how many data still not used

LSI3144A_FIFO_unused_read( ) to read the remained data number.

LSI3144A_FIFO_status_read( ) to read the FIFO status, such as full, empty or near empty (threshold occurs).

To use the simple FIFO compare function you must go through the basic compare function procedures as 9.7 Basic compare function does and then setup the FIFO with the FIFO operation functions. Don't forget to specify the first compare position to start FIFO operation by ***LSI3144A_compare_value_set*** (position only) or by ***LSI3144A_direction_compare_value_set*** (position and direction). Refer the flow chart 'FIFO Compare function' on p.27.

Same as other compare mode, you must enable/disable the FIFO compare function by: ***LSI3144A_compare_control_set.***

Refer p.27 Simple FIFO compare function flow for detail of programming sequence.

- **LSI3144A_FIFO_clear**

Format : **u32 status = LSI3144A_FIFO_clear(u8 CardID,u8 axis)**

Purpose: To clear (reset) FIFO (position and PWM FIFO).

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axis	u8	0: X	1: Y (not available)
		2: Z	3: A (not available)

Note: FIFO function only available for X, Z axes.

- **LSI3144A_FIFO_set**

Format : **u32 status = LSI3144A_FIFO_set (u8 CardID,u8 axis, u16 PWM_FIFO[1024], i32 position_FIFO[1024], u8 rel_abs, u16 size)**

Purpose: To fill position and PWM FIFO for comparison.

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axis	u8	0:X	1:Y(not available)
		2:Z	3:A(not available)
PWM_FIFO	u16	Array of PWM_duty data, each data is 16bit integer type and the data range is 1~65535. Pulse Width=(1/33M)*PWM_duty and the data range is 1~65535. The PWM_freq is defined in ***LSI3144A_PWM_set( )***	
position_FIFO	i32	Array of position FIFO data, each data is 32bit integer type	
rel_abs	u8	0: RELATIVE, relative coordinate data 1: ABSOLUTE, absolute coordinate data	
size	u16	The input FIFO_data size (1~1024)	

Note:

1. position_FIFO is an array of under compare position data with the “size” length.
2. For the relative coordinate type, the result position data must fall in the range + 21474836487~ - 2147483648 to avoid error since the FIFO hardware use absolute data to compare and the data size is 32 bit.
3. The PWM_FIFO[n] and position_FIFO[n] are matched pair.

- **LSI3144A_FIFO_read**

Format : **u32 status =LSI3144A_FIFO_read(u8 CardID,u8 axis, i32 *value)**

Purpose: Read data from top of position FIFO

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axis	u8	0: X	1: Y (not available)
		2: Z	3: A (not available)

Output:

Name	Type	Description
value	i32	data at the top of (position) FIFO

Note:

1. The FIFO pointer will increase by one, i.e. one data pop out from FIFO.
2. The PWM FIFO do not provide any function to read back but the position FIFO pop up will also consume one PWM FIFO data.

- **LSI3144A_direction_FIFO_set**

Format : **u32 status = LSI3144A_direction_FIFO_set (u8 CardID,u8 axis,**
u16 PWM_FIFO[1024], u8 direction[1024],
i32 position_FIFO[1024], u8 rel_abs, u16 size)

Purpose: To fill position and PWM FIFO for comparison.

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axis	u8	0:X	1: not available
		2:Z	3: not available
PWM_FIFO[1024]	u16	Array of PWM_duty data, each data is 16bit integer type and the data range is 1~65535. Pulse Width=(1/33M)*PWM_duty and the data range is 1~65535. The PWM_freq is defined in ***LSI3144A_PWM_set( )***	
direction[1024]	u8	Array of direction tag of FIFO data, direction[0]~ direction[1023] as direction attribute of position_FIFO[0] ~ position_FIFO[1023] 0: up count 1: down count	

position_FIFO[1024]	i32	Array of position FIFO data, each data is 32bit integer type
rel_abs	u8	0: RELATIVE, relative coordinate data 1: ABSOLUTE, absolute coordinate data
size	u16	The input FIFO_data size (1~1024)

Note:

1. position_FIFO is an array of position data with the "size" length.
2. For the relative coordinate type, the result position data must fall in the range + 21474836487~ -2147483648 to avoid error since the hardware use absolute position to compare.
3. The PWM_FIFO[n], position_FIFO[n] and direction[n] are matched pair.

- **LSI3144A_direction_FIFO_read**

Format : u32 status =LSI3144A_direction_FIFO_read(u8 CardID,u8 axis, u8 *direction, i32 *position_FIFO, u8 *rel_abs,)

Purpose: Read data from top of position FIFO

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axis	u8	0: X	1: Y (not available)
		2: Z	3: A (not available)

Output:

Name	Type	Description
direction	u8	0: up count 1: down count
position_FIFO	i32	data at the top of (position) FIFO
rel_abs	u8	0: RELATIVE, relative coordinate data 1: ABSOLUTE, absolute coordinate data

Note:

1. The FIFO pointer will increase by one, i.e. one data pop out from FIFO.
3. The PWM FIFO do not provide any function to read back but the position FIFO pop up will also consume one PWM FIFO data.

- **LSI3144A_FIFO_threshold_set**

Format : **u32 status = LSI3144A_FIFO_threshold_set (u8 CardID,u8 axis, u16 threshold_value)**

Purpose: To set LSI3144A card's FIFO threshold value of high speed counter. While FIFO remained count reach the threshold, it will generate an almost empty event.

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axis	u8	0:X	1:Y(not available)
		2:Z	3:A(not available)
threshold_value	u16	1 ~ 1023	

Note: The remained FIFO data number equals the threshold value will generate the FIFO ALMOST_EMPTY status.

- **LSI3144A_FIFO_threshold_read**

Format : **u32 status = LSI3144A_FIFO_threshold_read (u8 CardID,u8 axis, u16 *threshold_value)**

Purpose: To set LSI3144A card's compare FIFO threshold value of high speed counter. While FIFO remained count reach the threshold, it will generate an almost empty event.

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axis	u8	0:X	1:Y(not available)
		2:Z	3:A(not available)

Output:

Name	Type	Description
threshold_value	u16	1 ~ 1023

- **LSI3144A_FIFO_unused_read**

Format : **u32 status =LSI3144A_FIFO_unused_read(u8 CardID,u8 axis, u32*value)**

Purpose: Read FIFO unused space (number)

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axis	u8	0: X	1: Y (not available)
		2: Z	3: A (not available)

Output:

Name	Type	Description
value	u32	unused space (number) total maximum space is 1023 data

- **LSI3144A_FIFO_status_read**

Format : **u32 status =LSI3144A_FIFO_status_read(u8 CardID,u8 axis,u8 index, u8 *flag)**

Purpose: Read FIFO flag

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axis	u8	0: X	1: Y (not available)
		2: Z	3: A (not available)
index	u8	0: FIFO ALMOST_EMPTY, the FIFO remained count reach the threshold. 1: FIFO FULL 2: FIFO EMPTY	

Output:

Name	Type	Description
flag	u8	0: not active 1: active

Note: Please check FIFO FULL flag before push any data into FIFO.

PWM FIFO compare mode

The LSI3144A also provides PWM FIFO, which is synchronous with position FIFO. While position FIFO pop up one data, the PWM FIFO also pop up one data to PWM function block.

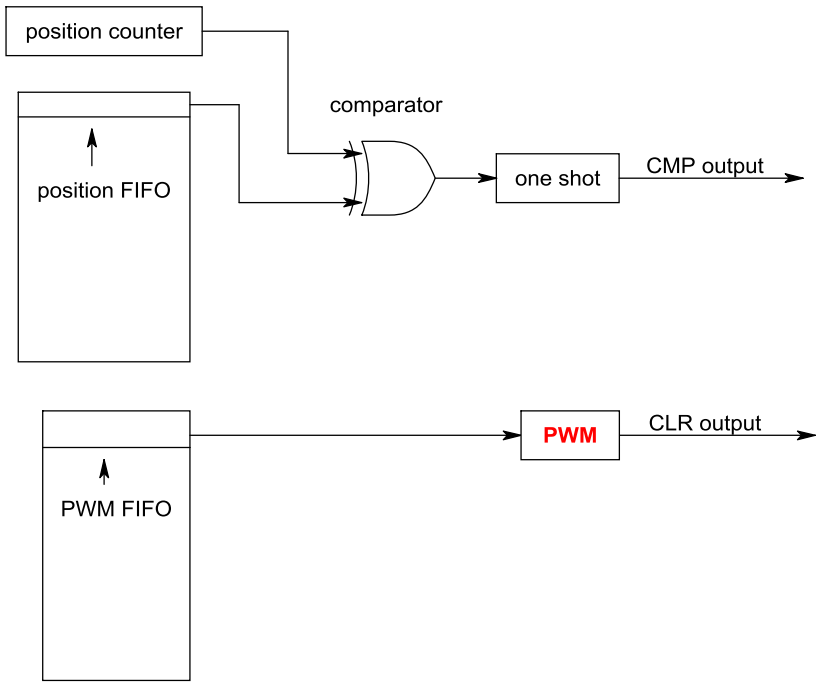


fig. 9.9-2 Function block of PWM FIFO compare mode

The PWM worked with position FIFO to adjust the power of laser or other power source, which will be varied with the position. To use the PWM FIFO compare function you must go through the basic compare function procedures as 9.7 Basic compare function does and then setup the FIFO with the FIFO operation functions. Don't forget to specify the first compare position to start FIFO operation by ***LSI3144A_compare_value_set*** (position only) or by ***LSI3144A_direction_compare_value_set*** (position and direction). Refer the flow chart 'FIFO Compare function' on p.27.

The next is setup the frequency at initial stage. Use:

LSI3144A_PWM_set( ) to setup the PWM frequency and duty.

LSI3144A_PWM_read( ) to read back for verification.

If there are no new PWM FIFO data to fill the PWM duty, the PWM will stay without change the duty.

To control the PWM enable or disable by:

LSI3144A_PWM_control_set( ) and read back for verification by:

LSI3144A_PWM_control_read( )

Same as other compare mode, you must enable/disable the FIFO compare function by:

LSI3144A_compare_control_set( ).

Refer p.28 PWM FIFO compare function flow for detail of programming sequence.

One shot FIFO compare mode

The one shot FIFO compare mode is almost the same with PWM FIFO compare mode except for the pulse output is one shot (only one trigger output and the pulse width defined by the PWM FIFO data).

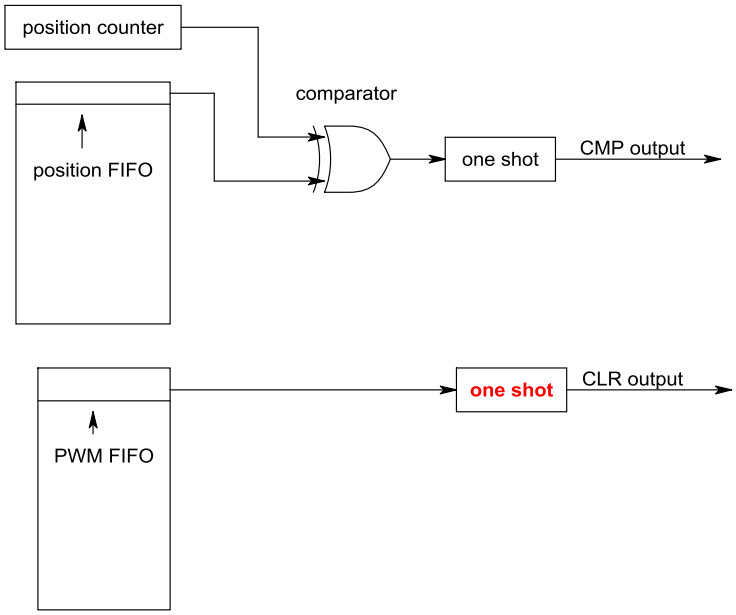


fig. 9.9-3 One shot FIFO compare mode

The one shot duration data comes from PWM FIFO and worked with position FIFO to generate the one shot pulse according to position. To use the one shot FIFO compare function you must go through the basic compare function procedures as 9.7 Basic compare function does and then setup the FIFO with the FIFO operation functions. Don't forget to specify the first compare position to start FIFO operation by ***LSI3144A_compare_value_set*** (position only) or by ***LSI3144A_direction_compare_value_set*** (position and direction). Refer the flow chart 'FIFO Compare function' on p.27.

The next is setup the frequency at initial stage. Use:

LSI3144A_CLR_oneshot_duration_set***( )*** to setup the first one shot duration and 1

LSI3144A_CLR_OUT_mode_set***( )*** to OUT_PULSE2 mode.

Same as other compare mode, you must enable/disable the FIFO compare function by:

LSI3144A_compare_control_set( ).

Refer p.29 One shot FIFO compare function flow for detail of programming sequence.

One shot FIFO compare with auto-increment mode

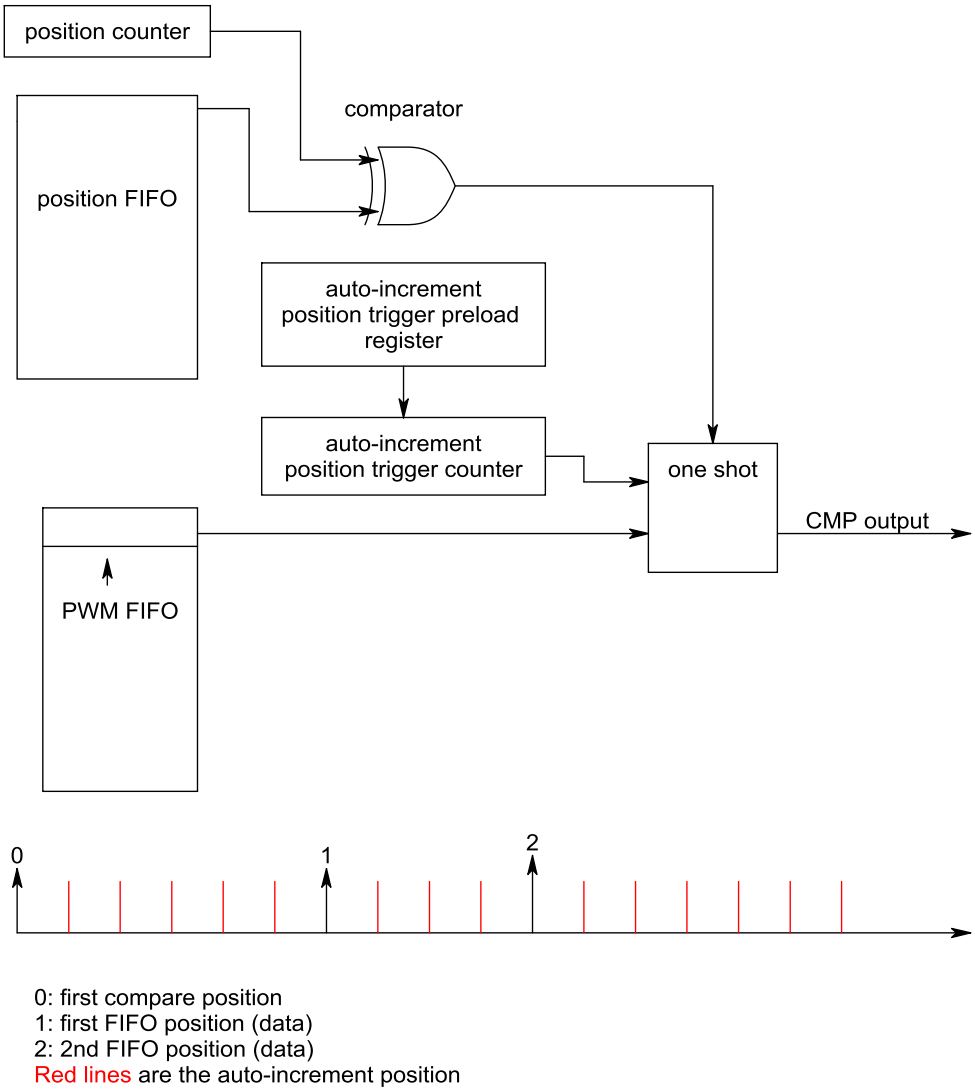


fig. 9.9-4 one shot FIFO compare with auto-increment mode

From the above diagram, you can see that the one-shot trigger can be generated by:

- the first compare position (one shot defined by ***LSI3144A_CMP_oneshot_duration_set***)
- the auto-increment compare position (keep the one-shot data as previous)
- the FIFO position (change the one shot time constant by ***LSI3144A_FIFO_set***)

To use one shot FIFO compare with auto-increment function, you must first program it as it is simple FIFO mode then an extra auto-increment data must program, using:

LSI3144A_position_trigger_set( ) to setup the preload register or auto-increment position counter, ***LSI3144A_position_trigger_read( )*** to read back for verification.

Refer p.30 One shot FIFO compare with auto-increment function flow for detail of programming sequence.

- **LSI3144A_position_trigger_set**

Format : u32 status = LSI3144A_position_trigger_set(u8 CardID,u8 axis, u8 index, u16 data)

Purpose: Set one shot FIFO's trigger pulse of auto-increment register.

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axis	u8	0:X	1:Y(not available)
		2:Z	3:A(not available)
index	u8	0 : position trigger counter 1 : position trigger preload	
data	u16	1~65535	

- **LSI3144A_position_trigger_read**

Format : u32 status = LSI3144A_position_trigger_read(u8 CardID,u8 axis,u8 index, u16 *data)

Purpose: Read one shot FIFO's trigger pulse of auto-increment register.

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axis	u8	0:X	1:Y(not available)
		2:Z	3:A(not available)
index	u8	0 : position trigger counter 1 : position trigger preload	

Output:

Name	Type	Description
data	u16	1 ~ 65535

9.10 Compare output mask function

Although the auto increment compare function and FIFO compare function are very convenient to generate trigger on customer's demand but some case we need to disable or enable the trigger on external hardware or software-preset request. The LSI3144A provides hardware input to gate the compare out trigger and also software to mask of the unwanted trigger.

To select the hard gate control (CLR_IN or INn) by:

LSI3144A_CMP_mask_source_set( ) and read back for verification by:

LSI3144A_CMP_mask_source_read( )

If you want to use the software to layout the rule to mask off the unwanted trigger, there are 3 sets of coordinate to use, first setup the coordinate you want:

LSI3144A_CMP_segment_set( ) and to verify by:

LSI3144A_CMP_segment_read( ).

Now you can define the interior or exterior of the coordinate you want to mask off the trigger:

LSI3144A_mask_off_set( ) and read back by:

LSI3144A_mask_off_read( )

Finally, you can enable or disable any of the segment of the segment mask function by:

LSI3144A_segment_control_set( ) and read back by:

LSI3144A_segment_control_read( )

- **LSI3144A_CMP_mask_source_set**

Format : **u32 status = LSI3144A_CMP_mask_source_set(u8 CardID,u8 axis, u8 source_sel)**

Purpose: To set the mask source of CMP output.

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axis	u8	0: X	1: Y
		2: Z	3: A
source_sel	u8	0: no mask to CMP output. 1: use CLR_IN to mask CMP output. (CMP output is enable/disabled by CLR_IN) 2: use general INn to mask CMP output. (CMP output is enable/disabled by INn)	

Note: For each axis CMP output and its mask

axis	Compare output	Mask input
X	xCMP	xCLR_IN or IN0
Y	yCMP	yCLR_IN or IN1
Z	zCMP	zCLR_IN or IN2
A	aCMP	aCLR_IN or IN3

- **LSI3144A_CMP_mask_source_read**

Format : **u32 status = LSI3144A_CMP_mask_source_read(u8 CardID,u8 axis, u8 *source_sel)**

Purpose: Read back the setting of mask source

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axis	u8	0: X	1: Y
		2: Z	3: A

Output:

Name	Type	Description
source_sel	u8	0: no mask to CMP_OUT. 1: use CLR_IN to mask CMP output. (CMP_OUT is enable/disabled by CLR_IN) 2: use general INn to mask CMP output. (CMP output is enable/disabled by INn)

- **LSI3144A_CMP_segment_set**

Format : **u32 status = LSI3144A_CMP_segment_set (u8 CardID,u8 axis,u8 index, i32 start, i32 stop)**

Purpose: To write the segment coordinate.

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axis	u8	0:X	1:Y
		2:Z	3:A
index	u8	0: Segment 0 1: Segment 1 2: Segment 2	
start	i32	Start position of mask off segment	
stop	i32	Stop position of mask off segment	

- **LSI3144A_CMP_segment_read**

Format : **u32 status = LSI3144A_CMP_segment_read (u8 CardID, u8 axis, u8 index, i32 *start, i32 *stop)**

Purpose: To read the segment coordinate.

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axis	u8	0:X	1:Y
		2:Z	3:A
index	u8	0: Segment 0 1: Segment 1 2: Segment 2	

Output:

Name	Type	Description
start	i32	Start position of mask off segment
stop	i32	Stop position of mask off segment

- **LSI3144A_mask_off_set**

Format : **u32 status = LSI3144A_mask_off_set (u8 CardID,u8 axis, u8 attribute)**

Purpose: To write the mask off attribute.

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axis	u8	0:X	1:Y
		2:Z	3:A
attribute	u8	0: mask off interior 1: mask off exterior	

Note:

1. Mask off interior means inside the segment coordinate, the CMP trigger output will not generate. But the compare function (auto increment or FIFO pop up) do not influenced.
2. Mask off exterior means outside the segment coordinate, the CMP trigger output will not generate. But the compare function (auto increment or FIFO pop up) do not influenced.

- **LSI3144A_mask_off_read**

Format : **u32 status = LSI3144A_mask_off_read (u8 CardID,u8 axis,u8 *attribute)**

Purpose: To read back the segment interior or exterior attribute.

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axis	u8	0:X	1:Y
		2:Z	3:A

Output:

Name	Type	Description
attribute	u8	0: mask off interior 1: mask off exterior

- **LSI3144A_segment_control_set**

Format : u32 status = LSI3144A_segment_control_set (u8 CardID, u8 axis, u8 index, u8 control)

Purpose: To write the segment control.

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axis	u8	0:X	1:Y
		2:Z	3:A
index	u8	0: Segment 0 1: Segment 1 2: Segment 2	
control	u8	0:disable 1:enable	

- **LSI3144A_segment_control_read**

Format : u32 status = LSI3144A_segment_control_read (u8 CardID,u8 axis,u8 index, u8 *control)

Purpose: To read the segment control.

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axis	u8	0:X	1:Y
		2:Z	3:A
index	u8	0: Segment 0 1: Segment 1 2: Segment 2	

Output:

Name	Type	Description
control	u8	0:disable 1:enable

9.11 Miscellaneous function

Sometimes you have signal trigger source to latch / load all the axes' counter,

LSI3144A_com_trigger_control( ) will do.

To verify the in-coming pulse counting direction by:

LSI3144A_counter_direction_read( )

- **LSI3144A_com_trigger_control**

Format : u32 status = LSI3144A_com_trigger_control(u8 CardID, u8 control)

Purpose: To set the 4 axes, which are common triggered (latch/load) by X trigger (xLAH) input or triggered from respectively axis trigger input.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
control	u8	0: individual mode (Default value) triggered respectively 1:common triggered mode X axis trigger input (xLAH) will trigger to latch all 4 axes.

- **LSI3144A_counter_direction_read**

Format : u32 status = LSI3144A_counter_direction_read(u8 CardID, u8 axis, u8 *direction)

Purpose: To read LSI3144A card counter. The 32bit counter on the fly value will return.

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axis	u8	0: X	1: not available
		2: Z	3: not available

Output:

Name	Type	Description
direction	u8	0: up count 1: down count

9.12 PWM function

PWM function, as mention on 9.9 FIFO compare mode, it can also work as simple PWM function to output signal on CMP output.

To setup the frequency and duty use:

LSI3144A_PWM_set( ) to setup the PWM frequency and duty.

LSI3144A_PWM_read( ) to read back for verification.

To control the PWM enable/disable by:

LSI3144A_PWM_control_set( ) and read back for verification by:

LSI3144A_PWM_control_read( )

- **LSI3144A_PWM_set**

Format : **u32 status = LSI3144A_PWM_set(u8 CardID,u8 axis, u16 PWM_freq, u16 PWM_duty)**

Purpose: To set PWM_freq and PWM_duty

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axis	u8	0:X	1:Y(not available)
		2:Z	3:A(not available)
PWM_freq	u16	PWM_freq = 33M / frequency, the data range is 1~65535.	
PWM_duty	u16	PWM_duty = duty% * PWM_freq, the data range is 1~65535.	

Note:

1. PWM clock is based on 33MHz, say if you want your PWM period is 50us(20KHz), please put the PWM_freq = (33MHz / 20KHz) = 1650
2. PWM duty must less than PWM_freq for proper operation, from the example above, the PWM_duty value can be 1 ~ 1649.
   For 50% duty, the PWM_duty will be 50% * 1650 = 825
3. CLR_OUT pin will become dedicated PWM_OUT.

- **LSI3144A_PWM_read**

Format : **u32 status = LSI3144A_PWM_read(u8 CardID,u8 axis, u16 *PWM_freq, u16 *PWM_duty)**

Purpose: Read back the PWM_freq and PWM_duty.

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axis	u8	0:X	1:Y(not available)
		2:Z	3:A(not available)

Output:

Name	Type	Description
PWM_freq	u16	PWM_freq = 33M / frequency, the data range is 1~65535.
PWM_duty	u16	PWM_duty = duty% * PWM_freq, the data range is 1~65535.

- **LSI3144A_PWM_control_set**

Format : **u32 status = LSI3144A_PWM_control_set(u8 CardID, u8 axis, u8 PWM_control)**

Purpose: To enable/disable PWM function.

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axis	u8	0: X	1: Y
		2: Z	3: A
PWM_control	u8	0:disable PWM function. 1:enable PWM FIFO function.(X,Z only) 2:enable PWM function	

- **LSI3144A_PWM_control_read**

Format : **u32 status = LSI3144A_PWM_control_read(u8 CardID, u8 axis, u8 *PWM_control)**

Purpose: To read back setting of PWM control.

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axis	u8	0: X	1: Y
		2: Z	3: A

Output:

Name	Type	Description
PWM_control	u8	0:disable PWM function. 1:enable PWM FIFO function (X,Z only) 2:enable PWM function

9.13 Vector speed to PWM function

Some application needs the PWM output follows the vector speed to get the adequate power of different contouring speed. Say, for example, the laser cutter needs to cut the paper or other material under the contouring motion control. On the route, the speed maybe different, on the faster segment, the laser power will not enough to cut off but the slower segment the laser will burn the paper. To get rid of the problem, you need the laser power adjust to follow the vector speed (composed speed of the motion axes).

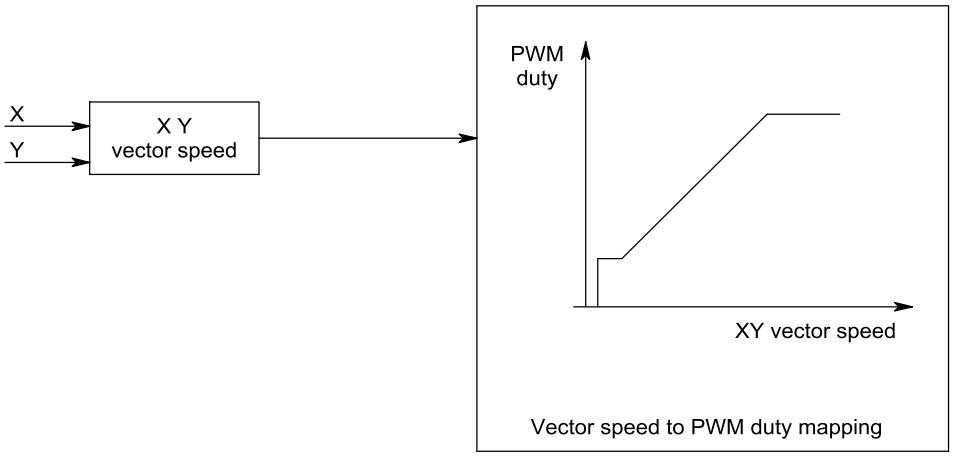


fig. 9.13-1 Function block diagram of vector speed to PWM duty mapping

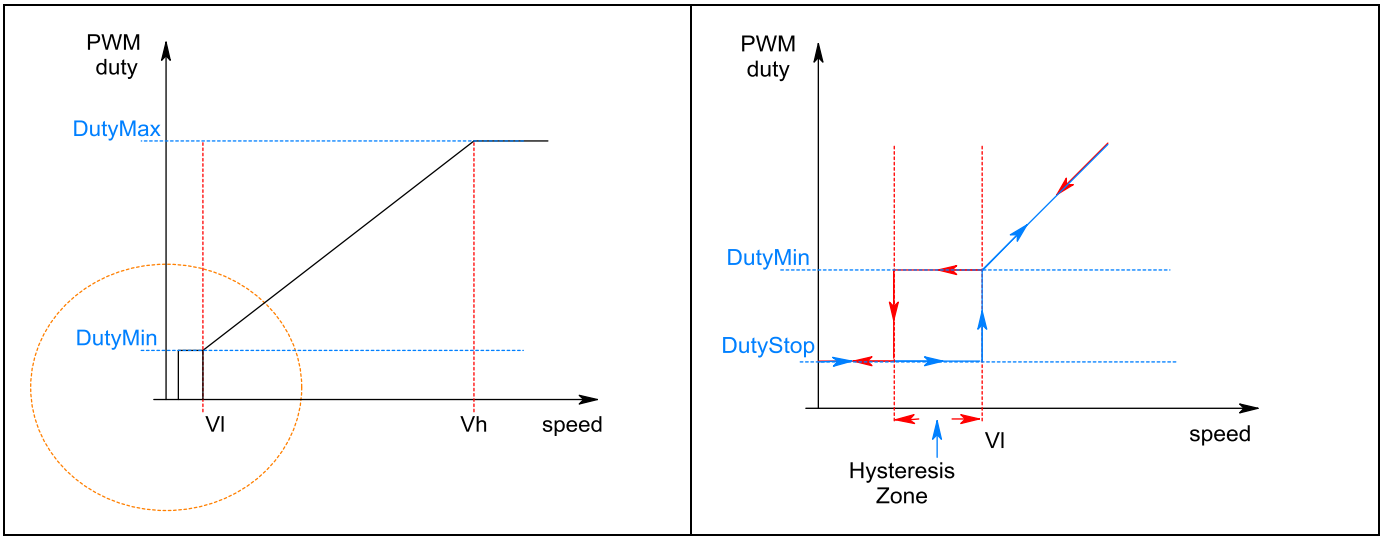


fig. 9.13-2 Definition of Vh,Vl, DutyMax, DutyMin, DutyStop and Hysteresis

LSI3144A provides PWM duty output varying with the vector speed of X,Y axes. Setup the parameters by: ***LSI3144A_PWM_motion_config_set( )*** and read back for verification by

LSI3144A_PWM_motion_config_read( )

To enable/disable the vector speed to PWM function by:

LSI3144A_PWM_motion_control_set( ) and read back for verification by

LSI3144A_PWM_motion_control_read( )

On the start vector speed to PWM function, please also remember to start the counter function of X,Y axes by ***LSI3144A_counter_control_set( )*** to normal counter mode.

To stop, on the reverse procedures, stop counter first then stop PWM motion function.

Refer p.31 Vector speed to PWM function flow for detail of programming sequence.

Note: Using the vector speed to PWM function will occupies the on card timer and all the timer functions are not available while vector speed to PWM function in use.

- **LSI3144A_PWM_motion_config_set**

Format : **u32 status = LSI3144A_PWM_motion_config_set(u8 CardID , _pwm_motion_config *config)**

Purpose: Setup parameters of X,Y vector speed to PWM

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by jumper setting

config	_pwm_motion_config	struct _pwm_motion_config
		{
		u32 Vh; //Input high speed(pps)
		u32 Vl; //Input low speed(pps)
		u16 freq; //PWM freq data.
		u16 DutyMax; //Output high pwm duty
		//(0~65535)
		u16 DutyMin ; //Output low pwm duty
		//(0~65535)
		u16 DutyStop; //Output stop pwm
		//duty
		//(0~65535)
		u16 Hysteresis;//Hysteresis range
		//(0~Vl pps)
		}

Note: freq、DutyMax、DutyMin、DutyStop clock count based on 33MHz clock

- **LSI3144A_PWM_motion_config_read**

Format : **u32 status = LSI3144A_PWM_motion_config_read(u8 CardID , _pwm_motion_config *config)**

Purpose: Read back parameters of X,Y vector speed to PWM

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by jumper setting

Output:

Name	Type	Description

config	_pwm_motion_config	struct _pwm_motion_config
		{
		u32 Vh; //Input high speed(pps)
		u32 Vl; //Input low speed(pps)
		u16 freq; //PWM freq data.
		u16 DutyMax; //Output high pwm duty
		//(0~65535)
		u16 DutyMin ; //Output low pwm duty
		//(0~65535)
		u16 DutyStop; //Output stop pwm
		//duty
		//(0~65535)
		u16 Hysteresis;//Hysteresis range
		//(0~Vl pps)
		}

- **LSI3144A_PWM_motion_control_set**

Format : **u32 status = LSI3144A_PWM_motion_control_set(u8 CardID , u8 control)**

Purpose: Enable/disable vector speed to PWM function

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by jumper setting
enable	u8	0 : disable 1 : enable

Note: Enable PWM motion control will occupy on card timer and do not use timer function while this function is enabled.

- **LSI3144A_PWM_motion_control_read**

Format : **u32 status = LSI3144A_PWM_motion_control_read(u8 CardID , u8 *control)**

Purpose: Read back status of Enable/disable vector speed to PWM function

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by jumper setting

Output:

Name	Type	Description
enable	u8	0 : disable 1 : enable

9.14 Timer function

There two 1us time base timer on card. Setup the timer constant or override the time constant on the fly by:

LSI3144A_timer_set( ) and read back

LSI3144A_timer_read( )

To start or stop timer by:

LSI3144A_timer_start( )

LSI3144A_timer_stop( )

- **LSI3144A_timer_set**

Format : u32 status = LSI3144A_timer_set (u8 CardID,u8 axis, u8 index, u32 time_constant)

Purpose: To setup timer operation mode or update timer

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY switch	
axis	u8	0:X	1:Y(not available)
		2:Z	3:A(not available)
index	u8	0=timer preset: preset value for the next timer cycle. preset value based on 1us clock T= (timer_constant +1) * 1us 1=Timer value: override value for the current working cycle. timer value based on 1us clock	
time_constant	u32	32 bit value	

Note:

1. If you also enable the timer interrupt, the period T must at least longer than the system interrupt response time else the system will be hanged by excess interrupts.

- **LSI3144A_timer_read**

Format : **u32 status = LSI3144A_timer_read (u8 CardID,u8 axis,u8 index, u32 *time_constant)**

Purpose: To read timer operation mode or update timer

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY switch	
axis	u8	0:X	1:Y(not available)
		2:Z	3:A(not available)
index	u8	0=timer preset: preset value for the next timer cycle. preset value based on 1us clock T= (timer_preset +1) * 1us 1=Timer value on the fly	

Output:

Name	Type	Description
time_constant	u32	32 bit value

- **LSI3144A_timer_start**

Format : **u32 status = LSI3144A_timer_start (u8 CardID,u8 axis)**

Purpose: To start timer operation mode

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY switch	
axis	u8	0:X	1:Y(not available)
		2:Z	3:A(not available)

- **LSI3144A_timer_stop**

Format : **u32 status = LSI3144A_timer_stop (u8 CardID,u8 axis)**

Purpose: To stop timer operation mode

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY switch	
axis	u8	0:X	1:Y(not available)
		2:Z	3:A(not available)

9.15 Interrupt function

There are 9 interrupt sources for your quick response application,

1. FIFO ALMOST_EMPTY
2. FIFO FULL
3. FIFO EMPTY
4. counter compare equal occurred, CMP output
5. TIMER time up
6. external trigger latch / load occurred
7. hardware counter clear occurred
8. counter carry occurred
9. counter borrow occurred

To use the interrupt service, the first step mask off the unwanted interrupt source by:

LSI3144A_IRQ_mask_set( ) and verify by

LSI3144A_IRQ_mask_read( )

Then tell the O.S. your IRQ service routine by:

LSI3144A_IRQ_process_link( )

The interrupt status can be checked in polling or in interrupt service routine (but not recommend in both)

LSI3144A_IRQ_status_read( )

Finally, enable or disable interrupt by:

LSI3144A_IRQ_enable( )

LSI3144A_IRQ_disable( )

- **LSI3144A_IRQ_mask_set**

Format : u32 status = LSI3144A_IRQ_mask_set (u8 CardID,u8 axis, u16 mask)

Purpose: Mask off interrupt source

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by Rotary SW	
axis	u8	0:X	1:Y
		2:Z	3:A
mask	u16	b0=1, FIFO ALMOST_EMPTY can generate IRQ. b1=1, FIFO FULL can generate IRQ. b2=1, FIFO EMPTY can generate IRQ. b3=1, CMP output can generate IRQ. b4=1, TIMER can generate IRQ. b5=1, external latch/load can generate IRQ. b6=1, hardware homing can generate IRQ. b7=1, counter carry occurred can generate IRQ. b8=1, counter borrow occurred can generate IRQ. If corresponding bit =0, mask off the interrupt	

- **LSI3144A_IRQ_mask_read**

Format : u32 status = LSI3144A_IRQ_mask_read (u8 CardID,u8 axis,u16 *mask)

Purpose: read back interrupt mask Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by Rotary SW	
axis	u8	0:X	1:Y
		2:Z	3:A

Output:

Name	Type	Description
mask	u16	b0=1, FIFO ALMOST_EMPTY can generate IRQ. b1=1, FIFO FULL can generate IRQ. b2=1, FIFO EMPTY can generate IRQ. b3=1, CMP output can generate IRQ. b4=1, TIMER can generate IRQ. b5=1, external latch/load can generate IRQ. b6=1, hardware homing can generate IRQ. b7=1, counter carry occurred can generate IRQ. b8=1, counter borrow occurred can generate IRQ. If corresponding bit =0, mask off the interrupt

- **LSI3144A_IRQ_process_link**

Format : u32 status = LSI3144A_link_IRQ_process

(u8 CardID, void (__stdcall *callbackAddr(u8 CardID))

Purpose: To link the interrupt event with the callback function.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
callbackAddr	void	the address of your callback function

- **LSI3144A_IRQ_status_read**

Format : u32 status = LSI3144A_IRQ_status_read (u8 CardID,u8 axis,u16 *Event_status)

Purpose: read back interrupt mask Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by Rotary SW	
axis	u8	0:X	1:Y
		2:Z	3:A

Output:

Name	Type	Description
Event_status	u16	b0=1, FIFO ALMOST_EMPTY generates IRQ. b1=1, FIFO FULL generates IRQ. b2=1, FIFO EMPTY generates IRQ. b3=1, CMP output generates IRQ. b4=1, TIMER generates IRQ. b5=1, external latch/load generates IRQ. b6=1, hardware homing generates IRQ. b7=1, counter carry occurred generates IRQ. b8=1, counter borrow occurred generates IRQ.

Note: ***LSI3144A_IRQ_status_read( )*** will also clears the status after reading.

Interrupt status availability table for LSI3144A

status function	Availability			
	X	Y	Z	A
b0: FIFO ALMOST_EMPTY	Y	NO	Y	NO
b1: FIFO FULL	Y	NO	Y	NO
b2: FIFO EMPTY	Y	NO	Y	NO
b3: CMP output	Y	Y	Y	Y
b4: TIMER time up	Y	NO	Y	NO
b5: external latch/load	Y	Y	Y	Y
b6: hardware homing	Y	Y	Y	Y
b7: counter carry occurred	Y	Y	Y	Y
b8: counter borrow occurred	Y	Y	Y	Y

- **LSI3144A_IRQ_enable**

Format : u32 status = LSI3144A_IRQ_enable(u8 CardID, HANDLE *phEvent)

Purpose: To enable interrupt.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW

Output:

Name	Type	Description
phEvent	HANDLE	The returned handle of event

- **LSI3144A_IRQ_disable**

Format : u32 status = LSI3144A_IRQ_disable(u8 CardID)

Purpose: To disable interrupt.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW

9.16 Latch FIFO

The LSI3144A provides a hardware external trigger –LAH, to latch the counter on the fly. You can configure by ***LSI3144A_input_polarity_set( )*** to set up the polarity of the LAH trigger input. Use it as single latch function (refer 9.6 Counter latch / load mode). But in some case, if the latch signal comes very often, the CPU will spend much time to deal with the counter and maybe you will lose some record. The latch FIFO provides you with a high priority software simulation to save latched data in FIFO.

To prepare the environment of latch FIFO, use

LSI3144A_Latch_FIFO_initial( ) and start / stop the operation by

LSI3144A_Latch_FIFO_control_set( ).

If you want to verify the control status, read by the setting by:

LSI3144A_Latch_FIFO_control_read( )

To check the data availability,

LSI3144A_Latch_FIFO_status_read( ) will do.

To read back the FIFO by using:

LSI3144A_Latch_FIFO_data_read( )

Note: You can only use either latch FIFO function or traditional single latch function. The latch FIFO will occupy some resource of traditional single latch function.

- **LSI3144A_Latch_FIFO_initial**

Format : **u32 status = LSI3144A_Latch_FIFO_initial(u8 CardID , u8 axis)**

Purpose: Initial Latch FIFO parameter and environment.

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axis	u8	0:X(not available)	1:Y
		2:Z	3:A(not available)

- **LSI3144A_Latch_FIFO_control_set**

Format : **u32 status = LSI3144A_Latch_FIFO_control_set(u8 CardID , u8 axis , u8 control)**

Purpose: Set Latch FIFO control. (to start or stop the software FIFO)

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axis	u8	0:X(not available)	1:Y
		2:Z	3:A(not available)
control	u8	0 : Disable 1 : Enable	

- **LSI3144A_Latch_FIFO_control_read**

Format : **u32 status = LSI3144A_Latch_FIFO_control_read(u8 CardID , u8 axis , u8 *control)**

Purpose: Read back Latch FIFO control.

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axis	u8	0:X(not available)	1:Y
		2:Z	3:A(not available)

Output:

Name	Type	Description
control	u8	0 : Disable 1 : Enable

- **LSI3144A_Latch_FIFO_status_read**

Format : **u32 status = LSI3144A_Latch_FIFO_status_read(u8 CardID , u8 axis , u16 *counter , u8 *alarm)**

Purpose: Read Latch FIFO status.

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axis	u8	0:X(not available)	1:Y
		2:Z	3:A(not available)

Output:

Name	Type	Description
counter	u16	number of available FIFO data
alarm	u8	0 : data is ok. 1 : data overlap.

- **LSI3144A_Latch_FIFO_data_read**

Format : **u32 status = LSI3144A_Latch_FIFO_data_read(u8 CardID , u8 axis , u16 counter , u32 data[4096])**

Purpose: Read Latch FIFO data.

Parameters:

Input:

Name	Type	Description	
CardID	u8	assigned by DIP/ROTARY SW	
axis	u8	0:X(not available)	1:Y
		2:Z	3:A(not available)
counter	u16	number of FIFO data to be read	

Output:

Name	Type	Description
data[4096]	u32	Latch FIFO data.

9.17 Security function

From the dll version 3.0 and later, we remove the software key function owing to some customers complained about the card locked on some unknown occasion. We only remain the functions to comply with the existing programs but the returned value always true.

Since LSI3144A is a general purpose card, anyone who can buy from the market. Your program is the fruit of your intelligence, un-authorized copy maybe prevent by the security function enabled.

You can use

LSI3144A_password_set( ) to set password and start the security function. Use

LSI3144A_password_change( ) to change it.

If you don't want to use security function after the password being setup,

LSI3144A_password_clear( ) will reset to the virgin state.

Once the password is set, any function call of the dll's (except for the security functions) will be blocked until the

LSI3144A_security_unlock( ) unlock the security.

You can also use

LSI3144A_security_status_read( ) to check the current status of security.

Note:

Any attempt to unlock the software security function with wrong passwords more than 10 time will "dead lock" the card. We also suggest locking the card under the demo program (comes with the card) and unlocking when your application program starts. Lock and unlock cycles is limited by the semiconductor's life read/write cycles.

- **LSI3144A_password_set**

Format : **u32 status = LSI3144A_password_set(u8 CardID,u16 password[5])**

Purpose: To set password and if the password is not all "0", security function will be enabled.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
password[5]	u16	Password, 5 words

Note: If the password is all "0", the security function is disabled.

- **LSI3144A_password_change**

Format : **u32 status = LSI3144A_password_change(u8 CardID, u16 Oldpassword[5], u16 password[5])**

Purpose: To replace old password with new password.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
Oldpassword [5]	u16	The previous password
password[5]	u16	The new password to be set

- **LSI3144A_password_clear**

Format : **u32 status = LSI3144A_password_clear(u8 CardID,u16 password[5])**

Purpose: To clear password, to set password to all "0", i.e. disable security function.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
password[5]	u16	The password previous set

- **LSI3144A_security_unlock**

Format : **u32 status = LSI3144A_security_unlock(u8 CardID,u16 password[5])**

Purpose: To unlock security function and enable the further operation of this card

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW
password[5]	u16	The password previous set

- **LSI3144A_security_status_read**

Format : **u32 status = LSI3144A_security_status_read(u8 CardID,u8 *lock_status, u8 *security_enable)**

Purpose: To read security status for checking if the card security function is unlocked.

Parameters:

Input:

Name	Type	Description
CardID	u8	assigned by DIP/ROTARY SW

Output:

Name	Type	Description
lock_status	u8	0: security unlocked 1: locked 2: dead lock (must return to original maker to unlock)
security_enable	u8	0: security function disabled 1: security function enabled

10. Dll list

	Function Name	Description
1	LSI3144A_initial( )	LSI3144A initial
2	LSI3144A_close( )	LSI3144A close
3	LSI3144A_info( )	Read the I/O address and card type
4	LSI3144A_input_polarity_set( )	Set input point polarity
5	LSI3144A_input_polarity_read( )	Read back polarity of input point
6	LSI3144A_input_debounce_set( )	Set input point debounce time
7	LSI3144A_input_debounce_read( )	Read back input point debounce time
8	LSI3144A_input_read( )	Read LSI3144A card's input status
9	LSI3144A_output_polarity_set( )	Set output polarity
10	LSI3144A_output_polarity_read( )	Read back polarity of output point
11	LSI3144A_output_set( )	Set/reset of output point
12	LSI3144A_output_read( )	Read back status of output point
13	LSI3144A_HOMING_mode_set( )	Setup the HOMING mode of counter
14	LSI3144A_HOMING_mode_read( )	Read back the HOMING mode
15	LSI3144A_HOMING_flag_read( )	Read HOMING flag
16	LSI3144A_HOMING_software( )	Software clear counter
17	LSI3144A_CLR_OUT_mode_set( )	Set CLR_OUT output mode
18	LSI3144A_CLR_OUT_mode_read( )	Read back CLR_OUT output mode
19	LSI3144A_CLR_oneshot_duration_ set( )	Set the one shot duration time for all CLR_OUT (clear output)
20	LSI3144A_CI_mode_set( )	Set counter input mode
21	LSI3144A_CI_mode_read( )	Read back counter input mode
22	LSI3144A_counter_control_set( )	Counter operation/stop control
23	LSI3144A_counter_control_read( )	Read back operation/stop control setting
24	LSI3144A_counter_set( )	Load value into counter
25	LSI3144A_counter_read( )	Read counter data on the fly
26	LSI3144A_counter_preset( )	Set value into preset buffer for counter external load
27	LSI3144A_counter_preset_read( )	Read back preset data
28	LSI3144A_latch_mode_set( )	To assign the hardware external trigger input function
29	LSI3144A_latch_mode_read( )	Read back external trigger setting
30	LSI3144A_latch_control_set( )	To enable/disable latch (load) function
31	LSI3144A_latch_control_read( )	Read back latch control setting
32	LSI3144A_latch_flag_read( )	To read latch flag, which accused by nLAH trigger

		input
33	LSI3144A_latched_value_read( )	Read counter latched value
34	LSI3144A_compare_mode_set( )	Set up counter compare mode
35	LSI3144A_compare_mode_read( )	Read back setting of compare mode
36	LSI3144A_compare_value_set( )	Set the compare value
37	LSI3144A_compare_value_read( )	Read the compare value
38	LSI3144A_direction_compare_value_set( )	Set the compare value with direction tag
39	LSI3144A_direction_compare_value_read( )	Read the compare value with direction tag
40	LSI3144A_CMP_mode_set( )	Set the CMP (compare out) mode
41	LSI3144A_CMP_mode_read( )	Read back setting of CMP (compare out) mode
42	LSI3144A_CMP_oneshot_duration_set( )	Setup CMP output pulse width
43	LSI3144A_CMP_oneshot_duration_read( )	Read back setting of CMP output pulse width
44	LSI3144A_compare_CMP_OUT_set( )	To set the CMP_OUT parameter
45	LSI3144A_compare_CMP_OUT_read( )	To read back the CMP_OUT parameter
46	LSI3144A_compare_control_set( )	Enable/disable compare function
47	LSI3144A_compare_control_read( )	Read back setting of compare control
48	LSI3144A_compare_increment_set( )	Load the incremental value
49	LSI3144A_compare_increment_read( )	Read the incremental value
50	LSI3144A_FIFO_clear( )	Clear (reset) FIFO
51	LSI3144A_FIFO_set( )	Fill (load) FIFO (position and PWM)
52	LSI3144A_FIFO_read( )	Read data from top of position FIFO
53	LSI3144A_direction_FIFO_set( )	To fill position and PWM FIFO of high speed counter.
54	LSI3144A_direction_FIFO_read( )	Read data from top of position FIFO
55	LSI3144A_FIFO_threshold_set( )	Set the FIFO alarm threshold
56	LSI3144A_FIFO_threshold_read( )	Read back FIFO alarm threshold setting
57	LSI3144A_FIFO_unused_read( )	Read the remained data number
58	LSI3144A_FIFO_status_read( )	Read the FIFO status
59	LSI3144A_position_trigger_set( )	Set one shot FIFO’s trigger pulse of auto-increment register.
60	LSI3144A_position_trigger_read( )	Read one shot FIFO’s trigger pulse of auto-increment register.
61	LSI3144A_CMP_mask_source_set( )	Set the mask source of CMP_OUT

62	LSI3144A_CMP_mask_source_ read( )	Read back the setting of mask source
63	LSI3144A_CMP_segment_set( )	Setup the segment coordinate
64	LSI3144A_CMP_segment_read( )	Read back the segment coordinate
65	LSI3144A_mask_off_set( )	Setup mask off interior or exterior method
66	LSI3144A_mask_off_read( )	Read back mask off setting
67	LSI3144A_segment_control_set( )	Enable / disable segment mask control
68	LSI3144A_segment_control_read( )	Read back setting of segment mask control
69	LSI3144A_com_trigger_control( )	Set up X axis as common trigger source
70	LSI3144A_counter_direction_read( )	To read LSI3144A card counter. The 32bit counter on the fly value will return
71	LSI3144A_PWM_set( )	Set the PWM frequency and duty
72	LSI3144A_PWM_read( )	Read the PWM value
73	LSI3144A_PWM_control_set( )	Enable/disable of PWM function
74	LSI3144A_PWM_control_read( )	Read back the PWM control setting
75	LSI3144A_PWM_motion_config_ set( )	Setup parameters of X,Y vector speed to PWM
76	LSI3144A_PWM_motion_config_ read( )	Read back parameters of X,Y vector speed to PWM
77	LSI3144A_PWM_motion_control_ set( )	Enable/disable vector speed to PWM function
78	LSI3144A_PWM_motion_control_ read( )	Read back status of Enable/disable vector speed to PWM function
79	LSI3144A_timer_set( )	Set up timer constant
80	LSI3144A_timer_read( )	Read back timer data on the fly
81	LSI3144A_timer_start( )	Timer starts operation
82	LSI3144A_timer_stop( )	Timer stops operation
83	LSI3144A_IRQ_mask_set( )	Setup interrupt source mask
84	LSI3144A_IRQ_mask_read( )	Read back setting of interrupt source mask
85	LSI3144A_IRQ_process_link( )	Link IRQ process to system
86	LSI3144A_IRQ_status_read( )	Read back the interrupt flags
87	LSI3144A_IRQ_enable( )	Enable interrupt
88	LSI3144A_IRQ_disable( )	Disable interrupt
89	LSI3144A_Latch_FIFO_initial( )	Initial Latch FIFO
90	LSI3144A_Latch_FIFO_control_set( )	Latch FIFO Control set
91	LSI3144A_Latch_FIFO_control_read( )	Latch FIFO Control read
92	LSI3144A_Latch_FIFO_status_read( )	Latch FIFO status read
93	LSI3144A_Latch_FIFO_data_read( )	Latch FIFO data read
94	LSI3144A_password_set( )	Set password

95	LSI3144A_password_change( )	Change password
96	LSI3144A_password_clear( )	Clear password
97	LSI3144A_security_unlock( )	Unlock the card
98	LSI3144A_security_status_read( )	Read back the lock status

11. LSI3144A Error code summary

11.1 LSI3144A Error code table

Error Code	Symbolic Name	Description
0	JSDRV_NO_ERROR	No error.
2	JSDRV_INIT_ERROR	Initial error
3	JSDRV_UNLOCK_ERROR	Security unclock failure
4	JSDRV_LOCK_COUNTER_ERROR	Dead lock, unclock failure more than 10 times
5	SDRV_SET_SECURITY_ERROR	Password overwrite error
100	DEVICE_IO_ERROR	Device drive error
101	JSDRV_NO_CARD	No card find error
102	JSDRV_DUPLICATE_ID	Card duplicate error
300	JSLSI_ID_ERROR	CardID setting error, CardID doesn’t match the DIP SW setting
301	JSLSI_COUNTER_MODE_ERROR	LSI3144A_set_counter_mode(),"mode" parameter out of range.
302	JSLSI_QUADRATURE_TIMES_ERROR	LSI3144A_set_quadrature_times(),"time" parameter out of range.
303	JSLSI_LATCH_CTRL_ERROR	LSI3144A_latch_control(),"control" parameter out of range.
304	JSLSI_LATCH_MODE_ERROR	LSI3144A_latch_mode(),"mode" parameter out of range.
305	JSLSI_POINT_ERROR	"point" parameter out of range.
306	JSLSI_AXIS_ERROR	"axis" parameter out of range.
307	JSLSI_CLR_OUT_MODE_ERROR	LSI3144A_set_clr_out_mode(),"mode" parameter out of range.
308	JSLSI_HOME_MODE_ERROR	LSI3144A_set_hard_homing(),"mode" parameter out of range.
309	JSLSI_POLARITY_ERROR	"polarity" parameter out of range.
310	JSLSI_ON_OFF_ERROR	"on_off" parameter out of range.
311	JSLSI_TRIGGER_CTRL_ERROR	LSI3144A_com_trigger_control(),"control" parameter out of range.
312	JSLSI_COMPARE_OUT_MODE_ERROR	setting of compare out mode error
313	JSLSI_FIFO_FULL_ERROR	push into new data while FIFO full
314	JSLSI_FIFO_EMPTY_ERROR	pop out data while FIFO empty
315	JSLSI_OTHER_PAR_ERROR	parameter error
400	JSLSI_DRIVER_NOT_SUPPORT	driver not support interrupt function.

401	JSLSI_VALUE_ERROR	data value error (out of range)
402	JSLSI_SORUCE_ERROR	IRQ source error
403	JSLSI_INDEX_ERROR	compare mask off segment index error
404	JSLSI_MODE_ERROR	CLR output mode error
405	JSLSI_PARAMETER_ERROR	command parameter error