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

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

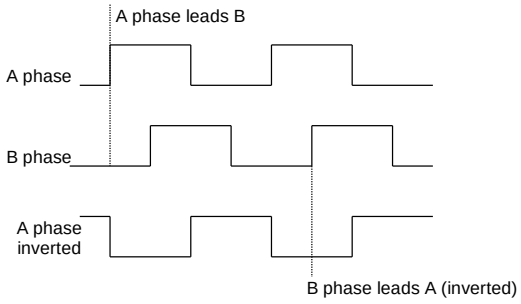
Input debounce time

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.

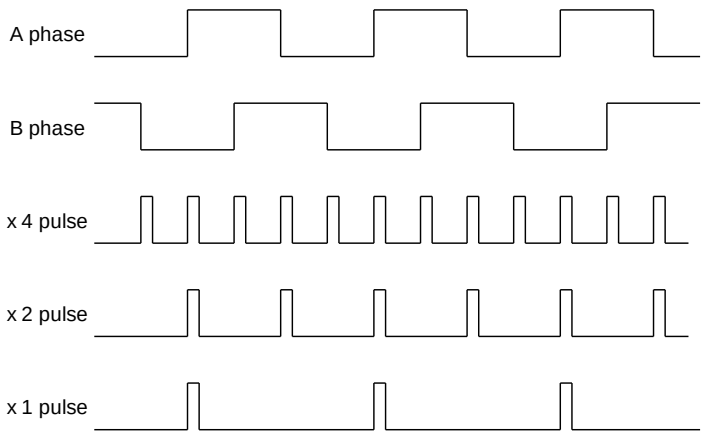


From left diagram, you can see A phase change polarity is equivalent to change A phase from lead state to lag state.

Signal input type

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

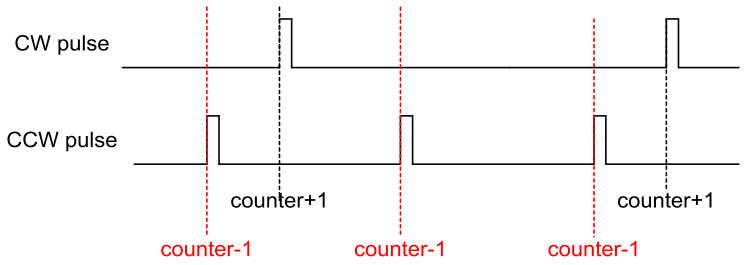
Quadrature input type



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.

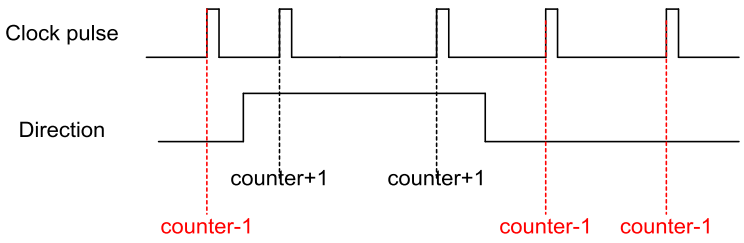
On the other hand, if B phase leads A phase, the counter will be down count.

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)



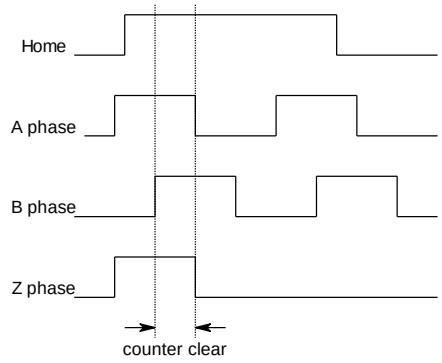
The left diagram shown that Clock and Direction pulses. Any Clock pulse input will increase counter by 1 while the Direction signal is make and any Clock pulse input will decrease counter by 1 while Direction signal is break.

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”.

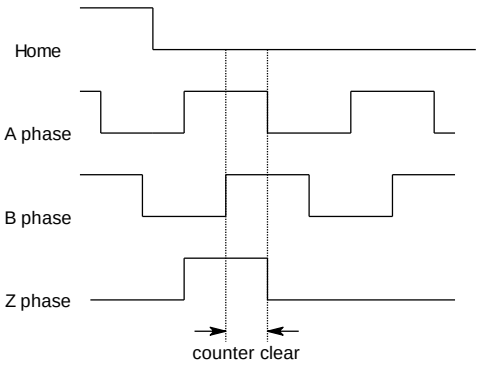
There are several modes to do homing:

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



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

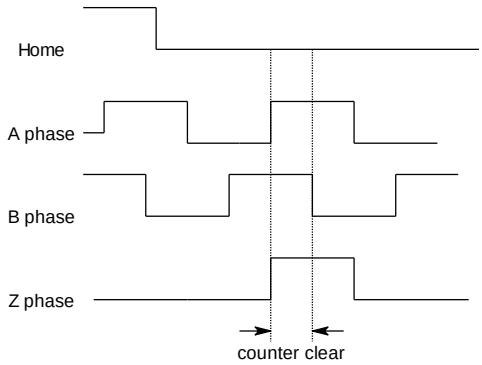
2. counter clear at first A,B,Z active after HOME turn to inactive and up count



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).

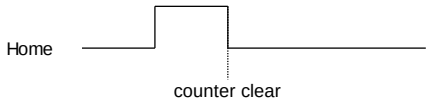
3. counter clear at first A,B,Z active after HOME turn to inactive and down count



The counter will be cleared while the following conditions meet:

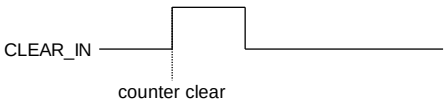
1. the HOME switch turns into “BREAK” state.
2.first A,B Z are all “MAKE” and counter down count (suppose B phase leads A phase).

4. counter clear at tailing edge of HOME



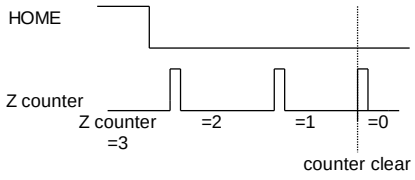
The counter will be cleared at the tailing edge of the HOME switch.

5. counter clear at rising edge of CLEAR_IN



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

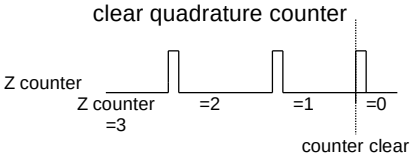
6. Trailing edge of HOME starts Z phase counter and count down to "0" clear quadrature counter



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.

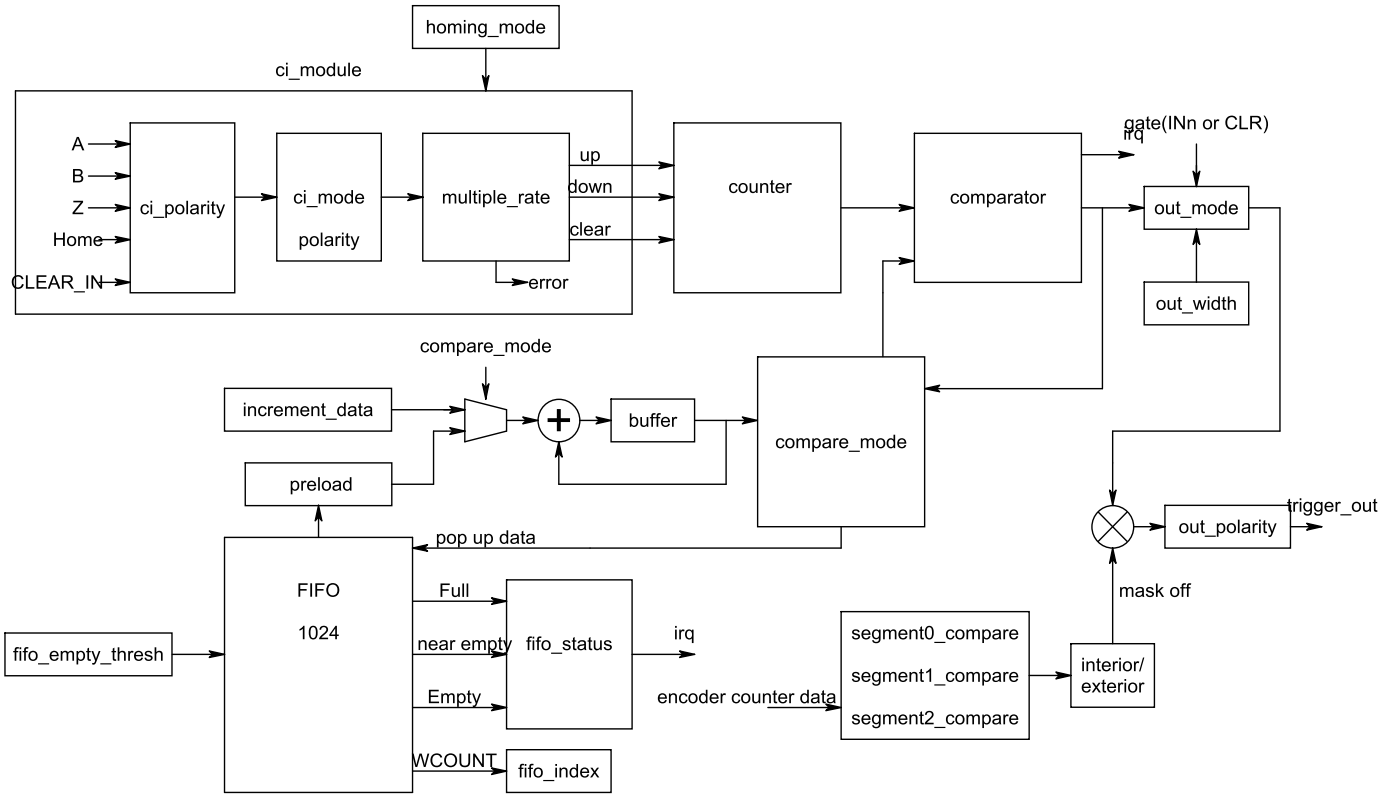


fig 6.1 counter function block diagram

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)

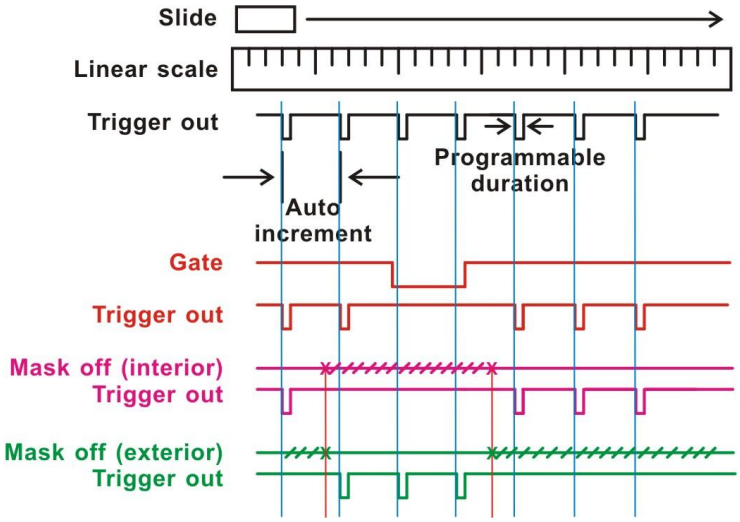


fig 6.2 Auto increment compare and compare output mask function

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.

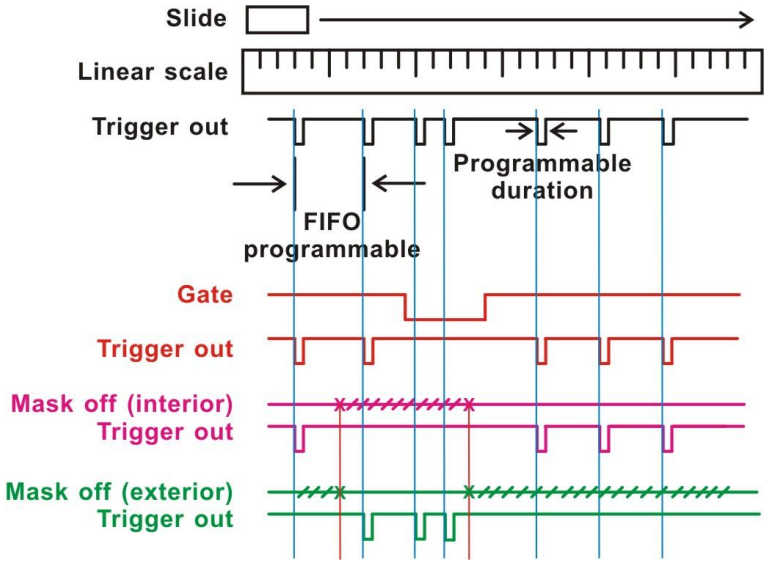


fig. 6.3 FIFO compare and compare output mask function

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.

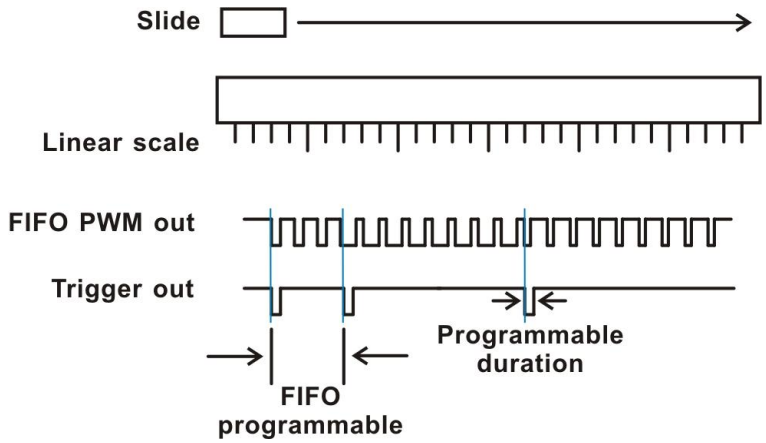


fig 6.4 FIFO compare and PWM FIFO function

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.

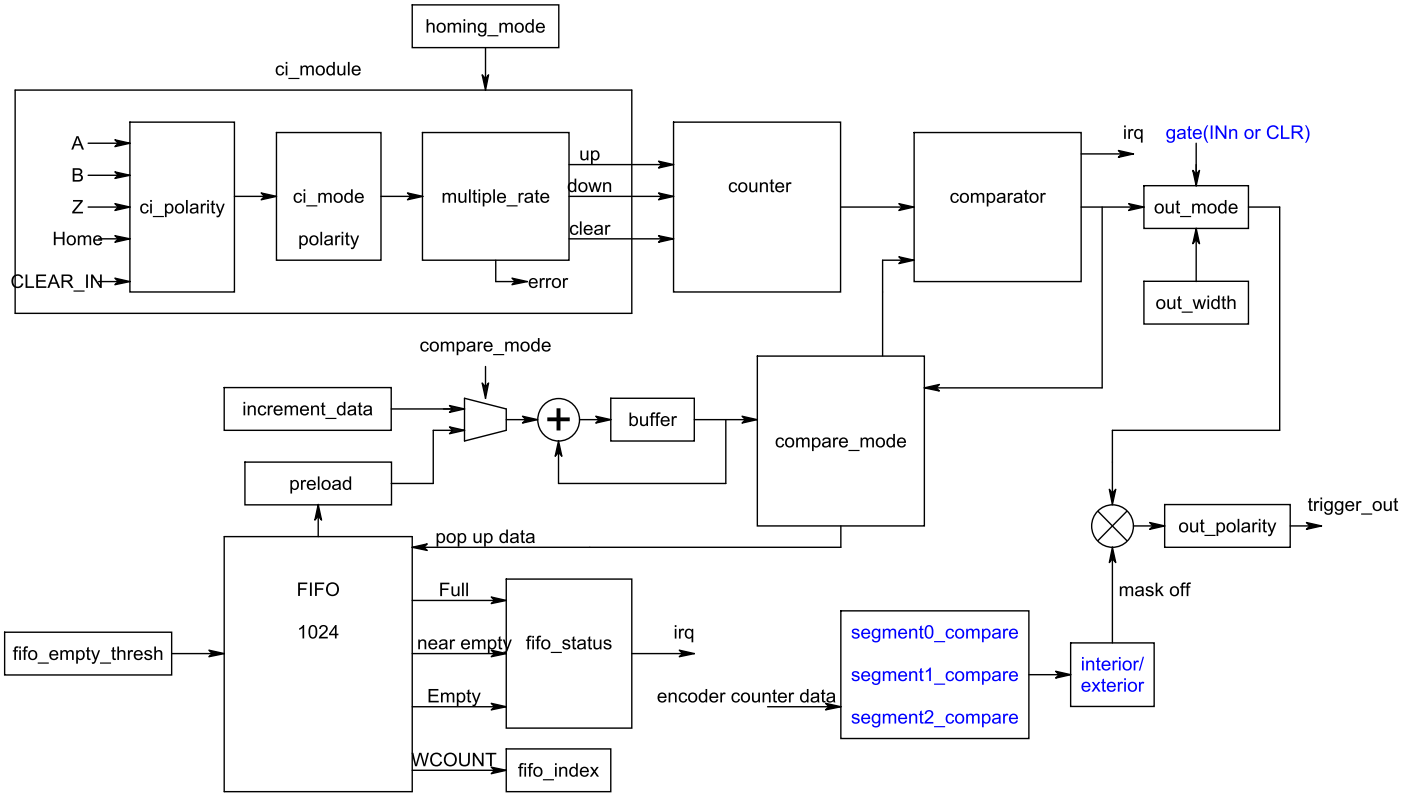


fig 6.5 segment mask off and external gate

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

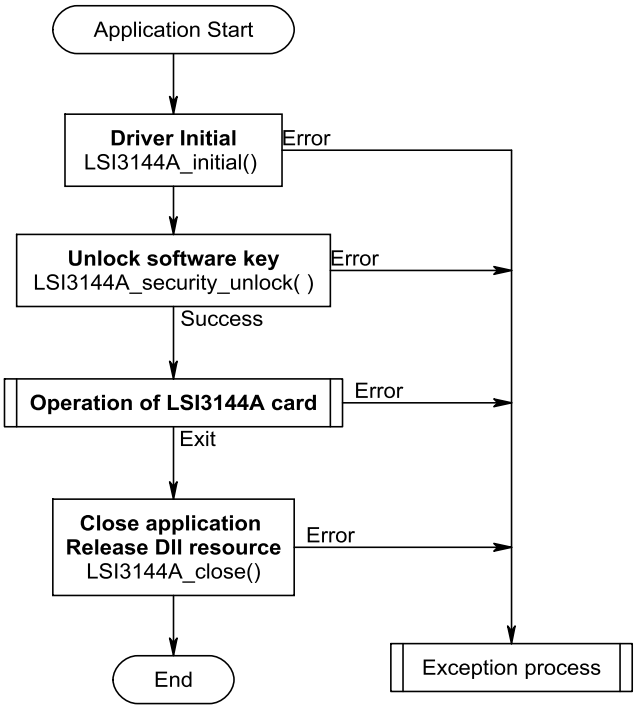


fig. 8-1 main application flow

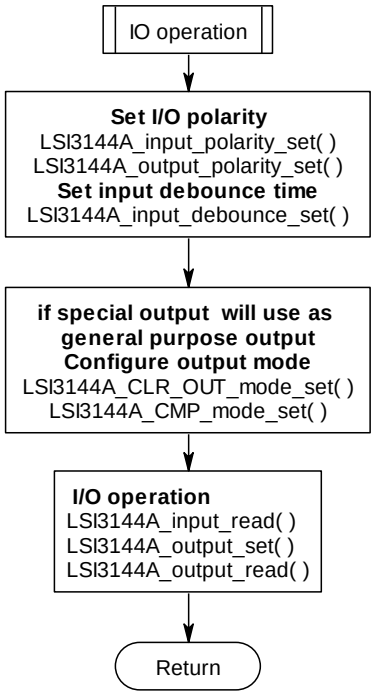


fig. 8-2 I/O operation flow

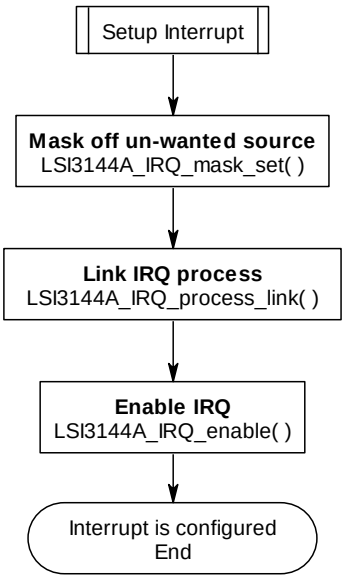


fig. 8-3 Interrupt set-up flow

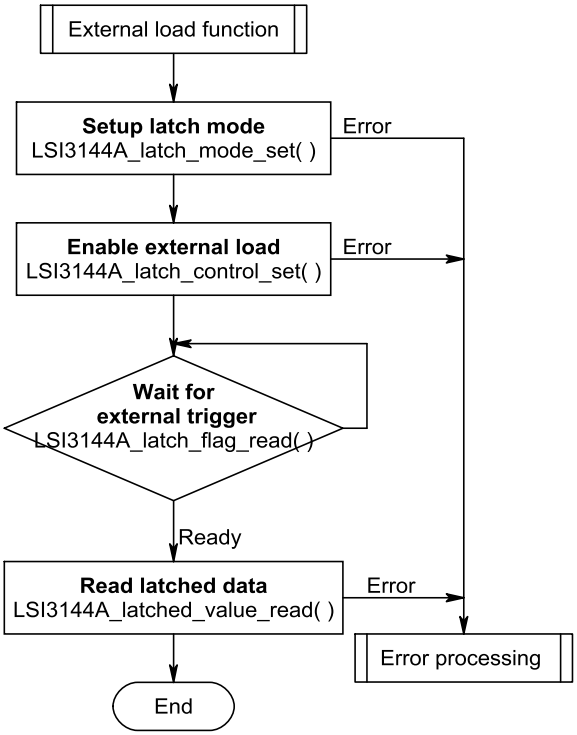


fig. 8-4 External load function flow