LS7366R Library User Guide

This user guide will help describe the usage of the LS/CSI LS7366R-S Quadrature Encoder Counter library developed for Arduino. This library creates an object that makes interfacing with the encoder counter easier allowing for interface through simple functions rather than through bare-metal SPI programming.

Constructors

Default constructor

The default constructor for the LS7366R only requires two parameters and configures the encoder counter with the settings most widely applicable to most users. These default settings include:

- No Glitch Filter Clock Division (Clock Division = 1)
- Free Run Encoder Counting
- 2-Byte wide Counter Register
- 4x Quadrature counting.

The use of this constructor is:

LS7366R <object name>(<chip select pin>, <count enable pin>);

Custom Configuration Constructor

The Custom Configuration Constructor allows the user full access to the mode registers at counter initialization. These open up options for the user for everything such as quadrature count resolution and counter width to interrupt configuration. The two 8-bit mode registers can be accessed through additional parameters to the constructor.

The use of this constructor is:

LS7366R <object name>(<chip select pin>, <count enable pin>, <MDR0 settings>, <MDR1 settings>);

The MDR Settings are set using the table below as a guide and the settings are used as such:

<MDR0 settings> = MDR0 X4_QUAD | FREE_RUN | DISABLE_IDX | ASYN_IDX | CLK_DIV_1

Leaving a field out of the settings list will cause that setting to be set to its default.

Table 1: MDR0 Settings. This register defaults to 0 at power up

MDR0 Settings		
Section	**Function**	**Constant**
Counts per Quadrature Cycle	**Non-Quadrature (A = clk, B = direction)**	**NON_QUAD**
	x1 Quadrature Mode (1 count/cycle)	X1_QUAD
	X2 Quadrature Mode (2 counts/cycle)	X2_QUAD
	X4 Quadrature Mode (4 counts/cycle)	X4_QUAD
Count Mode	**Free-Running count mode**	**FREE_RUN**
	Single Cycle count mode*	SINGLE_CYCLE
	Range Limit mode*	RANGE_LIMIT
	Mod-n Count mode*	MOD_N_CNT
Index Mode	**Disable index**	**DISABLE_IDX**
	Load DTR into Counter on index	LOAD_CNTR
	Reset Counter on index	RST_CNTR
	Save Counter to OTR on index	LOAD_OTR
Index Synchronization	**Asynchronous Index***	**ASYN_IDX**
	Synchronous Index*	SYN_IDX
Glitch Filter Clock Division	**Filter Clock Division = 1**	**CLK_DIV_1**
	Filter Clock Division = 2	CLK_DIV_2

*experienced users only; please refer to the LS7366R datasheet for more information

Bold values are the default values at power up

Table 2: MDR1 Settings. This register defaults to 0 at power up

MDR1 Settings		
Section	**Function**	**Constant**
Counter Bytes	**4-byte counter mode** 3-byte counter mode 2-byte counter mode 1-byte counter mode	**FOUR_BYTE** THREE_BYTE TWO_BYTE ONE_BYTE
Counter Enable	**Enable counter** Disable counter	**ENABLE_CNT** DISABLE_CNT
Index Interrupt	Interrupt on Index Pulse **Not Enabled**	FLAG_IDX **-**
Compare Interrupt	Interrupt when Counter = DTR* **Not Enabled**	FLAG_CMP **-**
Borrow Interrupt	Interrupt when Counter Underflows (0 -> max) **Not Enabled**	FLAG_BW **-**
Carry-Over Interrupt	Interrupt when Counter overflows (max->0) **Not Enabled**	FLAG_CY **-**

*experienced users only; please refer to the LS7366R datasheet for more information
Bold values are the default values

Discussion of Settings

- **Counts per Quadrature Cycle:**
  The counts per quadrature cycle setting determines how many events per quadrature cycle will increment (or decrement) the counter. In a quadrature cycle there are four events: both clock edges on both inputs.
  - On **x4 Quadrature** mode all four clock edges affect the counter.
  - On x2 Quadrature mode, only the rising edges (of falling edges, LS/CSI is not clear on which) will cause the counter to increment or decrement.
  - For x1 Quadrature mode, only a complete quadrature cycle will cause the counter to increment or decrement.
  - Finally, non-quadrature mode is used for non-quadrature inputs, where channel A is used to increment (or decrement) the clock and channel B determines whether the clock should be incremented or decremented.

- **Count Mode:**
  The count mode controls how the counter runs through counter division and range limiting. The settings other than Free Run may provide useful in specific applications.
    - In **Free-Run Count** mode the counter will continuously count past overflows, underflows.
    - In Single-cycle count mode the counter will run operate normally until the counter overflows or underflows. At this point the counter will pause counting until handled.
    - In Range Limit mode the counter will operate the same as single cycle mode except the counter's limits will be from 0 to the value stored in DTR and the count will resume once the direction resumes away from the boundary.
    - In Modulo-N count mode, this functionality is unknown. From the datasheet: (input count clock frequency is divided by a factor of (n+1), where n = DTR, in both up and down directions)
- **Index Mode:**
  The Index mode determines the functionality of the Index pulse when it arrives. Note that this does not impact the interrupt setting for the index; rather it enables special settings for the index to be carried out asynchronously. These functions may prove helpful during homing depending on your application
    - The **Disable Index** setting does nothing on index pulse
    - The Load Counter setting will place the value in DTR into the counter on index pulse. The Reset Counter setting clears the Counter on index pulse
    - The Load OTR setting loads the value in the counter into the OTR register to be read by the readOTR() function.
- **Index Synchronization:**
  This functionality is documented as existing in the datasheet but the true functionality is unknown.
    - **Asynchronous Index**
    - Synchronous Index: listed as overridden in non-quadrature mode.
- **Glitch Filter Clock Division**
  This sets the clock division of the counter's input glitch filter. A higher division factor will determine more precise readings at the cost of a lower maximum count frequency. Enable the higher clock division for high noise environments to provide better noise immunity.
    - **Filter clock division = 1:** 40MHz Glitch filter clock input
    - Filter clock division = 2: 20MHz Glitch filter clock input

- **Counter Bytes:**
  This setting sets the number of bytes used in the counter. The full resolution is a 32-bit unsigned number, but not all applications require this much space or may desire the ability to count as high. Keep in mind that the Arduino Uno can handle 32-bit numbers but prefers 16-bit integers which will result in faster calculations.
    - **Four Byte Mode** enables the full 32-bit range of the counter 0->4.2 Billion
    - Three Byte Mode allows for a range of 0->16 Million
    - Two Byte Mode allows for a range of 0->65 thousand
    - One Byte Mode allows for a range of 0-> 256
- **Counter Enable:**
  Enables or disables the counter in software. When single-cycle count mode is used this will resume counting. At this point this is allowed in the settings, but in the current implementation is not useful. In a future implementation this functionality will be better integrated.
    - Enable Counter: resumes counting if paused
    - Disable Counter: freezes counting if running
- **Interrupts**:
  Interrupts will cause the LFLAG pin to pull its output low signaling an interrupt has occurred. Calling the readSTR() function will return the status register and clear the interrupt. The LFLAG interrupt is an open-drain latching interrupt which should be connected to an interrupt-enabled pin with a pull-up resistor. See the readSTR() documentation below for more information on retrieving the interrupt cause.
    - Index Interrupt: Interrupt on an index pulse
    - Index Compare Interrupt: Interrupt when the counter equals the value in DTR
    - Borrow Interrupt: Interrupt when the counter underflows or goes below 0
    - Carry-over Interrupt: Interrupt when the counter overflows or goes above its max

Functions

unsigned long readEncoder()

Reads the encoder's current count and returns the result as an unsigned long

void clearEncoder()

Resets the Encoder's current count to 0

void enableEncoder()

Pulls the counter's count enable pin high to resume counting after being paused or after power up

void disableEncoder()

Pulls the counter's count enable pin low to pause counting

void writeDTR(unsigned long <value to write>)

Writes the parameter value to the DTR register for use with the Compare Interrupt, Range Limit counting, Mod-N counting, and presetting the counter register on index pulse

byte readSTR()

Reads then clears the status register and returns the status register as the return value. This function also clears the LFLAG latching interrupt. The table below discusses bit masks for the Status register for if statements.

Name	Description	Bit Mask Name
Carry Latch	Latch set when CNTR overflows	CY_MASK
Borrow Latch	Latch set when CNTR underflows	BW_MASK
Compare Latch	Latch set when CNTR = DTR	CMP_MASK
Index Latch	Latch set by Index Pulse	IDX_MASK
Count Enabled	0: count disabled, 1: count enabled	CEN_MASK
Power Loss Indicator	1: if status has not been cleared since last power up	PLS_MASK
Count Direction	0: count down, 1: count up	UD_MASK
Sign Bit	Mostly Unknown at this time	S_MASK

Keep in mind that multiple bits can be set at once. The above bits masks can be used as follows:

```
If (<object name>.readSTR() & CY_MASK) {
        // code for when carry-over has occurred
}
```

An interrupt that was triggered by the LS7366R can be cleared through reading the status register. If multiple interrupts are enabled ensure you check all applicable bits as many may have triggered before you read the register. Do not call this function in an ISR.

unsigned long readOTR()

Coming soon

void setCounter(unsigned long)

Coming soon