LSI/CSI

LS7366R

LSI Computer Systems, Inc. 1235 Walt Whitman Road, Melville, NY 11747 (631) 271-0400 FAX (631) 271-0405

32-BIT QUADRATURE COUNTER WITH SERIAL INTERFACE

June 2012

GENERAL FEATURES:

- Operating voltage: 3V to 5.5V (V_{DD} - V_{SS})
- 5V count frequency: 40MHz
- 3V count frequency: 20MHz
- 32-bit counter (CNTR).
- 32-bit data register (DTR) and comparator.
- 32-bit output register (OTR).
- Two 8-bit mode registers (MDR0, MDR1) for programmable functional modes.
- 8-bit instruction register (IR).
- 8-bit status register (STR).
- Latched Interrupt output on Carry or Borrow or Compare or Index.
- Index driven counter load, output register load or counter reset.
- Internal quadrature clock decoder and filter.
- x1, x2 or x4 mode of quadrature counting.
- Non-quadrature up/down counting.
- Modulo-N, Non-recycle, Range-limit or Free-running modes of counting
- 8-bit, 16-bit, 24-bit and 32-bit programmable configuration synchronous (SPI) serial interface
- **LS7366R** (DIP), **LS7366R-S** (SOIC), **LS7366R-TS** (TSSOP) **- See Figure 1 -**

SPI/MICROWIRE (Serial Peripheral Interface):

- Standard 4-wire connection: MOSI, MISO, SS/ and SCK.
- Slave mode only.

GENERAL DESCRIPTION:

LS7366R is a 32-bit CMOS counter, with direct interface for quadrature clocks from incremental encoders. It also interfaces with the index signals from incremental encoders to perform variety of marker functions.

For communications with microprocessors or microcontrollers, it provides a 4-wire SPI/MICROWIRE bus.The four standard bus I/Os are SS/, SCK, MISO and MOSI. The data transfer between a microcontroller and a slave LS7366R is synchronous. The synchronization is done by the SCK clocks supplied by the microcontroller. Each transmission is organized in blocks of 1 to 5 bytes of data. A transmission cycle is intitiated by a high to low transition of the SS/ input. The first byte received in a transmission cycle is always an instruction byte, whereas the second through the fifth bytes are always interpreted as data bytes. A transmission cycle is terminated with the low to high transition of the SS/ input. Received bytes are shifted in at the MOSI input, MSB first, with the leading edges (high transition) of the SCK clocks. Output data are shifted out on the MISO output, MSB first, with the trailing edges (low transition) of the SCK clocks.

PIN ASSIGNMENT
TOP VIEW

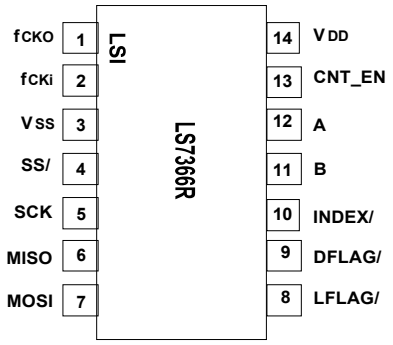


FIGURE 1

Read and write commands cannot be combined. For example, when the device is shifting out read data on MISO output, it ignores the MOSI input, even though the SS/ input is active. SS/ must be terminated and reasserted before the device will accept a new command.

The counter can be configured to operate as 1, 2, 3 or 4-byte counter. When configured as an n-byte counter, the CNTR, DTR and OTR are all configured as n-byte registers, where n = 1, 2, 3 or 4.
The content of the instruction/data identity is automatically adjusted to match the n-byte configuration. For example, if the counter is configured as a 2-byte counter, the instruction “write to DTR” expects 2 data bytes following the instruction byte. If the counter is configured as a 3-byte counter, the same instruction will expect 3 bytes of data following the instruction byte.

Following the transfer of the appropriate number of bytes any further attempt of data transfer is ignored until a new instruction cycle is started by switching the SS/ input to high and then low.

The counter can be programmed to operate in a number of different modes, with the operating characteristics being written into the two mode registers MDR0 and MDR1. Hardware I/Os are provided for event driven operations, such as processor interrupt and index related functions.

I/O Pins:
Following is a description of all the input/output pins.

A (Pin 12) **B** (Pin 11)
Inputs. A and B quadrature clock outputs from incremental encoders are directly applied to the A and B inputs of the LS7366R. These clocks are ideally 90 degrees out-of-phase signals. A and B inputs are validated by on-chip digital filters and then decoded for up/down direction and count clocks.
In non-quadrature mode, A serves as the count input and B serves as the direction input (B = high enables up count, B = low enables down count). In non-quadrature mode, the A and B inputs are not filtered internally, and are instantaneous in nature.

INDEX/ (Pin 10)
Input. The INDEX/ is a programmable input that can be driven directly by the Index output of an incremental encoder. It can be programmed via the MDR0 to function as one of the following:
LCNTR (load CNTR with data from DTR), RCNTR (reset CNTR), or LOTR (load OTR with data from CNTR). Alternatively, the INDEX input can be masked out for "no functionality".

In quadrature mode, the INDEX/ input can be configured to operate in either synchronous or asynchronous mode. In the synchronous mode the INDEX/ input is sampled with the same filter clock used for sampling the A and the B inputs and must satisfy the phase relationship in which the INDEX/ is in the active level of Logic 0 during a minimum of a quarter cycle of both A and B High or both A and B Low. In non-quadrature mode, the INDEX/ input is unconditionally set to the asynchronous mode. In the asynchronous mode, the INDEX/ input is not sampled and can be applied in any phase relationship with respect to A and B.

f_{CKi} (Pin 2)**, f_{CK0}** (Pin 1)
Input, Output. A crystal connected between these 2 pins generates the basic clock for filtering the A, B and INDEX/ inputs in the quadrature count mode. Instead of a crystal the **f_{CKi}** input may also be driven by an external clock.

The frequency at the f_{CKi} input is either divided by 2 (if MDR0 <B7> = 1) or divided by 1 (if MDR0 <B7> = 0) for the filter circuit. For proper filtering of the A, B and the Index/ inputs the following condition must be satisfied:

$$f_f \geq 4f_{QA}$$

Where f_f is the internal filter clock frequency derived from the f_{CKi} in accordance with the status of MDR0 <B7> and f_{QA} is the maximum frequency of Clock A in quadrature mode. In non-quadrature count mode, f_{CKi} is not used and should be tied off to any stable logic state.

SS/ (Pin 4)
A high to low transition at the SS/ (Slave Select) input selects the LS7366R for serial bi-directional data transfer; a low to high transition disables serial data transfer and brings the MISO output to high impedance state. This allows for the accommodation of multiple slave units on the serial I/O.

CNT_EN (Pin 13)
Input. Counting is enabled when CNT_EN input is high; counting is disabled when this input is low. There is an internal pull-up resistor on this input.

LFLAG/ (Pin 8), **DFLAG/** (Pin 9)
Outputs. LFLAG/ and DFLAG/ are programmable outputs to flag the occurences of Carry (counter overflow), Borrow (counter underflow), Compare (CNTR = DTR) and INDEX. The LFLAG/ is an open drain latched output. In contrast, the DFLAG/ is a push-pull instantaneous output. The LFLAG/ can be wired in multi-slave configuration, forming a single processor interrupt line. When active LFLAG/ switches to logic 0 and can be restored to the high impedence state only by clearing the status register, STR. In contrast, the DFLAG/ dynamically switches low with occurences of Carry, Barrow, Compare and INDEX conditions.

The configuration of LFLAG/ and DFLAG/ are made through the control register MDR1.

MOSI (RXD) (Pin 7)
Input. Serial output data from the host processor is shifted into the LS7366R at this input.

MISO (TXD) (Pin 6)
Output. Serial output data from the LS7366R is shifted out on the MISO (Master In Slave Out) pin. The MISO output goes into high impedance state when SS/ input is at logic high, providing multiple slave-unit serial outputs to be wire-ORed.

SCK (Pin 5)
Input. The SCK input serves as the shift clock input for transmitting data in and out of LS7366R on the MOSI and the MISO pins, respectively. Since the LS7366R can operate only in the slave mode, the SCK signal is provided by the host processor as a means for synchronizing the serial transmission between itself and the slave LS7366R.

REGISTERS:
The following is a list of LS7366R internal registers:

Upon power-up the registers DTR, CNTR, STR, MDR0 and MDR1 are reset to zero.

DTR. The DTR is a software configurable 8, 16, 24 or 32-bit input data register which can be written into directly from MOSI, the serial input. The DTR data can be transferred into the 32-bit counter (CNTR) under program control or by hardware index signal. The DTR can be cleared to zero by software control. In certain count modes, such as modulo-n and range-limit, DTR holds the data for "n" and the count range, respectively. In compare operations, whereby compare flag is set, the DTR is compared with the CNTR.

CNTR. The CNTR is a software configurable 8, 16, 24 or 32-bit up/down counter which counts the up/down pulses resulting from the quadrature clocks applied at the A and B inputs, or alternatively, in non-quadrature mode, pulses applied at the A input. By means of IR intructions the CNTR can be cleared, loaded from the DTR or in turn, can be transferred into the OTR.

OTR. The OTR is a software configuration 8, 16, 24 or 32-bit register which can be read back on the MISO output. Since instantaneous CNTR value is often needed to be read while the CNTR continues to count, the OTR serves as a convenient dump site for instantaneous CNTR data which can then be read without interfering with the counting process.

STR. The STR is an 8-bit status register which stores count related status information.

CY	BW	CMP	IDX	CEN	PLS	U/D	S
7	6	5	4	3	2	1	0

CY: Carry (CNTR overflow) latch
BW: Borrow (CNTR underflow) latch
CMP: Compare (CNTR = DTR) latch
IDX: Index latch
CEN: Count enable status: 0: counting disabled, 1: counting enabled

PLS: Power loss indicator latch; set upon power up
U/D: Count direction indicator: 0: count down, 1: count up
S: Sign bit. 1: negative, 0: positive

IR. The IR is an 8-bit register that fetches instruction bytes from the received data stream and executes them to perform such functions as setting up the operating mode for the chip (load the MDR) and data transfer among the various registers.

B7	B6	B5	B4	B3	B2	B1	B0

B2 B1 B0 = XXX (Don't care)
B5 B4 B3 = 000: Select none
= 001: Select MDR0
= 010: Select MDR1
= 011: Select DTR
= 100: Select CNTR
= 101: Select OTR
= 110: Select STR
= 111: Select none
B7 B6 = 00: CLR register
= 01: RD register
= 10: WR register
= 11: LOAD register

The actions of the four functions, CLR, RD, WR and LOAD are elaborated in Table 1.

TABLE 1

Number of Bytes	OP Code	Register	Operation
1	CLR	MDR0	Clear MDR0 to zero
		MDR1	Clear MDR1 to zero
		DTR	None
		CNTR	Clear CNTR to zero
		OTR	None
		STR	Clear STR to zero
2 to 5	RD	MDR0	Output MDR0 serially on TXD (MISO)
		MDR1	Output MDR1 serially on TXD (MISO)
		DTR	None
		CNTR	Transfer CNTR to OTR, then output OTR serially on TXD (MISO)
		OTR	Output OTR serially on TXD (MISO)
		STR	Output STR serially on TXD (MISO)
2 to 5	WR	MDR0	Write serial data at RXD (MOSI) into MDR0
		MDR1	Write serial data at RXD (MOSI) into MDR1
		DTR	Write serial data at RXD (MOSI) into DTR
		CNTR	None
		OTR	None
1	LOAD	STR	None
		MDR0	None
		MDR1	None
		DTR	None
		CNTR	Transfer DTR to CNTR in "parallel"
		OTR	Transfer CNTR to OTR in "parallel"

MDR0. The MDR0 (Mode Register 0) is an 8-bit read/write register that sets up the operating mode for the LS7366R. The MDR0 is written into by executing the "write-to-MDR0" instruction via the instruction register. Upon power up MDR0 is cleared to zero. The following is a breakdown of the MDR bits:

B7	B6	B5	B4	B3	B2	B1	B0

B1 B0 = 00: non-quadrature count mode. (A = clock, B = direction).
= 01: x1 quadrature count mode (one count per quadrature cycle).
= 10: x2 quadrature count mode (two counts per quadrature cycle).
= 11: x4 quadrature count mode (four counts per quadrature cycle).

B3 B2 = 00: free-running count mode.
= 01: single-cycle count mode (counter disabled with carry or borrow, re-enabled with reset or load).
= 10: range-limit count mode (up and down count-ranges are limited between DTR and zero, respectively; counting freezes at these limits but resumes when direction reverses).
= 11: modulo-n count mode (input count clock frequency is divided by a factor of (n+1), where n = DTR, in both up and down directions).

B5 B4 = 00: disable index.
= 01: configure index as the "load CNTR" input (transfers DTR to CNTR).
= 10: configure index as the "reset CNTR" input (clears CNTR to 0).
= 11: configure index as the "load OTR" input (transfers CNTR to OTR).

B6 = 0: Asynchronous Index
= 1: Synchronous Index (overridden in non-quadrature mode)
B7 = 0: Filter clock division factor = 1
= 1: Filter clock division factor = 2

MDR1. The MDR1 (Mode Register 1) is an 8-bit read/write register which is appended to MDR0 for additional modes. Upon power-up MDR1 is cleared to zero.

B7	B6	B5	B4	B3	B2	B1	B0

B1 B0 = 00: 4-byte counter mode
= 01: 3-byte counter mode
= 10: 2-byte counter mode.
= 11: 1-byte counter mode

B2 = 0: Enable counting
= 1: Disable counting
B3 = : not used
B4 = 0: NOP
= 1: FLAG on IDX (B4 of STR)
B5 = 0: NOP
= 1: FLAG on CMP (B5 of STR)
B6 = 0: NOP
= 1: FLAG on BW (B6 of STR)
B7 = 0: NOP
= 1: FLAG on CY (B7 of STR)

NOTE: Applicable to both LFLAG/ and DFLAG/

ABSOLUTE MAXIMUM RATINGS:
(All voltages referenced to Vss)

Parameter	Symbol	Values	Unit
DC Supply Voltage	V_{DD}	+7.0	V
Input Voltage	V_{IN}	Vss - 0.3 to V_{DD} + 0.3	V
Operating Temperature	T_A	-25 to +80	^oC
Storage Temperature	T_{STG}	65 to +150	^oC

DC Electrical Characteristics. (T_A = -25˚C to +85°C)

Parameter	Symbol	Min.	TYP	Max.	Unit	Remarks
Supply Voltage	V_{DD}	3.0	-	5.5	V	-
Supply Current	I_{DD}	300	400	450	µA	V_{DD} = 3.0V
	I_{DD}	700	800	950	µA	V_{DD} = 5.0V
Input Voltages						
f_{CKi}, Logic high	V_{CH}	2.3	-	-	V	V_{DD} = 3.0V
	V_{CH}	3.7	-	-	V	V_{DD} = 5.0V
f_{CKi}, Logic Low	V_{CL}	-	-	0.7	V	V_{DD} = 3.0V
	V_{CL}	-	-	1.3	V	V_{DD} = 5.0
All other inputs, Logic High	V_{AH}	2.1	-		V	V_{DD} = 3.0V
	V_{AH}	3.5	-		V	V_{DD} = 5.0V
All other inputs, Logic Low	V_{AL}	-	-	0.5	V	V_{DD} = 3.0V
	V_{AL}	-	-	1.0	V	V_{DD} = 5.0V
Input Currents:						
CNT_EN Low	I_{IEL}	-	3.0	5.0	µA	V_{AL} = 0.7V, V_{DD} = 3.0V
	I_{IEL}	-	10.0	15.0	µA	V_{AL} = 1.2V, V_{DD} = 5.0V
CNT_EN High	I_{IEH}	-	1.0	3.0	µA	V_{AH} = 1.9V, V_{DD} = 3.0V
	I_{IEH}	-	6.0	9.0	µA	V_{AH} = 3.2V, V_{DD} = 5.0V
All other inputs, High or Low	-	-	0	0	µA	-
Output Currents:						
LFLAG, DFLAG Sink	I_{OFL}	-1.3	-2.0	-	mA	V_{OUT} = 0.5V, V_{DD} = 3.0V
	I_{OFL}	-3.2	-4.0	-	mA	V_{OUT} = 0.5V, V_{DD} = 5.0V
LFLAG Source	-	0	0	-	mA	Open Drain Output
DFLAG Source	I_{OFH}	1.0	1.8	-	mA	V_{OUT} = 2.5V, V_{DD} = 3.0V
	I_{OFH}	2.8	3.6	-	mA	V_{OUT} = 4.5V, V_{DD} = 5.0V
f_{CKO} Sink	I_{OCL}	-1.3	-2.0	-	mA	V_{OUT} = 0.5V, V_{DD} = 3.0V
	I_{OCL}	-3.2	-4.0	-	mA	V_{OUT} = 0.5V, V_{DD} = 5.0V
f_{CKO} Source	I_{OCH}	1.3	2.0	-	mA	V_{OUT} = 2.5V, V_{DD} = 3.0V
	I_{OCH}	3.2	4.0	-	mA	V_{OUT} = 4.5V, V_{DD} = 5.0V
TXD/MISO:						
Sink	I_{OML}	-1.5	-2.4	-	mA	V_{OUT} = 0.5V, V_{DD} = 3.0V
	I_{OML}	-3.8	-4.8	-	mA	V_{OUT} = 0.5V, V_{DD} = 5.0V
Source	I_{OMH}	1.5	2.4	-	mA	V_{OUT} = 2.5V, V_{DD} = 3.0V
	I_{OMH}	3.8	4.8	-	mA	V_{OUT} = 4.5V, V_{DD} = 5.0V

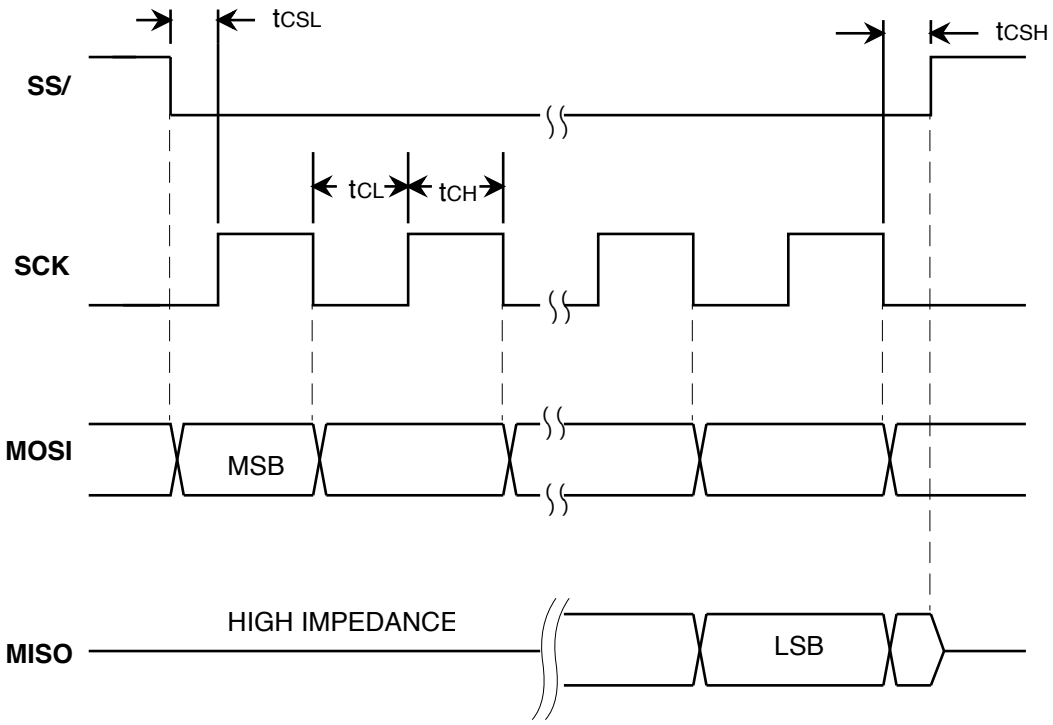
Transient Characteristics. (T_A = -25˚C to +85˚C, V_{DD} = 5V ± 10%)

Parameter	Symbol	Min. Value	Max.Value	Unit	Remarks
(See Fig. 2)					
SCK High Pulse Width	t_{CH}	100	-	ns	-
SCK Low Pulse Width	t_{CL}	100	-	ns	-
SS/ Set Up Time	t_{CSL}	100	-	ns	-
SS/ Hold Time	t_{CSH}	100	-	ns	-
Inter Command SS/ high	t_{CSI}	100	-	ns	-
Quadrature Mode					
(See Fig. 5, 7 & 8)					
f_{CKI} High Pulse Width	t_1	12	-	ns	-
f_{CKI} Pulse Width	t_2	12	-	ns	-
f_{CKI} Frequency	f_{FCK}	-	40	MHz	-
Effective Filter Clock f_F Period	t_3	25	-	ns	$t_3 = t_1+t_2$, MDR0 <7> = 0
	t_3	50	-	ns	$t_3 = 2(t_1+t_2)$, MDR0 <7> = 1
Effective Filter Clock f_F frequency	f_F	-	40	MHz	$f_F = 1/t_3$
Quadrature Separation	t_4	26	-	ns	$t_4 > t_3$
Quadrature Clock Pulse Width	t_5	52	-	ns	$t_5 \geq 2t_3$
Quadrature Clock frequency	f_{QA}, f_{QB}	-	9.6	MHz	$f_{QA} = f_{QB} < 1/4t_3$
Quadrature Clock to Count Delay	t_{Q1}	$4t_3$	$5t_3$	-	-
x1 / x2 / x4 Count Clock Pulse Width	t_{Q2}	12	-	ns	$t_{Q2} = (t_3)/2$
Index Input Pulse Width	t_{id}	32	-	ns	$t_{id} > t_4$
Index Set Up Time	t_{is}	-	5	ns	-
Index Hold Time	t_{ih}	-	5	ns	-
Quadrature clock to DFLAG/ or LFLAG/ delay	t_{fl}	$4.5t_3$	$5.5t_3$	ns	-
DFLAG/ output width	t_{fw}	26	-	ns	$t_{fw} = t_4$

Parameter	Symbol	Min. Value	Max.Value	Unit	Remarks
Non-Quadrature Mode					
(See Fig. 6 & 9)					
Clock A - High Pulse Width	t_6	12	-	ns	-
Clock A - Low Pulse Width	t_7	12	-	ns	-
Direction Input B Set-up Time	t_{8S}	12	-	ns	-
Direction Input B Hold Time	t_{8H}	10	-	ns	-
CNT_EN Set-up Time	t_{9S}	12	-	ms	-
CNT_EN Hold Time	t_{9H}	12	-	ms	-
Clock Frequency (non-Mod-N)	f_A	-	40	MHz	$f_A = (1/(t_6 + t_7))$
Clock to DFLAG/ or LFLAG/ delay	t_{10}	20	-	ns	-
DFLAG/ output width	t_{11}	12	-	ns	$t_{10} = t_7$

Transient Characteristics. (T_A = -25˚C to +85˚C, V_{DD} = 3.3V ± 10%)

Parameter	Symbol	Min. Value	Max.Value	Unit	Remarks
(See Fig. 2)					
SCK High Pulse Width	t_{CH}	120	-	ns	-
SCK Low Pulse Width	t_{CL}	120	-	ns	-
SS/ Set Up Time	t_{CSL}	120	-	ns	-
SS/ Hold Time	t_{CSH}	120	-	ns	-
Inter Command SS/ high	t_{CSI}	120	-	ns	-
Quadrature Mode					
(See Fig. 5, 7 & 8)					
f_{CKI} High Pulse Width	t_1	24	-	ns	-
f_{CKI} Pulse Width	t_2	24	-	ns	-
f_{CKI} Frequency	f_{FCK}	-	20	MHz	-
Effective Filter Clock fF Period	t_3	50	-	ns	$t_3 = t_1+t_2$, MDR0 <7> = 0
	t_3	100	-	ns	$t_3 = 2(t_1+t_2)$, MDR0 <7> = 1
Effective Filter Clock f_F frequency	f_F	-	20	MHz	$f_F = 1/ t_3$
Quadrature Separation	t_4	52	-	ns	$t_4 > t_3$
Quadrature Clock Pulse Width	t_5	105	-	ns	$t_5 \geq 2t_3$
Quadrature Clock frequency	f_{QA}, f_{QB}	-	4.5	MHz	$f_{QA} = f_{QB} < 1/4t_3$
Quadrature Clock to Count Delay	t_{Q1}	$4t_3$	$5t_3$	-	-
x1/x2/x4 Count Clock Pulse Width	t_{Q2}	25	-	ns	$t_{Q2} = (t_3)/2$
Index Input Pulse Width	t_{id}	60	-	ns	$t_{id} > t_4$
Index Set Up Time	t_{is}	-	10	ns	-
Index Hold Time	t_{ih}	-	10	ns	-
Quadrature clock to DFLAG/ or LFLAG/ delay	t_{fl}	$4.5t_3$	$5.5t_3$	ns	-
DFLAG/ output width	t_{fw}	52	-	ns	$t_{fw} = t_4$
Non-Quadrature Mode					
(See Fig. 6 & 9)					
Clock A - High Pulse Width	t_6	24	-	ns	-
Clock A - Low Pulse Width	t_7	24	-	ns	-
Direction Input B Set-up Time	t_{8S}	24	-	ns	-
Direction Input B Hold Time	t_{8H}	24	-	ns	-
Clock Frequency (non-Mod-N)	f_A	-	20	MHz	$f_A = (1/(t_6 + t_7))$
Clock to DFLAG/or LFLAG/ delay	t_9	40	-	ns	-
DFLAG/ output width	t_{10}	24	-	ns	$t_{10} = t_7$



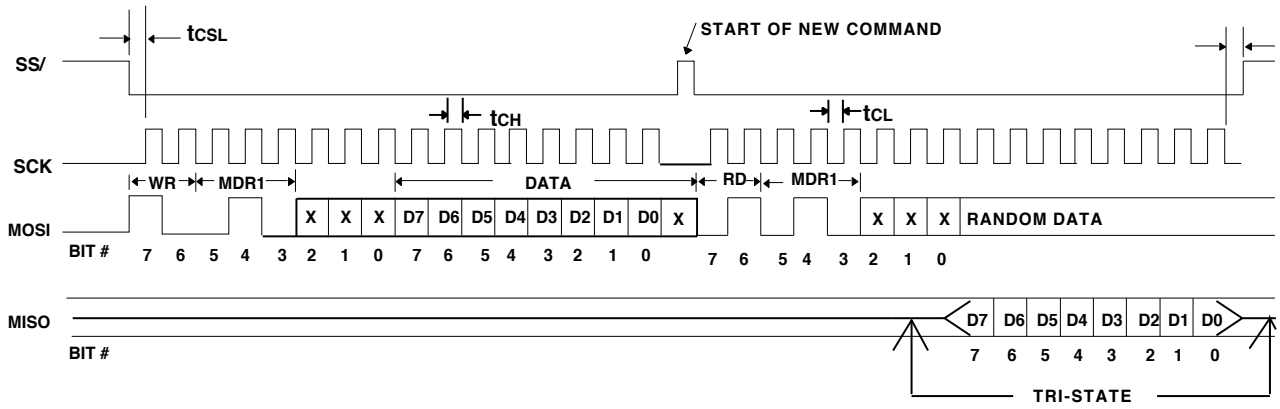
Note 1. The SPI port of the host MCU must be set up as follows:

1. SPI master mode.
2. SCK idle state = low
3. Clock edge for MOSI data shift = high to low
4. Clock edge for input data (MISO) sample by the Processor = low to high (or bit middle)

Note 2. To conform with the multibyte transmission protocol of LS7366R, the SS/ output port of the MCU may require direct manipulation by the application program.

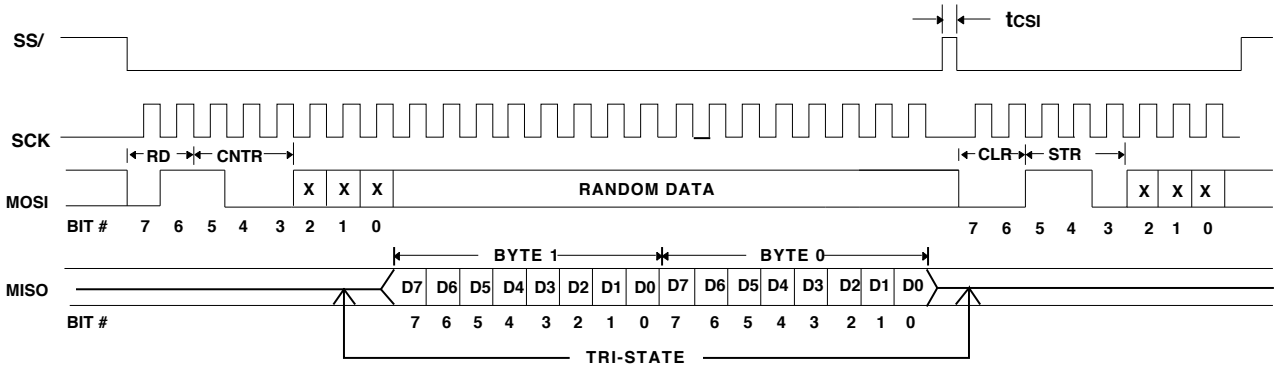
FIGURE 2. SPI TIMINGS

7366R-041906-7



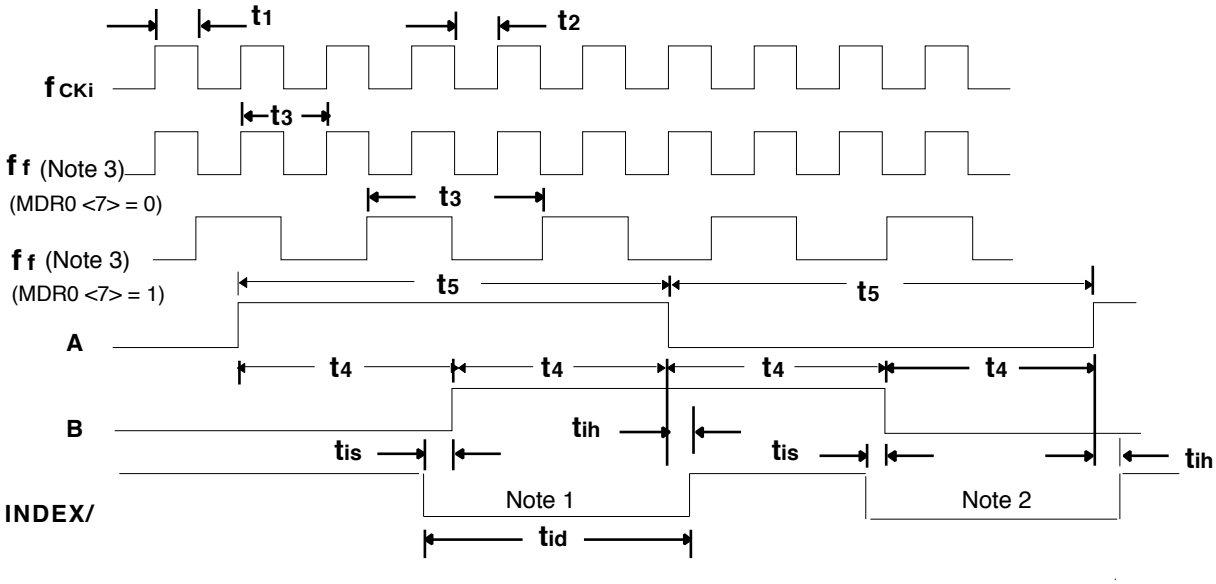
NOTE: Write to MDR1 followed by Read from MDR1 operation

FIGURE 3. WR MDR1 - RD MDR1



NOTE: Read CNTR (in 2-byte configuration) followed by CLR STR operation.

FIGURE 4. RD CNTR - CLR STR



Note 1. Synchronous index coincident with both A and B high.
Note 2. Synchronous index coincident with both A and B low..
Note 3. f_F is the internal effective filter clock.

FIGURE 5. f_{CKI}, A, B and INDEX

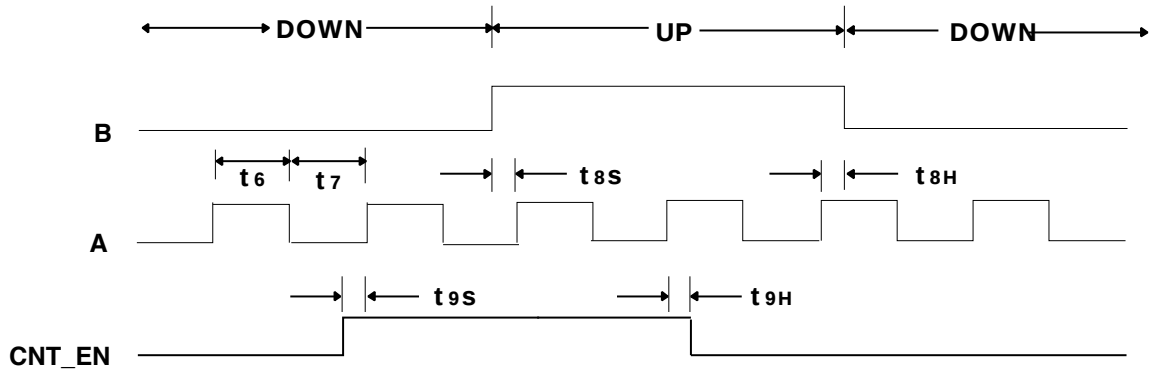
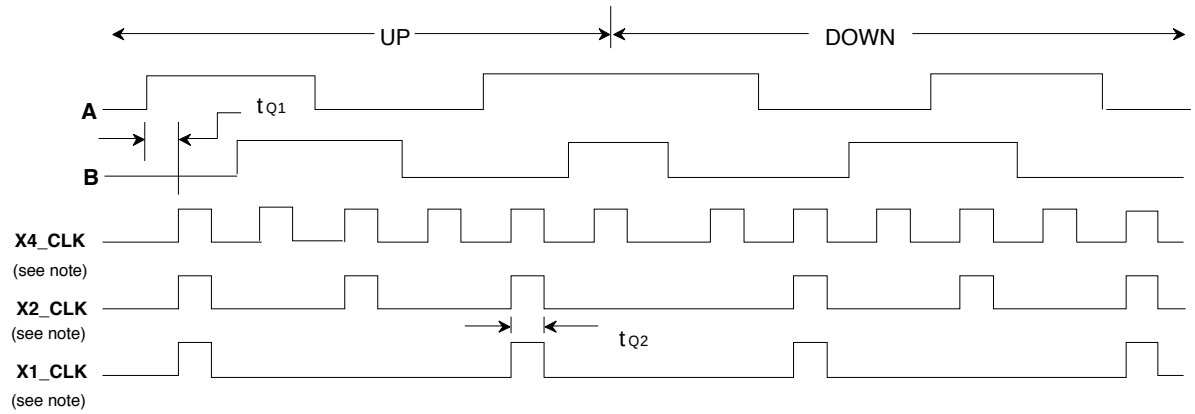
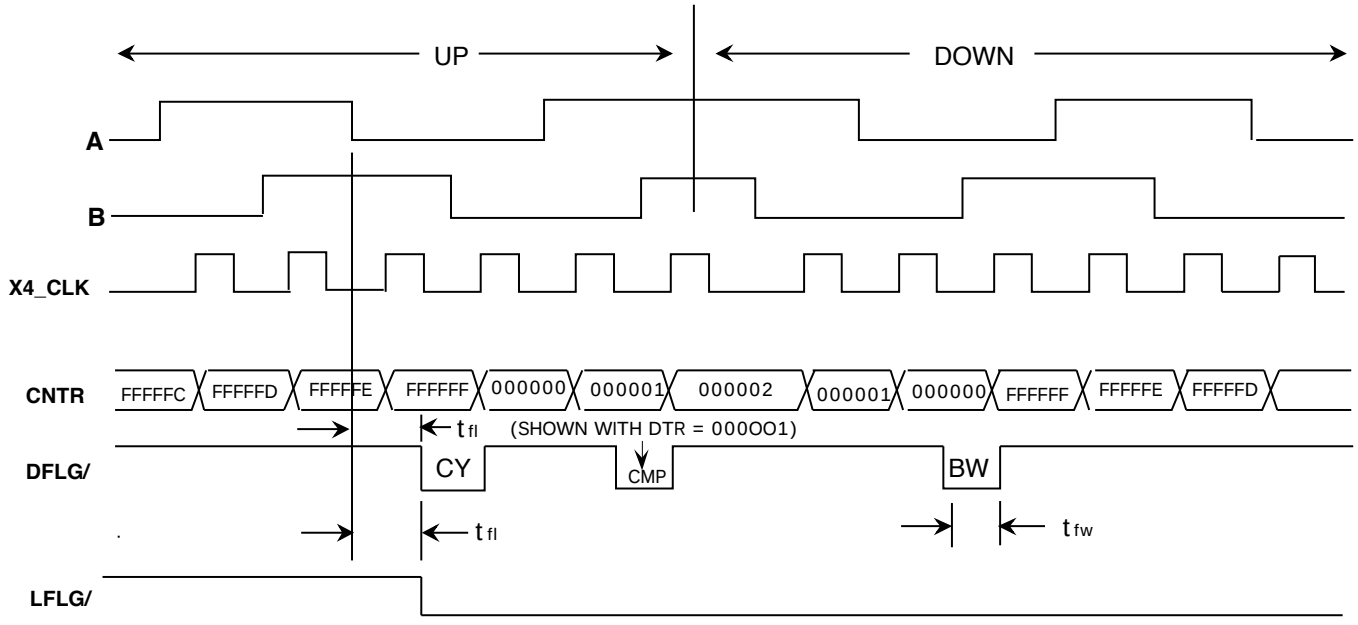


FIGURE 6. COUNT (A) AND DIRECTION (B) INPUTS IN NON-QUADRATURE MODE



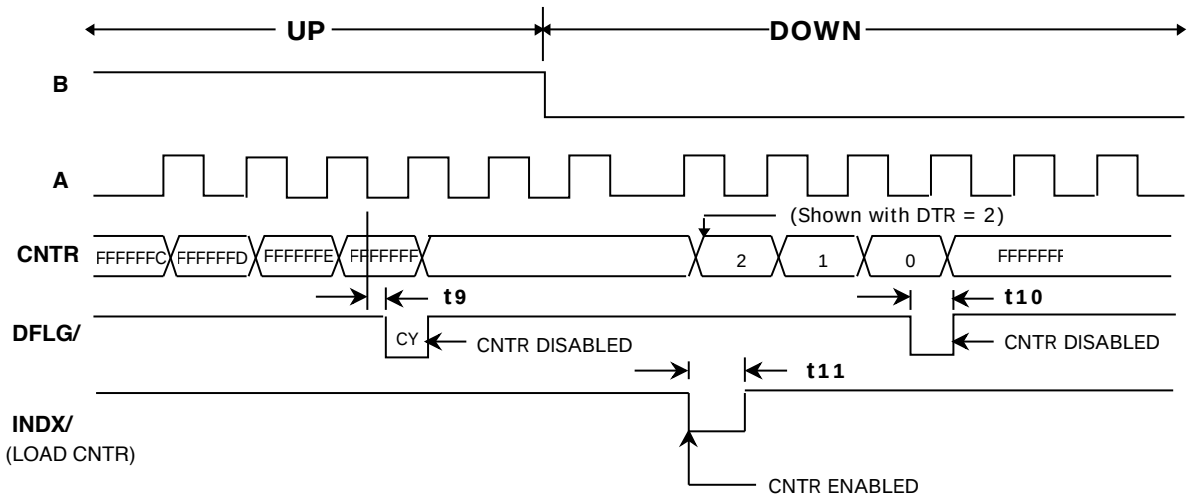
Note: x1, x2, and x4 CLKs are internal up/down clocks derived from filtered and decoded quadrature clocks.

FIGURE 7. A/B QUADRATURE CLOCKS vs INTERNAL COUNT CLOCKS



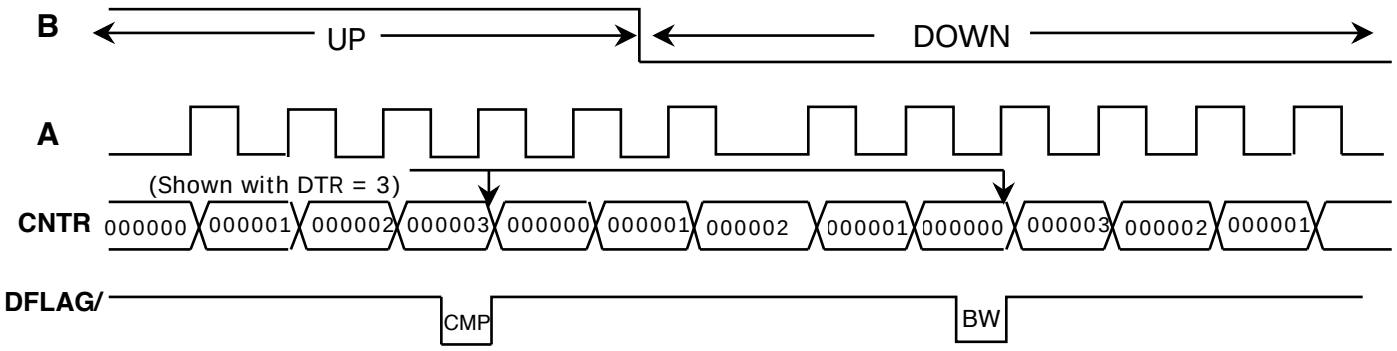
Note: CNTR values are indicated in 3-byte mode

FIGURE 8. QUADRATURE CLOCKS vs FLAG OUTPUTS



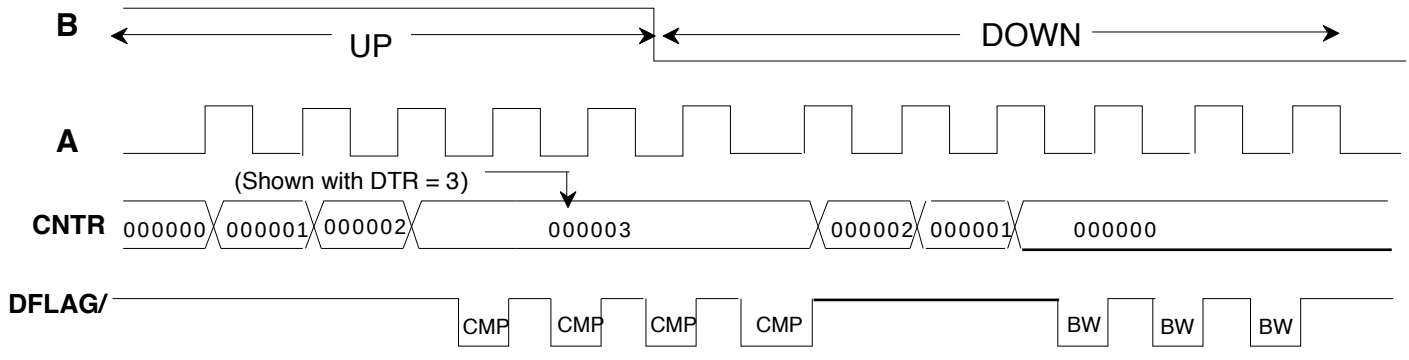
NOTE: CNTR values are indicated in 2-byte mode

FIGURE 9. SINGLE-CYCLE, NON-QUADRATURE



NOTE: CNTR values are indicated in 1-byte mode

FIGURE 10. MODULO-N, NON-QUADRATURE



NOTE: CNTR values are indicated in 1-byte mode

FIGURE 11. RANGE-LIMIT, NON-QUADRATURE

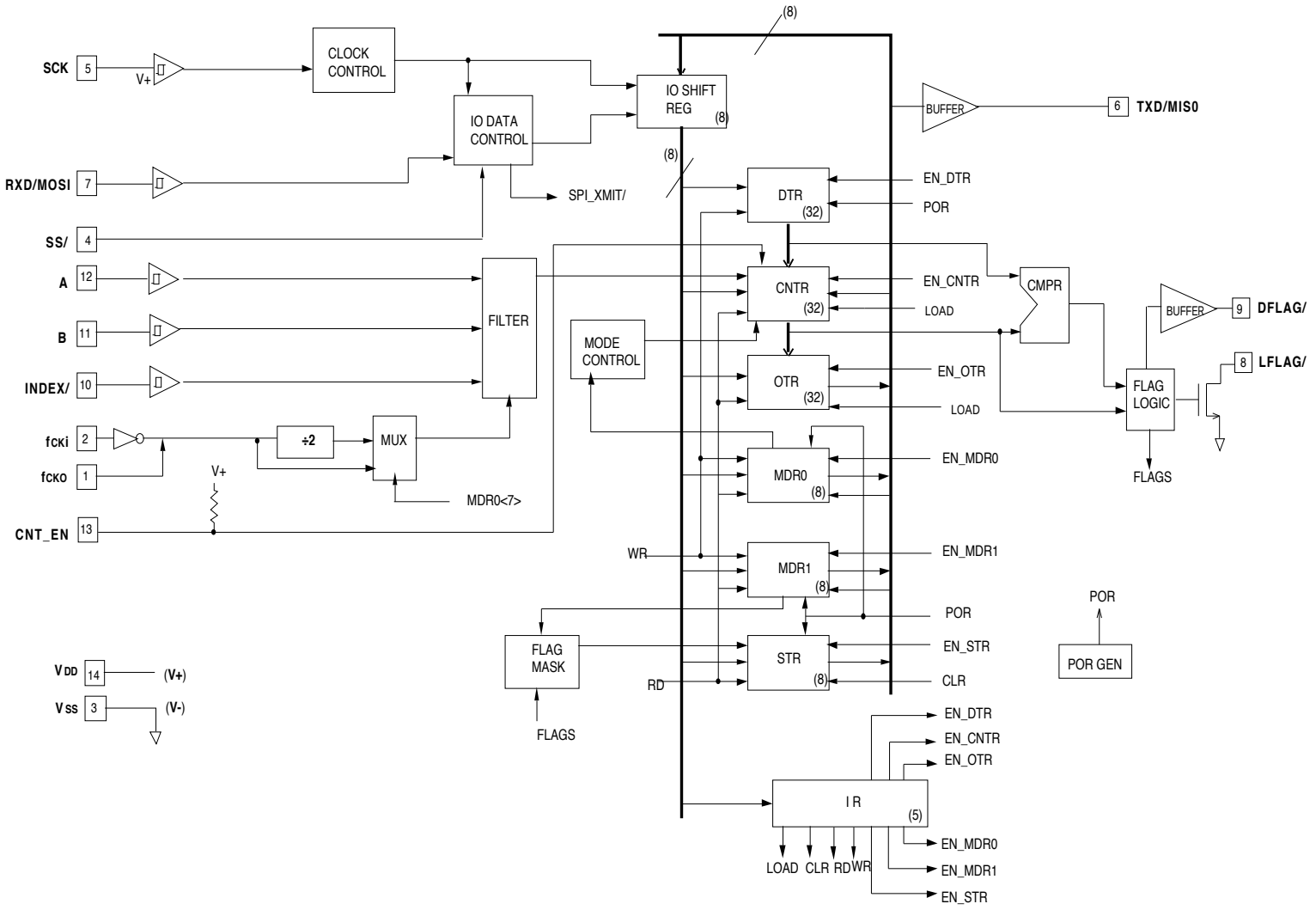
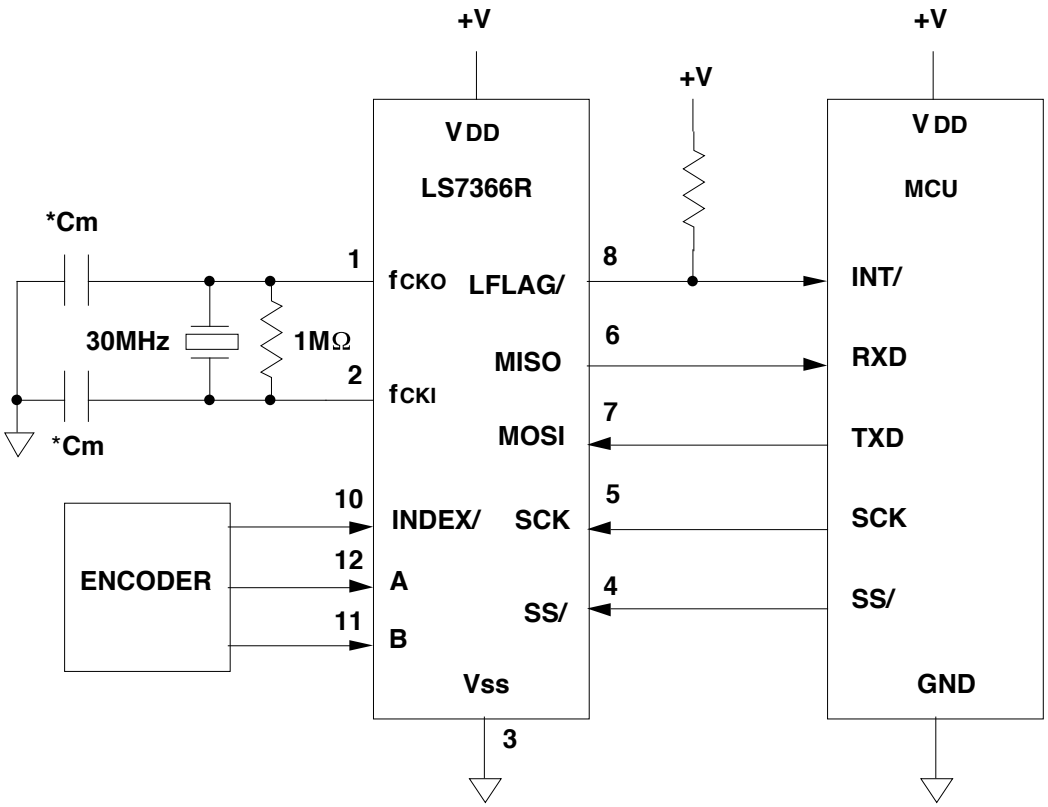


FIGURE 12. LS7366R BLOCK DIAGRAM



*NOTE: Cm is chosen according to the following equations:

$Cm = 2(C_L - C_P) - 10pF$, where
C_L = manufacturer's specified crystal load capacitance.
C_P = PCB parasitic capacitance seen by the crystal.

FIGURE 13. GENERAL I/O CONNECTIONS

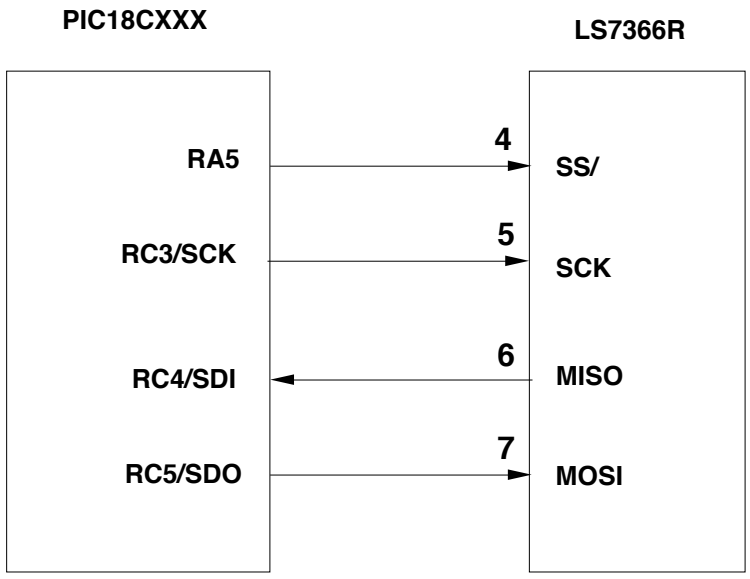


FIGURE 14. PIC18C TO LS7366R

;Sample routines for PIC18CXXX interface

```
/*****************************************************************************************************************/
 ;Initialize PIC18Cxxx portc in LS7366 compatible SPI
 ;Setup: master mode, SCK idle low, SDI/SDO datashift on high to low transition of SCK
 ;SS/ assertion/deassertion made with direct manipulation of RA5

          ;Initialize portc

          CLRF      PORTC          ;Clear portc
          CLRF      LATC           ;Clear data latches
          MOVLW     0x10           ;RC4 is input, RC3 & RC5 are outputs
          MOVWF     TRISC          ;RC3=CLK, RC4=SDI, RC5=SDO
          BCF       TRISA, 5       ;RA5=output
          BSF       PORTA, 5       ;RA5=SS/=high
          CLRF      SSPSTAT        ;SMP=0 => SDI data sampled at mid-data
          BSF       SSPSTAT, CKE   ;CKE=1 => data shifts on active to idle SCK transitions
          MOVLW     0x21           ;SPI mode initialization data
          MOVWF     SSPCON         ;Master  mode, CLK/16, CKP=0 => CLK idles low
                                   ;data shifted on active to idle CLK edge

/*****************************************************************************************************************/

          ; WR_MDR0

          BSF       PORTA, 5       ;SS/=high
          BCF       PORTA, 5       ;SS/=low
          MOVLW     0x88           ;LS7366 WR_MDR0 command
          MOVWF     SSPBUF         ;Transmit command byte
LOOP1     BTFSS     SSPSTAT, BF    ;Transmission complete with BF flag set?
          BRA       LOOP1          ;No, check again
          MOVF      SSPBUF, W      ;Dummy read to clear BF flag.
          MOVLW     0xA3           ;MDR0  data:fck/2, synchronous index. index=rcntr, x4
          MOVWF     SSPBUF         ;Transmit data
LOOP2     BTFSS     SSPSTAT, BF    ;BF set?
          BRA       LOOP2          ;No, check again
          BSF       PORTA, 5       ;SS/=high

/*****************************************************************************************************************/

          ;RD_MDR0

          BSF       PORTA, 5       ;SS/=high
          BCF       PORTA, 5       ;SS/=low
          MOVLW     0x48           ;LS7366 RD_MDR0 command
          MOVWF     SSPBUF         ;Transmit command byte
LOOP1     BTFSS     SSPSTAT, BF    ;BF flag set?
          BRA       LOOP1          ;No, check again
          MOVWF     SSPBUF         ;Send dummy byte to generate clock & receive data
LOOP2     BTFSS     SSPSTAT, BF    ;BF flag set?
          BRA       LOOP2          ;No, check again
          MOVF      SSPBUF, W      ;Recieved data in WREG.
          MOVWF     RXDATA         ;Save received data in RAM
          BSF       PORTA, 5       ;SS/=high
```

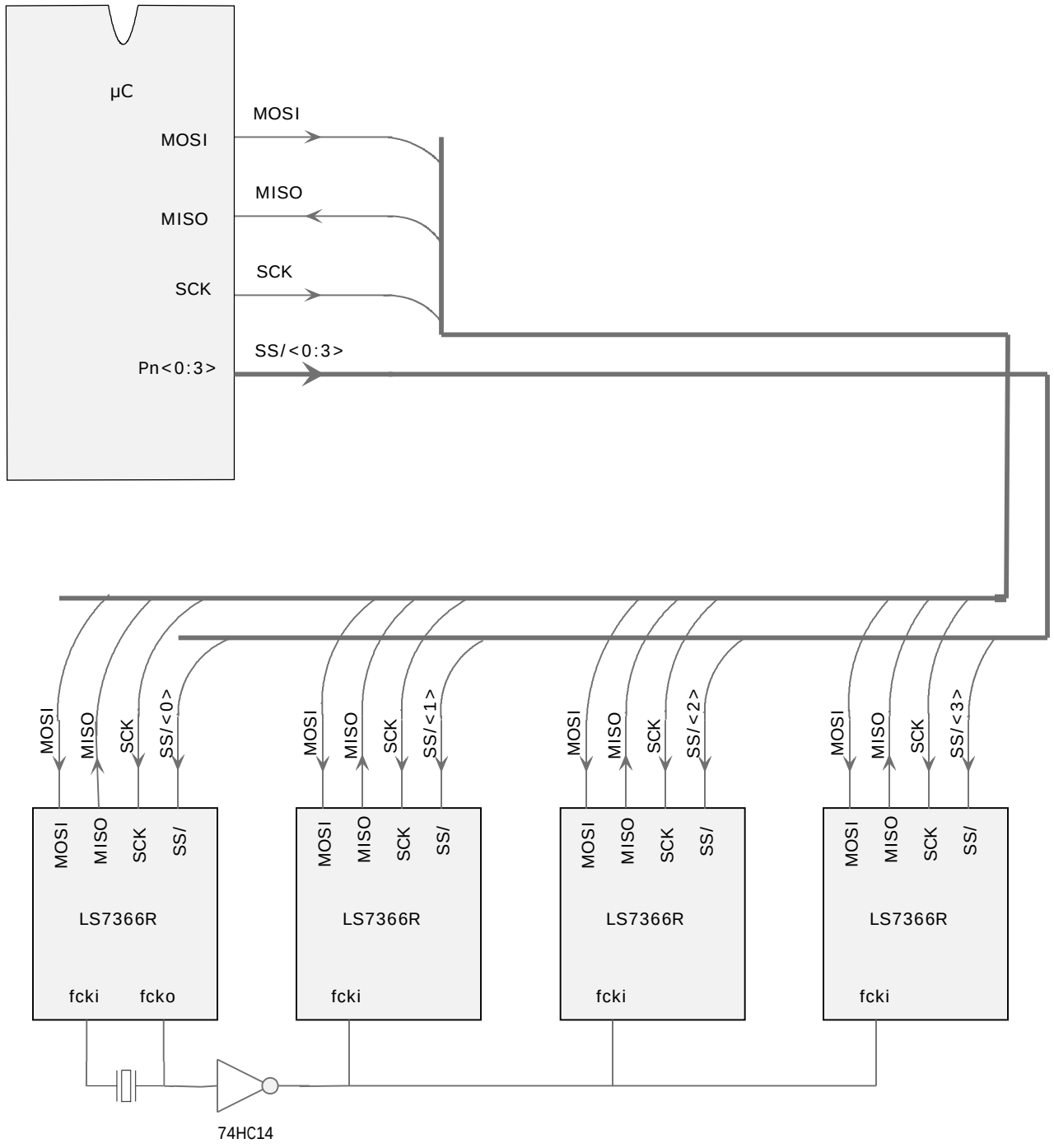


Fig 15. Multiple devices sharing common SPI bus.

```
//LS7366R to 90s8515 and ATmega32 SPI interface example

/********Add appropriate header files here**********/

/***MDR0 configuration data - the configuration byte is formed with***
  ***single segments taken from each group and ORing all together.***/

//Count modes
#define NQUAD 0x00              //non-quadrature mode
#define QUADRX1 0x01            //X1 quadrature mode
#define QUADRX2 0x02            //X2 quadrature mode
#define QUADRX4 0x03            //X4 quadrature mode

//Running modes
#define FREE_RUN 0x00
#define SINGE_CYCLE 0x04
#define RANGE_LIMIT 0x08
#define MODULO_N 0x0C

//Index modes
#define DISABLE_INDX 0x00       //index_disabled
#define INDX_LOADC 0x10         //index_load_CNTR
#define INDX_RESETC 0x20        //index_rest_CNTR
#define INDX_LOADO 0x30         //index_load_OL
#define ASYNCH_INDX 0x00        //asynchronous index
#define SYNCH_INDX 0x80         //synchronous index

//Clock filter modes
#define FILTER_1 0x00           //filter clock frequncy division factor 1
#define FILTER_2 0x80           //filter clock frequncy division factor 2

/* **MDR1 configuration data; any of these***
  ***data segments can be ORed together***/

//Flag modes
#define NO_FLAGS 0x00           //all flags disabled
#define IDX_FLAG 0x10;          //IDX flag
#define CMP_FLAG 0x20;          //CMP flag
#define BW_FLAG 0x40;           //BW flag
#define CY_FLAG 0x80;           //CY flag

//1 to 4 bytes data-width
#define BYTE_4 0x00;            //four byte mode
#define BYTE_3 0x01;            //three byte mode
#define BYTE_2 0x02;            //two byte mode
#define BYTE_1 0x03;            //one byte mode

//Enable/disable counter
#define EN_CNTR 0x00;           //counting enabled
#define DIS_CNTR 0x04;          //counting disabled

/* LS7366R op-code list */
#define CLR_MDR0 0x08
#define CLR_MDR1 0x10
#define CLR_CNTR 0x20
#define CLR_STR 0x30
#define READ_MDR0 0x48
#define READ_MDR1 0x50
```

```
#define READ_CNTR   0x60
#define READ_OTR    0x68
#define READ_STR    0x70

#define WRITE_MDR1 0x90
#define WRITE_MDR0 0x88
#define WRITE_DTR   0x98
#define LOAD_CNTR   0xE0
#define LOAD_OTR    0xE4

#define Slave_Select_Low PORTB &= ~(1 << PB4)
#define Slave_Select_High PORTB |= (1 << PB4)

 /*Configure and initialize the SPI on PortB of uC*/

void init_spi_master(void)
{
     SPCR = (0<<SPE);                  // Disable SPI until PortB configuration

 /* Port B (DDRB) PB7/SCK, PB5/MOSI, PB4/!SS outputs */

   DDRB = (1<<DDB7)|(1<<DDB5)|(1<<DDB4);                  // Define Outputs
   Slave_Select_High;                                     // Disable Slave Select

 /*** SPCR configuration***
  *** CPOL = 0,  CPHA = 0, Mode0***
  *** DORD = 0, MSB first***
  *** MSTR = 1, Master***
  ***SPE = 1, SPI enabled***
  ***SCK frequency = fosc/4***/

SPCR = (1<<SPE)|(1<<MSTR);
    }

void load_rst_reg(unsigned char op_code)          //dataless write command
{
   unsigned char spi_data;
   Slave_Select_High;                // Keep SS/ High for LS7366 deselect
   Slave_Select_Low;                 // Switch SS/ low for new command
   SPDR = op_code;                   // Send command to LS7366
   while (!(SPSR & (1<<SPIF)))     // Wait for end of the transmission
   {
   };
   spi_data = SPDR;                  // Reset SPIF
   Slave_Select_High;                // Switch SS/ high for end of command
}

void singleByteWrite(unsigned char op_code, unsigned char data)  //single byte write command
{
   unsigned char spi_data;
   Slave_Select_High;                  // Keep SS/ High for LS7366 deselect
   Slave_Select_Low;                   // Switch SS/  low for new command
   SPDR = op_code;                     // Send command  to LS7366
   while (!(SPSR & (1<<SPIF)))       // Wait for end of the transmission
   {
   };
   spi_data = SPDR;                    // Reset SPIF
   SPDR = data;                        // Send data to be written to LS7366 register
   while (!(SPSR & (1<<SPIF)))       // Wait for end of the transmission
```

```
 {
   };
   spi_data = SPDR;                  // Reset SPIF

 /*additional bytes can be sent here for multibyte write, e.g., write_DTR*/



  Slave_Select_High;               // Switch SS/ high for end of command
}

 void singleByteRead(unsigned char op_code)           //single byte read command
{
   unsigned char spi_data;
   Slave_Select_High;                // deselect the the LS7366
   Slave_Select_Low;                 // Switch SS/ low for new command
   SPDR = op_code;                   // send op_code for read  to LS7366
   while (!(SPSR & (1<<SPIF)))     // Wait for end of  transmission
   {
   };
   spi_data = SPDR;                  // Reset SPIF
   SPDR = 0xFF;                      // Start dummy transmission to read data from LS7366
   while (!(SPSR & (1<<SPIF)))     // Wait for end of the transmission
   {
   };
   spi_data = SPDR;                  // Reset SPIF

/*additional bytes can be received here for multibyte read, e.g., read_OTR*/

   Slave_Select_High;                // Switch SS/ high for end of command
   return spi_data;
}

//following example instantiates all macros defined above
 int main(void)
{
   init_spi_master;
   load_rst_reg(CLR_CNTR);
   singleByteWrite(WRITE_MDR0, QUADRX4|FREE_RUN|INDX_LOADC|SYNCH_INDX|FILTER_2);
   singleByteWrite(WRITE_MDR1, IDX_FLAG|CMP_FLAG|BYTE_2|EN_CNTR);
   singleByteRead(READ_MDR0);
   singleByteRead(READ_MDR1);
   return 0;
 }
```