LSI/CSI = LS7466

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

24-BIT 2-AXES PROGRAMMABLE QUADRATURE COUNTER WITH SERIAL (SPI) INTERFACE

DECEMBER 2023

GENERAL FEATURES:

- Operating voltage: 3V to 5.5V
- Up to 40MHz count frequency
- Two mode-registers for functional programmability
- 24-bit multimode counter (CNTR)
- 24-bit Input Data Register (IDR) for CNTR upload
- 24-bit Output Data Register (ODR) for CNTR download
- 24-bit digital comparator for IDR to CNTR data compare
- Dynamic (DSTR) and latched (SSTR) status registers
- Quadrature (A, B) clock and Index (Z) inputs with digital filters
- Index driven load and reset operations of CNTR, ODR, DSTR and SSTR
- 8-bit, 16-bit or 24-bit programmable configurations
- Programmable count modes:
  Quadrature (X1, X2, X4), non-quadrature, mod-N, non-recycle, range-limit and free-run
- Common SPI I/O's for addressing both axes.
- 16-pin SOIC and TSSOP packages

GENERAL DESCRIPTION:

LS7466 is a monolithic CMOS 24-bit programmable counter. It consists of two identical functional modules to interface with X and Y axes encoders simultaneously. Each block consists of the following registers: MCR0, MCR1, IDR, ODR, CNTR, DSTR and SSTR. MCR0 and MCR1 controls the functional modes. Data written into IDR is used to set limits to the counter (CNTR) range in several ways. CNTR can be uploaded into ODR for instantaneous or future read. DSTR dynamically follows the counter status in terms of carry, borrow etc. The instantaneous state of the DSTR can be latched into SSTR for future inspection.

Each axis can independently be configured to operate in 8-bit, 16-bit or 24-bit modular structures forcing IDR, CNTR and ODR into that configuration. Programmable modes include: X1/X2/X4 bi-directional quadrature or non-quadrature. Either of these modes can further be combined with Free-Run, Non-Recycle, Mod-N and Range-Limit modes.

A common SPI module establishes the communication between a host uC and the X and Y axis functional modules for bidirectional data transfers. The SPI module consists of the standard 4-wire IO's namely, SS/, SCK, MOSI and MISO. Only mode0 bus protocol is supported.

PIN ASSIGNMENT
TOP VIEW

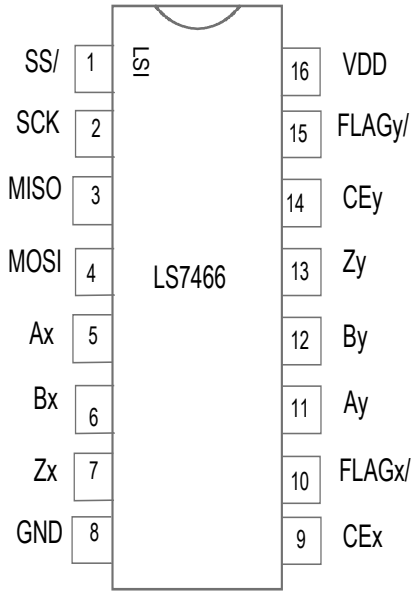


Fig 1

LS7466 functions in the slave mode only. Any communication between LS7466 and a host controller is initiated by the host controller by bringing the SS/ input low followed by an instruction byte serially transmitted on the MOSI bus line. The instruction byte consists of an OP_CODE field for functional instruction, an address field for axis selection and another address field for register selection. The instruction byte gets loaded into the Instruction Register (IR) and executed at the completion of a communication cycle. A communication cycle consists of 1 to 4 consecutive bytes initiated by a SS/ low transition and completed when the SS/ switches high.

I/O PINS:
SS/ (pin 1). A high to low transition at this input selects the device for serial bidirectional data transfer. A low to high transition terminates the serial data transfer and brings the MISO output into high impedance state. This allows for multiple slave units to be on the bus.

SCK (pin 2). Clocks applied to this input is used to shift serial data in and out of LS7466 on the MOSI and MISO pins respectively. Since LS7466 operates in the slave mode only, SCK clocks must be provided by the host processor.

MISO (pin3). Serial output data from LS7466 is shifted out on this output. MISO output goes into high impedance state when SS/ input is at logic high allowing for multiple devices to share the bus.

MOSI (pin 4). Output data from host processor is serially shifted into LS7466 at this input.

Ax (pin 5), **Bx** (pin6). X-axis count inputs. In quadrature mode A and B clocks from incremental encoders are directly applied to the A and B inputs. These clocks are ideally 90° out of phase signals. Ax and Bx inputs are digitally filtered for noise suppression and decoded for count clocks and up/down direction. With A leading B count UP is selected, with A lagging B count DOWN is selected. In non-quadrature mode Ax serves as the count input (counter advances at the falling edge) and Bx as the up/down direction control input with B =1 selecting UP and Bx = 0 selecting DOWN count modes. In non-quadrature mode Ax and Bx inputs are not filtered but reshaped with input Schmitt trigger buffers.

Ax, Bx inputs' functional configuration is made by MCR0 register.

Zx (pin 7). X-axis index input Zx is a programmable input to function as one of the following:

- LCNT: (CNTR <= IDR)
- RCNT: (CNTR <= 0)
- RDST: (DSTR <= 0)
- LSST: (SSTR <= DSTR)
- LODR: (ODR <= CNTR)
- LODR_RCNT: (LODR at leading edge and RCNT at trailing edge transitions)

The Zx input is programmable to be either high active or low active. In either case the input is level sensitive. An exception to this rule is made when Zx is configured as LODR_RCNT input. In this mode LODR and RCNT operations are performed with the leading and trailing edges of the Zx input irrespective of the selected active levels.

In the quadrature mode, Zx input is digitally filtered with the same internal clock used for filtering Ax and Bx inputs. In non-quadrature mode the Zx input is not filtered.

Zx input's functional configuration is made by MCR0 register.

GND (pin 8). Supply voltage negative terminal

CEx (pin 9). X-axis count enable input. A logic high at the CEx input enables the Ax and Bx inputs for counting. A logic low disables the Ax and Bx inputs. The CEx input has an internal pull-up.

FLAGx/ (pin 10). The FLAGx/ is programmable output to produce an output signal for one or all of the following events:
- CY (CARRY: indicates CNTR overflow)
- BW (BORROW: indicates CNTR underflow)
- INDX (indicates Z input at active level)
- EQL (indicates CNTR = IDR).

(Note. In mod-N, non-recycle and range-limit modes EQL is generated in up count direction only. In free-run mode EQL is generated in both up and down directions.)

FLAGx/ output can be configured to function in either dynamic or latched mode. In the dynamic mode a low going pulse is produced at the output with the occurrence of CY or BW or EQL. In contrast the occurrence of INDX produces a steady low level coincident with the active state of the Z input. INDX output masks out a coincident CY, BW or EQL.

In the latched mode any occurrence of CY, BW, EQL or INDX will set the FLAGx/ output low. The output will remain low until cleared by a RST_DSTR command.
(Note. RST_DSTR command resets both the flag latch and DSTR register together)

In latched mode the FLAG/ is an open drain output requiring an external pull-up resistor returned to the positive supply rail. This allows for the FLAG/ outputs from multiple devices to be connected together to form a single interrupt for the host processor.

In the dynamic mode the FLAG/ is a push-pull output.

FLAGx/ output's functional configuration is made by MCR1 register.

Ay (pin 11), **By** (pin 12), **Zy** (pin 13), **CEy** (pin 14) and **FLAGy/** (pin 15). All these I/O pins belong to the Y-axis and have identical functions as those in the x-axis.

REGISTERS:

IR: The IR is an 8-bit register which fetches instruction bytes from the received data stream at the MOSI input and executes them to configure device functional modes and other dynamic operations listed in table 1. The first byte received after the SS/ input switches low is always an instruction byte and gets lodged into the IR for execution.

IR

B[7]	B[6]	B[5]	B[4]	B[3]	B[2]	B[1]	B[0]

B[7:6] = 00: RST: Reset register
= 01: RD: Read register
= 10: WR: Write into register
= 11: LOAD: Upload to register
B[5:3] = 000: select none
= 001: select MCR0
= 010: select MCR1
= 011: select IDR
= 100: select CNTR
= 101: select ODR
= 110: select SSTR
= 111: select DSTR
B[2:1] = 00: select axis-x
= 01: select axis-y
= 1x: select both x and y axes (ignored for RD)
B[0] = 0: NOP
= 1: Simultaneously transfer DSTR to SSTR whenever a RD_CNTR is executed

TABLE 1

OP CODE	REGISTER	OPERATION
RST	MCR0	Reset MCR0 to 0
	MCR1	Reset MCR1 to 0
	IDR	None
	CNTR	Reset CNTR to 0
	ODR	None
	DSTR	Reset DSTR to 0
	SSTR	Reset SSTR to 0
RD	MCR0	Output MCR0 serially on MISO
	MCR1	Output MCR1 serially on MISO
	IDR	None
	CNTR	Transfer CNTR to ODR, then output ODR serially on MISO
	ODR	Output ODR serially on MISO
	DSTR	None
	SSTR	Output SSTR serially on MISO
WR	MCR0	Write serial data received at MOSI input into MCR0
	MCR1	Write serial data received at MOSI input into MCR1
	IDR	Write serial data received at MOSI input into IDR
	CNTR	None
	ODR	None
	DSTR	None
	SSTR	None
LOAD	MCR0	None
	MCR1	None
	IDR	None
	CNTR	Transfer IDR to CNTR
	ODR	Transfer CNTR to ODR
	DSTR	None
	SSTR	Transfer DSTR to SSTR

MCR0: The MCR0 is a **read/write** control register for configuring some of the functional modes of the device. At power up MCR0 is cleared to 0 and must be loaded with the proper configuration data for intended functionalities.

MCR0

B[7]	B[6]	B[5]	B[4]	B[3]	B[2]	B[1]	B[0]

B[7] = 0: Z input active level is logic 0.
= 1: Z input active level is logic 1.
B[6:4] = 000: disable Z input
= 001: configure Z as LCNT input. (CNTR <= IDR).
= 010: configure Z as RCNT input. (reset CNTR).
= 011: configure Z as RDST input. (reset DSTR).
= 100: configure Z as LSST input. (SSTR <= DSTR).
= 101: configure Z as LODR input. (ODR <= CNTR). *
= 110: configure Z as both RCNT and LODR input. In this mode Z input is edge sensitive; leading edge transition of Z performs a LODR operation and trailing edge transition performs a RCNT operation.
= 111: disable Z input
B[3:2] = 00: Free-Running count mode.
= 01: Non-Recycle count mode. **
= 10: Range-Limit count mode. **
= 11: Modulo-N count mode. **
B[1:0] = 00: Non-Quadrature count mode. (A = count clock, B = up/down control)
= 01: X1 Quadrature count mode. (1 count per quad)
= 10: X2 Quadrature count mode. (2 count per quad)
= 11: X4 Quadrature count mode. (4 count per quad)

* (A LODR (ODR <= CNTR) operation is always accompanied by a LSST (SSTR <= DSTR) operation)

** Definition of count modes:

● Non-Recycle mode: CNTR freezes at CNTR = (IDR + 1) with generation of EQL at the FLAG/ output in the **up**-count direction and at CNTR = -1 with generation of BW in **down**-count direction. Counting is re-enabled with a LOAD_CNTR or with a RST_CNTR.

● Range-Limit mode: Up and down count ranges are limited between IDR (high) and 0 (low) respectively. CNTR freezes at these limits in up and down directions respectively. Counting resumes when the direction is reversed.

● Modulo-N: input count clock frequency, fc is divided by (N+1) in both up and down directions where, N = IDR. The resultant frequency, fo = fc/(N+1) is made available at the FLAG/ output when MCR1 register's bits B[5] and B[6] are set to 1.

MCR1: MCR1 is a **read/write** control register which sets the bit lengths for registers IDR, CNTR and ODR to 8-bit (1-byte) or 16-bit (2-byte) or 24-bit (3-byte) modular structures. MCR1 is also used to enable or disable the following signals at the FLAG/ output: CY, BW, EQL and INDX.

MCR1

B[7]	B[6]	B[5]	B[4]	B[3]	B[2]	B[1]	B[0]

B[7] = 0: Disable CY at FLAG/ output
= 1: Enable CY at FLAG/ output

MCR1 (cntd.)
B[6] = 0: Disable BW (borrow) at FLAG/ output
= 1: Enable BW (borrow) at FLAG/ output
B[5] = 0: Disable EQL (CNTR = IDR) at FLAG/ output
= 1: Enable EQL (CNTR = IDR) at FLAG/ output
B[4] = 0: Disable INDX (Z input active) at FLAG/ output
= 1: Enable INDX (Z input active) at FLAG/ output
B[3] = 0: FLAG/ output is dynamic
= 1: FLAG/ output is latched
B[2] = 0: Enable counting
= 1: Disable counting
B[1:0] = 00: 3-byte mode enabled
= 01: 2-byte mode enabled
= 1x: 1-byte mode enabled (x = don't care)

Upon power up MCR1 is reset to 0 and must be loaded with proper configuration data for intended functionalities.

IDR: The Input Data Register, IDR is a **write-only** register with programmable bit length to function as 1-byte, 2-byte or 3-byte register. IDR can be uploaded to CNTR to preset the CNTR. IDR also functions as the repository of data in Non-Recycle, Range-Limit and Modulo-N modes as the upper limit of the count in the CNTR.

IDR

B[23:16]	B[15:8]	B[7:0]
Byte2	Byte1	Byte0

CNTR: The CNTR is an up/down synchronous counter module with programmable bit lengths in 1-byte, 2-byte or 3-byte structures. CNTR receives the count pulses from the A input in non-quadrature mode and decoded up/down count pulses derived from the A and B clocks in the quadrature count mode. CNTR can be downloaded from IDR and uploaded to ODR with LOAD commands or index (Z) signal. CNTR is a **read-only** register. Upon powerup the CNTR is cleared to 0.

CNTR

B[23:16]	B[15:8]	B[7:0]
Byte2	Byte1	Byte0

ODR: The Output Data Register, ODR is a **read-only** register with programmable bit lengths to function in 1-byte, 2-byte or 3-byte structures. Instantaneous CNTR value can be uploaded to ODR with a LOAD command or an index (Z) signal. Upon powerup the ODR is cleared to 0.

ODR

B[23:16]	B[15:8]	B[7:0]
Byte2	Byte1	Byte0

DSTR: DSTR is an 8-bit status register which records dynamic events and holds them until cleared by a RST (reset) command. DSTR cannot be read directly. For reading DSTR It must be transferred into SSTR which is accessible for read. Upon powerup DSTR is cleared to 0.

DSTR

SGN	U/D	CY	BW	EQL	INDX	CEN	PLS
B[7]	----------						B[0]

SGN = 0: CNTR data is a positive number
= 1: CNTR data is a negative number

DSTR (cntd.)
U/D = 0: counting down
= 1: counting up
CY = 1: a carry has occurred
BW = 1: a borrow has occurred
EQL = 1: a CNTR = IDR event has occurred
INDX = 1: Z input is or was in active level
DSTR (cntd.)
CEN = 0: counting is currently disabled
= 1: counting is currently enabled
PLS = 1: a power reset occurred

SGN, U/D and CEN bits are not latched and display instantaneous states. In contrast CY, BW, EQL, INDX and PLS bits are latched and must be cleared with a RST_DSSTR command to register the changing status.

SSTR: SSTR is an 8-bit **read-only** status register with a bit-for-bit mapping of the DSTR. The DSTR instantaneous status can be saved in the SSTR for polling without interfering with dynamic operation of the DSTR. The transfer of DSTR to SSTR can be made under command control or with the index signal originating from the Z input. Under command control an automatic transfer of DSTR to SSTR can be made whenever a RD_CNTR command is executed (see IR description). This allows for an accurate correlation between the status bits and the CNTR data.

SSTR

SGN	U/D	CY	BW	EQL	INDX	CEN	PLS
B[7]	----------						B[0]

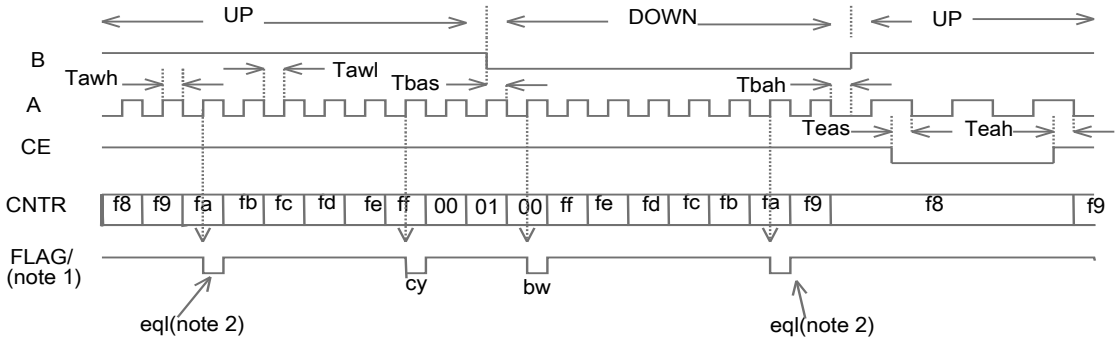
All SSTR bits have identical meaning as those for DSTR. Unlike DSTR however, all SSTR bits are latched and gets updated only when a DSTR to SSTR transfer takes place.
Upon powerup SSTR is cleared to 0.

Absolute maximum ratings:			
Parameter	Symbol	Value	Unit
Supply Voltage	VDD	+7.0	Volts
Voltage at any input/output pin	Vin	Vss-0.3 to VDD + 0.3	Volts
Operating temperature	Ta	-40 to +85	°C
Storage temperature	Ts	-65 to +150	°C

DC electrical characteristics @ Ta = -25° to +85°, VDD = 5.0V (unless specified otherwise)						
Parameter	Symbol	Min	Typ	Max	Unit	Conditions
Supply voltage	VDD	3.0	5.0	5.5	Volts	
Supply current	IDD	-	700	800	uA	VDD = 3.0V
		-	1.4	1.8	mA	VDD = 5.0V
Logic low: A, B, Z, SS/, SCK, MOSI	Vil	-	-	0.3*VDD	Volts	
Logic high: A, B, Z, SS/, SCK, MOSI	Vih	0.7VDD	-	-	Volts	
Logic low: CE input	Vcil	-	-	0.4*VDD	Volts	
Logic high: CE input	Vcih	0.6*VDD	-	-	Volts	
Input current: CE input low	Icl	-10	-15	-30	uA	Vi = 0.3*VDD, VDD = 3V to 5V
Input current: CE input high	Ich	1	-	4	uA	Vi = 0.7*VDD, VDD = 3V to 5V
Output current: MISO output sink	Imosnk	1.5	2.4	-	mA	Vo = 0.5V, VDD = 3V
		3.8	4.8	-	mA	Vo = 0.5V, VDD = 5V
Output current: MISO output source	Imosrc	-1.5	-2.4	-	mA	Vo = 2.5V, VDD = 3V
		-3.8	-4.8	-	mA	Vo = 4.5V, VDD = 5V
Output current: FLAG/ output sink	Ifosnk	1.3	2.0	-	mA	Vo = 0.5V, VDD = 3V
		3.2	4.0	-	mA	Vo = 0.5V, VDD = 5V
Output current: FLAG/ output source	Ifosrc	-1.0	-1.8	-	mA	Vo = 2.5V, VDD = 3V
		-2.8	-3.6	-	mA	Vo = 4.5V, VDD = 5V

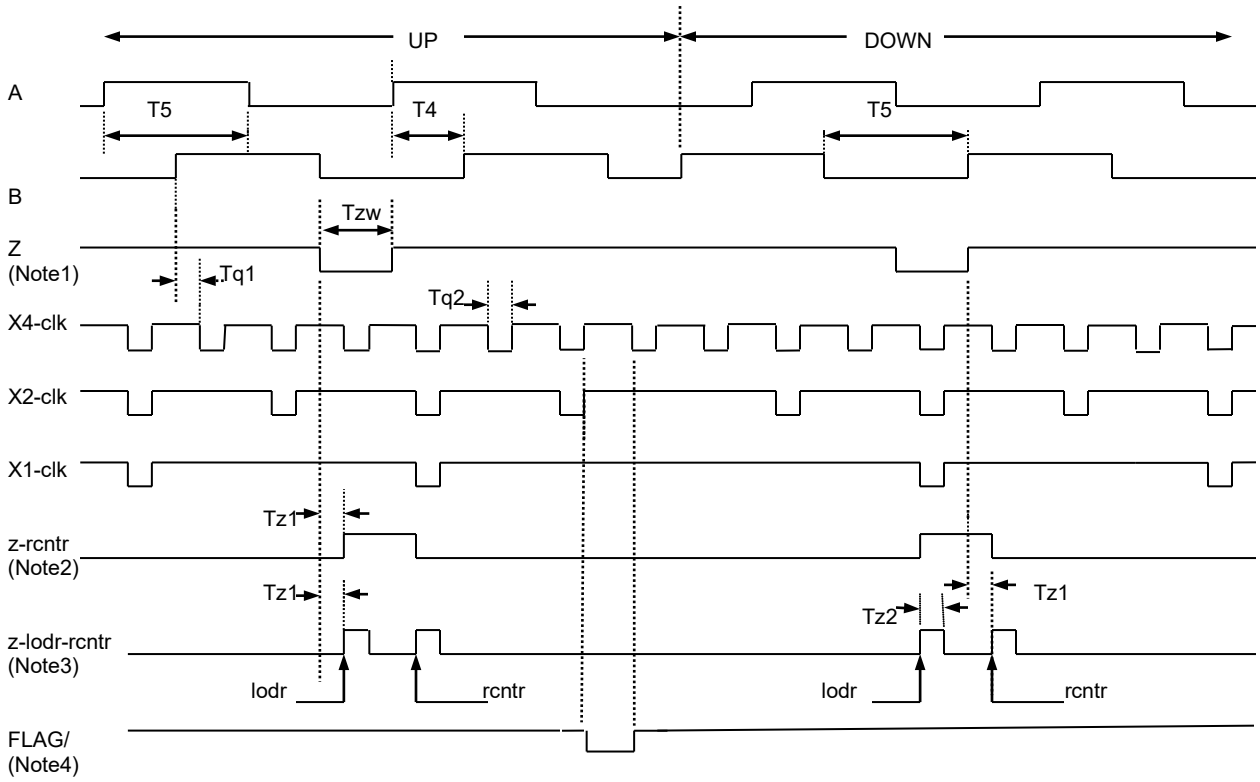
Transient Characteristics: @ Ta = -25° to +85°, VDD = 5.0V (unless specified otherwise)						
Parameter	Symbol	Min	Typ	Max	Unit	Conditions
SCK input pulse width: high	Tcwh	100	-	-	ns	
SCK input pulse width: low	Tcwl	100	-	-	ns	
SS/ input setup time	Tss	100	-	-	ns	
SS/ input hold time	Tsh	100	-	-	ns	
SS/ input inter-command high duration	Tswh	100	-	-	ns	
Non-quadrature mode (see figs 2 and 3)						
Count input A pulse width: high	Tawh	15	-	-	ns	
Count input A pulse width: low	Tawl	10	-	-	ns	
Direction input B setup time	Tbas	15	-	-	ns	
Direction input B hold time	Tbah	10	-	-	ns	
CE input setup time	Teas	15	-	-	ns	
CE input hold time	Teah	15	-	-	ns	
Count input A frequency (free run mode)	fa	-	-	40	MHz	
Input A to FLAG/ delay	Tdaf	20	-	-	ns	
FLAG/ output pulse width (non-latch mode)	Tfw	15	-	-	ns	Tfw = Tawh
Z input pulse width	Tzw	30	-	-	ns	Z not in LODR_RCNTR mode
Z input pulse width	Tzw	130	-	-	ns	Z in LODR_RCNTR mode
Quadrature mode (see figs 3)						
A to B to A separation	T4	130	-	-	ns	
A, B pulse width	T5	260	-	-	ns	
A, B frequency	fqab	-	-	2.0	MHz	
A or B to x1/x2/x4 count clock delay	Tq1	-	130	190	ns	
x1/x2/x4 count clock pulse width	Tq2	-	35	-	ns	
FLAG/ output pulse width (non-latch mode)	Tqfg	-	T4 - 35	-	ns	X4 mode
		-	2*T4 - 35	-	ns	X2 mode
		-	4*T4 - 35	-	ns	X1 mode
Z input pulse width	Tzw	130	-	-	ns	
Z high or low transition to z-lodr and r-cntr pulse delay	250		-	380	ns	Z input in LODR_RCNTR configuration

Transient Characteristics: @ Ta = -25° to +85°, VDD = 3.0V (unless specified otherwise)						
Parameter	Symbol	Min	Typ	Max	Unit	
SCK input pulse width: high	Tcwh	120	-	-	ns	
SCK input pulse width: low	Tcwl	120	-	-	ns	
SS/ input setup time	Tss	120	-	-	ns	
SS/ input hold time	Tsh	120	-	-	ns	
SS/ input inter-command high duration	Tswh	120	-	-	ns	
Non-quadrature mode (see figs 2 and 3)						
Count input A pulse width: high	Tawh	40	-	-	ns	
Count input A pulse width: low	Tawl	40	-	-	ns	
Direction input B setup time	Tbas	24	-	-	ns	
Direction input B hold time	Tbah	24	-	-	ns	
CE input setup time	Teas	24	-	-	ns	
CE input hold time	Teah	24	-	-	ns	
Count input A frequency (free run mode)	fa	-	-	12.5	MHz	
Input A to FLAG/ delay	Tdaf	40	-	-	ns	
FLAG/ output pulse width (non-latch mode)	Tfw	40	-	-	ns	Tfw = Tawh
Quadrature mode (see figs 3)						
A to B to A separation	T4	200	-	-	ns	
A, B pulse width	T5	400	-	-	ns	
A, B frequency	fqab	-	-	1.3	MHz	
A or B to x1/x2/x4 count clock delay	Tq1	-	200	300	ns	
x1/x2/x4 count clock pulse width	Tq2	-	50	-	ns	
FLAG/ output pulse width (non-latch mode)	Tqfg	-	T4 - 50	-	ns	X4 mode
		-	2*T4 - 50	-	ns	X2 mode
		-	4*T4 - 50	-	ns	X1 mode
Z input pulse width	Tzw	200	-	-	ns	
Z high or low transition to z-lodr and r-cntr pulse delay	400		-	600	ns	Z input in LODR_RCNTR configuration



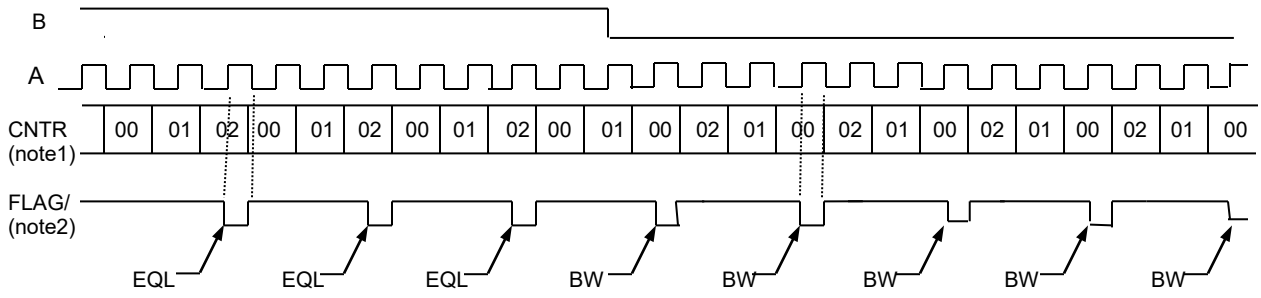
Note1. FLAG/ configured to output EQL, CY and BW in non-latch mode
Note2. In Range-Limit, Non-Recycle and Modulo-n modes EQL is generated in UP count direction only
Note3. Shown in 1-byte configuration with IDR arbitrarily chosen to be “fa” and CNTR starting at “f8”.

Fig 2. A, B, CE and FLAG/ in non-quadrature mode



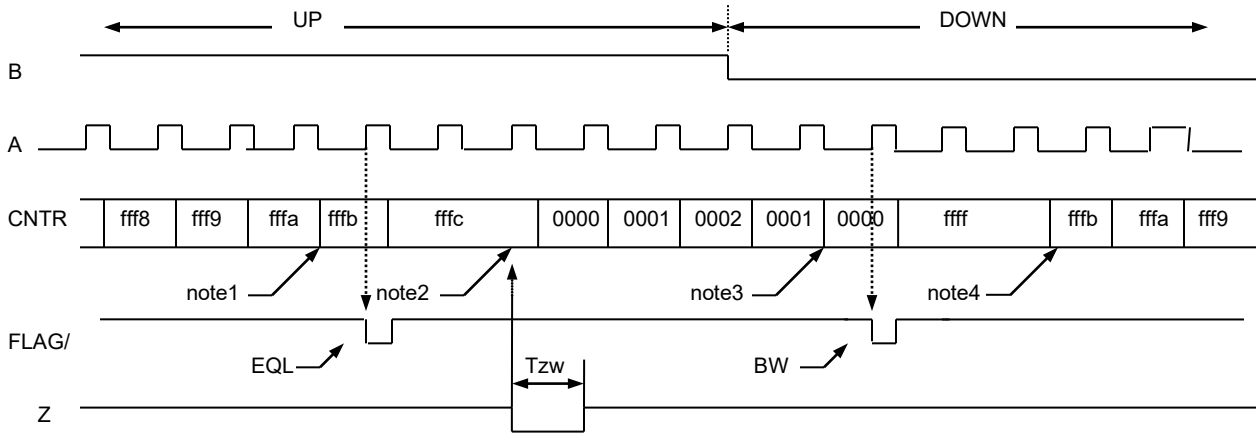
Note1. Z input is shown in “low active” configuration. For “high active” configuration the Z input waveform is inverted.
Note2. Internal RCNTR signal originating from input Z
Note3. Internal LODR and RCNT signals from Z input when the Z input is configured for LODR-RCNTR function.
Note4. FLAG/ output for CY or BW or EQL in non-latched mode (shown for x4 mode)
Note5. x1-clk, x2-clk and x4-clk are internal count clocks in x1 and x2 and x4 modes.
Note6. CNTR advances on the falling edge of x1 or x2 or x4 clk.

Fig 3. A, B and Z inputs in quadrature count mode



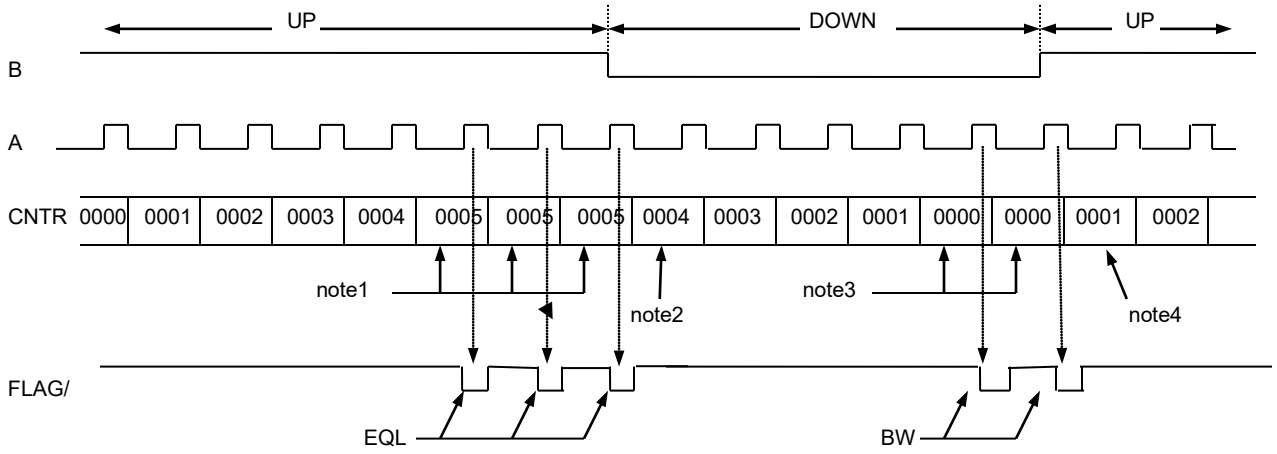
Note1. IDR is assumed to be set to 02 (n = 02) causing a divide-by-3 modulo-n count scale
Note2. FLAG/ is configured to output EQL and BW in non-latch mode

Fig 4. Modulo-n, non-quadrature mode



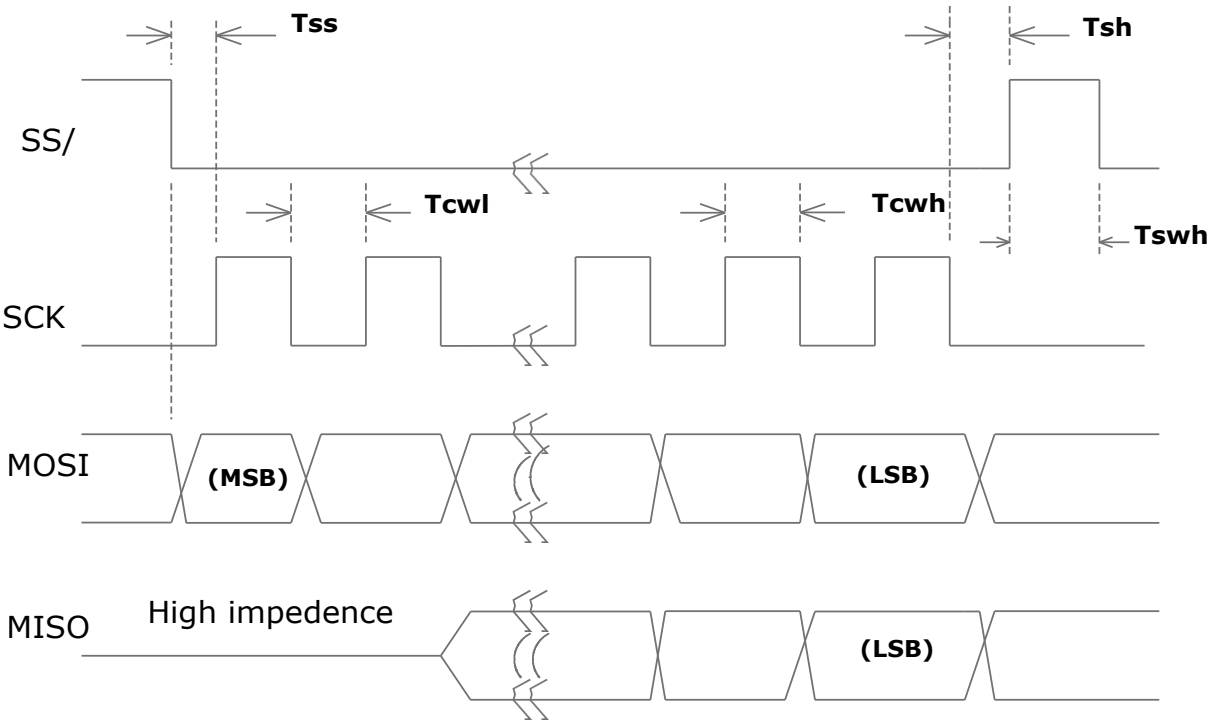
Note1. CNTR disabled at CNTR = IDR (set to fffb in 2-byte mode) with generation of a single EQL on the FLAG/ output and freezes at fffc.
Note2. CNTR re-enabled with a RCNTR operation via the Z input.
Note3. CNTR disabled at CNTR = 0000 with generation of a single BW on the FLAG/ output and freezes at ffff.
Note4. CNTR re-enabled with a load CNTR operation.
Note5. FLAG/ configured to output EQL and BW in non-latch mode

Fig 5. Non-Recycle, non-quadrature mode



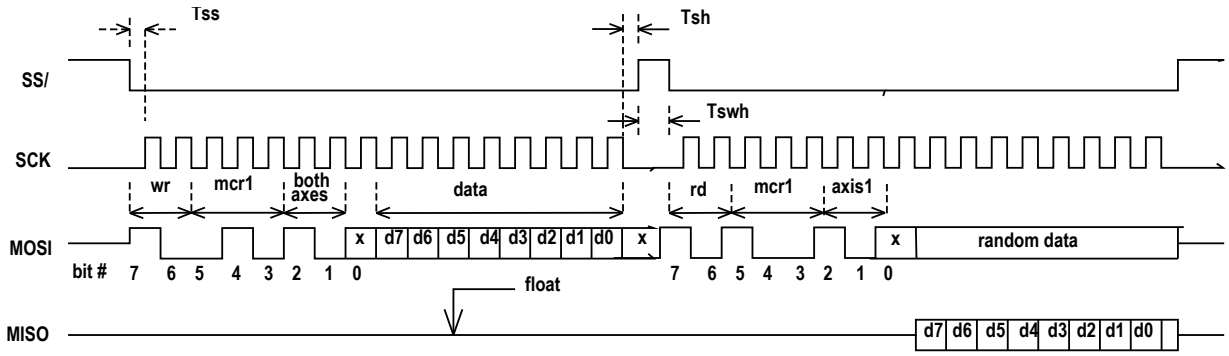
Note1. In UP direction CNTR freezes at CNTR = IDR (set to 0005 in example) with repeated generation of EQL.
Note2. CNTR re-enabled with direction reversal.
Note3. In DOWN direction CNTR freezes at CNTR = 0000 with repeated generation of BW.
Note4. CNTR re-enabled with direction reversal.
Note5. FLAG/ configured to output EQL and BW in non-latch mode

Fig 6. Range-limit , non-quadrature mode



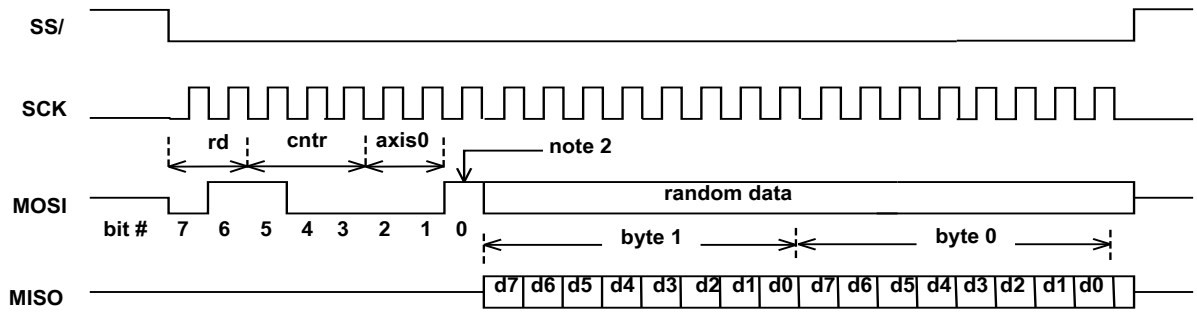
Note. The SPI port of the host MCU must be set up in Mode 0 with following characteristics:
1. MCU in master mode.
2. SCK idle state = low
3. SCK edge for (MOSI) input data shift = high to low
4. SCK edge for (MISO) output data shift = high to low

Fig 7. SPI bus protocol in Mode 0



Note 1. This example shows a WR command followed by a RD command
Note 2. WR command accesses MCR1 registers in both axes for write
Note 3. RD command accesses MCR1 register in axis1 for read.

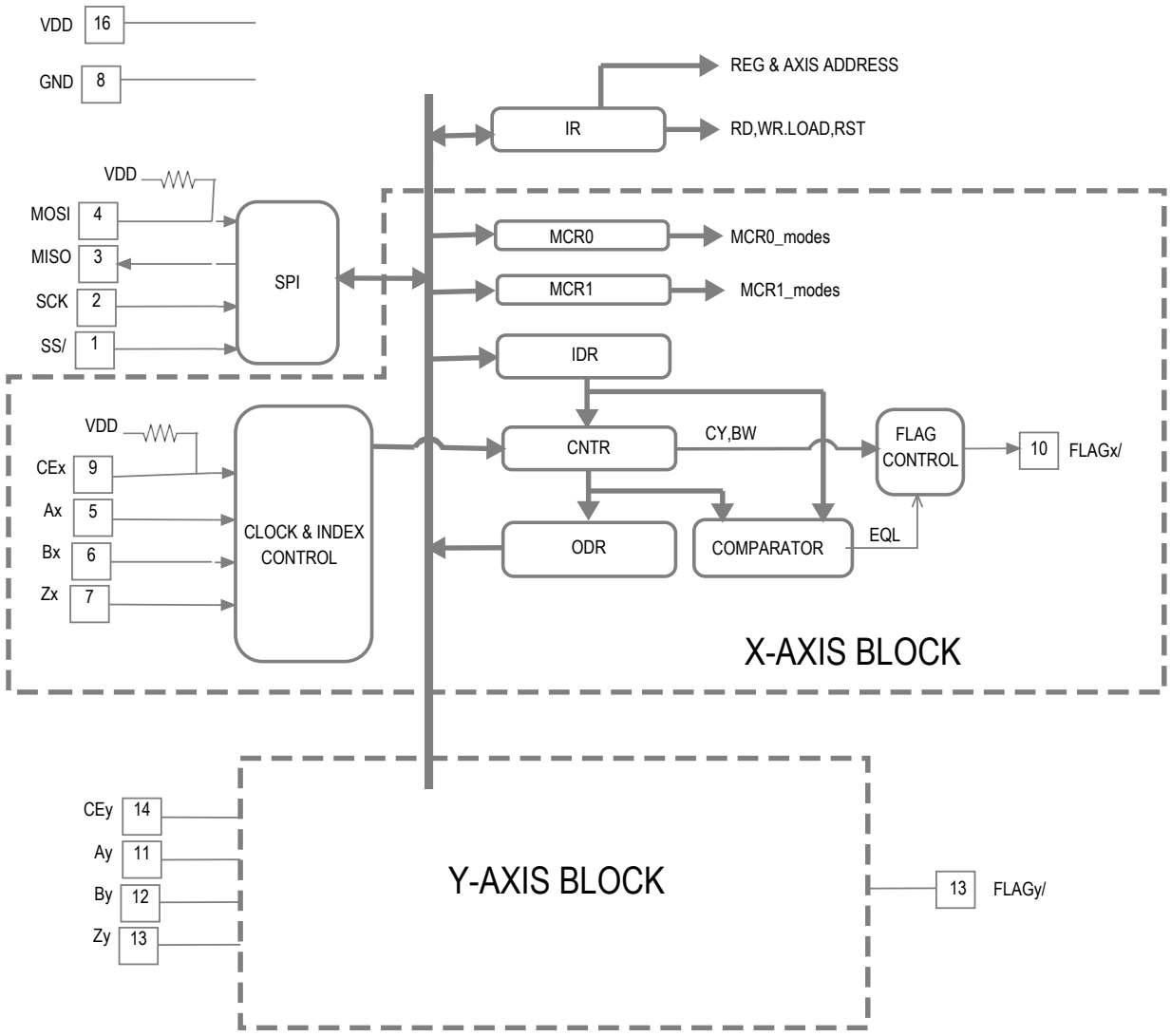
Fig 8. WR_MCR1-RD_MCR1



Note1. This example shows a RD command to read CNTR in axis0 in 2-byte configuration.
Note2. IR bit0 = 1 causes an auto transfer of DSTR to SSTR when the RD command is executed.

Fig 9. RD_CNTR

7466-020123-10



Note 1. All inputs have ESD protection circuits (not shown).
Note 2. Following inputs have Schmitt trigger buffers (not shown): MOSI, MISO, SS/ SCK, Ax, Bx, Zx, Ay, By, and Zy.

Fig 10. LS7466 Block diagram

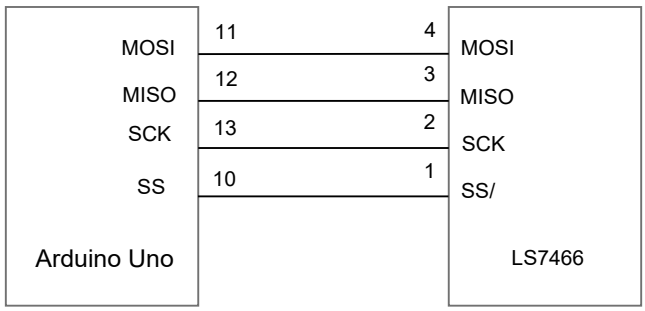


Fig 10. Arduino Uno to LS7466 interface

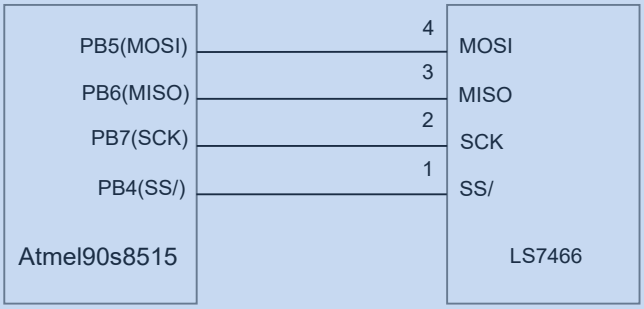


Fig 11. Atmel90s8515 to LS7466 interface

```
//LS7466 to 90s8515 SPI interface example

/******** Add appropriate header files here*****
*#include <iom32v.h>
*#include <90s8515.h>
*#include <macros.h>
*#include <delay.h>
*****************************************/

/* MCR0 configuration data - the byte is formed with ONE segment
 taken from each group and ORing them 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 NON_RECYCLE 0x04
#define RANGE_LIMIT  0x08
#define MODULO_N     0x0C

//Index modes
#define DISABLE_INDX 0x00       //index_disabled
#define INDX_LCNT 0x10          //index_load_CNTR
#define INDX_RCNT 0x20          //index_rest_CNTR
#define INDX_RDST 0x30          //index_reset_DSTR
#define INDX_LSST 0x40          //index_load_SSTR
#define INDX_LODR 0x50          //index_load_ODR
#define INDX_LODR_RCNT 0x60     //index_LODR&RCNT

#define ACTV_LOW_INDX 0x00      //active low index
#define ACTV_HIGH_INDX 0x80     //active high index
```

```
/* MCR1 configuration data; any of these data segments can be ORed together */
//Flag modes
#define NO_FLAGS 0x00          //no flag
#define INDX_FLAG 0x10         //INDX flag
#define EQL_FLAG 0x20          //EQL flag
#define BW_FLAG 0x40           //BW flag
#define CY_FLAG 0x80           //CY flag
#define DYNAMIC_flag 0X00      //FLAG/ out is dynamic
#define LATCHED_FLAG 0x08      //FLAG/ out is latched

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

//data width
#define BYTE_3 0x00            //3-byte mode
#define BYTE_2 0x01            //2-byte mode
#define BYTE_1 0x02            //1-byte mode

/* LS7466 op-code list */
#define RST_MCR0x    0x08      //reset x-axis MCR0
#define RST_MCR0y    0x0a      //reset y-axis MCR0
#define RST_MCR0xy   0x0c      //reset both-axis MCR0
#define RST_MCR1x    0x10      //reset x-axis MCR1
#define RST_MCR1y    0x12      //reset y-axis MCR1
#define RST_MCR1xy   0x14      //reset both-axis MCR1
#define RST_CNTRx    0x20      //reset x-axis CNTR
#define RST_CNTRy    0x22      //reset y-axis CNTR
#define RST_CNTRxy   0x24      //reset both-axis CNTR
#define RST_SSTRx    0x30      //reset x-axis SSTR
#define RST_SSTRy    0x32      //reset y-axis SSTR
#define RST_SSTRxy   0x34      //reset both-axis SSTR

#define RD_MCR0x   0x48        //read x-axis MCR0
#define RD_MCR0y   0x4a        //read y-axis MCR0
#define RD_MCR1x   0x50        //read x-axis MCR1
#define RD_MCR1y   0x52        //read y-axis MCR1
#define RD_CNTRx   0x60        //read x-axis CNTR
#define RDC_LDSx   0x61        //read CNTR and load SSTR in x-axis
#define RD_CNTRy   0x62        //read y-axis CNTR
#define RDC_LDSy   0x63        //read CNTR and load SSTR in y-axis
#define RD_ODRx    0x68        //read x-axis ODR
#define RD_ODRy    0x6a        //read y-axis ODR
#define RD_SSTRx   0x70        //read x-axis SSTR
#define RD_SSTRy   0x72        //read y-axis SSTR
```

```
#define WR_MCR0x   0x88        //write to x-axis MCR0
#define WR_MCR0y   0x8a        //write to y-axis MCR0
#define WR_MCR0xy  0x8c        //write to both-axis MCR0
#define WR_MCR1x   0x90        //write to x-axis MCR1
#define WR_MCR1y   0x92        //write to y-axis MCR1
#define WR_MCR1xy  0x94        //write to both-axis MCR1
#define WR_IDRx    0x98       //write to x-axis IDR
#define WR_IDRy    0x9a       //write to y-axis IDR
#define WR_IDRxy   0x9c       //write to bth-axis IDR

#define LOAD_CNTRx   0xe0      //load x-axis CNTR
#define LOAD_CNTRy   0xe2      //load y-axis CNTR
#define LOAD_CNTRxy  0xe4      //load x-axis CNTR
#define LOAD_ODRx    0xe8      //load x-axis ODR
#define LOAD_ODRy    0xea      //load y-axis ODR
#define LOAD_ODRxy   0xec      //load both-axis ODR
#define LOAD_SSTRx   0xf0      //load x-axis SSTR
#define LOAD_SSTRy   0xf2      //load y-axis SSTR
#define LOAD_SSTRxy  0xf4      //load both-axis SSTR

#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, SPI mode0 operation
    CPHA = 0, SPI Mode 0 operation
    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)
{
   unsigned char spi_data;
   Slave_Select_High;          // Keep SS/ High for LS7466 deselect
   Slave_Select_Low;           // Switch SS/  low for new command

   /* Send command to LS7466 */
   SPDR = op_code;                  // Send command
   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 singleByteWR(unsigned char op_code, unsigned char data)
{
   unsigned char spi_data;
   Slave_Select_High;          // Keep SS/ High for LS7466 deselect
   Slave_Select_Low;           // Switch SS/  low for new command

 /* Send command to LS7466 */
   SPDR = op_code;             // Send command
   while (!(SPSR & (1<<SPIF)))     // Wait for end of the transmission
   {
   };
   spi_data = SPDR;            // Reset SPIF

   /* Send data to be written to LS7466 Register */
   SPDR = data;                // Send data
   while (!(SPSR & (1<<SPIF)))     // Wait for end of the transmission
   {
   };
   spi_data = SPDR;            // Reset SPIF
   /*additional bytes can be sent here for multibyte WR, e.g.,to IDR*/
   Slave_Select_High;          // Switch SS/ high for end of command
}

void singleByteRD(unsigned char op_code)
{
   unsigned char spi_data;
   Slave_Select_High;          // deselect the the LS7466
   Slave_Select_Low;           // Switch SS/ low for new command
```

```
 /* Send RD OP Code */
   SPDR = op_code;              // Start the transmission
   while (!(SPSR & (1<<SPIF)))     // Wait for end of  transmission
   {
   };
   spi_data = SPDR;             // Reset SPIF

   /* Dummy Transfer to RD Data */
   SPDR = 0xFF;                 // Start the transmission for RD from LS7466
   while (!(SPSR & (1<<SPIF)))     // Wait for end of the transmission
   {
   };
   spi_data = SPDR;             // Reset SPIF
   /*additional bytes can be received here for multibyte RD, e.g.,from ODR*/
   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(RST_CNTRxy);
   singleByteWR(WR_MCR0xy, QUADRX4|FREE_RUN|INDX_LCNT|ACTV_LOW_INDX);
   singleByteWR(WR_MCR1xy, IDX_FLAG|EQL_FLAG|DYNAMIC_FLAG|EN_CNTR|BYTE_1);
   singleByteRD(RD_MCR0x);
   singleByteRD(RD_MCR1x);
   singleByteRD(RDC_LDSy);
   return 0;
}
```