Умножитель

- Знаковое и беззнаковое умножение
- Режим с накоплением
- 8, 16, 24, 32-битные операнды
- Режим насыщения
- Умножение дробных чисел

Умножитель

- Размер операндов определяется адресом при их записи, тип операции — адресом записи 1 операнда

- Регистры 1 операнда:
- MPY = MPY32L, беззнаковое, биты 0...15
- MPYS = MPYS32L, знаковое, биты 0...15
- MAC = MAC32L, беззнаковое с накоплением, 0...15
- MACS = MACS32L, знаковое с накоплением, биты 0...15
- MPY32H, беззнаковое, биты 16...31
- MPYS32H, знаковое,биты 16...31
- MAC32H, беззнаковое с накоплением,биты 16...31
- MACS32H, знаковое с накоплением,биты 16...31

Умножитель

- Регистры 2 операнда:
- OP2 – запуск операции умножения с 16-битным операндом 2 (биты 0...15)
- OP2L – запуск операции умножения с 32-битным операндом 2 (биты 0...15)
- OP2H – продолжение операции умножения с 32-битным операндом 2 (биты 16...31)

- Последняя запись (32L или 32H) перед записью 2 операнда определяет ширину 1 операнда
- Запись в OP2H без предшествующей OP2L

- 64-битный результат – 16-битные регистры RES0...RES3. RES0 = RESLO, RES1 = RESHI

Умножитель

Operation (OP1 × OP2)	Result Ready in MCLK Cycles					After
	RES0	RES1	RES2	RES3	MPYC Bit	
8/16 × 8/16	3	3	4	4	3	OP2 written
24/32 × 8/16	3	5	6	7	7	OP2 written
8/16 × 24/32	3	5	6	7	7	OP2L written
	N/A	3	4	4	4	OP2H written
24/32 × 24/32	3	8	10	11	11	OP2L written
	N/A	3	5	6	6	OP2H written

- *8/16-битные: результат доступен сразу после записи 2 операнда. В случае косвенной адресации требуется операция NOP*
- *24/32-битные: аналогично*

Умножитель

- **Поля SUMEXT и MPYC позволяют определить перенос либо знак результата в зависимости от операции**

Mode	SUMEXT		MPYC	
MPY	SUMEXT is always 0000h.		MPYC is always 0.	
MPYS	SUMEXT contains the extended sign of the result.		MPYC contains the sign of the result.	
	00000h	Result was positive or zero	0	Result was positive or zero
	0FFFFh	Result was negative	1	Result was negative
MAC	SUMEXT contains the carry of the result.		MPYC contains the carry of the result.	
	0000h	No carry for result	0	No carry for result
	0001h	Result has a carry	1	Result has a carry
MACS	SUMEXT contains the extended sign of the result.		MPYC contains the carry of the result.	
	00000h	Result was positive or zero	0	No carry for result
	0FFFFh	Result was negative	1	Result has a carry

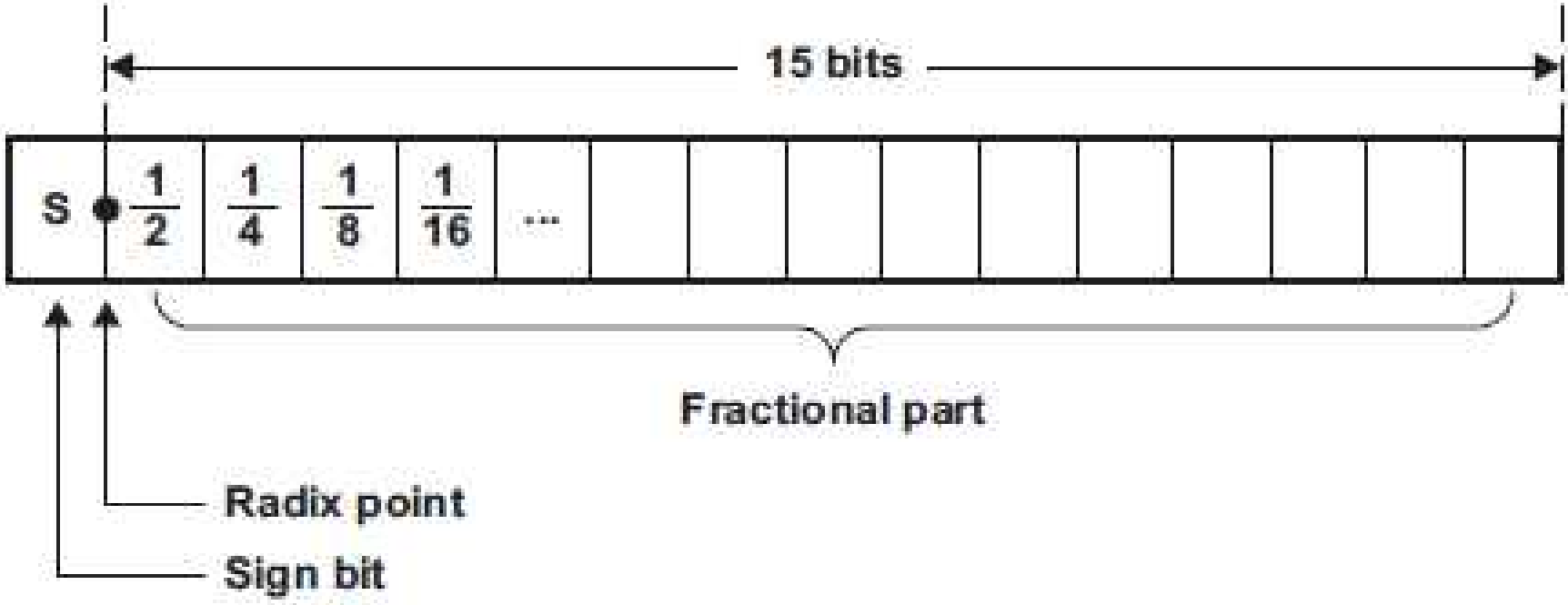
Умножитель

- *При операции MACS переполнение автоматически не определяется*
- *Определить его можно, если перенос MPYC отличается от знака SUMEXT*

```
; 16x16 Unsigned Multiply
MOV #01234h,&MPY ;Load 1st operand
MOV #05678h,&OP2 ;Load 2nd operand
; ... ; Process results
```

Умножитель

- MPYFRAC = 1 устанавливает режим умножения дробной части
- При умножении Q15 чисел, результат Q31, чтение RES1 дает результат в Q15
- При умножении Q31 чисел результат Q31 в регистрах RES2 и RES3



Умножитель

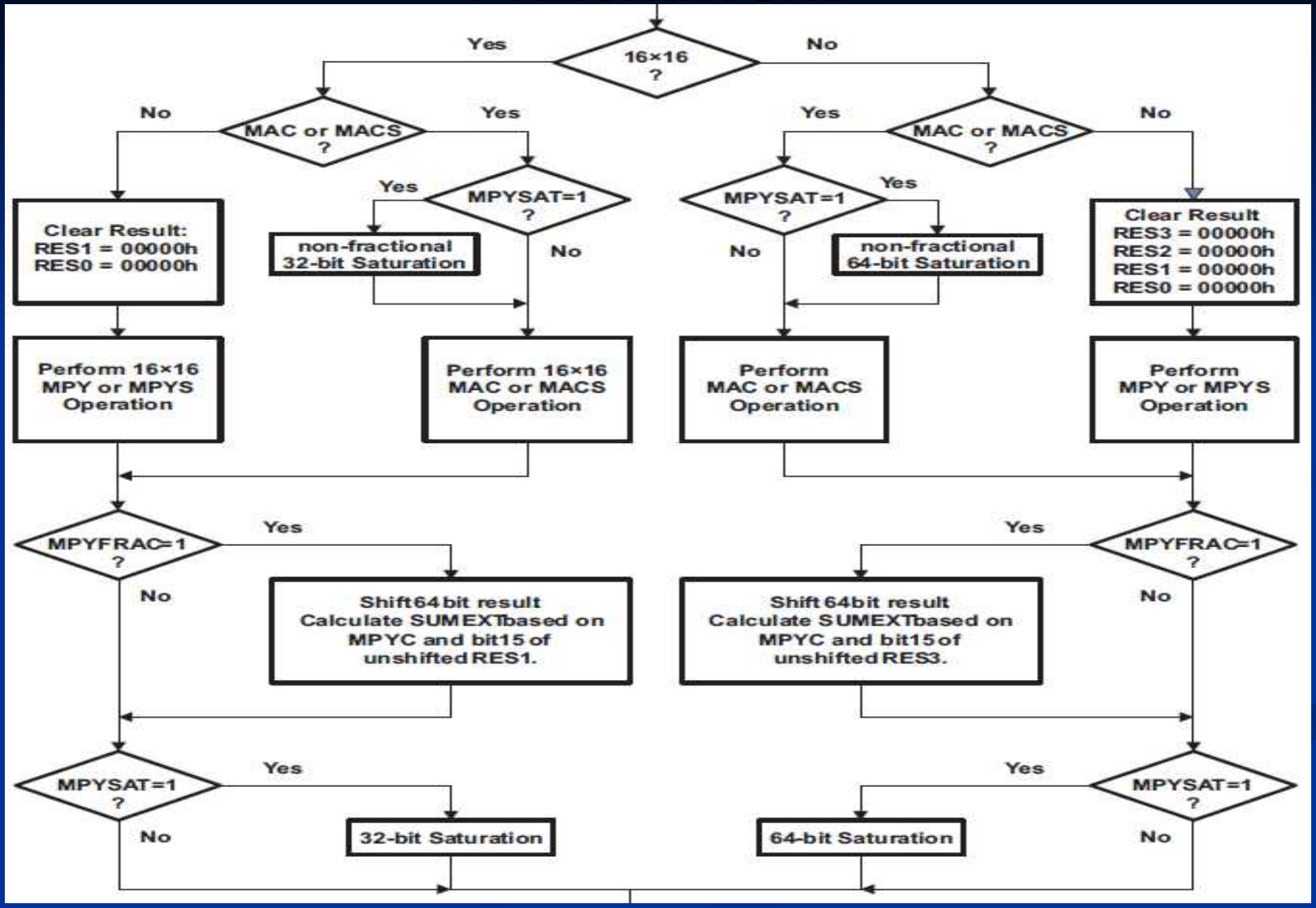
- Режим насыщения: MPYSAT = 1
- При переполнении устанавливается максимально возможное значение
- При 16-разрядных операциях режим влияет только на биты 0...31 (RES0 и RES1)
- При операциях MAC и MACS смешивание последовательностей с 16-разрядными операциями и с 32-разрядными операциями дает непредсказуемый результата

Умножитель

- В режиме насыщения результат доступен только после вычисления RES3

Operation (OP1 × OP2)	Result Ready in MCLK Cycles					After
	RES0	RES1	RES2	RES3	MPYC Bit	
8/16 × 8/16	3	3	N/A	N/A	3	OP2 written
24/32 × 8/16	7	7	7	7	7	OP2 written
8/16 × 24/32	7	7	7	7	7	OP2L written
	4	4	4	4	4	OP2H written
24/32 × 24/32	11	11	11	11	11	OP2L written
	6	6	6	6	6	OP2H written

Умножитель


16×16 ?
Yes
No
MAC or MACS ?
No
Yes
MAC or MACS ?
Yes
No
MPYSAT=1 ?
Yes
No
MPYSAT=1 ?
Yes
No
Clear Result: RES1 = 00000h RES0 = 00000h
non-fractional 32-bit Saturation
non-fractional 64-bit Saturation
Clear Result RES3 = 00000h RES2 = 00000h RES1 = 00000h RES0 = 00000h
Perform 16×16 MPY or MPYS Operation
Perform 16×16 MAC or MACS Operation
Perform MAC or MACS Operation
Perform MPY or MPYS Operation
MPYFRAC=1 ?
Yes
No
MPYFRAC=1 ?
Yes
No
Shift 64 bit result Calculate SUMEXT based on MPYC and bit15 of unshifted RES1.
Shift 64 bit result Calculate SUMEXT based on MPYC and bit15 of unshifted RES3.
MPYSAT=1 ?
Yes
No
MPYSAT=1 ?
Yes
No
32-bit Saturation
64-bit Saturation

Умножитель

- Если после запись OP перед записью OP2 возникает прерывание, в котором используется умножитель, то текущий режим теряется и результат операции непредсказуем
- Для избегания этого:
- Запрещать прерывания на время использования умножителя
- Не использовать умножитель в обработчике прерывания
- Выполнять сохранение и восстановление состояния умножителя

Умножитель

- Регистры умножителя, за исключением регистров операндов и результата:
- SUMEXT — регистр расширения суммы
- MPY32CTL0 — регистр управления умножителем

Умножитель. Регистр управления

Поле	Биты	Назначение
MPYDLY32	9	Задержка записи до получения результата (0 -64 бит, 1- 32бит)
MPYDLYWRTEN	8	Разрешение режима задержки записи
MPYOP2_32	7	Разрядность операнда 2 (0- 16, 1-32)
MPYOP1_32	6	Разрядность операнда 1 (0- 16, 1-32)
MPYMx	4-5	Режим (00- MPY, 01- MPYS, 10- MAC, 11- MACS)
MPYSAT	3	Разрешение режима насыщения
MPYFRAC	2	Разрешение дробной части
MPYC	0	Флаг переноса

Последовательный интерфейс

- USCI — Universal Serial Communication Interface
- 2 канала USCI_A
  - Режим UART (Universal asynchronous receiver/transmitter)
  - IrDA
  - Режим SPI (Serial Peripheral Interface)
  - Автоматическое определение скорости LIN
- 2 канала USCI_B
  - Режим I^2C (Inter-Integrated Circuit)
  - Режим SPI

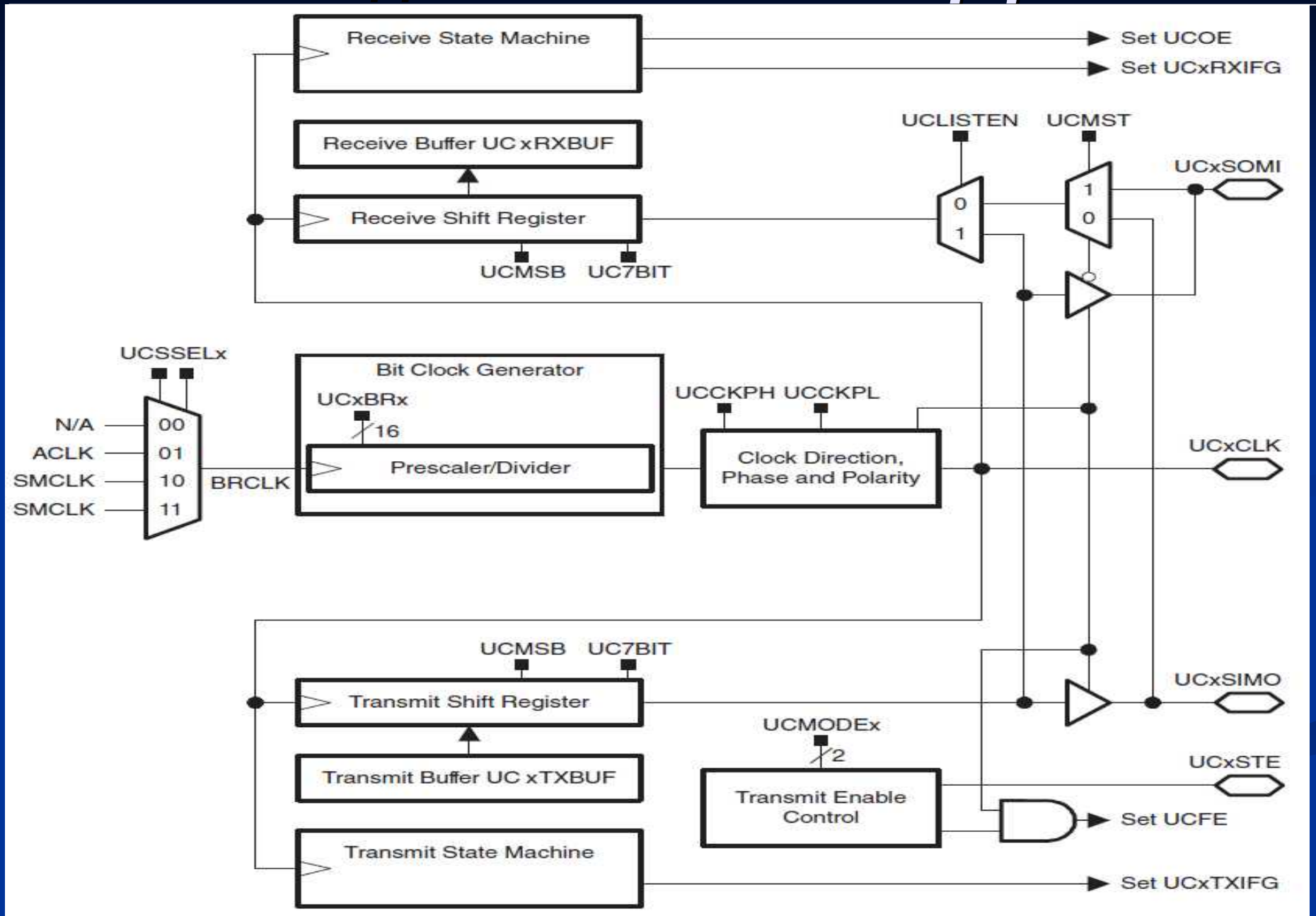
Последовательный интерфейс

- SPI — синхронный дуплексный интерфейс
- 3 или 4 линии: UCxSIMO, UCxSOMI, UCxCLK, and UCxSTE
- 7 или 8 бит данных
- Режим обмена: LSB или MSB первым
- Режим Master / Slave
- Независимые для приема и передачи сдвиговые регистры
- Отдельные буферные регистры для приема и передачи

Последовательный интерфейс

- Непрерывный режим передачи
- Выбор полярности синхросигнала и контроль фазы
- Программируемая частота синхросигнала в режиме Master
- Независимые прерывания на прием и передачу
- Операции режима Slave в LPM4

Последовательный интерфейс



Последовательный интерфейс

- Сигналы интерфейса:
- UCxSIMO — Slave In, Master Out
- UCxSOMI — Slave Out, Master In
- UCxCLK — тактовый сигнал, выставляется Master-устройством
- UCxSTE — Slave Transmit Enable. В 4-битном протоколе используется для нескольких Master устройств на одной шине. В 3-битном не используется

Последовательный интерфейс

- Передача данных начинается при помещении данных в регистр UCxTXBUF
- Данные помещаются в сдвиговый регистр, если он пуст, что начинает передачу по линии UCxSIMO
- Флаг прерывания UCTXIFG устанавливается при перемещении данных в сдвиговый регистр и сигнализирует об освобождении буферного регистра, а не окончания передачи
- UCTXIFG требует разрешений UCTXIE и GIE, автоматически сбрасывается при записи в UCxTXBUF

Последовательный интерфейс

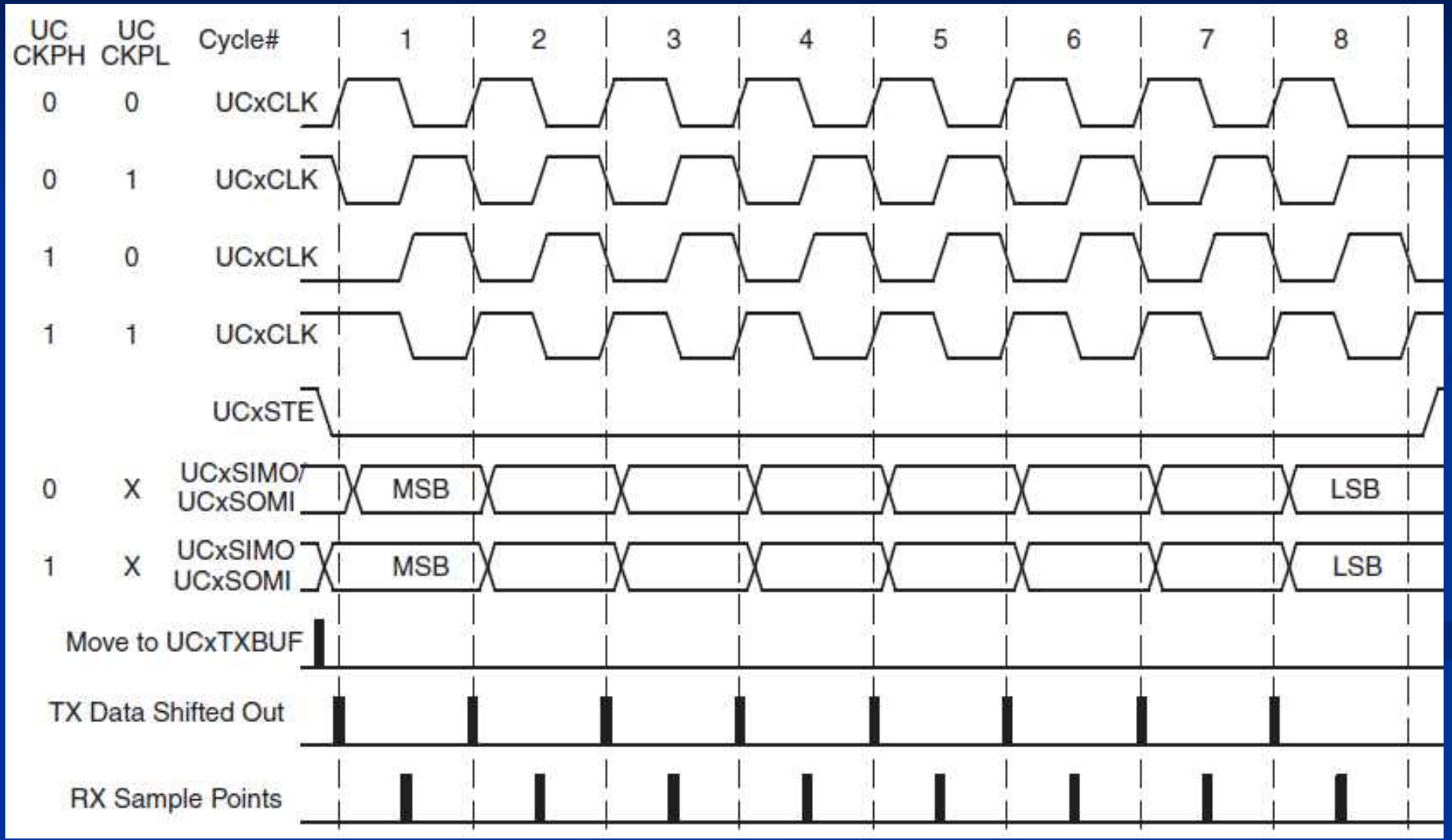
- Прием данных по линии UCxSOMI начинается с помещения данных в сдвиговый регистр по спаду синхросигнала
- Как только символ передан, данные из сдвигового регистра помещаются в UCxRXBUF
- После этого устанавливается флаг прерывания UCRXIFG, что сигнализирует об окончании передачи
- UCRXIFG требует разрешений UCRXIE и GIE, автоматически сбрасывается при чтении UCxRXBUF

Последовательный интерфейс

- Сброс бита UCSWRST разрешает работу модуля USCI
- Для Master-устройства тактовый генератор готов к работе, но начинает генерировать сигнал только при записи в регистр UCxTXBUF
- Для Slave-устройства тактовый генератор отключен, а передача начинается с выставлением тактового сигнала Master-устройством
- Наличие передачи определяется флагом UCBUSY = 1

Последовательный интерфейс

- Поля полярности UCCKPL и фазы UCCKPH определяют 4 режима синхронизации бит



Последовательный интерфейс

- Если UCMST = 1, для тактирования используется генератор USCI, входная частота выбирается битами UCSSELx
- 16 бит UCBRx (регистры UCxxBR1 и UCxxBR0) определяют делитель BRCLK входной тактовой частоты USCI
- $F_{BitClock} = f_{BRCLK} / UCBRx$

- Начальные адреса регистров:
- USCI_A0 — 05C0h, USCI_A1 — 0600h
- USCI_B0 — 05E0h, USCI_B1 — 0620h

Последовательный интерфейс

Регистр	Адрес	Назначение
UCAxCTL0	05C0h	Регистры управления
UCAxCTL1	05C1h	
UCAxBR0	05C6h	Управление скоростью передачи
UCAxBR1	05C7h	
UCAxSTAT	05CAh	Регистр состояния
UCAxRXBUF	05CCh	Буфер приемника
UCAxTXBUF	05CEh	Буфер передатчика
UCAxIE	05DCh	Разрешение прерываний
UCAxIFG	05DDh	Флаги прерываний
UCAxIV	05DEh	Вектор прерываний

Последовательный интерфейс

Регистр	Биты	Поле	Назначение
UCAxCTL0	7	UCCKPH	Выбор фазы Ти (0 — изменение по первому перепаду, 1 — захват)
	6	UCCKPL	Выбор полярности Ти (0 — активный - высокий)
	5	UCMSB	Выбор порядка передачи: 0 — LSB, 1- MSB
	4	UC7BIT	Разрядность: 0 — 8, 1 — 7
	3	UCMST	Режим: 0 — Slave, 1 - Master

- ***Здесь и далее красное поле обозначает, что изменения возможны лишь при UCSWRST = 1***

Последовательный интерфейс

Регистр	Биты	Поле	Назначение
UCAxCTL0	1-2	UCMODEx	Выбор синхронного режима: 00 – 3pin SPI, 01 – 4pin SPI + STE активен высокий, 10 – 4pin SPI + STE активный низкий, 11 - I^2C
	0	UCSYNC	Выбор синхронного режима (=1)
UCAxCTL1	6-7	UCSSELx	Выбор источника Ти: 01 — ACLK, 10,11 - SMCLK
	0	UCSWRST	Разрешение программного сброса: 1 — логика интерфейса переводится в состояние сброса

Последовательный интерфейс

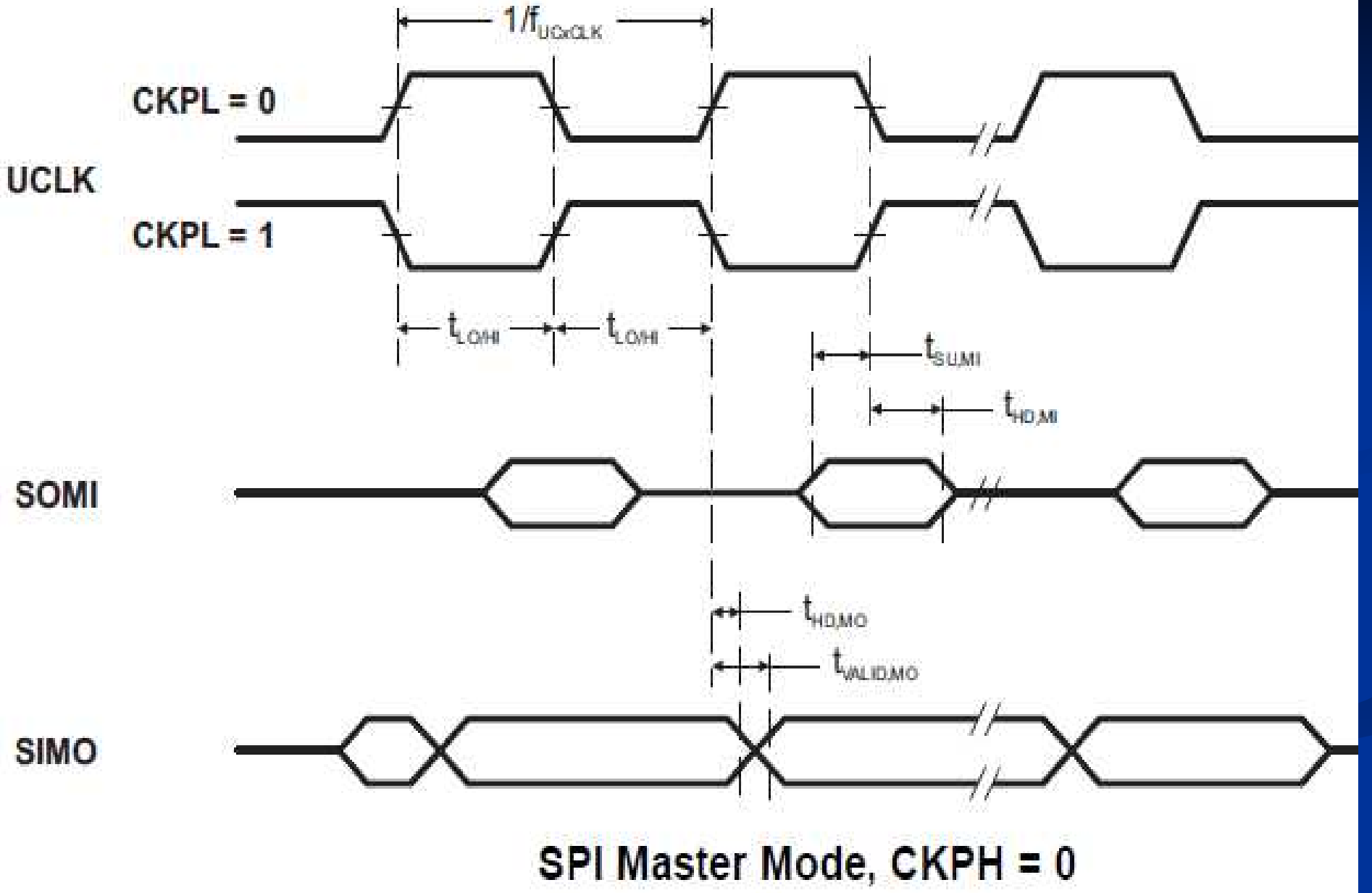
Регистр	Биты	Поле	Назначение
UCAxBR0	0-7	UCBRx	Младший байт делителя частоты
UCAxBR1	0-7	UCBRx	Старший байт делителя частоты
UCAxSTAT	7	UCLISTEN	Режим прослушивания — передача передается на прием
	6	UCFE	Флаг ошибки фрейма. При конфликте на шине в 4-bit
	5	UCOE	Флаг ошибки перезаписи. Устанавливается, если происходит запись в регистр UcxRXBUF до чтения предыдущего значения
	0	UCBUSY	Флаг приема/передачи

Последовательный интерфейс

Регистр	Биты	Поле	Назначение
UCAxRXBUF	0-7	UCRXBUFx	Буфер приемника
UCAxTXBUF	0-7	UCTXBUFx	Буфер передатчика
UCAxIE	1	UCTXIE	Разрешение пре-ния передачи
	0	UCRXIE	Разрешение прерывания приема
UCAxIFG	1	UCTXIFG	Флаг пр-я передачи
	0	UCRXIFG	Флаг пр-я приема
UCAxIV	15-0	UCIVx	Вектор прерываний

- ***Для интерфейса USCI_Bx поля и регистры аналогичны***
- ***После сброса SPI в режиме 3-pin***

Последовательный интерфейс

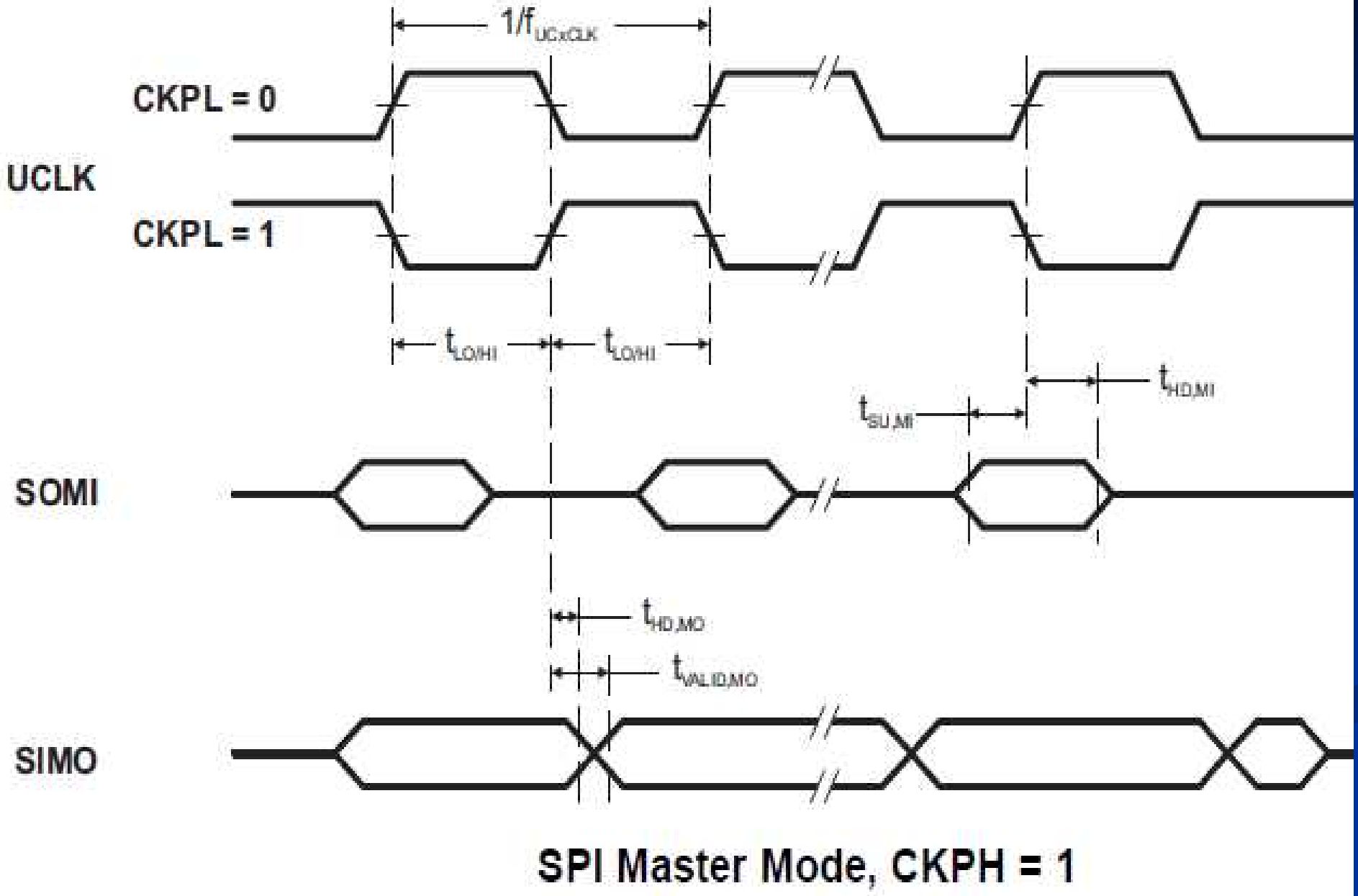


SPI Master Mode, CKPH = 0

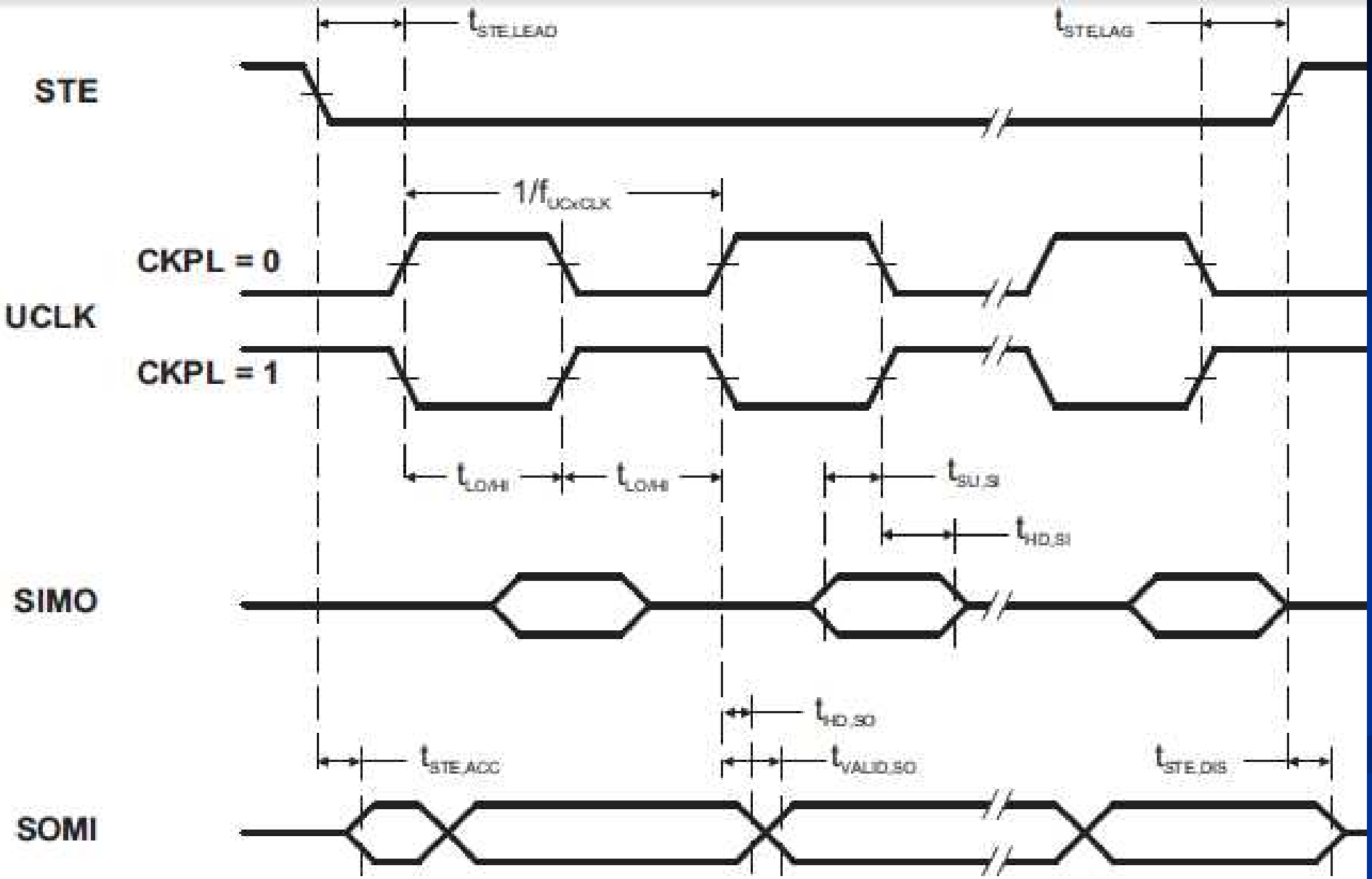
Последовательный интерфейс

- $f_{UCxCLK} = 1/2t_{LO/HI}$;
- $t_{LO/HI} \geq max(t_{VALID, MO(Master)} + t_{SU, SI(Slave)}, t_{SU, MI(Master)} + t_{VALID, SO(Slave)})$
- $t_{SU, MI}$ — время установки данных на входе SOMI — мин 25 — 55 нс в зависимости от режима
- $t_{HD, MI}$ — время удержания данных на входе SOMI — мин 0 нс
- $t_{VALID, MO}$ — время появления действительных данных на выходе SIMO после фронта CLK — макс 15-20 нс в зависимости от режима
- $t_{HD, MO}$ — время удержания данных на выходе SIMO. При отрицательных значениях данные становятся недействительными до фронта CLK. Мин -8...-10 нс

Последовательный интерфейс

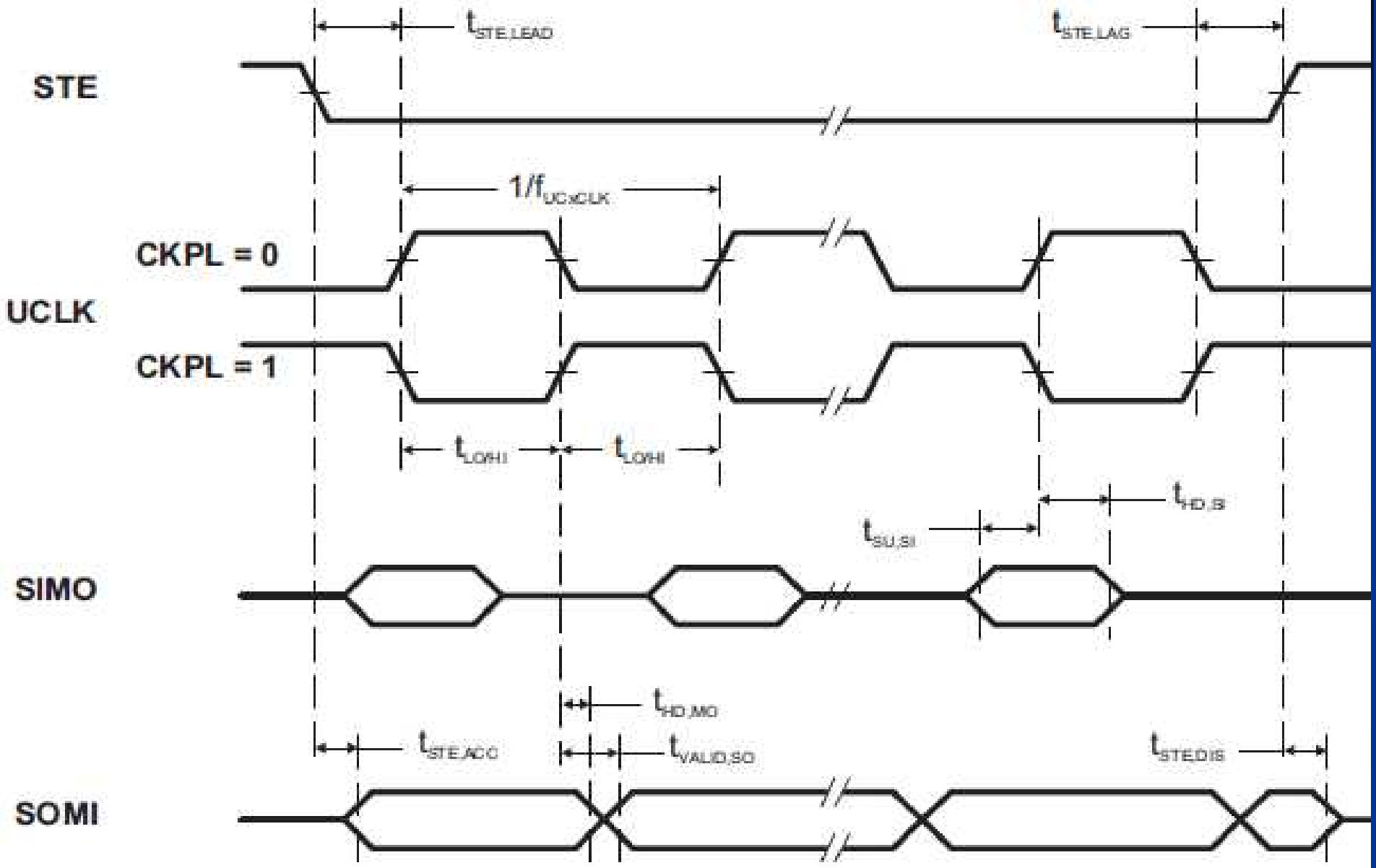


Последовательный интерфейс



SPI Slave Mode, CKPH = 0

Последовательный интерфейс



SPI Slave Mode, CKPH = 1

Последовательный интерфейс

- $t_{SU, SI}$ — время установки данных на входе SIMO — мин 2 — 5 нс в зависимости от режима
- $t_{HD, SI}$ — время удержания данных на входе SIMO — мин 5 нс
- $t_{VALID, SO}$ — время появления действительных данных на выходе SOMI после фронта CLK — макс 40-76 нс в зависимости от режима
- $t_{HD, SO}$ — время удержания данных на выходе SOMI - мин 8 — 18 нс

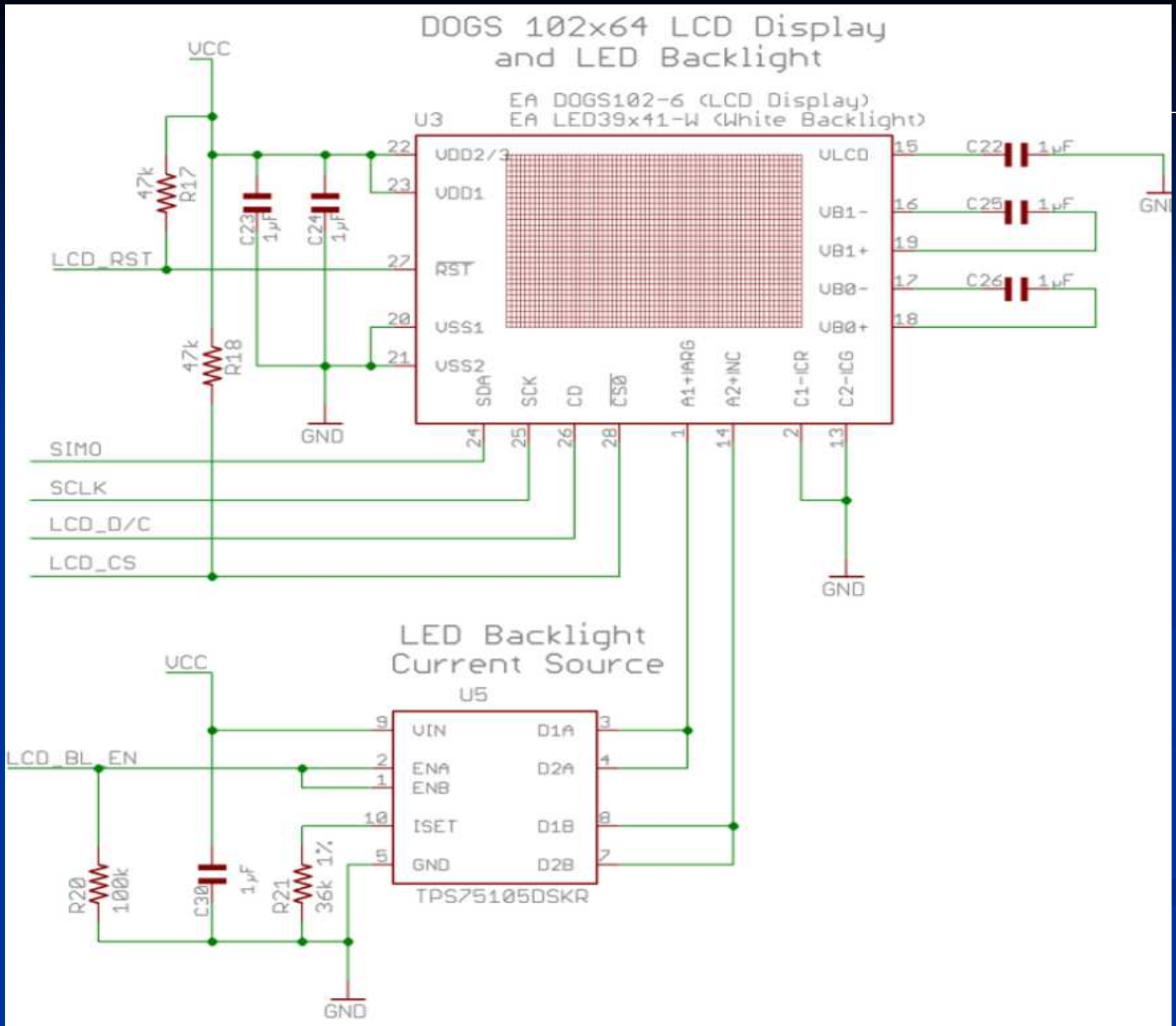
Последовательный интерфейс

- $t_{STE, LEAD}$ — время установки STE до начала первого CLK — мин 6 — 11 нс в зависимости от режима
- $t_{STE, LAG}$ — время последнего CLK до снятия STE — мин 3 нс
- $t_{STE, ACC}$ — время появления первых данных на выходе SOMI после установки STE — макс 30-66 нс в зависимости от режима
- $t_{STE, DIS}$ — время перевода выхода SOMI в высокоимпедансное состояние после снятия STE - макс 13 — 30 нс в зависимости от режима

ЖКИ

- EA DOGS102W-6 — ЖКИ дисплей 102x64 пиксела
- EA LED39x41-W - подсветка
- UC1701 — контроллер ЖКИ 65x132
- Ток потребления 250 мкА
- Частота тактирования до 33 МГц при 3,3 В
- Контроллер поддерживает 2 параллельных режима 8-бит и SPI последовательный режим
- Контроллер поддерживает также и чтение данных, однако в SPI режиме только запись
- Двухпортовая статическая DDRAM

ЖКИ


DOGS 102x64 LCD Display
and LED Backlight
EA DOGS102-6 (LCD Display)
EA LED39x41-W (White Backlight)
U3
VCC
47k
R17
C23
1µF
C24
1µF
LCD_RST
47k
R18
GND
VDD2/3
VDD1
RST
VSS1
VSS2
VLCD
VB1-
VB1+
VB0-
VB0+
C22
C25
C26
SDA
SCK
CD
CS0
A1+IARG
A2+INC
C1-ICR
C2-ICG
SIMO
SCLK
LCD_D/C
LCD_CS
LED Backlight
Current Source
U5
VCC
LCD_BL_EN
VIN
ENA
ENB
ISET
GND
D1A
D2A
D1B
D2B
TPS75105DSKR
R20
100k
C30
1µF
R21
36k 1%
GND

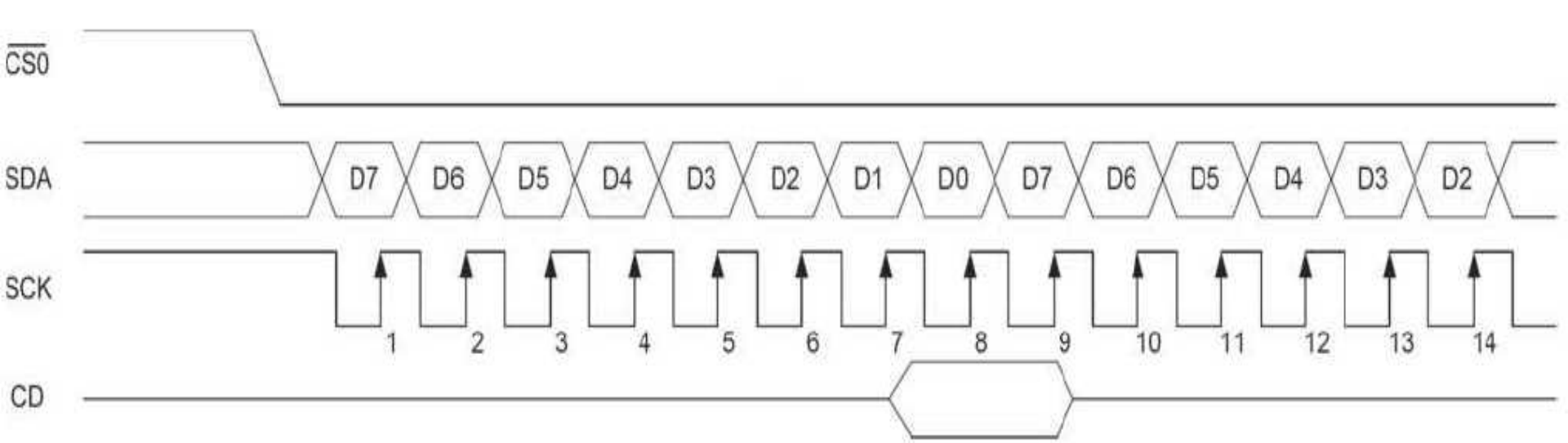
ЖКИ

по схеме (по модулю)	по MSP430F5529
LCD_RST (RST)	P5.7 / TB0.1
SIMO (SDA)	P4.1 / PM_UCB1SIMO / PM_UCB1SDA
SCLK (SCK)	P4.3 / PM_UCB1CLK / PM_UCA1STE
LCD_D/C (CD)	P5.6 / TB0.0
LCD_CS (CS0)	P7.4 / TB0.2
LCD_BL_EN (ENA, ENB)	P7.6 / TB0.4

ЖКИ

- LCD_RST (RST) сброс (=0)
- SIMO (SDA) — SIMO данные
- SCLK (SCK) — синхросигнал
- LCD_D/C (CD) — команда (=0) / данные (=1)
- LCD_CS (CS0) — выбор устройства (=0)
- LCD_BL_EN (ENA, ENB) — включение подсветки

ЖКИ



- *Только режим записи, формат MSB*
- *Сигнал CD защелкивается на бите данных D0 текущего фрейма*
- *Чтение данных по фронту синхросигнала*

ЖКИ. Набор команд

Command		Command Code								Function
		CD	D7	D6	D5	D4	D3	D2	D1	D0	
(1)	Write Data Byte	1	data bit D[7..0]								Write one byte to memory
(4)	Set Column Address LSB	0	0	0	0	0	CA[3..0]				Set the SRAM column address CA=0..131
	Set Column Address MSB		0	0	0	1	CA[7..4]				
(5)	Set Power Control	0	0	0	1	0	1	PC[2..0]			PC0: 0=Booster OFF; 1=Booster ON PC1: 0=Regulator OFF; 1=Regulator ON PC2: 0=Follower OFF; 1=Follower ON
(6)	Set Scroll Line	0	0	1	SL[5..0]						Set the display startline number SL=0..63
(7)	Set Page Address	0	1	0	1	1	PA[3..0]				Set the SRAM page address PA=0..7
(8)	Set VLCD Resistor Ratio	0	0	0	1	0	0	PC[5..3]			Configure internal resistor ratio PC=0..7
(9)	Set Electronic Volume	0	1	0	0	0	0	0	0	1	Adjust contrast of LCD panel PM=0..63
			0	0	PM[5..0]						
(10)	Set All Pixel On	0	1	0	1	0	0	1	0	C1	C1=0: show SRAM content C1=1: Set all SEG-Drivers to ON
(11)	Set Inverse Display	0	1	0	1	0	0	1	1	C0	C0=0: show normal SRAM content C0=1: show inverse SRAM content
(12)	Set Display Enable	0	1	0	1	0	1	1	1	C2	C2=0: disable Display (sleep) C2=1: enable Display (exit from sleep)
(13)	Set SEG direction	0	1	0	1	0	0	0	0	MX	MX=0: normal SEG 0..131 MX=1: mirror SEG 131..0
(14)	Set COM direction	0	1	1	0	0	MY	0	0	0	MY=0: normal COM 0..63 MY=1: mirror COM 63..0
(15)	System Reset	0	1	1	1	0	0	0	1	0	System Reset
(17)	Set LCD Bias Ratio	0	1	0	1	0	0	0	1	BR	BR: 0=1/9; 1=1/7
(25)	Set Adv. Program Control 0	0	1	1	1	1	1	0	1	0	TC: Temp. comp. 0= -0.05; 1= -0,11%/°C WC: Column wrap around 0=0FF; 1=ON WP: Page wrap around 0=0FF; 1=ON
			TC	0	0	1	0	0	WC	WP	

ЖКИ

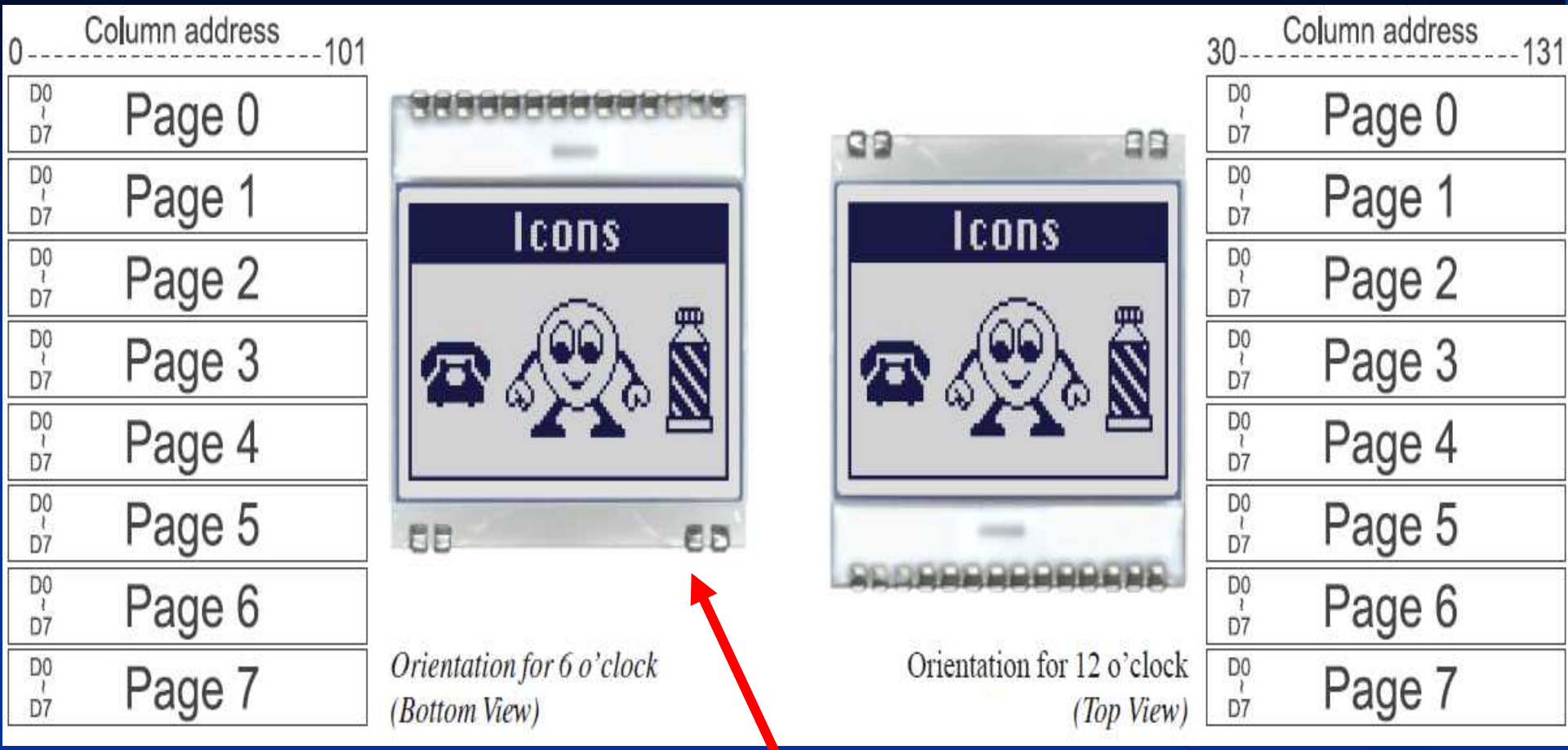
- значения по умолчанию (после сброса):
- CA [7..0] = 00000000 — адрес столбца
- PC [2..0] = 000 — питание усилителя и других элементов отключено
- SL [5..0] = 000000 — начало экрана на 0 строке (без скроллинга)
- PA [3..0] = 0000 — адрес страницы
- PC [5..3] = 100 — внутренний резистор
- PM [5..0] = 100000 — регулировка контраста
- C1 = 0 — отображение содержимого памяти
- C0 = 0 — обычный, не инверсный режим
- C2 = 0 — дисплей отключен (Sleep)

ЖКИ

- значения по умолчанию (после сброса):
- MX = 0 — не зеркальное отображение по X
- MY = 0 — не зеркальное отображение по Y
- BR = 0 — смещение (компенсация) питания 1/9
- TC = 1 — температурная компенсация -0,11% /C
- WC = 0 — циклический адрес колонки выключен
- WP = 0 — циклический адрес страницы выключен

- Поля, отмеченные красным, программный сброс не устанавливает

ЖКИ. Ориентация экрана



ЖКИ

PA[3:0]	0	Line AddeCss		MY=0 SL=0	MY=0 SL=16	MY=1 SL=0	MY=1 SL=0	MY=1 SL=25	MY=1 SL=25
0000	D0	00H	Page 0	C1	C49	C64	C48	C25	C9
	D1	01H		C2	C50	C63	C47	C24	C8
	D2	02H		C3	C51	C62	C46	C23	C7
	D3	03H		C4	C52	C61	C45	C22	C6
	D4	04H		C5	C53	C60	C44	C21	C5
	D5	05H		C6	C54	C59	C43	C20	C4
	D6	06H		C7	C55	C58	C42	C19	C3
	D7	07H		C8	C56	C57	C41	C18	C2
0001	D0	08H	Page 1	C9	C57	C56	C40	C17	C1
	D1	09H		C10	C58	C55	C39	C16	---
	D2	0AH		C11	C59	C54	C38	C15	---
	D3	0BH		C12	C60	C53	C37	C14	---
	D4	0CH		C13	C61	C52	C36	C13	---
	D5	0DH		C14	C62	C51	C35	C12	---
	D6	0EH		C15	C63	C50	C34	C11	---
	D7	0FH		C16	C64	C49	C33	C10	---
0010	D0	10H	Page 2	C17	C1	C48	C32	C9	---
	D1	11H		C18	C2	C47	C31	C8	---
	D2	12H		C19	C3	C46	C30	C7	---
	D3	13H		C20	C4	C45	C29	C6	---
	D4	14H		C21	C5	C44	C28	C5	---
	D5	15H		C22	C6	C43	C27	C4	---
	D6	16H		C23	C7	C42	C26	C3	---

- *MX = 0, MY = 0, SL = 0*
- *Page 0 Seg 1 : 11100000b, Page 0 Seg 25: 00110011*

ЖКИ

	D7	2FH												C48	C32	C17	C1	C42	C26
0110	D0	30H												C49	C33	C16	---	C41	C25
	D1	31H												C50	C34	C15	---	C40	C24
	D2	32H												C51	C35	C14	---	C39	C23
	D3	33H												C52	C36	C13	---	C38	C22
	D4	34H								Page 6				C53	C37	C12	---	C37	C21
	D5	35H												C54	C38	C11	---	C36	C20
	D6	36H												C55	C39	C10	---	C35	C19
	D7	37H												C56	C40	C9	---	C34	C18
0111	D0	38H												C57	C41	C8	---	C33	C17
	D1	39H												C58	C42	C7	---	C32	C16
	D2	3AH												C59	C43	C6	---	C31	C15
	D3	3BH												C60	C44	C5	---	C30	C14
	D4	3CH								Page 7				C61	C45	C4	---	C29	C13
	D5	3DH												C62	C46	C3	---	C28	C12
	D6	3EH												C63	C47	C2	---	C27	C11
	D7	3FH												C64	C48	C1	---	C26	C10
1000	D0	40H								Page 8				CIC	CIC	CIC	CIC	CIC	CIC
																65	49	65	49
																MUX			

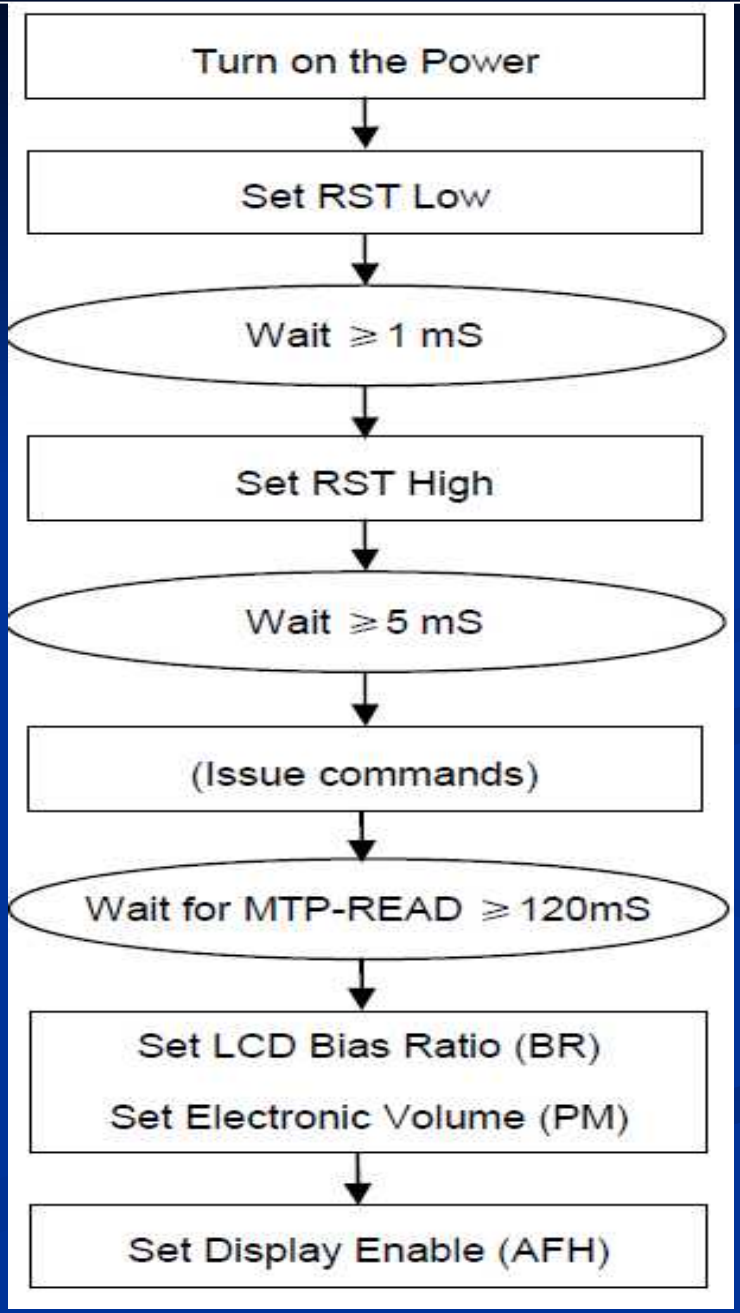
MX														
0	SEG1	SEG2	SEG3	SEG4	SEG5	SEG6	SEG7	SEG8		SEG128	SEG129	SEG130	SEG131	SEG132
1	SEG132	SEG131	SEG130	SEG129	SEG128	SEG127	SEG126	SEG125		SEG5	SEG4	SEG3	SEG2	SEG1

ЖКИ. Пример инициализации

Initialisation example (bottom view)												
Command		CD	D7	D6	D5	D4	D3	D2	D1	D0	Hex	Remark
(6)	Set Scroll Line	0	0	1	0	0	0	0	0	0	$40	Display start line 0
(13)	Set SEG direction	0	1	0	1	0	0	0	0	1	$A1	SEG reverse *)
(14)	Set COM direction	0	1	1	0	0	0	0	0	0	$C0	Normal COM0~COM63
(10)	Set All Pixel On	0	1	0	1	0	0	1	0	0	$A4	Disable -> Set All Pixel to ON
(11)	Set Inverse Display	0	1	0	1	0	0	1	1	0	$A6	Display inverse off
(17)	Set LCD Bias Ratio	0	1	0	1	0	0	0	1	0	$A2	Set Bias 1/9 (Duty 1/65)
(5)	Set Power Control	0	0	0	1	0	1	1	1	1	$2F	Booster, Regulator and Follower on
(8)	Set VLCD Resistor Ratio	0	0	0	1	0	0	1	1	1	$27	Set Contrast
(9)	Set Electronic Volume	0	1	0	0	0	0	0	0	1	$81	
			0	0	0	1	0	0	0	0	$10	
(25)	Set Adv. Program Control 0	0	1	1	1	1	1	0	1	0	$FA	Set Temperature compensation curve to -0.11%/°C
			1	0	0	1	0	0	0	0	$90	
(12)	Set Display Enable	0	1	0	1	0	1	1	1	1	$AF	Display on

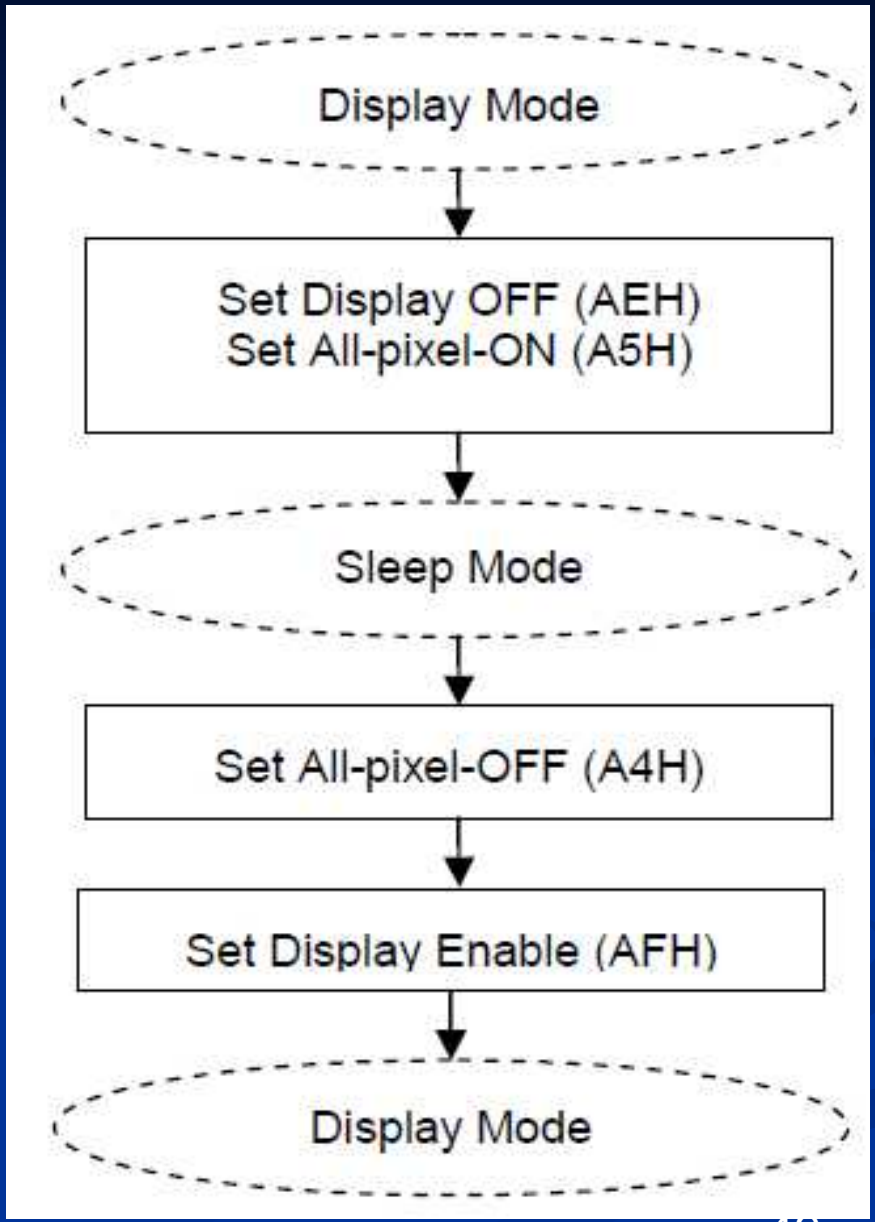
ЖКИ

- ***Контроллер ЖКИ при формировании сигнала сброса требует ожидания 5-10 мс, при включении питания ожидания не требуется***



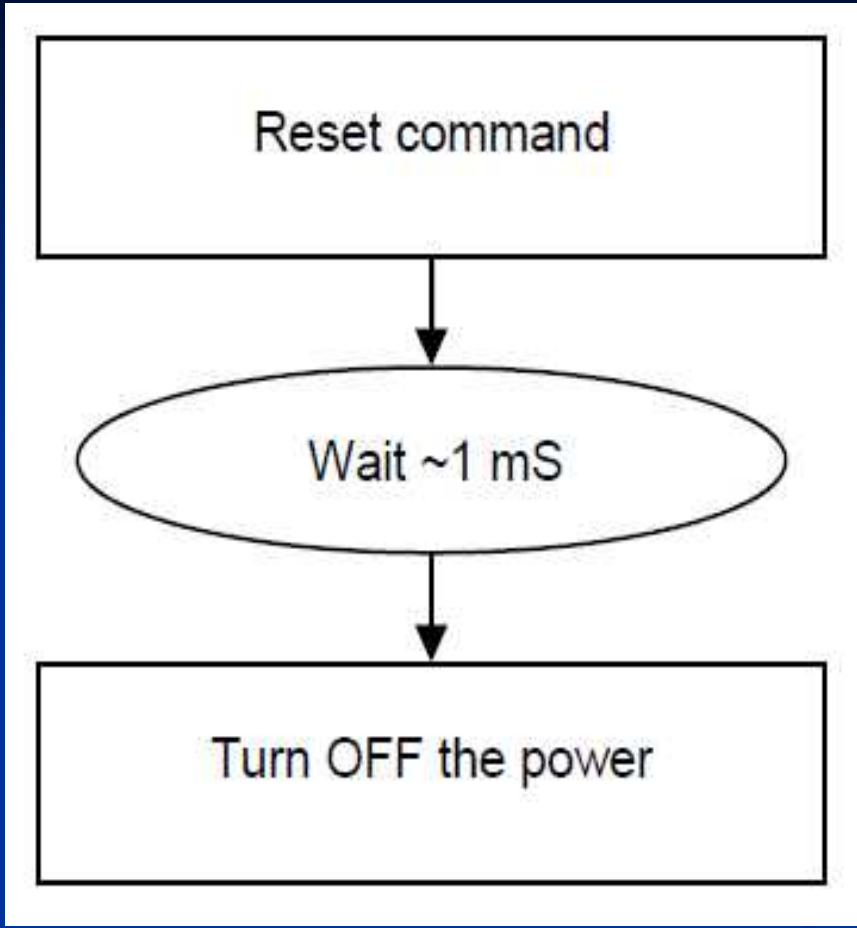
ЖКИ

- ***Вход — выход в режим ожидания***

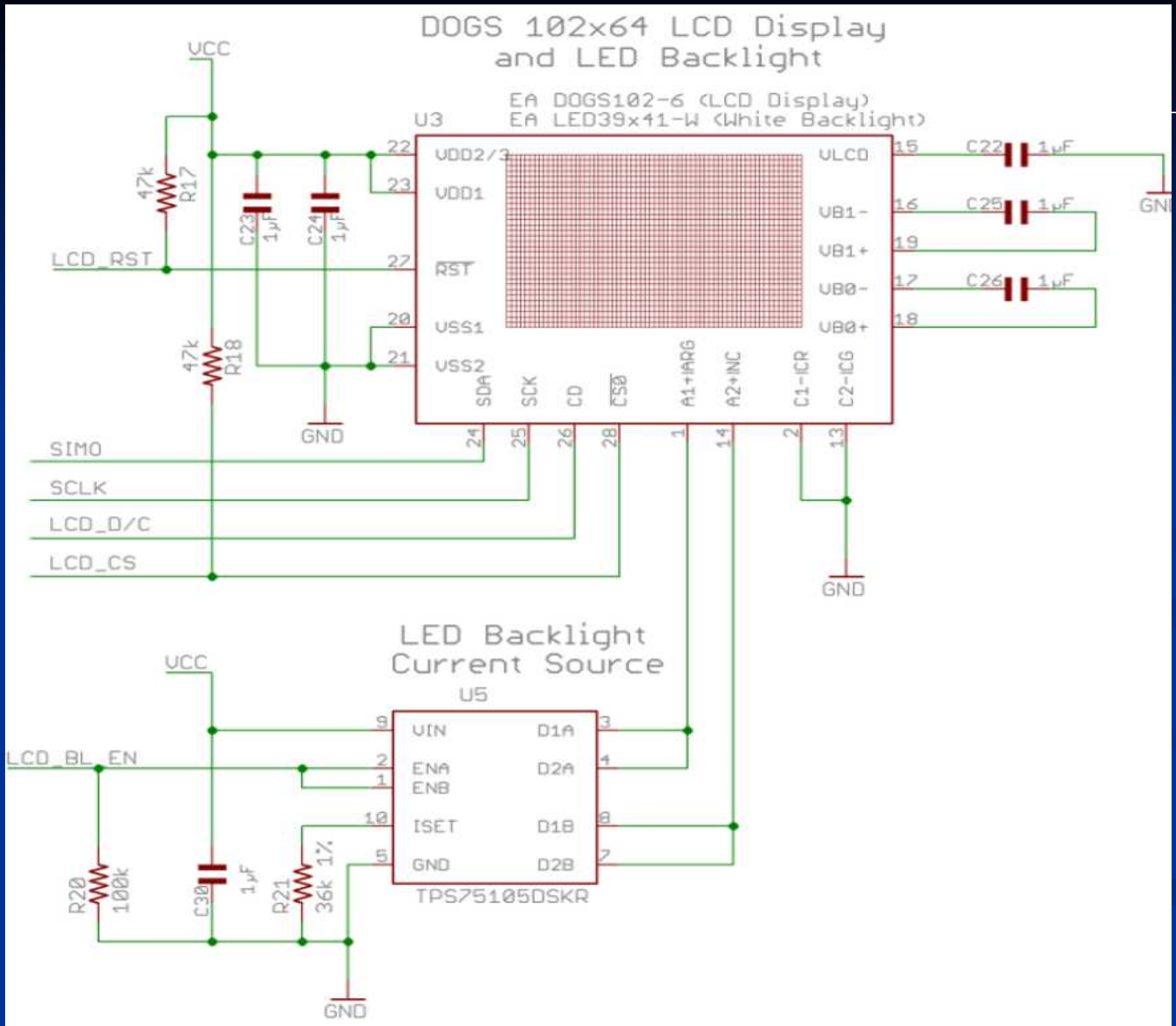


ЖКИ

- ***Выключение***



ЖКИ


DOGS 102x64 LCD Display
and LED Backlight
EA DOGS102-6 (LCD Display)
EA LED39x41-W (White Backlight)
U3
VCC
47k
R17
C23
1µF
C24
1µF
LCD_RST
47k
R18
GND
VDD2/3
VDD1
RST
VSS1
VSS2
VLCD
VB1-
VB1+
VB0-
VB0+
C22
C25
C26
SDA
SCK
CD
CS0
A1+IARG
A2+INC
C1-ICR
C2-ICG
SIMO
SCLK
LCD_D/C
LCD_CS
LED Backlight
Current Source
U5
VCC
LCD_BL_EN
VIN
ENA
ENB
ISET
GND
D1A
D2A
D1B
D2B
TPS75105DSKR
R20
100k
C30
1µF
R21
36k 1%
GND

ЖКИ

по схеме (по модулю)	по MSP430F5529
LCD_RST (RST)	P5.7 / TB0.1
SIMO (SDA)	P4.1 / PM_UCB1SIMO / PM_UCB1SDA
SCLK (SCK)	P4.3 / PM_UCB1CLK / PM_UCA1STE
LCD_D/C (CD)	P5.6 / TB0.0
LCD_CS (CS0)	P7.4 / TB0.2
LCD_BL_EN (ENA, ENB)	P7.6 / TB0.4

ЖКИ

- LCD_RST (RST) сброс (=0)
- SIMO (SDA) — SIMO данные
- SCLK (SCK) — синхросигнал
- LCD_D/C (CD) — команда (=0) / данные (=1)
- LCD_CS (CS0) — выбор устройства (=0)
- LCD_BL_EN (ENA, ENB) — включение подсветки

Пример. ЖКИ. SPI (USCI_B1)

```
#include <msp430.h>
void DOGS102_SPI(unsigned char byte1);
int main(void) {
    WDTCTL = WDTPW | WDTHOLD; // Stop watchdog timer
    // P5.6 & P5.7 (DOGS102 pin CD & RST) set as output
    P5DIR |= (BIT7 | BIT6);
    // P7.4 & P7.6 (DOGS102 pin CS & ENA) set as output
    P7DIR |= (BIT4 | BIT6);
// P4.1 & P4.3 (DOGS102 pin CDA/SIMO & SCK) set as output
    P4DIR |= (BIT1 | BIT3);
// P7.4 & P7.6 (DOGS102 pin CS & ENA) no select, on bkLED
    P7OUT |= (BIT4 | BIT6);
    // device mode: P4.1 & P4.3 is UCB1SIMO & UCB1CLK
    P4SEL |= (BIT1 | BIT3);
    // P5.7 (DOGS102 pin RST) set "0" is reset
    P5OUT &= ~BIT7;
    __delay_cycles(25000);
    // P5.7 (DOGS102 pin RST) set "1" is no reset
    P5OUT |= BIT7;
    __delay_cycles(125000);
```

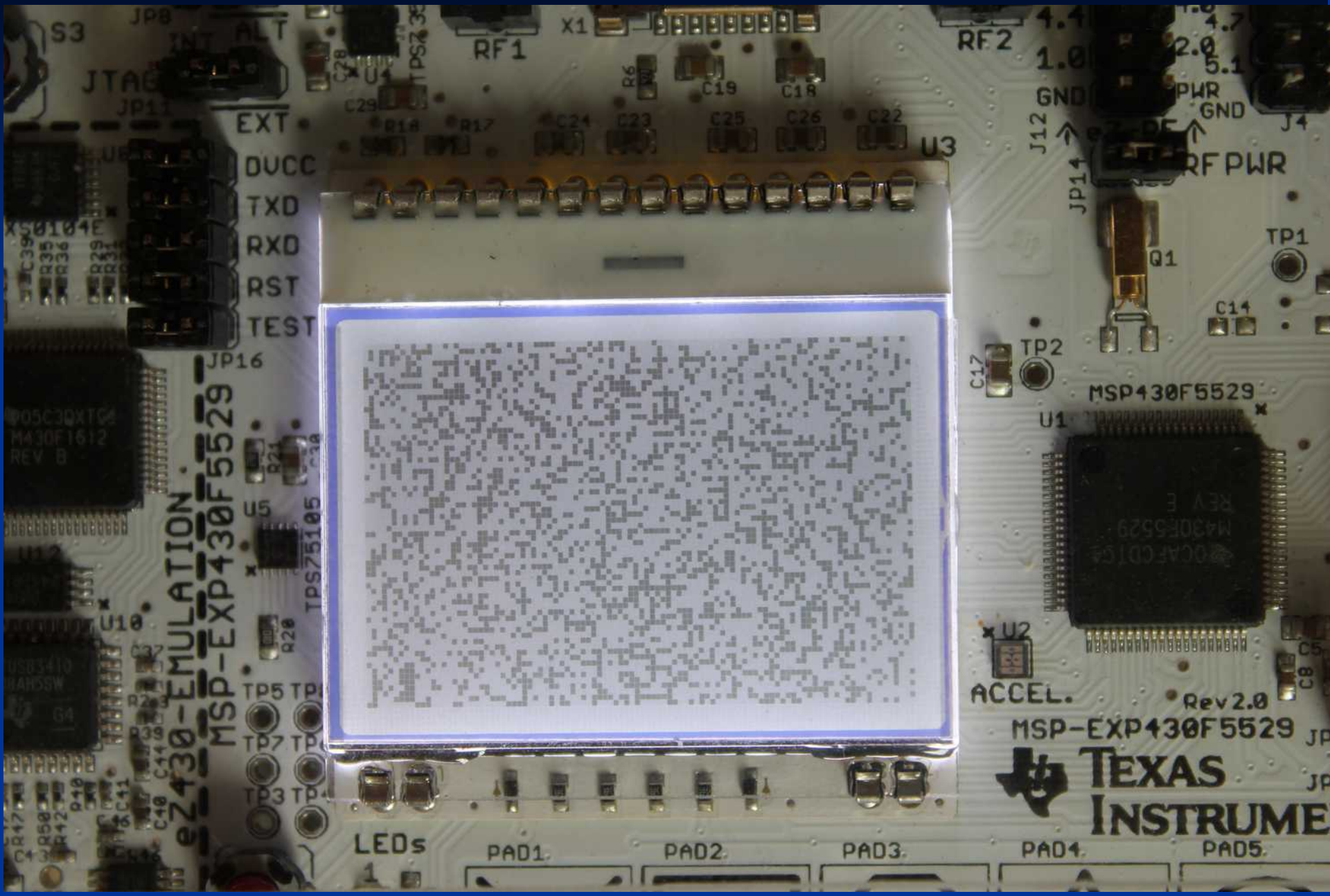
Пример. ЖКИ. SPI (USCI_B1)

```
    UCB1CTL1 |= UCSWRST;//Set SPI mode: reset logic is on
// Set SPI mode: master, MSB first, data latch on rising
    UCB1CTL0 |= UCSYNC | UCMST | UCMSB | UCCKPH;
    // Set SPI mode: SMCLK for clock, keep reset
    UCB1CTL1 |= UCSWRST | UCSSEL__SMCLK;
    UCB1BR0 = 0x30;   // Set SPI mode: low byte division
    UCB1BR1 = 0;      // Set SPI mode: high byte division
    UCB1CTL1 &= ~UCSWRST;//Reset SPI==Start SPI interface

    P5OUT &= ~BIT6;   // DOGS102 format: command
    DOGS102_SPI(0x2F);// SPI transmit.Command: Power on
    DOGS102_SPI(0xAF);// SPI transmit.Command: Display on
    // SPI transmit. Command: Set LSB column address
    DOGS102_SPI(0);
    // SPI transmit. Command: Set MSB column address
    DOGS102_SPI(0x14);
    // SPI transmit. Command: Set page address
    DOGS102_SPI(0xB4);
    P5OUT |= BIT6;        // DOGS102 format: data
    DOGS102_SPI(0xFF);   // SPI transmit. 8 pixels are set
```

Пример. ЖКИ. SPI (USCI_B1)

```
    // Enter LPM0, enable interrupts
    __bis_SR_register(LPM0_bits + GIE);
    __no_operation();   // For debugger
    return 0;
}

void DOGS102_SPI(unsigned char byte1)
{
    // Wait TXIFG == TXBUF is ready for new data
    while (!(UCB1IFG & UCTXIFG));
// P7.4 (DOGS102 pin CS) set "0" is start SPI operation
    P7OUT &= ~BIT4;
    UCB1TXBUF = byte1;          // Start SPI transmit
    // Wait until USCI_B1 SPI interface is no longer busy
    while (UCB1STAT & UCBUSY);
    // P7.4 (DOGS102 pin CS)set "1" is stop SPI operation
    P7OUT |= BIT4;
}
```

Пример. ЖКИ. SPI (USCI_B1)



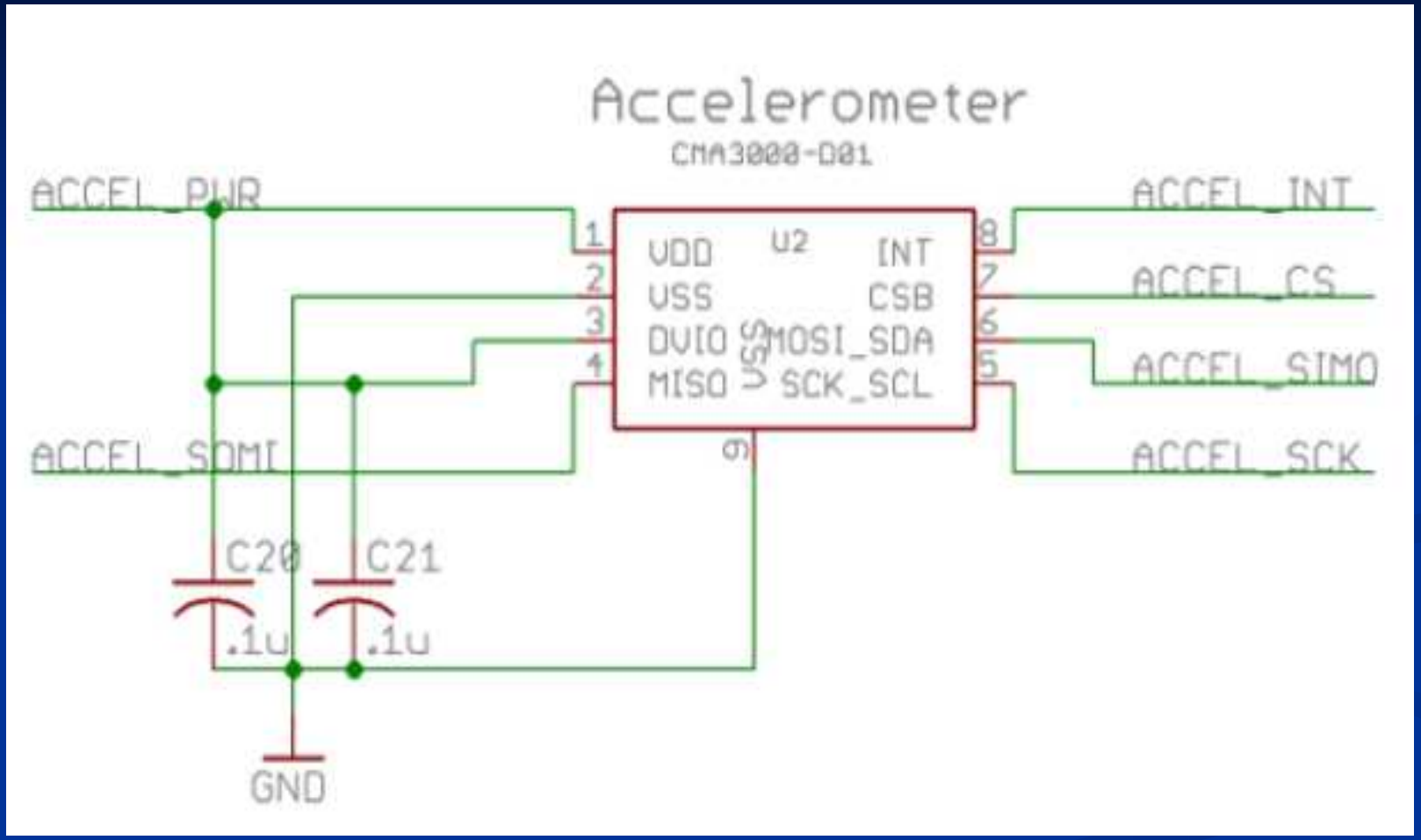
Пример. ЖКИ. SPI (USCI_B1)

Задача 4.

4.1. С какой частотой выставляются биты данных на SPI?

4.2. Почему не заработал LCD?

Акселерометр



Акселерометр

по схеме (по модулю)	по MSP430F5529
ACCEL_PWR (VDD, DVIO)	P3.6 / TB0.6
ACCEL_SOMI (MISO)	P3.4 / UCA0RXD / UCA0SOMI
ACCEL_INT (INT)	P2.5 / TA2.2
ACCEL_CS (CSB)	P3.5 / TB0.5
ACCEL_SIMO (MOSI_SDA)	P3.3 / UCA0TXD / UCA0SIMO
ACCEL_SCK (SCK_SCL)	P2.7 / UCB0STC / UCA0CLK

Акселерометр

- 3-координатный акселерометр с цифровым выходом CMA3000-D01
- Диапазон измерений задается программно (2g, 8g)
- Питание 1.7 — 3.6 В
- Интерфейс SPI или I^2C задается программно
- Частота отсчетов (10, 40, 100, 400 Гц) задается программно
- Режим сна — ток потребления 3 мкА
- Ток потребления при 10 отсчетах — 7 мкА, при 400 отсчетах — 70 мкА
- Максимальная частота 500 КГц

Акселерометр

- Разрешение 18 mg (при диапазоне 2g), 71mg (при диапазоне 8g)
- Чувствительность 56 точек / g (при 2g), 14 точек / g (при 8g)
- Режим обнаружения движения
- Режим обнаружения свободного падения

Акселерометр

- Режим измерения:
- Доступна фильтрация нижних частот
- 2g — 400 Гц, 100 Гц
- 8g — 400 Гц, 100 Гц, 40 Гц
- Прерывание при готовности новых данных
- Прерывание может быть отключено
- Флаг прерывания сбрасывается автоматически при чтении данных

Акселерометр

- Режим определения свободного падения:
- Доступна фильтрация нижних частот
- 2g — 400 Гц, 100 Гц
- 8g — 400 Гц, 100 Гц
- Прерывание при обнаружении свободного падения
- Пороги срабатывания (время, ускорение) могут изменяться

Акселерометр

- Режим определения движения:
- Фильтрация по полосе пропускания 1,3 — 3,8 Гц
- 8g — 10 Гц
- Прерывание при обнаружении движения
- Пороги срабатывания (время, ускорение) могут изменяться
- Может быть установлен режим перехода в режим измерения 400 Гц после обнаружения движения

Акселерометр

- Сброс формируется внутренней цепью. После сброса читаются калибровочные и конфигурационные данные. Бит PERR=0 регистра STATUS определяет успешность чтения
- Запись последовательности 02h, 0Ah, 04h в RSTR регистр выполняет программный сброс
- После инициализации по сбросу автоматически переходит в режим отключенного питания. Состояние регистров данных сохраняется
- Программно этот режим устанавливается битами MODE = 000b или 111b в CTRL регистре

Акселерометр

Регистр	Адрес	Назначение
WHO_AM_I	0h	R, идентификационный регистр
REVID	1h	R, версия ASIC
CTRL	2h	RW, управление
STATUS	3h	R, состояние
RSTR	4h	RW, регистр сброса
INT_STATUS	5h	R, состояние прерывания
DOUTX	6h	R, регистр данных канала X
DOUTY	7h	R, регистр данных канала Y
DOUTZ	8h	R, регистр данных канала Z
MDTHR	9h	RW, регистр порога ускорения режима детекции движения

Акселерометр

Регистр	Адрес	Назначение
MDFFTMR	Ah	RW, регистр порога времени для режимов детекции движения и свободного падения
FFTHR	Bh	RW, регистр порога ускорения режима детекции свободного падения
I2C_ADDR	Ch	R, адрес устройства для протокола I^2C

Акселерометр

Регистр	Биты	Поле	Назначение
CTRL	7	G_RANGE	Диапазон. 0 — 8g, 1 - 2g
	6	INT_LEVEL	Активный уровень сигнала прерывания: 0 - высокий, 1 - низкий
	5	MDET_EXIT	Переход в режим измерения после детекции движения
	4	I2C_DIS	Выбор интерфейса I^2C: 0 — разрешен, 1 - запрещен
	1-3	MODE[2..0]	Режим
	0	INT_DIS	Запрещение прерывания (1 - отключен)

Акселерометр

- Режим:
- 000 — отключено питание
- 001 — измерение, 100 Гц
- 010 — измерение, 400 Гц
- 011 — измерение, 40 Гц
- 100 — детекция движения, 10 Гц
- 101 — детекция свободного падения, 100 Гц
- 110 — детекция свободного падения, 400 Гц
- 111 — отключено питание

Акселерометр

Регистр	Биты	Поле	Назначение
STATUS	3	PORST	Флаг состояния сброса. Чтение всегда сбрасывает в 0
	0	PERR	Флаг ошибки четности EEPROM
RSTR	0-7	RSTR	Запись 02h, 0Ah, 04h выполняет сброс устройства
INT_STATUS	2	FF_DET	Флаг детекции свободного падения
	0-1	MDET[1..0]	Флаг детекции движения: 00 — нет, 01 - X , 10 - Y, 11 - Z

Акселерометр

- **Физические уровни бит (в mg) регистра данных**

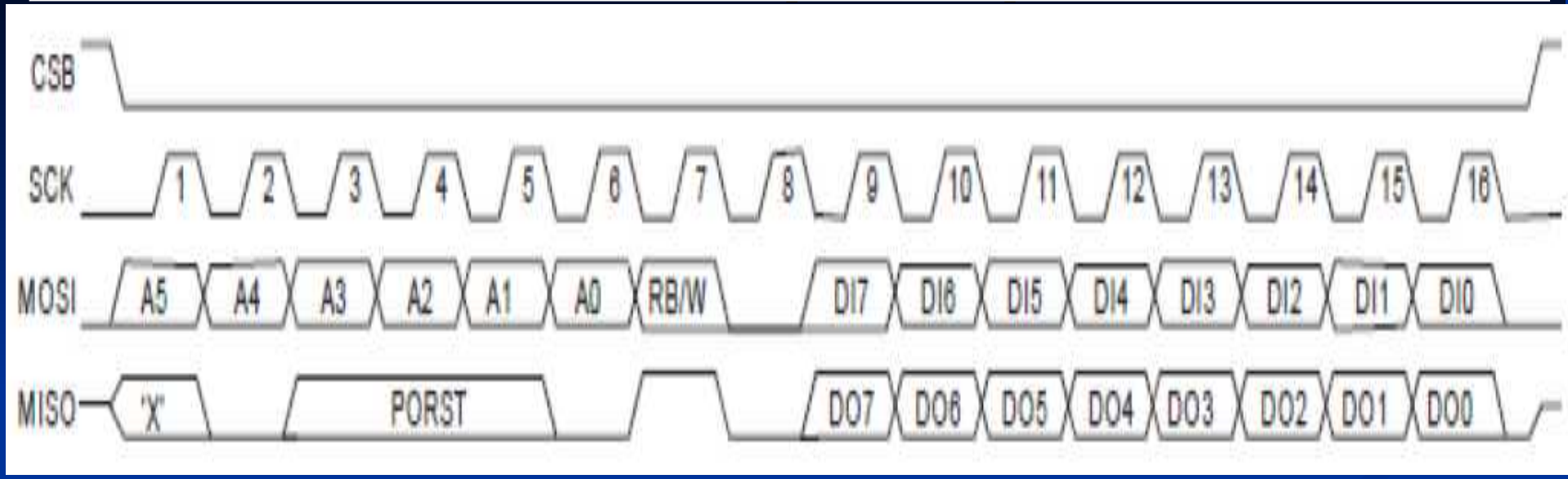
Range	G_RANGE	Output sample rate	B7	B6	B5	B4	B3	B2	B1	B0
2g	1	400 Hz, 100 Hz	s	1142	571	286	143	71	36	1/56 = 18 mg
2g	1	40 Hz, 10 Hz	s	4571	2286	1142	571	286	143	1/14 = 71 mg
8g	0	400 Hz, 100 Hz	s	4571	2286	1142	571	286	143	1/14 = 71 mg
8g	0	40 Hz, 10 Hz	s	4571	2286	1142	571	286	143	1/14 = 71 mg

s = sign bit

Акселерометр

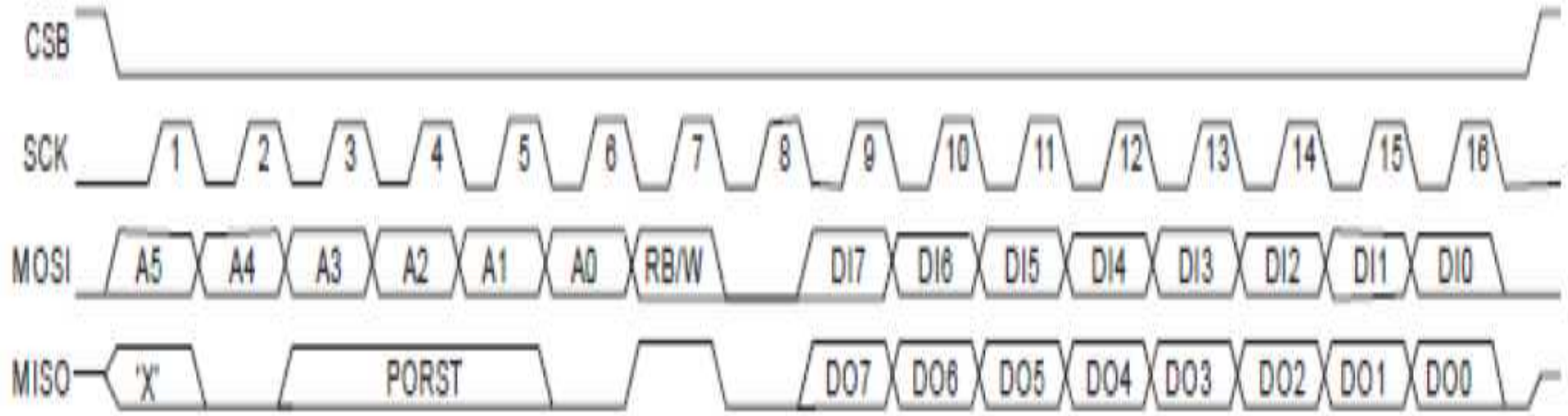
- Выбор интерфейса (SPI или I^2C) при помощи сигнала выбора кристалла
- I^2C может быть отключен программно
- Акселерометр всегда работает в ведомом (Slave) режиме
- Подключение 4-проводное

Акселерометр



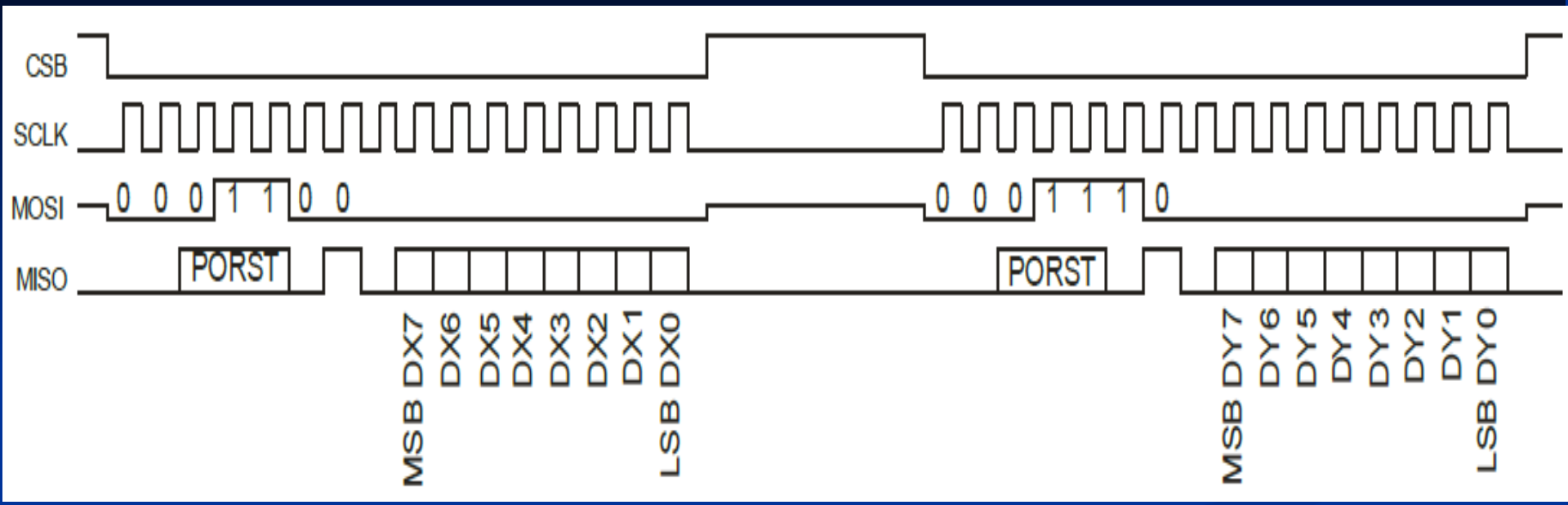
- *Фрейм содержит 2 байта (16 бит)*
- *Первый байт (первые 6 бит) содержит адрес регистра и тип операции (R/W, 7 бит), 8 бит = 0*
- *Второй байт содержит данные (при записи), и что угодно (при чтении)*
- *Данные заносятся в регистр по переднему фронту*

Акселерометр



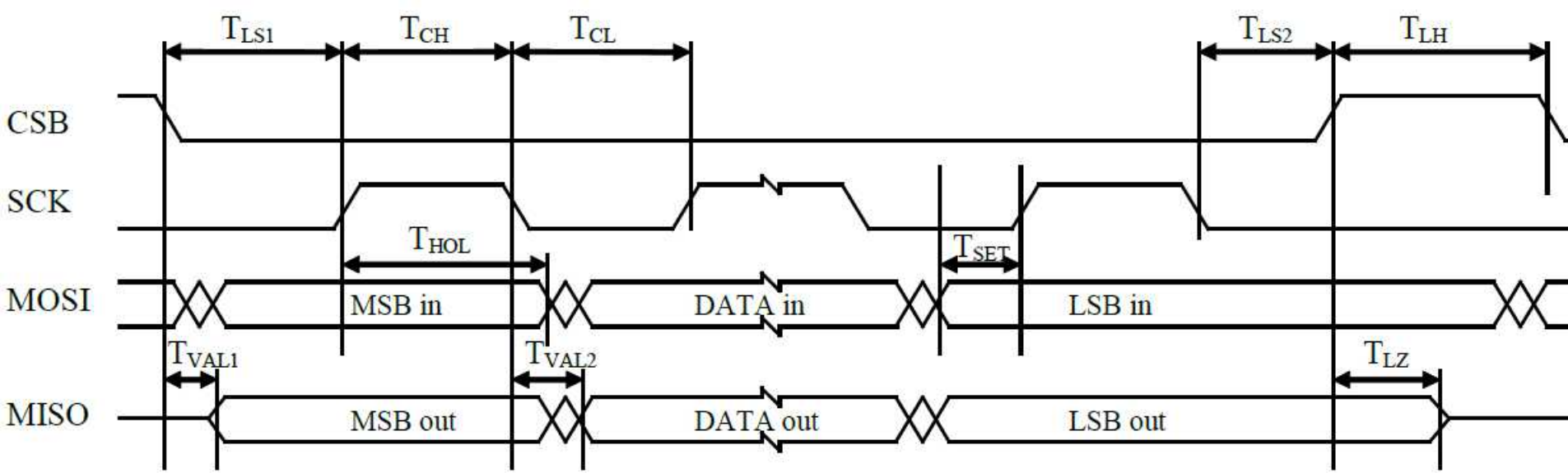
- *На линии MISO первый бит не определен, второй — 0, потом 3 бита статуса сброса, далее 010*
- *Второй байт содержит данные*
- *При высоком CSB, MISO в высокоимпедансном состоянии*
- *Данные выставляются на линию по заднему фронту, поэтому читать линию надо по переднему фронту*

Акселерометр



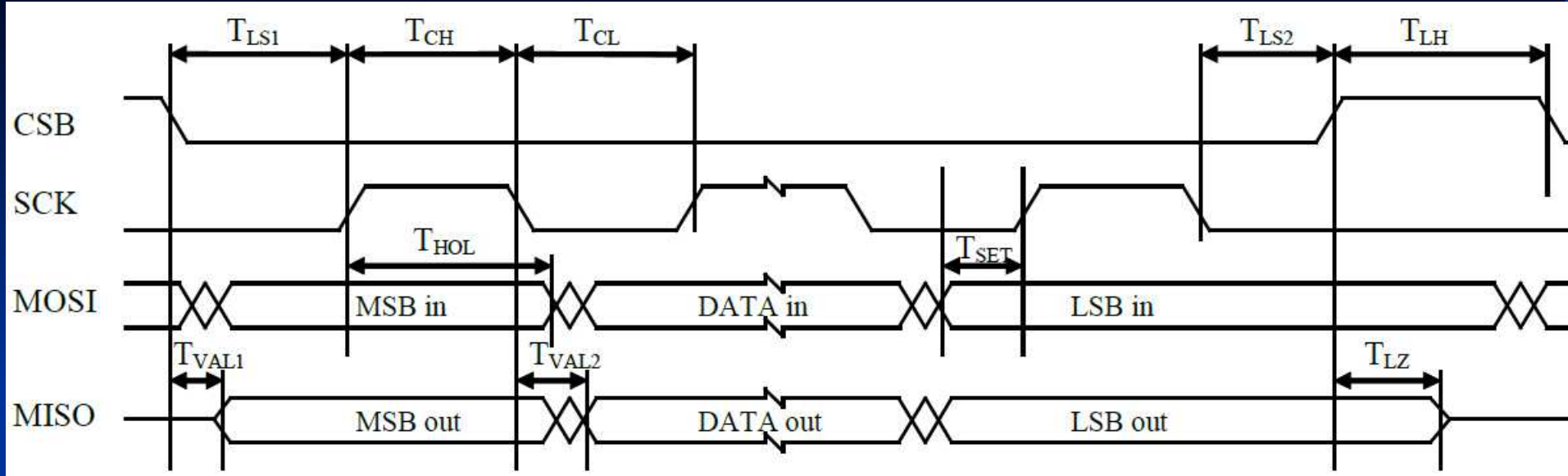
- *Пример операции чтения*

Акселерометр



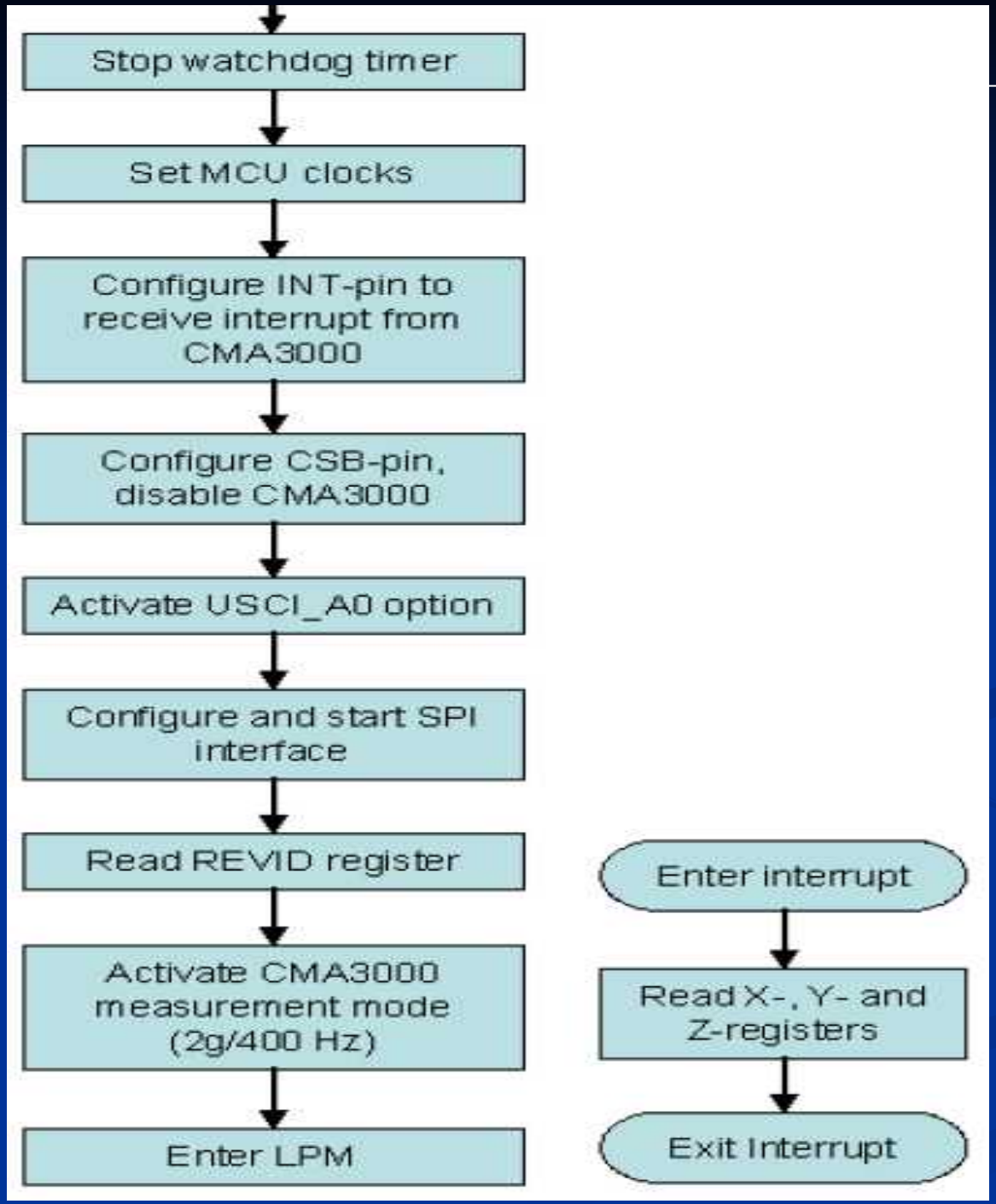
- T_{LS1} — от CSB до SCK мин 0,8 мкс
- T_{LS2} — от SCK до CSB мин 0,8 мкс
- T_{CL} — низкий SCK мин 0,8 мкс
- T_{CH} — высокий SCK мин 0,8 мкс
- T_{SET} — время установки данных (до SCK) мин 0,5 мкс
- T_{HOL} — время удержания данных (от SCK до изменения MOSI) мин 0,5 мкс

Акселерометр

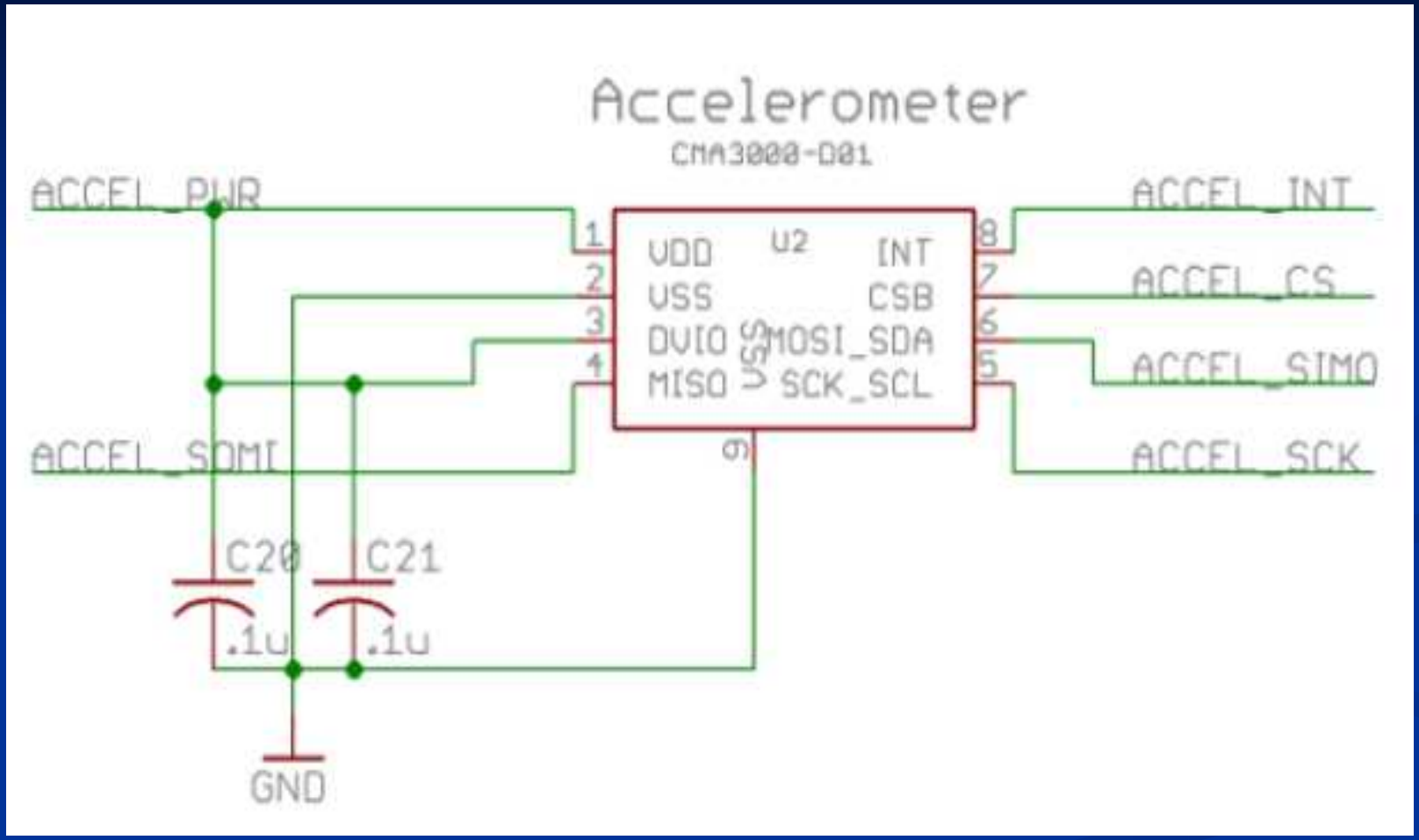


- T_{VAL1} — от CSB до стабилизации MISO макс 0,5 мкс
- T_{LZ} — от снятия CSB до высокоимпедансного MISO макс 0,5 мкс
- T_{VAL2} — от спада SCK до стабилизации MISO макс 0,75 мкс
- T_{LH} — задержка между SPI циклами (высокий CSB) мин 22 мкс

Акселерометр



Акселерометр



Акселерометр

по схеме (по модулю)	по MSP430F5529
ACCEL_PWR (VDD, DVIO)	P3.6 / TB0.6
ACCEL_SOMI (MISO)	P3.4 / UCA...
ACCEL_INT (INT)	P2.5 / TA2.2
ACCEL_CS (CSB)	P3.5 / TB0.5
ACCEL_SIMO (MOSI_SDA)	P3.3 / UCA0TXD / UCA0SIMO
ACCEL_SCK (SCK_SCL)	P2.7 / UCB0STC / UCA0CLK

Пример. SPI. Акселерометр

```
#include <msp430.h>

char cma3000_SPI(unsigned char byte1, unsigned char
byte2);

int main(void) {
WDTCTL = WDTPW | WDTHOLD;   // Stop watchdog timer
P1DIR |= BIT0;              // P1.0 (LED1) set as output
P8DIR |= (BIT1 | BIT2); // P8.1 & P8.2(LED2 & LED3)output
P1OUT &= ~BIT0;             // LED1 off
P8OUT &= ~(BIT1 | BIT2);// LED2 & LED3 off
P2DIR &= ~BIT5;             // P2.5 (cma3000 pin INT) input
P2OUT |= BIT5;// P2.5 (cma3000 pin INT) pull-up resistor
P2REN |= BIT5;// P2.5 (cma3000 pin INT) enable resistor
P2IE |= BIT5; // P2.5 (cma3000 pin INT) interrupt enable
// P2.5 (cma3000 pin INT) edge for interrupt: low-to-high
P2IES &= ~BIT5;
P2IFG &= ~BIT5;// P2.5 (cma3000 pin INT) clear int flag
```

Пример. SPI. Акселерометр

```
P3DIR |= BIT5;     // P3.5 (cma3000 pin CSB) set as output
// P3.5 (cma3000 pin CSB) set "1" is disable cma3000
P3OUT |= BIT5;
P2DIR |= BIT7;     // P2.7 (cma3000 pin SCK) set as output
P2SEL |= BIT7;     // device mode: P2.7 is UCA0CLK
// P3.3 & P3.6 (cma3000 pin MOSI, PWR) set as output
P3DIR |= (BIT3 | BIT6);
P3DIR &= ~BIT4;    // P3.4 (cma3000 pin MISO) set as input
// device mode: P3.3 - UCA0SIMO, P3.4 - UCA0SOMI
P3SEL |= (BIT3 | BIT4);
// P3.6 (cma3000 pin PWR) set "1" is power cma3000
P3OUT |= BIT6;

UCA0CTL1 |= UCSWRST; // Set SPI mode: reset logic is on
// Set SPI mode: master, MSB first, data latch on rising
UCA0CTL0 |= UCSYNC | UCMST | UCMSB | UCCKPH;
// Set SPI mode: SMCLK for clock, keep reset
UCA0CTL1 |= UCSWRST | UCSSEL__SMCLK;
```

Пример. SPI. Акселерометр

```
UCA0BR0 = 0x30;    // Set SPI mode: low byte division
UCA0BR1 = 0;       // Set SPI mode: high byte division
UCA0MCTL = 0;      // Set SPI mode: no modulation
UCA0CTL1 &= ~UCSWRST;// Reset SPI == Start SPI interface

// Start SPI transmit. cma3000 format:
// 1 = REVID register, 0 - read, 0 - predefined
// Second byte for read operation can be any
cma3000_SPI(0x4, 0);
__delay_cycles(1250); // ????

// SPI: 10 = CTRL register, 1 - write, 0 - predefined
// cma3000 CTRL register set: 2g, disable I2C, 400 Hz
cma3000_SPI(0xA, BIT7 | BIT4 | BIT2);
__delay_cycles(25000); // ????

// Enter LPM0, enable interrupts
__bis_SR_register(LPM0_bits + GIE);
__no_operation();              // For debugger
return 0; }
```

Пример. SPI. Акселерометр

```
char cma3000_SPI(unsigned char byte1, unsigned char
byte2) {
char indata;
// P3.5 (cma3000 pin CSB) set "0" is start SPI operation
P3OUT &= ~BIT5;
indata = UCA0RXBUF; // ??????
// Wait TXIFG == TXBUF is ready for new data
while (!(UCA0IFG & UCTXIFG)) ;
UCA0TXBUF = byte1; // Start SPI transmit. Send first byte
// Wait RXIFG == RXBUF have new data
while (!(UCA0IFG & UCRXIFG));
indata = UCA0RXBUF; // ??????
// Wait TXIFG == TXBUF is ready for new data
while (!(UCA0IFG & UCTXIFG));
UCA0TXBUF = byte2; // Start SPI transmit. Send second byte
// Wait RXIFG == RXBUF have new data
while (!(UCA0IFG & UCRXIFG));
```

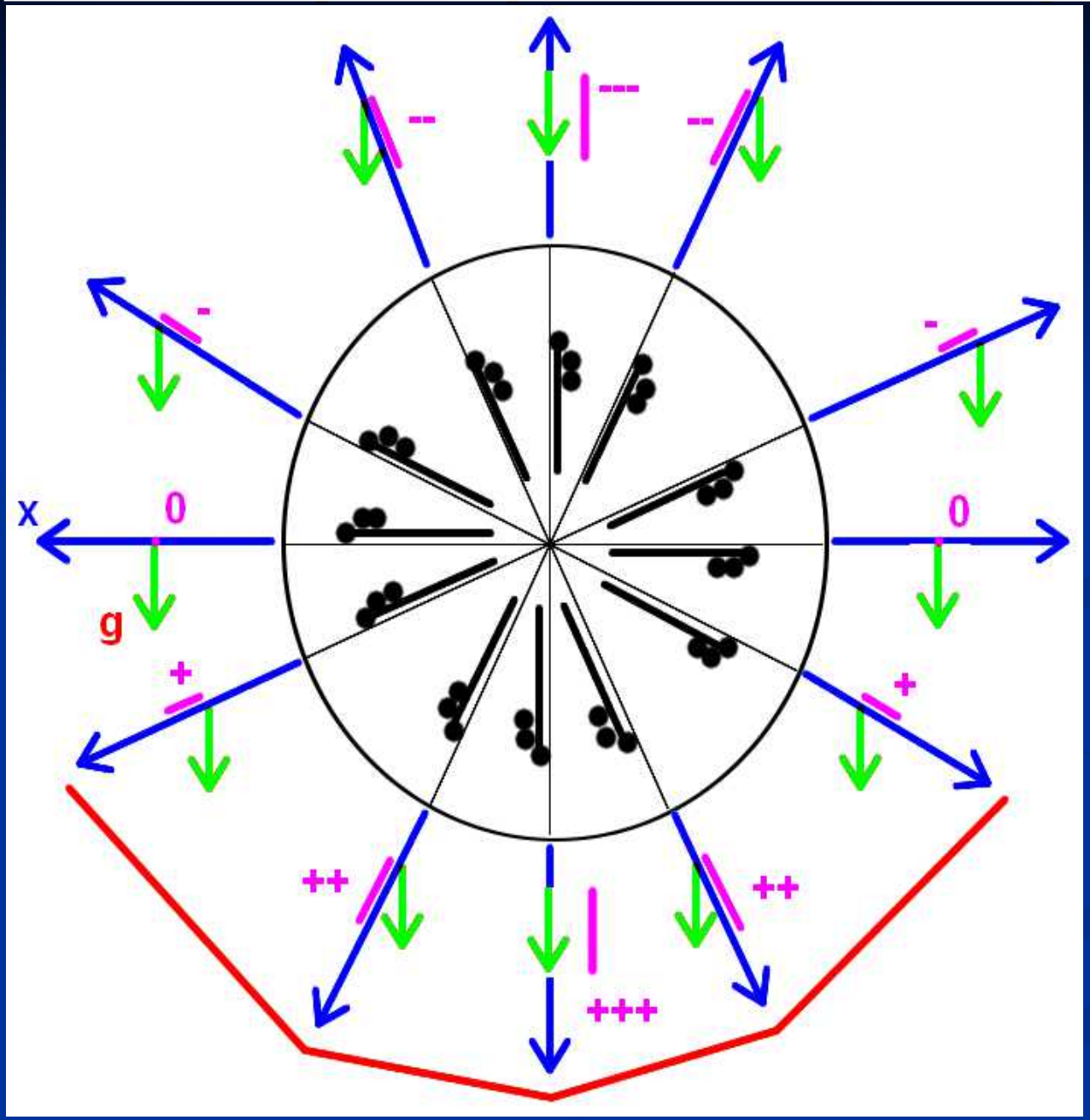
Пример. SPI. Акселерометр

```
// Read SPI data from accel. in 2 byte in read command
// ????? in write command
indata =UCA0RXBUF;
// Wait until USCI_A0 SPI interface is no longer busy
while (UCA0STAT & UCBUSY) ;
// P3.5 (cma3000 pin CSB) set "1" is stop SPI operation
P3OUT |= BIT5;
return indata;
}

// Port 2 interrupt service routine == cma3000 interrupt
#pragma vector=PORT2_VECTOR
__interrupt void PORT2_ISR(void)
{
 char dx, dy, dz;
 if (P2IN & BIT5)  {
 P1OUT &= ~BIT0;              // LED1 off
 P8OUT &= ~(BIT1 | BIT2);   // LED2 & LED3 off
```

Пример. SPI. Акселерометр

```
// Start SPI transmit. cma3000 format:
// 110 = DOUTX register, 0 - read, 0 - predefined
// Second byte for read operation can be any
dx = cma3000_SPI(0x18, 0);
__delay_cycles(1250); // ????
// SPI: 111 = DOUTY register, 0 - read, 0 - predefined
dy = cma3000_SPI(0x1C, 0);
__delay_cycles(1250); // ????
// SPI: 1000 = DOUTZ register, 0 - read, 0 - predefined
dz = cma3000_SPI(0x20, 0);
__delay_cycles(1250); // ????
// LED1 on if acceleration on X more than threshold
if (dx > 32) P1OUT |= BIT0;
// LED2 on if acceleration on Y more than threshold
if (dy > 32) P8OUT |= BIT1;
// LED3 on if acceleration on Z more than threshold
if (dz > 32) P8OUT |= BIT2;
P2IFG &= ~BIT5;          // reset interrupt flag
    }
}
```

Пример. SPI. Акселерометр



Пример. SPI. Акселерометр

Видео

05. Accelerometer_3

Что хотели получить

Пример. SPI. Акселерометр

Видео

05. Accelerometer_0

Что на самом деле вышло

Пример. SPI. Акселерометр

Задача 4.

4.3. Какая тактовая частота поступает на CLK вход акселерометра (какая частота передачи бит)?

4.4. Почему плата не реагирует на перемещение?

4.5. Зачем __delay_cycle?

4.6. Зачем при передаче данных в акселерометр выполняется чтение данных?