KS480 LCD controller/driver

480 segment, KickStart™ protocol, intelligent LCD front panel controller and driver, with SPI and i2c interface

GPEG International Ltd

Revision History

Date	Rev#	Description	Page

Contents

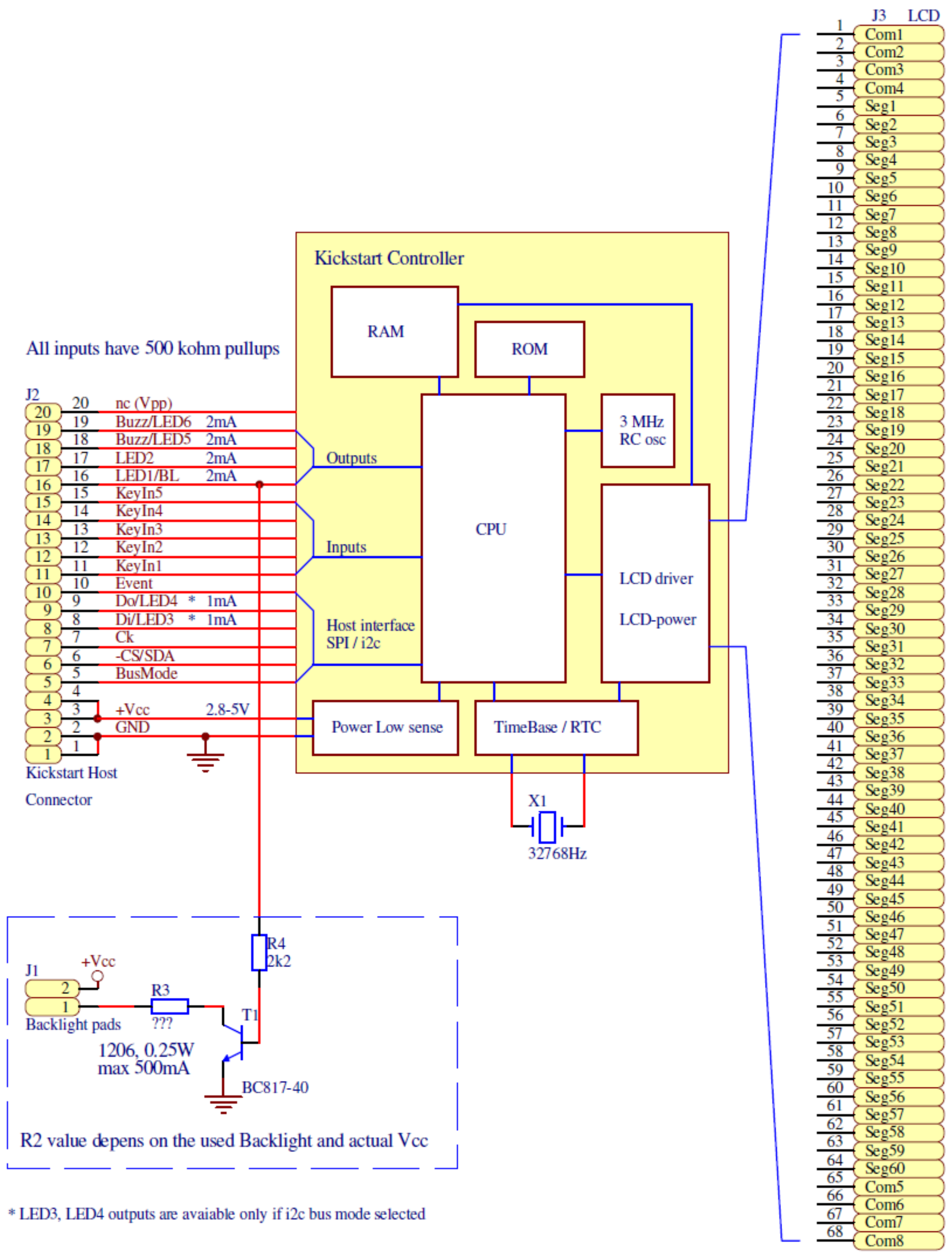
Features

- Selectable duty cycle: 1/4, 1/5, 1/6, 1/8
- 60 segment x 8 (max) common
- 480 max segment count
- SPI and I2C interface
- Operating Voltage 2.7 - 5V
- Low power mode (sleep 30uA) with LCD on and keypress monitoring, RTC running, command wakeup
- Ultra-low power mode (deepsleep <1uA), only command wakeup
- Built-in realtime clock
- Built-in 7/14/16 segment character font table (HexaDec + ASCII)
- Built-in Bar-converter
- Built-in oscillator and external 32.768kHZ crystal
- Built-in symbol-animation (disk rotation, etc), Blink symbol or text
- Text string and bargraph printing
- Text scrolling (max 255 char)
- Intuitive and fast to implement high level command set
- Backlight control (On/Off)
- Buzzer output with variable frequencies
- I/O for external LEDs
- RealTime Clock with auto printing, event generation on every sec
- Key inputs with event generation on keypresses
- Standard die package or chip-on-film (COF) available

General Description

The SK480 is an intelligent LCD controller driver with capability up-to 480 segments. The device can be configured to operate in any LCD system with its programmable duty cycle. Interface via SPI and i2C is supported, along with a simple, high level command structure, significantly reducing programming time and errors. The SK480 is used in conjunction with the kickstart on-line tool (www.gpeg.kickstart.com) eliminating the need for the traditional LCD pin lookup table, time consuming font tables and power hungry animated elements.

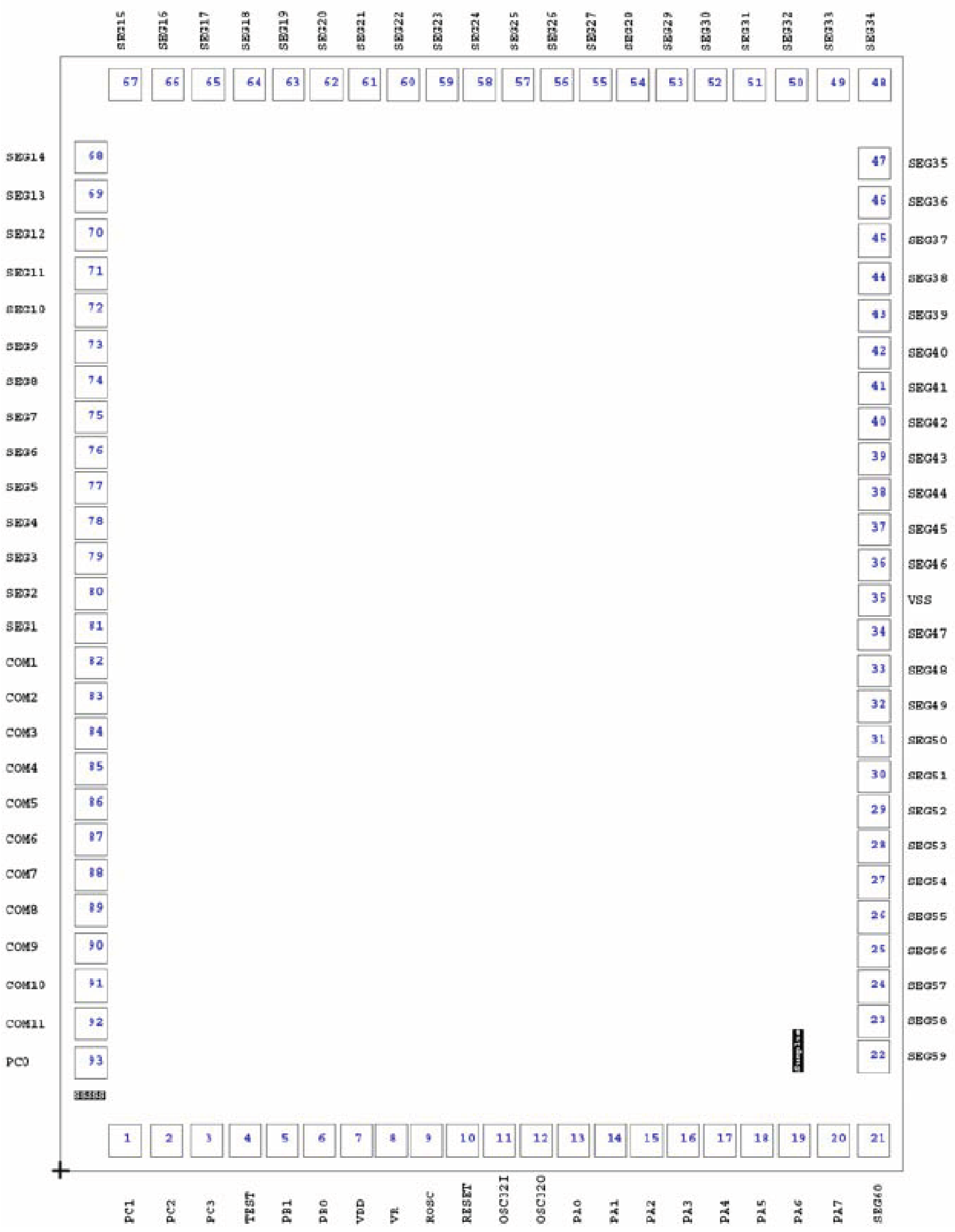
Block Diagram of COF version



* LED3, LED4 outputs are avaiable only if i2c bus mode selected

Notes:

Pad assignment



Chip Size: 2740x 3590 um

Note: the IC substrate should be connected to Vss on the PCB layout

Absolute maximum ratings

Characteristics	Symbol	Ratings
DC supply voltage	VDD	<6.0V
Input voltage range	Vin	-0.5V to VDD +0.5V
Operating Temperature Range	Ta	0°C to +60°C
Storage Temperature range	Tsto	-50°C to +150°C

Note: stress beyond these given may cause operational errors and damage to the device. For normal operating conditions see DC/AC Characteristics.

DC Characteristics

DC Characteristics (VDD=3.0V, Ta=25°C)

Characteristics	Symbol	Limit			Unit	Test Condition
		Min.	Typ.	Max.		
Operating Voltage	VDD	2.6	-	3.6	V	For 2 cell battery
Operating Current	Iop		1.3		mA	3V, no load
Deepsleep Current	Ideepsl			1.0	µA	VDD=3V, display OFF
Sleep Current	Isleep			20.0	µA	VDD=3V, display ON, 32768 XTL ON
Input High level	Vih	0.7 VDD				
Input Low level	Vil			0.2 VDD		

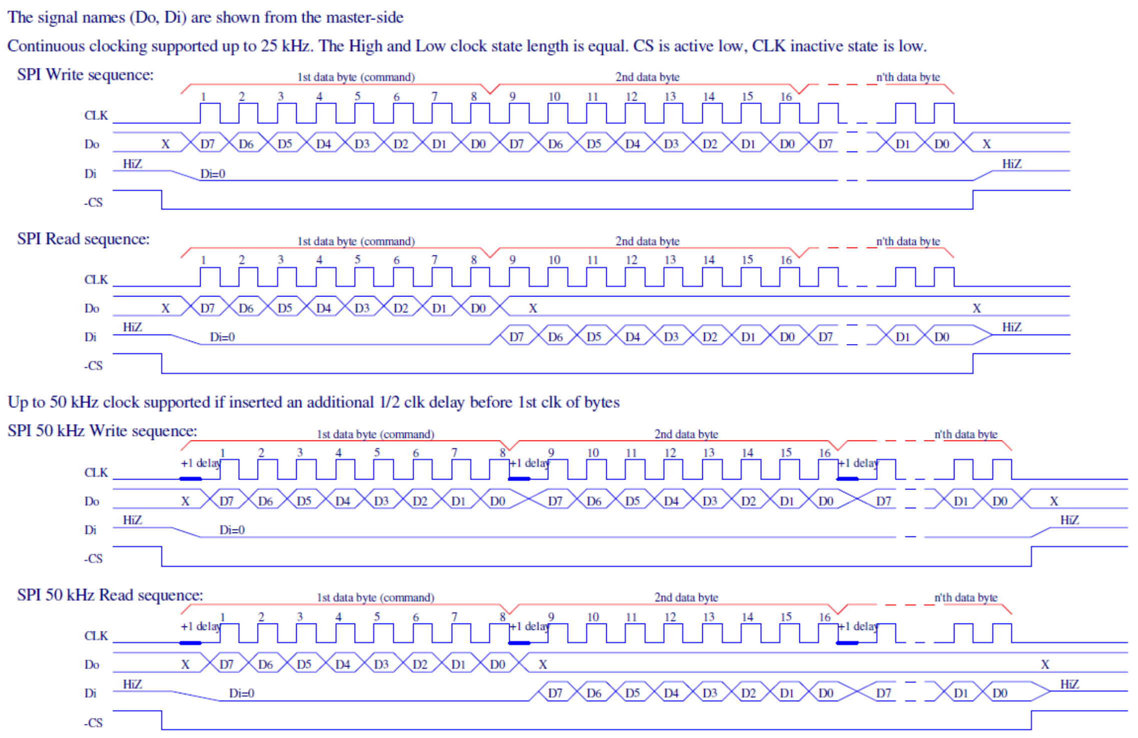
DC Characteristics (VDD=5V, Ta=25°C)

Characteristics	Symbol	Limit			Unit	Test Condition
		Min.	Typ.	Max.		
Operating Voltage	VDD	3.6	-	5.5	V	For 2 cell battery
Operating Current	Iop		3.1		mA	@ 5V, no load
Deepsleep Current	Ideepsl			1.0	µA	VDD=5V, display OFF
Sleep Current	Isleep			30.0	µA	VDD=5V, display ON, 32768 XTL ON
Input High level	Vih	0.7 VDD				
Input Low level	Vil			0.2 VDD		

AC Characteristics

Symbol	Parameter	Test condition		Min	Typ	Max	Unit
		VDD	Condition				
Fclk	i2c, SPI ck	-	-	10	50	50	kHz
TsdaL	i2c-start			10			us
CSL	SPI			10			us

4 Wire serial SPI


The signal names (Do, Di) are shown from the master-side

Continuous clocking supported up to 25 kHz. The High and Low clock state length is equal. CS is active low, CLK inactive state is low.

SPI Write sequence:

SPI Read sequence:

Up to 50 kHz clock supported if inserted an additional 1/2 clk delay before 1st clk of bytes

SPI 50 kHz Write sequence:

SPI 50 kHz Read sequence:

Notes:

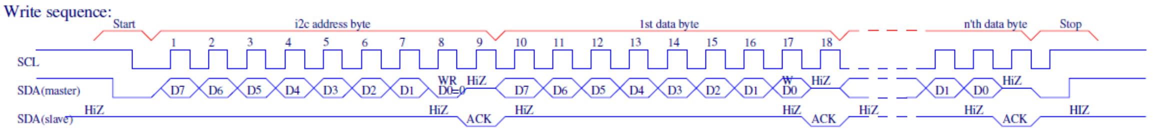
- CS monitored while command pending. The Host can terminate the command sequence by removing CS.
- While command receiving, Kickstart stops executing all other internal functions.
- If too slow clock used or clock stopped within command, Kickstart will WDT reset after 1 sec to clear power-on state. Each bytes should be finished under 1 sec to avoid watchdog reset
- The data should be stable while the clock is high
- The Host is always the master, Kickstart is the slave on the bus.

IIC Interface

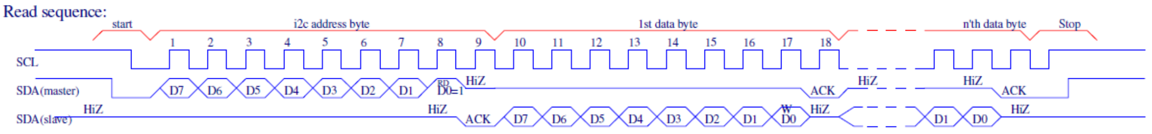
SDA(master) and SDA(slave) are phisically one line, separated only for functional clarity in the diagrams below

Continuous clocking supported up to 25 kHz. The High and Low clock state length is equal.

i2c Write sequence:

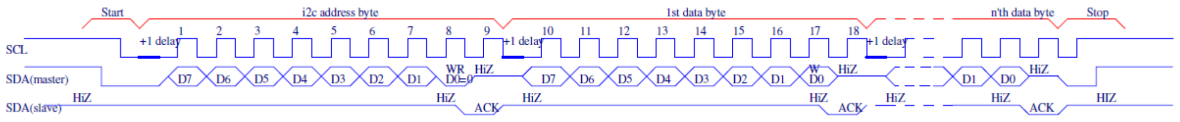


i2c Read sequence:

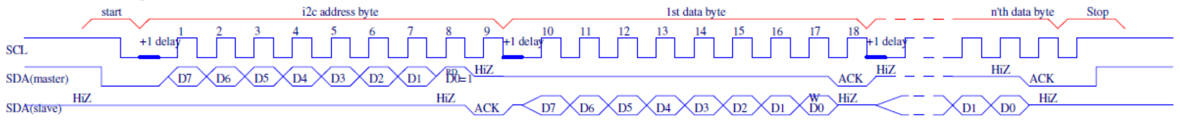


Up to 50 kHz clock supported if inserted an additional 1/2 clk delay before 1st clk of bytes

i2c 50 kHz Write sequence:



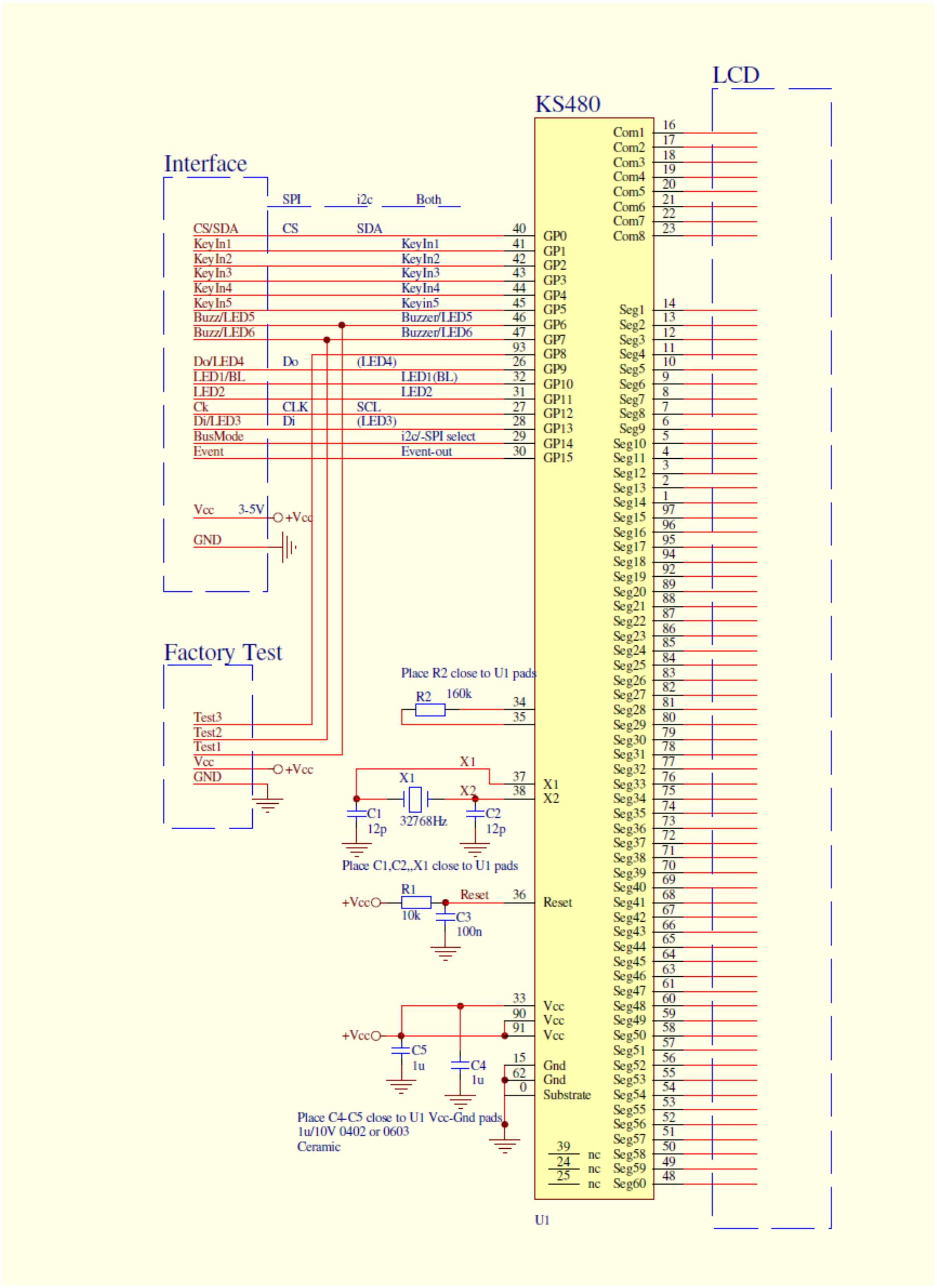
i2c 50 kHz Read sequence:



Notes:

- Repeated start condition not supported!.
- Stop condition not detected while command pending.
  The Host always should provide all clocks for the entire command sequence.
- While command receiving, Kickstart stops executing all other internal functions.
- If clock stopped, Kickstart will WDT reset after 1 sec to clear power-on state.
- The Host is always the master, Kickstart is the slave on the bus.
- The data should be stable while the clock is high (except the start/stop conditions)

Application Circuit



Command Set Summary

Kickstart Host interface protocol, V0.4, PRELIMINARY

Kickstart supports i2c (2 wires) and SPI (4 wires) bus modes for Host interface.
Interface select: by "BusMode" pin in host connector. 1=i2c, 0=SPI.
Max clock speed: 50 kHz, see timing diagrams for details.
Kickstart suspends all other internal processes while receiving the command.
For preventing Kickstart from hanging if the host stops the interface clock, 1 sec Watchdog timeout used.
The WDT restarts if an byte received. If the byte not completed within 1 sec, causes WDT reset to clear PowerOn state.
In this case all data cleared, the Host should initialize Kickstart again.

At the command start, the Host should check Kickstart ready/busy state, see below.
Kickstart can be busy either:

- After command received until executes it,
- during 2 Hz timer event handling (RTC, animations...).

i2c description

Addresses:

-Kickstart i2c Write address = 0xCA	(i2c: upper 7 bit = 0x65, W: bit0=0)
-Kickstart i2c Read address = 0xCB	(i2c: upper 7 bit = 0x65, R: bit0=1)

W: Host writes command data, R: Host reads status data

Write command:
- W: StartCond
- W: Address-W byte, check ACK ok, else Kickstart busy
- W: CMD byte
- W: Length of Payload byte
- W: Payload[] bytes
- W: Checksum byte: 0x80 + (8 bit sum of CMD, Length, Payload[] bytes)
- W: StopCond

All received bytes will be acknowledged (ACK) by Kickstart.
If Address byte not ACK-ed, then Kickstart is busy. Send StopCond and repeat the command until ACK ok.

Read status:
- W: StartCond
- W: Address-R byte. Check ACK ok, else Kickstart busy: stop, repeat the read.
- R: Payload[] bytes (fix 8 bytes: defined in command details section: Status read)
- R: Checksum byte: 0x80 + (8 bit sum Payload[] bytes)
- W: StopCond

Address will be acknowledged (ACK).
The host may, but must not ACK the Payload and Chksum bytes.
If Address byte not ACK, then Kickstart is busy. Send StopCond and repeat the command until ACK ok.
Repeated StartCond not allowed!

StopCond not detected during the command! The host always should complete the CMD (send all clocks),
else timeout reset happens after 1 sec.

If more devices are on this i2c bus:
The Host should not use higher than Kickstart-max clock speed in this bus,
else StartCond/address mistakes occours!
All SDA falling edge will cause Kickstart Wakeup-from-sleep for detect StartCond.
If it is StartCond, Kickstart remains in runmode untill the i2c address byte received (Icc=1.3mA)

SPI Description

W: Host writes command data, R: Host reads status data

Write command:
- CS=0: Startes the CMD
- wait for Do goes low, else Kickstart busy (remove CS and repeat)
- W: CMD byte,
- W: Length of Payload byte
- W: Payload[] bytes
- W: Checksum byte: 0x80 + (8 bit sum of CMD, Length, Payload[] bytes)
- CS=1: Stops the CMD

Read command and parameters (payload):
- CS=0: Startes the CMD
- Wait for Do goes low, else Kickstart busy (remove CS and repeat)
- W: CMD byte (Cmd_Statread)
- R: Payload bytes (fix 8 bytes: defined in command details section: Status read)
- R: Checksum byte: 0x80 + (8 bit sum Payload[] bytes)
- CS=1: Stops the CMD

The CS pin is monitored continuously, the Host can interrupt/terminate the pending command without complete it.
If the command was executable, the partial command will be ignored.
If it was direct-RAM command, the partialy stored data remains updated.

Command table

direct memory commands, codes: 0-15, executed in cmd thread (IT) :

code	name	description
0	Cmd_LCDRAM_W	write directly to LCD RAM
1	Cmd_Seg_Tbl	load LCD Segment table
2	Cmd_Obj_Tbl	load LCD Object table
3	Cmd_ScrollTxt	load text for autoscroll print, 1-255 char

normal commands, codes: 16-127, executed in main thread:
(command-codes may change in next doc version)

16	Cmd_LCD_cnf	set commons of glass, bias, object count at startup
17	--	
18	Cmd_Statread	read status bytes. Only for SPI mode! i2c: use i2c Read Address
19	Cmd_Reset	Reset Kicstart if received sucessfuly
20	Cmd_LCD_W	write 1 byte to LCD RAM
21	Cmd_Obj_W	write Obj (1-16 seg) without conversion (direct segment pattern)
22	Cmd_Set_Seg	Set 1 segment (On)
23	Cmd_Clr_Seg	Clr 1 segment (Off)
24	Cmd_7seg	print 1 character converted using built-in 7 segment font
25	Cmd_Bar	Print 1 Bar, converted using built-in Bar converter
26	--	
27	Cmd_Buzz	turn on/off buzzer, set output-mode, frequency
28	Cmd_LED	set LED1-6 On/Off (incl backlight)
29	Cmd_Attr	Set attribute of object(s): Static, Blink, AnimateFW, AnimateBW
30	Cmd_Contrast	Set LCD contrast level 0-15
31	Cmd_LCD_OnOff	turn On/Off the LCD driver (not affects the LCD-RAM content)
32	Cmd_RTC	set RTC mode, date/time, events, auto-print
33	Cmd_Sleep	set Kickstart power mode (Run/Sleep/Deepsleep)
34	Cmd_14seg	print 1 character converted using built-in 14 segment font
35	Cmd_16seg	print 1 character converted using built-in 16 segment font
36	Cmd_Shift_L	Shifts a group of objects (string) left by one object. The new obj is cleared
37	Cmd_Shift_R	Shifts a group of objects (string) right by one object. The new obj is cleared
38	Cmd_7seg_Str	print 1-30 character string converted using built-in 7 segment font
39	Cmd_Bar_Str	print 1-30 bars (bargraph) converted using built-in Bar converter
40	Cmd_14seg_Str	print 1-30 character string converted using built-in 7 segment font
41	Cmd_16seg_Str	print 1-30 character string converted using built-in 7 segment font
42	Cmd_Scroll2Hz	ctrl autoscroll text in window, stored by Cmd_ScrollTxt
43	Cmd_TestLED	at startup, test LEDs (error LEDs) enabled. Turns on/off test-LED functions

Command descriptions

Usage and features:
After power on, the LCD-tables should be downloaded for the actual LCD glass layout.
This tables are given by GPEG.
The size of tables depends on segment and object count.
Max 480 physical segments and 127 objects supported. An object may have 1-16 segments.
(max 510 logical segments possible, allows to create virtual objects using physical segments)
All command arguments in the descriptions below are unsigned bytes (8bit).

The generic command form (bytes): command, length, payload[length], checksum
exept Cmd_Seg_Tbl, where the length byte is payloadlength/4. (allows max 1020 bytes at once)

Kickstart **Standard objects** with internal conversions:

- 7 segment hexadecimal and some alpha chars
  (call: 0-15: 0123456789AbcdEF, 16-31 special, 32-127 ASCII where possible)
- 14 segment alphanumeric character
  (call: 0-15 as 7seg, 16-31 special, 32-127 ASCII, some exeptions due to 14seg)
- 16 segment alphanumeric character (call: 0-15 as 7seg, 16-31 special, 32-127 ASCII)
- (code 16-31: all above charsets have some special pattern, useful for animated symbols)
- - 1-16 segment bar (call: 0-16)

Kickstart **Standard object groups** with internal conversions:

- String (1 to 30 objects (7/14/16 segment characters, or bargraph)
- Date (RTC), 00:00:00 (YY:MM:DD) (the order of year/month/day determined by the Object table)
- Time (RTC), 00:00:00 (hh:mm:ss, 24h format)

Command details, argument list:

Status read :
This is a special command, the only one which reads data from Kickstart, and differs in SPI and i2c mode.
-i2c: write i2c Read Address (= 0xCB), then read 8 staus bytes and the checksum.
-SPI: write Cmd_Statread (= 18), then read 8 status bytes and the checksum.
the 8 status bytes:

0. Kickstart version
1. Keys (bit4..0: is set if Keyin5..1 pressed since last Status read)
2. Year (RTC year 1-99 since 2000 if RTC enabled)
3. Month (RTC month 1-12 if RTC enabled)
4. Day (RTC day of month 1-31 if RTC enabled)
5. DoW (RTC day of week, 1-7)
6. Hour (RTC hours 0-23 if RTC enabled)
7. Minute (RTC minutes 0-59 if RTC enabled)
8. Second (RTC month 0-59 if RTC enabled)

(RTC: month lengths and leap years handled correctly)

Cmd_LCDRAM_W: write directly to LCD RAM without using segment lookup table (useful for graphic area)
Executed in command thread, interrupt level.
Fastest access, writes LCD RAM byte by byte as they received in the command.
No internal processing performed after the command done.
syntax: 0, length, address, lcd_data[datalength], chksum

- length: datalength+1, 2-64, the byte count for transfer to LCD RAM, +1 the address byte.
- address: 0-63, the start address in LCD RAM (where to lcd_data[0] will be copied).
- lcd_data[]: segment pattern bytes

This command is useful if the user needs faster control over an continuous LCD RAM area,
mainly if an graphic area is defined.

note1:
LCD RAM is 64 bytes, all bits are associated with one LCD segment.
The Segment mapping will be attached later.

note2:
If internal 2 Hz periodic function (animation, scroll, blink) is enabled for an object,
which uses same LCD RAM bytes, then it will overwrite the bytes stored by this command if
the command interrupted just an pending 2 Hz event.

Cmd_Seg_Tbl: Load LCD Segment table at startup. Always the first thing if the Segment table used
Max 480 physical supported. Logical segments max 510, this allows to form some virtual objects.
Executed in command thread, interrupt level.
syntax: 1, length, data[datalength], chksum

- length = datalength/4, allows max 1020 bytes to transfer at once
- data[] given by GPEG

Cmd_Obj_Tbl: Load LCD Object table at startup. Use always after Cmd_Seg_Tbl if Objects used.
Executed in command thread, interrupt level.
syntax: 2, length, data[length], chksum

- length =
- data[] given by GPEG

Cmd_ScrollTxt: Load text for autoscroll printing, 1-255 char
Executed in command thread, interrupt level.
syntax: 3, length, text[length], chksum

- length: byte count of text
- text[]: string of Kickstart special chars (code 0-31) or ASCII characters (code 32-127).

note: may Start/Stop scrolling with Cmd_Scroll2Hz command after the text loaded

Cmd_LCD_cnf: set Duty (Commons of glass), Bias, Objcount at startup
syntax: 16, 3, Duty, Bias, Objcount, chksum

- Duty: 0=1/4, 1=1/5, 2=1/6, 3=1/8
- Bias: 1=1/4, 2=1/5, recommended=1/4, others prohibited!
- Objcount: 0-128, the count of objects in the Object table

Cmd_Statread: read status bytes in SPI mode, see details above in "Status read" session

Cmd_Reset: Reset Kicstart to power on state if received sucess
Syntax: 19, 0, chksum

Cmd_LCD_W: write 1 byte (may be change later to more) to the LCD RAM after the command received.
The main difference from the Cmd_LCDRAM_W (0):
Executed in the main thread after the command received, not in interrupt level.
Therefore slower, but have not conflicts with the 2 Hz events if an object refers to the same LCD RAM bytes.
syntax: 20, 2, address, lcd_byte, chksum
address: 0-63, the address in LCD RAM (where to lcd_byte will be copied)
lcd_byte: segment pattern byte
note: LCD RAM is 64 bytes, all bits associated to one LCD segment.
The Segment mapping will be attached later.
(maybe later change to: 20, length+1, address, lcd_data[length], chksum)

Cmd_Obj_W: write one Object without conversion (seg15..0 of the object)
syntax: 21, 3, ObjNum, SegmentBitsL, SegmentBitsH, chksum
ObjNum: ordinal number of target object (0-127)
SegmentBitsL: pattern for segment7..0
SegmentBitsH: pattern for segment15..8
note: always send 16 pattern bits, regardless of the segment count of the object, (padd with 0).

Cmd_Set_Seg: set 1 segment (seg On)
syntax: 22, 2, SegNumL, SegNumH, chksum
SegNumL, SegNumH: 2 byte ordinal number of the target segment (0-510)

Cmd_Clr_Seg: clear 1 segment (seg Off)
syntax: 23, 2, SegNumL, SegNumH, chksum
SegNumL, SegNumH: 2 byte ordinal number of the target segment (0-510)

Cmd_7seg: print one character to one 7seg digit
syntax: 24, 2, ObjNum, charcode, chksum
ObjNum: ordinal number of target object (0-127)
charcode: 0-15: 0123456789AbcdEF, 16-31 special, 32-127 ASCII where possible
Cmd_Bar: print Bar to LCD (1-16 seg)
syntax: 25, 2, ObjNum, Value, chksum
ObjNum: ordinal number of target object (0-127)
Value: 0-16, the segment count to set

Cmd_Buzz: turn on/off buzzer, set freq, time
syntax: 27, 3, Mode, Divider, Time, chksum
Mode: 0=Buzzer-Off, Mode<1..0>: HostPin19..18=Buzzer out (allows single out or bridge)
Divider: f=23438/(divider+1)
Time: 0=infinite, 1-255=duration in 0.5s increments. (not yet supported!)
Note:
The buzzer operates only in RUN mode, do not set Kickstart to Sleep while the buzzer is on.
HostPin19..18 are shared with LED6..5. If buzzer off, they are for LEDs. Buzzer overides LEDs.
The two pins may set individually by Mode<1..0> bits. If both = 0, buzzer off
!!! Time not yet supported!
Planned: 0.5s periodic beep, enable by Mode<7> bit

Cmd_LED: set LEDs On/Off (also the backlight)
syntax: 28, 1, LEDbits, chksum
LEDbits<5..0>: turns on/off the corresponding LED6..LED1 outputs.
Note1: If an shared LEDx output is configured to other usage (Di,Do,Buzzer), this output will not affected.
Note2: LED1 is the default backlight control output (transistor on FPC, max 400mA).

Cmd_Attr: Set attribute of object
syntax: 29, 3, ObjNum, Attr, Objcount, chksum
-ObjNum: ordinal number of target object (0-127)
-Attr<7..6>:
00=Static,
01=Blink (0.5s on, 0.5s off),
10=Animate Forward(2 Hz),
11=Animate Backward (2 Hz)
<5..0> not used
-Objcount: number of objects to set (objnum incremented)
Note: Animation (bit rotation) is performed by rotate left/right the pre-filled segment pattern of the object. Use Cmd_Obj_W for preset the pattern.
After Blink sets back to Static, reprint the object with the default pattern (else may be invisible)

Cmd_Contrast: Set LCD contrast level 0-15
syntax: 30, 1, Contrast, chksum
Contrast: 0-15, 0 = max (Vlcd = Vcc, default), 15 = min (Vlcd = 0.7 * Vcc)

Cmd_LCD_OnOff: LCD On/Off
syntax: 31, 1, LCDmode, chksum
LCDmode: 1=On, 0=Off
Note: If sleepmode used, and Keypress wakeup needed, do not turn off LCD, else Keyin5..1 can not wakeup Kickstart from sleep.
DeepSleep mode not affected (no key wakeup from deepsleep)

Cmd_RTC: set RTC time, events, autoprint
syntax:
32, 10, RTCconfig, Year, Month, Day, DoW, Hour, Minute, Second, Date_Obj, Time_Obj
-RTCconfig<7..0>: RTC configuration bits:
<7>: RTC_On, turns On/Off the entire RTC system
<6>: RTC_event_On, enable/disable event out pin. If On, sets the Event pin at every seconds. Host may read new data using Cmd_Statread.
The Event pin remains set until an Cmd_Statread received.
Note the RTC and keypress shares the same Event pin.
<5>: RTC_print_date, enable/disable auto print Date
<4>: RTC_print_time, enable/disable auto print Time
<3..0>: reserved, =0
-Year: 1-99 = 2001-2099
If Year=0, Kickstart ignores Date/Time bytes, only sets the RTCconfig
-Month: 1-12
-Day: 1-31 (day of month)
-DoW: 1-7 (day of week)
-Hour: 0-23
-Minute: 0-59
-Second: 0-59
-Date_Obj: 0-122, the object number of year upper digit, =0 if RTC_print_date=0
-Time_Obj: 0-122, the object number of hour upper digit, =0 if RTC_print_time=0

Auto print date/time details: (now 7seg and 14seg supported, 16seg not!)
Date:
If RTC_print_date=1, Kickstart will print the date to 6 digits. "YYMMDD"
The digit identified by Date_Obj will be the year teens, the next the year ones, month teens, month ones, day teens, day ones.
The digits will be reprinted only after this command, and if the value changed by the RTC. (eg. will not be reprinted every sec)
Of course the order on the glass can be any, but in the Object table must be defined as an continuos string.
The separators (colon, dot...) are not handled by the auto print date, and they must be defined outside of this string.

Day of week handled, but not auto printed.

- Time:
  If RTC_print_time=1, Kickstart will print the time to 6 digits.
  Format "hhmmss"
  The digit identified by Time_Obj will be the hour teens, the next the hour ones, minutes teens, minutes ones, seconds teens, seconds ones.
  The hours supports only 24-hours format (eg. no 12h AM/PM)
  The digits will be reprinted only after this command, and if the value changed by the RTC. (eg. sec digits reprinted every sec, the others only if changed)
  The separators (colon,dot...) are not handled by the auto print time, and they must be defined outside of this string.
  If blinking colon or dot required for separator, then it should be defined as an individual object, and set it to blinking using Cmd_Attr command.

The Event pin is useful for the host to detect if the clock changed.
Note the Keypresses also activates the Event pin. Use the Status read command to determine the reason of the event. The Event pin remains set until an Status read command received.

Do not enable auto print date/time before the Segments and Object tables loaded.
If Year=0, Kickstart ignores Date/Time bytes, only sets the RTCconfig,
this allows set printing on/off without set the time.
The full RTC is read by the Cmd_Statread.

Cmd_Sleep: (notready!) set Kickstart power mode (Run/Sleep/Deepsleep)
syntax: 33, 1, PWRmode, chksum
PWRmode: 0=Run, 1=Sleep, 2=Deepsleep
Note1:
Run is the full operation mode, 1.3mA
Sleep is reduced power mode, 30 uA. Command-receiving, LCD, Keyin-wakeup, RTC, 2Hz events working.
Deepsleep is fully stopped mode, 1uA , only Command-receiving possible (and wakes up).
Note2: If an event wakes up Kickstart from sleep, it operates in Run mode until its service executed.

Cmd_14seg: print one 14 seg char to LCD
syntax: 34, 2, ObjNum, Char, chksum
-ObjNum: ordinal number of target object (0-127)
-Char: value to print (0-18: 0123456789AbcdEF-'space'_ 32-127:ASCII)

Cmd_16seg: print one 16 seg char to LCD
syntax: 35, 2, ObjNum, Char, chksum
-ObjNum: ordinal number of target object (0-127)
-Char: value to print (0-18: 0123456789AbcdEF- _ 32-127:ASCII)

Cmd_Shift_L: Shift a group of objects (string) left << by one object.
syntax: 36, 2, ObjNum, ObjCount, chksum
-ObjNum: ordinal number of first object of group (0-127)
-ObjCount: count of objects to Shift
Note1: the last object cleared after operation
Note2: The objects can be any type, so Characters, Bars...

Cmd_Shift_R: Shift a group of objects (string) right >> by one object.
syntax: 37, 2, ObjNum, ObjCount, chksum
-ObjNum: ordinal number of first object of group (0-127)
-ObjCount: count of objects to Shift
Note1: the first object cleared after operation
Note2: The objects can be any type, so Characters, Bars...

Cmd_7seg_Str: print 7seg char string, max 30 char
syntax: 38, Lenght+1, StartObj, string[Length], chksum
-Length: the character count of the string
-StartObj: ordinal number of first object of string (0-127)
-string[]: characters to print (->converted by the Kicstart 7seg font table)

Cmd_Bar_Str: print bargraph (bar-string) max 30 bars
syntax: 39, Lenght+1, StartObj, bars[Length], chksum
-Length: the bar count of the string
-StartObj: ordinal number of first object of bargraph (0-127)
-bars[]: bar values to print (->converted by the Kicstart bar table)

Cmd_14seg_Str: print 14seg char string, max 30 char
syntax: 40, Lenght+1, StartObj, string[Length], chksum
-Length: the character count of the string
-StartObj: ordinal number of first object of string (0-127)
-string[]: characters to print (->converted by the Kicstart 14seg font table)

Cmd_16seg_Str: print 16seg char string, max 30 char
syntax: 41, Lenght+1, StartObj, string[Length], chksum
-Length: the character count of the string
-StartObj: ordinal number of first object of string (0-127)
-string[]: characters to print (->converted by the Kicstart 16seg font table)

Cmd_Scroll2Hz: autoscroll text in window. Text stored by Cmd_ScrollTxt.
This command operates on text already loaded using Cmd_ScrollTxt (1-255 char).
First fills the window with spaces, then scrolls in the text riht to left,
every 0.5s by one character. The text rolled infinite until stop received.
syntax: 42, 3, StartObj, ObjCount, TxtLength, chksum
-StartObj: ordinal number of first object of group (0-127)
-ObjCount: count of objects to print. (window width, 1-255)
-TxtLength: the full length of ScrollTxt string
to stop scrolling, send this command with ObjCount=0
The char type auto detected: 7seg/14seg/16seg depending on the segment count of the last obj.
If the window is shorter than the text, only a part of text will be shown at a time.
If the window is larger than the text, it appears multiple times in the window. (repeated)

Cmd_TestLED: Turns on/off test-LED functions
At startup, test LEDs (error LEDs and RTC-1sec) enabled, which is useful for developpers,
and helps to write the interface functions.
Disable it if the LEDs are not wanted. (or the LEDs are used)
Now i2c/SPI checksum error lites LED6.
If RTC enabled, LED2 blinks /sec.
syntax: 43, 1, On/Off, chksum
On/Off: 0=disabled, FFh=enabled

Chip-on-Flex Package information