ASCII commands supported by the ws2811.asm application:

The application assembly program ws2811.asm supports single ASCII character commands. These commands are managed by the software state machine SESMA and its related subroutines/tasks. Single ASCII character command is case-sensitive, **a** is different from **A**.

The device that communicates with the FPGA-UART either an Android phone/tablet through a Bluetooth interface or a terminal emulator in a computer through the USB-FTD232-UART sends and receives data based on these commands.

The Main Menu is the first menu from power up, it will be displayed on a terminal emulator after E and C commands are sent to the FPGA, in Android mode by default the echo is off same as clear screen. Suppose you are using a terminal emulator the following is an example of command execution:

```
Power up:
In terminal mode type: (no CR, not needed)
C
E
Display:
------Main Menu------
 a = application
 s = system setup
 d = debug
 v = version
 L = Load Program *.mem
 E = Echo ON
 e = echo OFF
 C = CLS ON
 c = CLS OFF
 I = IRQ mask
```

```
In terminal mode type: (no CR, not needed)
v
Display:
SW: 041022-01-xxxx-yyyy-(3f00)
a
Display:
------Application------
 c = set Channel (0 to f)
 f = select Foreground
 b = select Background
 ……… (More commands)………
 h = HELP
 x = exit
```

Now the new state is **Application** and there are new set of commands like **c**, it sets the channel number for the color setup.

```
In terminal mode type: (no CR, not needed)
c
Display:
Enter number: (type any number from 0 to F and CR|Enter|Return)
```

< (prompt)
In this program all the numbers entered in the data fields are hexadecimal; a routine to convert from decimal-hexadecimal and vice versa can easily be done.
In terminal mode type: (no CR, not needed)
x (exit or return to previous menu)
Display:

```
------Main Menu------
 a = application
 s = system setup
 d = debug
 v = version
 L = Load Program *.mem
 E = Echo ON
 e = echo OFF
 C = CLS ON
 c = CLS OFF
 I = IRQ mask
```

The command tells the state machine to go back to the Main Menu
Some commands return data back, this is the case of **h** or help command, it will return the current menu with some description. This is useful in terminal mode; in Android you must modify/change my code to be able to properly display the message buffer something that I'm not doing well. In Android, prior to display messages you must send the following commands **E** and **C**, to get messages from the FPGA-CPU.

Basically, a single command moves between different states like a typical hardware FSM, you can add more commands following the template structure.

The following are all the supported commands and its menus, return data is not showed but very intuitive.

To download a new program to the FPGA memory the following sequence is needed:

a. From Main Menu type **s**
b. In System Menu type **W** and enter **0**, this enables write to memory
c. Type **x** to return to Main Menu
d. Type **L** and a prompt '**<**' will be displayed, the FPGA is ready to accept a .mem file
e. Locate the send file command in your terminal , click on it and select prog_ram.mem
f. This prog_ram.mem is located by default in D:\Aprj_cmodA7_SL\ASM_compiler\Opus16_cores_xil
g. The system will start sending asterisks to the terminal and when it finish you must type: C and E to display the Main Menu. Automatically the system enters in write protection mode, a green LED on the FPGA board is light up.

Next pages are a list of commands and some description

```
------Main Menu------
 a = application
 s = system setup
 d = debug
 v = version
 L = Load Program *.mem
 E = Echo ON
 e = echo OFF
 C = CLS ON
 c = CLS OFF
 I = IRQ mask
```

Command:

a = jump to Application menu, this is the main application menu, it has all the commands to control the LED strips.

s = jump to System menu, this is menu for system setup.

d = jump to Debug menu, for advanced users, like a typical debugger, can write and read from internal and external memories, write and read registers and input/output ports, etc.

v = returns the application software version number, located inside the ws2811.asm module in ASM_compiler\Opus16_asm\PROG.

L = load the .mem file into program memory, because the code is protected from unwanted writes, it is needed a sequence to load a new program, it is explained later.

E = enables the processor to echo all the messages and commands, it is used in USB terminal mode.

e = disables the processor to echo messages, it is set by default and it is used in the Android application.

C = enables the processor to clear screen and set home cursor, same as E command it is used in USB terminal mode.

c = disables clear screen/home cursor.

I = interrupt enable, current version a 0 disable interrupts any other value different from 0 enables all interrupts.

```
------Application------
 c = set Channel (0 to f)
 f = select Foreground
 b = select Background
 s = Set color value per channel
 p = Pattern
 e = thEme
 d = Display theme
 m = theme Mode ---------------------→stretch, repeat
 i = Intensity (0-7)
 0 = blk
 1 = red
 2 = grn
 3 = yel
 4 = blu
 5 = mag
 6 = cya
 7 = wht
 8 = update colorset queue
 9 = reset colorset queue
 a = active pixels -----------------→ number of pixels to walk
 r = read buffer mode --------------→ walk up/walk dn/walk ud/blink
 t = timer (blink/dim/walk) --------→ general timer for effects
 j = dimm steps --------------------→ simple counter to dim channels
 n = Number of Pixels
 S = Save channel to virtual channel
 R = Restore virtual channel to channel
 P = external memory Page
 V = set Virtual channel, V=-1 then V=c
 k = display clock -----------------→ real time clock
 v = version
 h = HELP
 x = exit --------------------------→ return to Main Menu
```

Commands:

c = set channel for colors and themes, channel 1 is c=0, channel 16 is c=f, so it is from 0 to f

f = foreground color is the main color, **b** is the background color used in walking pixels and blinking effects. In walking/blinking pixels, the pixel color is defined by the foreground and walks/blinks over the background

s = set color to the selected channel, for example s=00304050(hex) 00(don't care) 30(GRN) 40(BLU) 50(RED)

p = pattern in custom mode will walk up/down the theme, if you select blink/walk/dimm (no custom) it will use the solid colors in all channels

e = theme will create the buffer with the selected item, in custom mode it will use the colors in the queue

d = refresh the colors, read the buffers to LED

m = theme mode controls how the buffer is filled, stretch or repeat, the first one divides the number of LEDs by the number of colors in the queue, the repeat , just repeat the color in the queue until reach the number of LEDs

0 to **7** = solid colors affects all channels, same as s=00xxyyzz

0 = s000000 Black
1 = s0000FF RED
2 = s00FF00 GREEN
4 = sFF0000 BLUE
7 = sFFFFFF White

8 = add the color to the queue and increments the color counter
9 = clears the queue counter
a = number of pixels to walk
r = used by the pattern to define how the buffer is read to create the walking mode
t = timer/counter for walking/blinking
j = timer/counter for dimmer
n = Number of Pixels depends on the LED strip length
S = Save channel to external RAM, selected channel is saved to the virtual channel defined by the V command, also the page memory is involved in the operation
R = Restore virtual channel from external RAM into channel local RAM
P = set the page for external memory, there are 8 pages from 0 to 7 the use of the page port depends on the size of the external RAM
V = virtual channel is the channel on the external RAM, when V is equal -1 (FFFF) or 8xxx (sign bit high) then V=c, virtual channel is equal to channel
k = display RTC (Real Time Clock), to setup the RTC you must go to the System Menu
x = always return to previous menu

```
------System Setup------
 t = trigger source (0 to 3)
 s = scope Source (0 to 3) -----------→ lower case
 l = PWM reset -----------------------→ letter l
 p = PWM period
 0 = PWM pulse width code 0
 1 = PWM pulse width code 1
 8 = 800Khz PWM
 4 = 400Khz PWM
 b = set Base reg address
 r = Read all registers --------------→ lower case
 w = Write/read port/register --------→ lower case
 n = Number of Pixels
 Y = Year
 M = Month
 D = day/DATE
 H = Hour
 U = minUte
 S = Sec -----------------------------→ upper case
 k = display clock
 R = target Reset --------------------→ upper case
 W = Write protect (2 or 0)-----------→ upper case
 v = display variables
 x = exit
```

---

t = set 2 bits on an output register in the FPGA, it is used to select **a** signal source for the scope or logic analyzer

s = set 2 bits on an output register in the FPGA, it is used to select **a** signal source for the scope or logic analyzer

l,**p**,**0**,**1**,**8** and **4** = control the PWM timing

W = 2 will protect the program code and the bootloader to be mistakenly written, ECODE is the end of the program code and BCODE is the start of the boot code, only variables are enabled for write. ECODE and BCODE and calculated automatically when you compile the asm file and our code must write to two output registers to make them effective, also a PCODE signal will overwrite the protection to allow downloads, W=0 will unprotect the program memory and bootloader.
r = read 16 registers from the base address, by default the base address register is 0.

b = set Base reg address, default is 0

r = Read all registers, read 16 regs at at time

w = write registers, full address is needed, for example to write to reg 100h is w100,55 all hex numbers, to read back set b=100 and then r to read the values of reg 100.

```
------ Debug Menu------
 A = memory Access mode
 B = memory Byte mode
 P = external memory Page
 m = Memory write/read
 d = Memory dump
 n = Next dump
 f = fill memory address/value (aaaa,vvvv)
 l = set length/increment (llll,iiii)
 w = Write/read port/register
 s = Set reg bit number
 c = Clear reg bit number
 t = Toggle reg bit number
 i = Pulse reg bit number
 a = and
 o = or
 e = xor
 b = set Base I/O/Registers address
 r = Read all Input ports/registers
 v = display variables
 h = HELP
 x = exit
```

A = access internal or external RAM, used by memory display like memory read/write, memory dump and fill memory

B = memory accress mode, default is 16 bits (word) program internal memory, the external RAM in the Cmod-A7 is byte access

P = page memory, 8 pages (0 to 7) 8x16K

m = memory write/read, example m100a,3a5b or m100a display 3a5b

d = dump 256 (defined by code) memory values, external or internal defined by **A** command

n = next 256 memory values

f = set the start address and the value, both must be 4 hexadecimal digits, this is the way was coded

l = set the length of the memory to be written and the increment if it is desired, same as **f** command full 4 hex digits for length and value

b,**r** = set base address and read 16 registers from the base address, by default the base address register is 0.

w = write registers, full address is needed, for example to write to reg 100h is w100,55 all hex numbers, to read back set b=100 and then r to read the values of reg 100

s,**c**,**t**,**i** = full register address and bit number from 0 to f, example s0100,1 address 100hex and bit #1

a,**o**,**e** = full address and value, to read back set the base address to the full address.

v = display variables defined by the program address

Help for application menu

```
------ H E L P ------
 Space key will refresh the screen
 Enter x to exit (previous menu)
 Enter values in HEXADECIMAL (0-9, a-f)

------ Commands ------
 b = background color
 c = channel strip (0 to f)
 d = display buffer
 f = foreground color

 a = active walking/length pixels
 r = read buffer mode, DN(0),UP(1),UD(2)
 t = timer for blink/dim/walk
 j = dimm steps

 p = selects a pattern
 p = 0 Walking Custom theme
 p = 1 Blinking LED
 p = 2 Walking LED
 p = 3 Dimm LED

 e = selects a theme
 e = 0 Custom Theme
 e = 1,2 XMas
 e = 3,4,5,6 colorful1
 e = 7,8,9,a,b Flags
 ....7:USA,8:IRE,9:ARG,a:ENG,b:ITA,

 s = selects color, enter GGBBRR in Hex
 i = color intensity for solid colors <0-7>
 m = Fill channel buffer mode
 ....0:stretch, 1:repeat
 8-9 = Update-Reset colorset counter/queue
 0-7 = selects predefined colors
..... 0:BLK 1:R 2:G 3:RG 4:B 5:RB 6:GB 7:WHT
```

Help for Debug menu

------ H E L P ------

```
Space key will refresh the screen
 Enter x to exit (previous menu)
 Enter values in HEXADECIMAL

------ Memory Commands ------
 A = Memory access (toggle)
 B = Memory byte mode (toggle)
 m = Memory read/write: (MEM ADDR)
 ___ [m]addr[RET] read and set address
 ___ [m]addr,val[RET] write value
 d = dump memory from address 0
 n = next 256 memory from last address
 f = fill memory with value + increment
 ___ addr,value = aaaa,vvvv (comma or space)
 l = memory length and increment
 ___ length,increment = llll,iiii (comma or space)
 ___ aaaa,vvvv,llll,iiii mandatory 4 digits

------ Register Commands ------
 w = Write Output Port/Register: (full address)
 ___ [w]addr[RET] read port
 ___ [w]addr,val[RET] delimeter comma
 ___ [w]addr val[RET] delimiter space
 ___ [w]addr'n'val[RET] one's complement
 ___ [w]addr'-'val[RET] two's complement

[s/c/t/i]regaddr,bit[RET] (bit=0 to F)
[a/o/e] regaddr,mask[RET]

 b = set register Base address
 r = Read 16 Registers (BASE ADDR + 0 to F)
```