深圳市拓普微科技开发有限公司
SHENZHEN TOPWAY TECHNOLOGY CO., LTD.

SMART LCD

Lua Functions Reference Note

Prepared by:	Checked by:	Approved by:
K.C.		
Date: 2020-05-14	Date:	Date:

Table of Content

1 Basic

Lua script is supported in some of the TOPWAY Smart LCD to enhance the flexibility.
Engineer may refine the Smart LCD functionality with script.

The provided Lua functionality support most of the features of Lua 5.3 except OS operation and file I/O. (please refer to lua.net for Lua 5.3 details)
It also extended with hmt library for Smart LCD variables access, function control and value return etc.

2 TOPWAY Smart LCD script and variable

2.1 Lua Scrip structure

Basic structure

```
-- define variable(s)
Script_Variable01 = 0x00


-- main loop
luamain = function(void)

return 0
end
```

Note:
- main loop run continually
- intensive loop in the main loop may affect the Smart LCD UI operations

Advance structure with hooks

```
-- define variable(s)
Script_Variable01 = 0x00


-- main loop
luamain = function(void)

return 0
end


-- touch key hook
tpkhook = function(page, id, state)

return 0
end


-- page hook
pagechangehook = function(pageid)

return 0
end
```

Note:
- main loop run continually
- intensive loop in the main loop may affect the Smart LCD UI operations
- hooks share the MCU time with the main loop. looping or long delay will affect the main loop
- “wait loop” is not allowed in the hooks.

2.2 Smart LCD VP variables

Mnemonic	Name	Memory Limit	Address Range
VP_STR	String Variable	1024(MAX)*(127+1)byte	0x000000~0x01FF80
VP_N16	16Bit Integer Variable	32512(MAX)*(2)byte	0x080000~0x08FDFE
VP_N32	32Bit Integer Variable	16128(MAX)*(4)byte	0x020000~0x02FEFC
VP_N64	64Bit Integer Variable	7936(MAX)*(8)byte	0x030000~0x03F7F8
VP_SYS	System Register Variable	256(MAX)*(1)byte	0xFFFF00~0xFFFFFF

Note. Please also refer to Smart LCD handbook for VP variables details

3 hmt library

3.1 hmt functions Summary

Category	Function Name	Descriptions
VP read/write	hmt.readvpstr(addr)	Read VP_STR value
	hmt.writevpstr(addr, string)	Write VP_STR value
	hmt.readvp16(addr)	Read VP_16 value
	hmt.writevp16(addr, value)	Write VP_16 value
	hmt.readvp32(addr)	Read VP_32 value
	hmt.readvp32f(addr)	Read VP_32 value (floating point value)
	hmt.writevp32(addr, value)	Write VP_32 value
	hmt.readvp64(addr)	Read VP_64 value
	hmt.readvp64f(addr)	Read VP_64 value (floating point value)
	hmt.writevp64(addr, value)	Write VP_64 value
	hmt.readvpreg(addr)	Read VP_SYS value
	hmt.writevpreg(addr, value)	Write VP_SYS value
System	hmt.changepage(pageid)	Display a pre-stored PAGE
	hmt.readpage()	Read Page ID
	hmt.gettick()	Read system time
	hmt.delayms(time)	Delay in ms(*1)
	hmt.runcmd(table, num)	Execute terminal command (without packet header and taill)
	hmt.bypass(is)	Disable Smart LCD terminal command execution (I/O) is=1 for disable; is=0 for enable(normal)
	hmt.crc16calc(table, num)	Generate a CRC value
UART0	hmt.uartisempty()	Check for UART0 buffer 1 for empty; 0 for not empty
	hmt.uartclearbuf()	Clear UART0 buffer
	hmt.getchar()	Read a byte form UART0 buffer
	hmt.putchar(char)	Send a byte to UART0
	hmt.uartsendbytes(table, num)	Send a num of byte to UART0
	hmt.uartlock()	Lock UART0 before Lua sending data to UART0 (*2, *3)
	hmt.uartunlock()	Unlock UART0 after Lua sending data to UART0 (*2, *3)
UART1	hmt.uart1isempty()	Check for UART1 buffer 1 for empty; 0 for not empty
	hmt.uart1clearbuf()	Clear UART1 buffer
	hmt.uart1getchar()	Read a byte form UART1 buffer
	hmt.uart1putchar(char)	Send a byte to UART1
	hmt.uart1sendbytes(table, num)	Send a num of byte to UART1
	hmt.uart1lock()	Lock UART1 before sending data to UART1 (*2, *3)
	hmt.uart1unlock()	Unlock UART1 after sending data to UART1 (*2, *3)

Note.
*1. long delay may affect the Smart LCD normal operation.
*2. It is necessary to lock the UART for Lua access, if hmt.bypass is not enabled.
*3. Generally hmt.uartlock() and hmt.uartunlock() should be used in pair.

3.2 VP read/write Functions

3.2.1 hmt.readvpstr

Function	hmt.readvpstr(addr)
Description	Read VP_STR value
Parameter	addr is the VP_STR address (*1)
Return Value	String

Example

Lua script	Descriptions

`local strvalue = ""`
`strvalue = hmt.readvpstr(0x000080)`
`print("strvalue=", strvalue)`
	Put "TOPWAY" into VP_STR 0x000080 via SmartLCD UI
	define a local variable strvalue and put "" inside
	copy value in VP_STR at 0x000080 to strvalue
	print to UART0
	it should find the following at the UART0
	strvalue=TOPWAY

3.2.2 hmt.writevpstr

Function	hmt.writevpstr(addr, string)
Description	Write a value to VP_STR
Parameter	addr is the VP_STR address (*1) string is the value to be written
Return Value	nil

Example

Lua script	Descriptions

`local strvalue = "ShenZhen"`
`hmt.writevpstr(0x000080, strvalue)`
`print("0x000080:", hmt.readvpstr(0x000080))`
	define a local variable strvalue and put "ShenZhen" inside
	copy the strvalue to VP_STR at 0x000080
	print to UART0
	it should find the following at the UART0
	0x000080:ShenZhen

3.2.3 hmt.readvp16

Function	hmt.readvp16(addr)
Description	Read VP_N16 value
Parameter	addr is the VP_N16 address (*1)
Return Value	Integer

Example

Lua script	Descriptions

`local N16value = 0`
`N16value = hmt.readvp16(0x080002)`
`print("N16value:", N16value)`
	Put 24 into VP_N16 0x080002 via SmartLCD UI
	define a local variable N16value and put 0 inside
	copy value in VP_N16 at 0x080002 to N16value
	print to UART0
	it should find the following at the UART0
	N16value:24

3.2.4 hmt.writevp16

Function	hmt.writevp16(addr, value)
Description	Write a value to VP_N16
Parameter	addr is the VP_N16 address(*1) value is the value to be written
Return Value	nil

Example

Lua script	Descriptions

`local N16value = 123`
`hmt.writevp16(0x080002, N16value)`
`print("0x080002:", hmt.readvp16(0x080002))`
	define a local variable N16value and put 123 inside
	copy the N16value to VP_N16 at 0x080002
	print to UART0
	it should find the following at the UART0
	0x080002:123

3.2.5 hmt.readvp32

Function	hmt.readvp32(addr)
Description	Read VP_N32 value
Parameter	addr is the VP_N32 address (*1)
Return Value	Integer

Example

Lua script	Descriptions

`local N32value = 0`
`N32value = hmt.readvp32(0x020004)`
`print("N32value:", N32value)`
	Put 1108410368 into VP_N32 0x020004 via SmartLCD UI
	define a local variable N32value and put 0 inside
	copy value in VP_N32 at 0x020004 to N32value
	print to UART0
	it should find the following at the UART0
	N32value:1108410368

3.2.6 hmt.readvp32f

Function	hmt.readvp32f(addr)
Description	Read VP_N32 value with floating value inside
Parameter	addr is the VP_N32 address (*1)
Return Value	Floating point

Example

Lua script	Descriptions

`local N32value = 0.0`
`N32value = hmt.readvp32f(0x020004)`
`print("N32value:", N32value)`
	Put a floating point value 36.25 into VP_N32 0x020004 via SmartLCD UI
	define a local variable N32value and put 0.0 inside
	copy floating point value in VP_N32 at 0x020004 to N32value
	print to UART0
	it should find the following at the UART0
	N32value:36.25

3.2.7 hmt.writevp32

Function	hmt.writevp32(addr, value)
Description	Write a value to VP_N32
Parameter	addr is the VP_N32 address(*1) value is the value to be written
Return Value	Nil

Example

Lua script	Descriptions

`local N32value = 54321`
`hmt.writevp32(0x080004, N32value)`
`print("0x080004:", hmt.readvp32(0x080004))`
	define a local variable N32value and put 54321 inside
	copy the N32value to VP_N32 at 0x080004
	print to UART0
	it should find the following at the UART0
	0x080004:54321

3.2.8 hmt.readvp64

Function	hmt.readvp64(addr)
Description	Read VP_N64 value
Parameter	addr is the VP_N64 address (*1)
Return Value	Integer

Example

Lua script	Descriptions

`local N64value = 0`
`N32value = hmt.readvp64(0x030000)`
`print("N64value:", N64value)`
	Put 4638029965185179976 into VP_N64 0x030000 via SmartLCD UI
	define a local variable N64value and put 0 inside
	copy value in VP_N64 at 0x030000 to N64value
	print to UART0
	it should find the following at the UART0
	N64value: 4638029965185179976

3.2.9 hmt.readvp64f

Function	hmt.readvp64f(addr)
Description	Read VP_N64 value with floating value inside
Parameter	addr is the VP_N64 address (*1)
Return Value	Floating point

Example

Lua script	Descriptions

`local N64value = 0.0`
`N64value = hmt.readvp64f(0x030000)`
`print("N64value:", N64value)`
	Put a floating point value 118.37 into VP_N64 0x030000 via SmartLCD UI
	define a local variable N64value and put 0.0 inside
	copy floating point value in VP_N64 at 0x030000 to N64value
	print to UART0
	it should find the following at the UART0
	N64value:118.37

3.2.10 hmt.writevp64

Function	hmt.writevp64(addr, value)
Description	Write a value to VP_N64
Parameter	addr is the VP_N64 address(*1) value is the value to be written
Return Value	Nil

Example

Lua script	Descriptions

`local N64value = 123456789`
`hmt.writevp64(0x030000, N64value)`
`print("0x030000:", hmt.readvp64(0x030000))`
	define a local variable N64value and put 123456789 inside
	copy the N64value to VP_N64 at 0x030000
	print to UART0
	it should find the following at the UART0
	0x030000:123456789

3.2.11 hmt.readvpreg

Function	hmt.readvpreg(addr)
Description	Read Smart LCD system register value (VP_SYS) value
Parameter	addr is the VP_SYS address (*1)
Return Value	Integer

Example

Lua script	Descriptions

`local Regvalue = 0`
`Regvalue = hmt.readvpreg(0xFFFF20)`
`print("Regvalue:", Regvalue)`
	define a local variable Regvalue and put 0 inside
	copy value in VP_SYS at 0xFFFF20 to Regvalue
	print to UART0
	it should find the following at the UART0
	Regvalue:8

3.2.12 hmt.writevpreg

Function	hmt.writevpreg(addr, value)
Description	Write a value to Smart LCD system register value (VP_SYS) value
Parameter	addr is the VP_SYS address(*1) value is the value to be written
Return Value	Nil

Example

Lua script	Descriptions

`local Regvalue = 18`
`hmt.writevpreg(0xFFFF20, Regvalue)`
`print("0xFFFF20:", hmt.readvpreg(0xFFFF20))`
	define a local variable Regvalue and put 18 inside
	copy the Regvalue to VP_SYS at 0xFFFF20
	print to UART0
	it should find the following at the UART0
	0xFFFF20:18

Note: *1. Please refer to Smart LCD VP variables section.

3.3 System Functions

3.3.1 hmt.changepage

Function	hmt.changepage(pageid)
Description	Change the Smart LCD UI display PAGE
Parameter	pageid is the target display PAGE to be shown
Return Value	Nil

Example

Lua script	Descriptions
`hmt.changepage(0x02)`	UI show up the PAGE 2

3.3.2 hmt.readpage

Function	hmt.readpage()
Description	Read the current display page id
Parameter	Nil
Return Value	Integer

Example

Lua script	Descriptions

`local pageid = 0`
`pageid = hmt.readpage()`
`print("pageid:", pageid)`
	SmartLCD UI is showing PAGE 3
	define a local variable pageid and put 0 inside
	copy the current display page id to pageid
	print to UART0
	it should find the following at the UART0
	pageid:3

3.3.3 hmt.gettick

Function	hmt.gettick()
Description	Read the system running time in ms
Parameter	Nil
Return Value	Integer

Example

Lua script	Descriptions

`local worktime = 0`
`worktime = hmt.gettick()`
`print("worktime:", worktime)`
	define a local variable worktime and put 0 inside
	copy the current system running time to worktime
	print to UART0
	it should find the following at the UART0
	worktime:334520 (The system ran for 334520ms)

3.3.4 hmt.delayms

Function	hmt.delayms(time)
Description	delay routine
Parameter	time is the delay in ms
Return Value	Nil

Example

Lua script	Descriptions
`hmt.delaymy(200)`	system will do nothing for 200ms and jump and go on

Note. long delay may affect the Smart LCD normal operation.

3.3.5 hmt.runcmd

Function	hmt.runcmd(table, num)
Description	Send a command directly as sending command to the Smart LCD terminal (no command head and command tail)
Parameter	table is the command byte array to be send num is the number of byte to be send
Return Value	Nil

Example

Lua script	Descriptions
`local cmdtab = {0x5F, 0x04}` `hmt.runcmd(cmdtab, 2)`	define a local variable array and put 0x5F, 0x04 inside use hmt.runcmd to send 2 byte of local variable array to Smart LCD it will adjust the backlight of the display to dim down

Note. 0xF5 is backlight control command; please also refer to Smart LCD handbook for command details

3.3.6 hmt.bypass

Function	hmt.bypass(is)
Description	Disable Smart LCD terminal command execution (I/O) And prevent data stream corruption while Lua accessing the UART0
Parameter	is=1 for disable; is=0 for enable(normal)
Return Value	Nil

Example

Lua script	Descriptions

`hmt.bypass(1)`
 	
`hmt.bypass(0)`
	Disable Smart LCD terminal command execution
	Send AA 5F 03 CC 33 C3 3C to Smart LCD terminal to dim down the backlight
	No response
	Enable Smart LCD terminal command execution
	Send AA 5F 03 CC 33 C3 3C to Smart LCD terminal to dim down the backlight
	Display dim down accordingly

Note. 0xF5 is backlight control command; please also refer to Smart LCD handbook for command details

3.3.7 hmt.crc16calc

Function	hmt.crc16calc(table, num)
Description	Generate a CRC value form given num of bytes
Parameter	table is the byte array to be input num is the number of byte to be taken form the table
Return Value	Integer

Example

Lua script	Descriptions

`local padcmd = {0x82, 0x00, 0x08, 0x05}`
`local crcreturn = 0`
`crcreturn = hmt.crc16calc(padcmd, 4)`
`print("crcreturn:", crcreturn)`
	define a local variable array and put 0x82, 0x00, 0x08, 0x05 inside
	define a local variable and put 0 inside
	calculate a CRC value of the 4 bytes form the array
	print to UART0
	it should find the following at the UART0
	crcreturn:6134316

3.4 UART0 functions

UART0 is the default command terminal of the Smart LCD. It may be RS-232C or UART(3.3V) depends on model. Lua print function also using UART0 for default output.
Some of the Smart LCD equipped with a second terminal UART1. It may be RS-232C or UART(3.3V) depends on model.

3.4.1 hmt.uartisempty

Function	hmt.uartisempty()
Description	Check the UART0 buffer status
Parameter	Nil
Return Value	Integer Return 1 for buffer empty; 0 for data in buffer

Example

Lua script	Descriptions

`local Uart0state = 0`
`Uart0state = hmt.uartisempty()`
`print("Uart0state:", Uart0state)`
	define a local variable Uart0state and put 0 inside
	copy the current UART is empty result to Uart0state
	print to UART0
	it should find the following at the UART0
	Uart0state:1 (as there is no data received form UART0)

3.4.2 hmt.uartclearbuf

Function	hmt.uartclearbuf()
Description	Clear the UART0 buffer
Parameter	Nil
Return Value	Nil

Example

Lua script	Descriptions
`hmt.uartclearbuf()` `print("Uart0state:", hmt.uartisempty())`	use command to clear the UART0 buffer print to UART0 it should find the following at the UART0 Uart0state:1 (as there is no data send to the UART0)

3.4.3 hmt.getchar

Function	hmt.getchar()
Description	Receive one byte of data form UART0 buffer
Parameter	Nil
Return Value	Integer

Example

Lua script	Descriptions

`local getdata = 0`
`getdata = hmt.getchar()`
`print("getdata:", getdata)`
	Send one byet of data 0x99 to the Smart LCD terminal
	define a local variable getdata and put 0 inside
	get one byte of data form the UART0 buffer and put it into getdata
	print to UART0
	it should find the following at the UART0
	getdata:153 (0x99 is 153(d))

3.4.4 hmt.putchar

Function	hmt.putchar(char)
Description	Send one byte of data to UART0
Parameter	char is the byte of data to be sent (it should be a byte value, not string char)
Return Value	Integer

Example

Lua script	Descriptions
`hmt.putchar(0x41)` `hmt.putchar(66)`	it should find the following at the UART0 0x41 0x42 (in hex) or AB (in ASCII)

3.4.5 hmt.uartsendbytes

Function	hmt.uartsendbytes(table, num)
Description	Send bytes of data to UART0
Parameter	table is the data byte array to be send num is the number of byte to be send
Return Value	Nil

Example

Lua script	Descriptions
`local sendbuff = {0xab, 0xcd, 0xef}` `hmt.uartsendbytes(sendbuff, 3)`	define a local variable array and put 0xab, 0xcd, 0xef inside use hmt.uartsendbytes to send 3 byte of local variable array to UART0 it should find the following at the UART0 0xab 0xcd 0xef (in hex)

3.4.6 hmt.uartlock

Function	hmt.uartlock()
Description	Lock the UART0 before Lua send data to UART0
Parameter	Nil
Return Value	Nil

Example

Lua script	Descriptions

`hmt.uartlock()`
`hmt.putchar(0xaa)`
`hmt.putchar(0x02)`
`hmt.putchar(0x31)`
 	
`hmt.uartunlock()`
	Lock the UART0 for Lua use
	Send 0xaa to UART0
	Send 0x02 to UART0
	Send 0x31 to UART0
	it should find the following at the UART0
	0xaa 0x02 0x31 (in hex)
	Release the UART0 for Smart LCD normal operation

Note.
It is necessary to lock the UART for Lua access, if hmt.bypass is not enabled.
Generally hmt.uartlock() and hmt.uartunlock() should be used in pair.

3.4.7 hmt.uartunlock

Function	hmt.uartunlock()
Description	unlock the UART0 after Lua send data to UART0
Parameter	Nil
Return Value	Nil

Example

Lua script	Descriptions

`hmt.uartlock()`
`hmt.putchar(0xaa)`
`hmt.putchar(0x02)`
`hmt.putchar(0x31)`
 	
`hmt.uartunlock()`
	Lock the UART0 for Lua use
	Send 0xaa to UART0
	Send 0x02 to UART0
	Send 0x31 to UART0
	it should find the following at the UART0
	0xaa 0x02 0x31 (in hex)
	Release the UART0 for Smart LCD normal operation

Note.
It is necessary to lock the UART for Lua access, if hmt.bypass is not enabled.
Generally hmt.uartlock() and hmt.uartunlock() should be used in pair.

3.5 UART1 functions

UART1 is the extend terminal of some of the Smart LCD. It may be RS-232C or UART(3.3V) depends on model.

3.5.1 hmt.uart1isempty

Function	hmt.uart1isempty()
Description	Check the UART1 buffer status
Parameter	Nil
Return Value	Integer Return 1 for buffer empty; 0 for data in buffer

Example

Lua script	Descriptions

`local Uart1state = 0`
`Uart1state = hmt.uart1isempty()`
`print("Uart1state:", Uart1state)`
	define a local variable Uart1state and put 0 inside
	copy the current UART is empty result to Uart1state
	print to UART0
	it should find the following at the UART0
	Uart1state:1 (as there is no data received form UART1)

3.5.2 hmt.uart1clearbuf

Function	hmt.uart1clearbuf()
Description	Clear the UART1 buffer
Parameter	Nil
Return Value	Nil

Example

Lua script	Descriptions
`hmt.uart1clearbuf()` `print("Uart1state:", hmt.uart1isempty())`	use command to clear the UART1 buffer print to UART0 it should find the following at the UART0 Uart1state:1 (as there is no data send to the UART0)

3.5.3 hmt.uart1getchar

Function	hmt.uart1getchar()
Description	Receive one byte of data form UART1 buffer
Parameter	Nil
Return Value	Integer

Example

Lua script	Descriptions

`local getdata = 0`
`getdata = hmt.uart1getchar()`
`print("getdata:", getdata)`
	Send one byet of data 0xaa to the UART1 terminal
	define a local variable getdata and put 0 inside
	get one byte of data form the UART1 buffer and put it into getdata
	print to UART0
	it should find the following at the UART0
	getdata:170 (0xaa is 170(d))

3.5.4 hmt.uart1putchar

Function	hmt.uart1putchar(char)
Description	Send one byte of data to UART1
Parameter	char is the byte of data to be sent (it should be a bin value, not string char)
Return Value	Integer

Example

Lua script	Descriptions
`hmt.uart1putchar(0x61)` `hmt.uart1putchar(97)`	it should find the following at the UART1 0x61 0x62 (in hex) or ab (in ASCII)

3.5.5 hmt.uart1sendbytes

Function	hmt.uart1sendbytes(table, num)
Description	Send bytes of data to UART1
Parameter	table is the data byte array to be send num is the number of byte to be send
Return Value	Nil

Example

Lua script	Descriptions
`local sendbuff = {0xab, 0xcd, 0xef}` `hmt.uart1sendbytes(sendbuff, 3)`	define a local variable array and put 0xab, 0xcd, 0xef inside use hmt.uart1sendbytes to send 3 byte of local variable array to UART1 it should find the following at the UART1 0xab 0xcd 0xef (in hex)

3.5.6 hmt.uart1lock

Function	hmt.uart1lock()
Description	Lock the UART1 before Lua send data to UART1
Parameter	Nil
Return Value	Nil

Example

Lua script	Descriptions

`hmt.uart1lock()`
`hmt.uart1putchar(0xaa)`
`hmt.uart1putchar(0x02)`
`hmt.uart1putchar(0x31)`
 	
`hmt.uart1unlock()`
	Lock the UART1 for Lua use
	Send 0xaa to UART1
	Send 0x02 to UART1
	Send 0x31 to UART1
	it should find the following at the UART1
	0xaa 0x02 0x31 (in hex)
	Release the UART1 for other functions

Note.
It is necessary to lock the UART for Lua access, if hmt.bypass is not enabled.
Generally hmt.uartlock() and hmt.uartunlock() should be used in pair.

3.5.7 hmt.uart1unlock

Function	hmt.uart1unlock()
Description	unlock the UART1 after Lua send data to UART1
Parameter	Nil
Return Value	Nil

Example

Lua script	Descriptions

`hmt.uart1lock()`
`hmt.uart1putchar(0xaa)`
`hmt.uart1putchar(0x02)`
`hmt.uart1putchar(0x31)`
 	
`hmt.uart1unlock()`
	Lock the UART1 for Lua use
	Send 0xaa to UART1
	Send 0x02 to UART1
	Send 0x31 to UART1
	it should find the following at the UART1
	0xaa 0x02 0x31 (in hex)
	Release the UART1 for other functions

Note.
It is necessary to lock the UART for Lua access, if hmt.bypass is not enabled.
Generally hmt.uartlock() and hmt.uartunlock() should be used in pair.

3.6 Main Loop and Hooks

3.6.1 luamain

Function	luamain = function(void)
Description	A loop routine run continuously. Looping cycle 10ms(MIN)
Parameter	Nil
Return Value	Nil

Example

Lua script	Descriptions

`luamain = function(void)`
`print("TOPWAY")`
 	
`return 0`
`end`
	Beginning of the luamain
	print to UART0
	it should find the following at the UART0
	TOPWAY
	no return value
	end of the luamain
	it will loop back to the beginning of the luamain continuously

3.6.2 tpkhook

Function	tpkhook = function(page, id, state)
Description	A hook routine that will be trigged while any of the touch key status that defined in the Smart LCD UI is being changed
Parameter	page is the touched touch key's PAGE id is the touch key's ID state is the statues of the touch key (0 for released; 1 for touched; 2 for moved out)
Return Value	return 0 tell Smart LCD UI continue to execute the touch key predefined operations return 1 tell Smart LCD UI not to execute the touch key predefined operations

Example

Lua script	Descriptions

`tpkhook = function(page, id, state)`
`print("PageID=", age,`
`"  TPKID=", id,`
`"  state=", state)`
`return 0`
`end`
	On page 0 touch the touch key (ID=2)
	Beginning of the tpkhopok
	print to UART0
	it should find the following at the UART0
	PageID=0 TPKID=2 state=1
	return 0 for UI normal continue run
	end of the tpkhook

3.6.3 pagechangehook

Function	pagechangehook = function(pageid)
Description	A hook routine that will be trigged when page changed (before showing the page)
Parameter	pageid is the target PAGE ID
Return Value	return 0 tell Smart LCD UI continue to showing up the target page return 1 tell Smart LCD UI staying at the previous page

Example

Lua script	Descriptions

`pagechangehook = function(pageid)`
`print("PageID=", pageid)`
 	
`return 0`
`end`
	On page 0 touch the touch key to jump to page 1
	Beginning of the pagechangehook
	print to UART0
	it should find the following at the UART0
	PageID=1
	return 0 for UI normal continue run
	end of the pagechangehook

4 Example

4.1 Built an alarm clock function with Lua

- set up a default alarm (alarmhour, alarmminute, alarmsecond) and alarm duration (alarmend)
- continue checking the Smart LCD compare the RTC with the alarm; if match then sound the buzzer on the Smart LCD.
- set up a touch key monitor, when it is touched, assign the alarm with value store in the smart LCD (those values, can be assign with the user interface)

```
-- define variable(s)
Alarmhour = 0x7
alarmminute = 0x14
alarmsecond = 0x0
alarmend = 0



-- main loop
luamain = function(void)
  if((hmt.readvpreg(0xffff13) == alarmhour) and (hmt.readvpreg(0xffff14) ==
  alarmminute) and (hmt.readvpreg(0xffff15) == alarmsecond))
  then
  alarmend = hmt.gettick() + 60000
  end
  if(alarmend > hmt.gettick())then
     local buzzercmd = {0x7a, 0x01, 0x01, 0x01, 0x05, 0x08}
     hmt.runcmd(buzzercmd,6)
     hmt.delayms(200)
  end
return 0
end



-- touch key hook
tpkhook = function(page, id, state)
   if((page == 1)and(id == 2)and(state == 1))then
      alarmhour = hmt.readvp16(0x080002)
      alarmminute = hmt.readvp16(0x080004)
      alarmsecond = hmt.readvp16(0x080006)
   end
   if((page == 1)and(id == 4)and(state == 1))then
      alarmend = hmt.gettick()
   end
return 0
end



-- page hook
pagechangehook = function(pageid)
return 0
end
```

Note: “wait loop” is not allowed in the script!

5 Appendix

5.1 hmt.crc16calc(table, num) reference

```
uint16_t const CRC16[256]={
/* 16: 8005 reflected */
  0x0000,0xc0c1,0xc181,0x0140,0xc301,0x03c0,0x0280,0xc241,
  0xc601,0x06c0,0x0780,0xc741,0x0500,0xc5c1,0xc481,0x0440,
  0xcc01,0x0cc0,0x0d80,0xcd41,0x0f00,0xcfc1,0xce81,0x0e40,
  0x0a00,0xcac1,0xcb81,0x0b40,0xc901,0x09c0,0x0880,0xc841,
  0xd801,0x18c0,0x1980,0xd941,0x1b00,0xdbc1,0xda81,0x1a40,
  0x1e00,0xdec1,0xdf81,0x1f40,0xdd01,0x1dc0,0x1c80,0xdc41,
  0x1400,0xd4c1,0xd581,0x1540,0xd701,0x17c0,0x1680,0xd641,
  0xd201,0x12c0,0x1380,0xd341,0x1100,0xd1c1,0xd081,0x1040,
  0xf001,0x30c0,0x3180,0xf141,0x3300,0xf3c1,0xf281,0x3240,
  0x3600,0xf6c1,0xf781,0x3740,0xf501,0x35c0,0x3480,0xf441,
  0x3c00,0xfcc1,0xfd81,0x3d40,0xff01,0x3fc0,0x3e80,0xfe41,
  0xfa01,0x3ac0,0x3b80,0xfb41,0x3900,0xf9c1,0xf881,0x3840,
  0x2800,0xe8c1,0xe981,0x2940,0xeb01,0x2bc0,0x2a80,0xea41,
  0xee01,0x2ec0,0x2f80,0xef41,0x2d00,0xedc1,0xec81,0x2c40,
  0xe401,0x24c0,0x2580,0xe541,0x2700,0xe7c1,0xe681,0x2640,
  0x2200,0xe2c1,0xe381,0x2340,0xe101,0x21c0,0x2080,0xe041,
  0xa001,0x60c0,0x6180,0xa141,0x6300,0xa3c1,0xa281,0x6240,
  0x6600,0xa6c1,0xa781,0x6740,0xa501,0x65c0,0x6480,0xa441,
  0x6c00,0xacc1,0xad81,0x6d40,0xaf01,0x6fc0,0x6e80,0xae41,
  0xaa01,0x6ac0,0x6b80,0xab41,0x6900,0xa9c1,0xa881,0x6840,
  0x7800,0xb8c1,0xb981,0x7940,0xbb01,0x7bc0,0x7a80,0xba41,
  0xbe01,0x7ec0,0x7f80,0xbf41,0x7d00,0xbdc1,0xbc81,0x7c40,
  0xb401,0x74c0,0x7580,0xb541,0x7700,0xb7c1,0xb681,0x7640,
  0x7200,0xb2c1,0xb381,0x7340,0xb101,0x71c0,0x7080,0xb041,
  0x5000,0x90c1,0x9181,0x5140,0x9301,0x53c0,0x5280,0x9241,
  0x9601,0x56c0,0x5780,0x9741,0x5500,0x95c1,0x9481,0x5440,
  0x9c01,0x5cc0,0x5d80,0x9d41,0x5f00,0x9fc1,0x9e81,0x5e40,
  0x5a00,0x9ac1,0x9b81,0x5b40,0x9901,0x59c0,0x5880,0x9841,
  0x8801,0x48c0,0x4980,0x8941,0x4b00,0x8bc1,0x8a81,0x4a40,
  0x4e00,0x8ec1,0x8f81,0x4f40,0x8d01,0x4dc0,0x4c80,0x8c41,
  0x4400,0x84c1,0x8581,0x4540,0x8701,0x47c0,0x4680,0x8641,
  0x8201,0x42c0,0x4380,0x8341,0x4100,0x81c1,0x8081,0x4040,
};



static __inline uint16_t rshiftu16(uint16_t value, int nb)
{
  return (uint16_t)((value >> nb) & ~((( uint16_t) 0x8000) >> (nb-1)));
}



uint16_t crc16_calc(unsigned char *q, int len)
{
  uint16_t crc = 0xffff;
  while (len-- > 0)
    crc=(rshiftu16(crc,8) ^ CRC16[(crc ^ *q++) & 0xff]);
  return crc;
}
```

6 Revisions

Rev.	Descriptions	by	Release Date
0.01	- Draft Release	C.J.	2019-04-28
1.00	- Preliminary New Release (Chinese)	H.C.B.	2019-12-12
1.01	- Format update - English Release	K.C.	2020-04-11
1.02	- 2.1 section add basic structure and notes - 3.6.2 refine descriptions and parameter	K.C.	2020-05-14