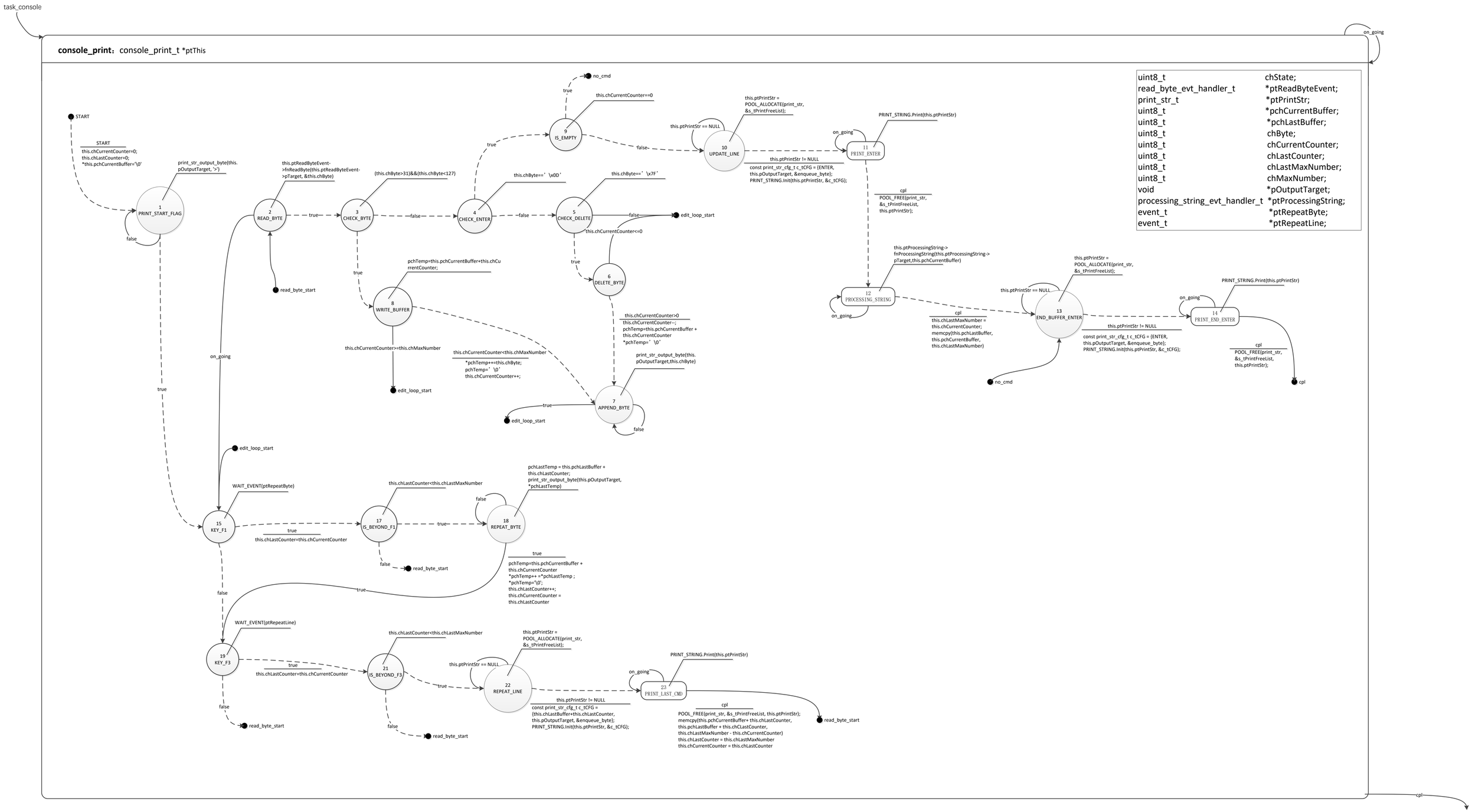
task_console
console_print：console_print_t *ptThis
on_going
START
START
this.chCurrentCounter=0;
this.chLastCounter=0;
*this.pchCurrentBuffer='\0'
1
PRINT_START_FLAG
false
true
print_str_output_byte(this.
pOutputTarget, '>')
2
READ_BYTE
true
read_byte_start
this.ptReadByteEvent-
>fnReadByte(this.ptReadByteEvent-
>pTarget, &this.chByte)
3
CHECK_BYTE
false
(this.chByte>31)&&(this.chByte<127)
true
8
WRITE_BUFFER
pchTemp=this.pchCurrentBuffer+this.chCu
rrentCounter;
this.chCurrentCounter>=this.chMaxNumber
edit_loop_start
4
CHECK_ENTER
false
true
9
IS_EMPTY
true
no_cmd
this.chCurrentCounter==0
false
this.chByte=='\x0D'
5
CHECK_DELETE
false
edit_loop_start
this.chCurrentCounter<=0
this.chByte=='\x7F'
true
6
DELETE_BYTE
this.chCurrentCounter>0
this.chCurrentCounter--;
pchTemp=this.pchCurrentBuffer +
this.chCurrentCounter
*pchTemp='\0'
this.chCurrentCounter<this.chMaxNumber
*pchTemp++=this.chByte;
pchTemp='\0'
this.chCurrentCounter++;
print_str_output_byte(this.
pOutputTarget,this.chByte)
7
APPEND_BYTE
true
edit_loop_start
false
10
UPDATE_LINE
this.ptPrintStr == NULL
this.ptPrintStr =
POOL_ALLOCATE(print_str,
&s_tPrintFreeList);
this.ptPrintStr != NULL
const print_str_cfg_t c_tCFG = {ENTER,
this.pOutputTarget, &enqueue_byte};
PRINT_STRING.Init(this.ptPrintStr, &c_tCFG);
11
PRINT_ENTER
on_going
PRINT_STRING.Print(this.ptPrintStr)
cpl
POOL_FREE(print_str,
&s_tPrintFreeList,
this.ptPrintStr);
12
PROCESSING_STRING
on_going
this.ptProcessingString:>
fnProcessingString(this.ptProcessingString:>
pTarget,this.pchCurrentBuffer)
cpl
this.chLastMaxNumber =
this.chCurrentCounter;
memcpy(this.pchLastBuffer,
this.pchCurrentBuffer,
this.chLastMaxNumber)
13
END_BUFFER_ENTER
this.ptPrintStr == NULL
this.ptPrintStr =
POOL_ALLOCATE(print_str,
&s_tPrintFreeList);
this.ptPrintStr != NULL
const print_str_cfg_t c_tCFG = {ENTER,
this.pOutputTarget, &enqueue_byte};
PRINT_STRING.Init(this.ptPrintStr, &c_tCFG);
no_cmd
14
PRINT_END_ENTER
on_going
PRINT_STRING.Print(this.ptPrintStr)
cpl
POOL_FREE(print_str,
&s_tPrintFreeList,
this.ptPrintStr);
cpl
uint8_t chState;
read_byte_evt_handler_t *ptReadByteEvent;
print_str_t *ptPrintStr;
uint8_t *pchCurrentBuffer;
uint8_t *pchLastBuffer;
uint8_t chByte;
uint8_t chCurrentCounter;
uint8_t chLastCounter;
uint8_t chLastMaxNumber;
uint8_t chMaxNumber;
void *pOutputTarget;
processing_string_evt_handler_t *ptProcessingString;
event_t *ptRepeatByte;
event_t *ptRepeatLine;
on_going
edit_loop_start
WAIT_EVENT(ptRepeatByte)
15
KEY_F1
true
this.chLastCounter=this.chCurrentCounter
17
IS_BEYOND_F1
true
this.chLastCounter<this.chLastMaxNumber
false
read_byte_start
18
REPEAT_BYTE
false
pchLastTemp = this.pchLastBuffer +
this.chLastCounter;
print_str_output_byte(this.pOutputTarget,
*pchLastTemp)
true
pchTemp=this.pchCurrentBuffer +
this.chCurrentCounter
*pchTemp++ =*pchLastTemp ;
*pchTemp='\0';
this.chLastCounter++;
this.chCurrentCounter =
this.chLastCounter
true
false
WAIT_EVENT(ptRepeatLine)
19
KEY_F3
true
this.chLastCounter=this.chCurrentCounter
21
IS_BEYOND_F3
true
this.chLastCounter<this.chLastMaxNumber
false
read_byte_start
22
REPEAT_LINE
this.ptPrintStr == NULL
this.ptPrintStr =
POOL_ALLOCATE(print_str,
&s_tPrintFreeList);
this.ptPrintStr != NULL
const print_str_cfg_t c_tCFG =
{this.chLastBuffer+this.chLastCounter,
this.pOutputTarget, &enqueue_byte};
PRINT_STRING.Init(this.ptPrintStr, &c_tCFG);
23
PRINT_LAST_CMD
on_going
PRINT_STRING.Print(this.ptPrintStr)
cpl
POOL_FREE(print_str, &s_tPrintFreeList, this.ptPrintStr);
memcpy(this.pchCurrentBuffer+ this.chLastCounter,
this.pchLastBuffer + this.chLastCounter,
this.chLastMaxNumber - this.chCurrentCounter)
this.chLastCounter = this.chLastMaxNumber
this.chCurrentCounter = this.chLastCounter
read_byte_start
cpl
processing_string
processing_string：print_buffer_t *ptThis, uint8_t *pchBuffer
on_going
uint8_t chState;
void *pTarget;
print_str_t *ptPrintStr;
START
this.ptPrintStr == NULL
this.ptPrintStr =
POOL_ALLOCATE(print_str,
&s_tPrintFreeList);
1
INIT_PRINT
this.ptPrintStr != NULL
const print_str_cfg_t c_tCFG = {pchBuffer,
this.pTarget, &enqueue_byte};
PRINT_STRING.Init(this.ptPrintStr, &c_tCFG);
on_going
2
PRINT_BUFFER
PRINT_STRING.Print(this.ptPrintStr)
cpl
POOL_FREE(print_str,
&s_tPrintFreeList,
this.ptPrintStr);
CPL