

START

this.pchTokensArray = pchTokens;

→ CHECK_TOKEN }

_NUMBER

this.hwTokens = hwTokens;

processing_string: void *pThis, uint8_t *pchTokens, uint16_t hwTokens

this.hwTokens>0;

on_going

chState;
*pTarget;

hwTokens;

*ptPrintStr;

uint8_t

uint16_t print_str_t

PRINT_STRING.Print(this.ptPrintStr)

PRINT_TOKEN_ENTER

print_str_output_byte(this.
pTarget,*pchTokens++)

OUTPUT_BYTE

this.ptPrintStr != NULL

const print_str_cfg_t c_tCFG = {ENTER, this.pTarget, &enqueue_byte};

PRINT_STRING.Init(this.ptPrintStr, &c_tCFG);

*pchTokens=='\0'

— — —true — —

this.ptPrintStr =

&s_tPrintFreeList);

POOL_ALLOCATE(print_str,

_END

END_STRING_ENTER

POOL_FREE(print_str, &s_tPrintFreeList, this.ptPrintStr);

this.hwTokens--;

this.ptPrintStr == NULL

processing_string

on_going

chState;

*pTarget;

*ptPrintStr;

uint8_t

print_str_t

PRINT_STRING.Print(this.ptPrintStr)

POOL_FREE(print_str,

&s_tPrintFreeList,

this.ptPrintStr);

void

PRINT_BUFFER

processing_string

processing_string: print_buffer_t *ptThis, uint8_t *pchBuffer

INIT_PRINT

this.ptPrintStr == NULL

this.ptPrintStr =

POOL_ALLOCATE(print_str, &s_tPrintFreeList);

this.ptPrintStr != NULL

const print_str_cfg_t c_tCFG = {pchBuffer,

PRINT_STRING.Init(this.ptPrintStr, &c_tCFG);

this.pTarget, &enqueue_byte};