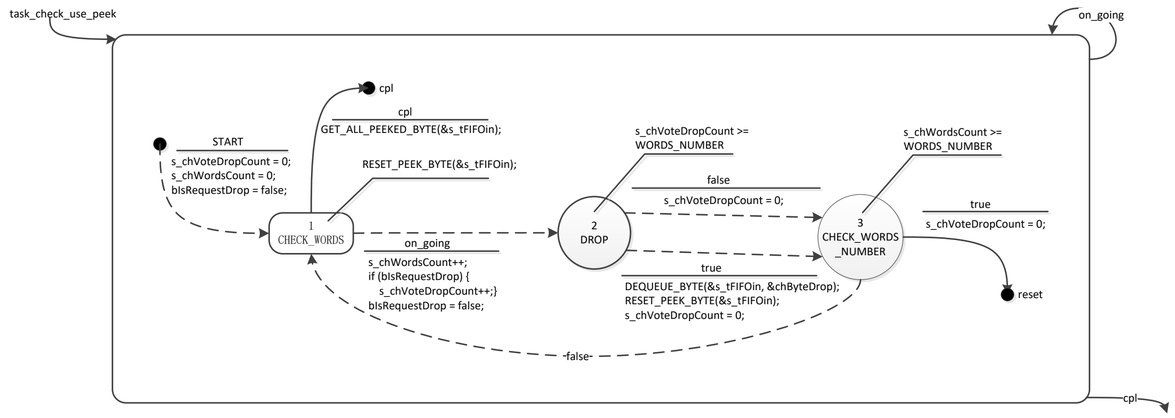
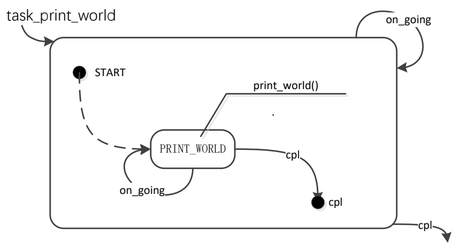
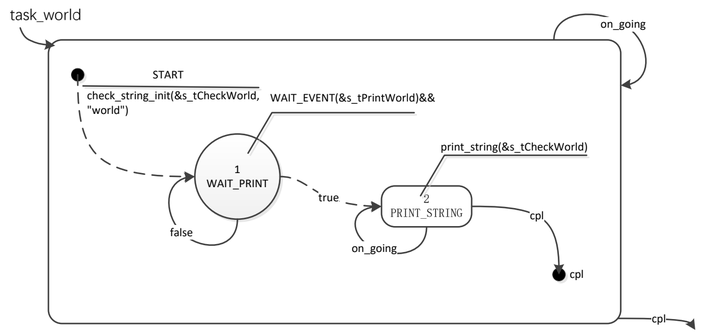
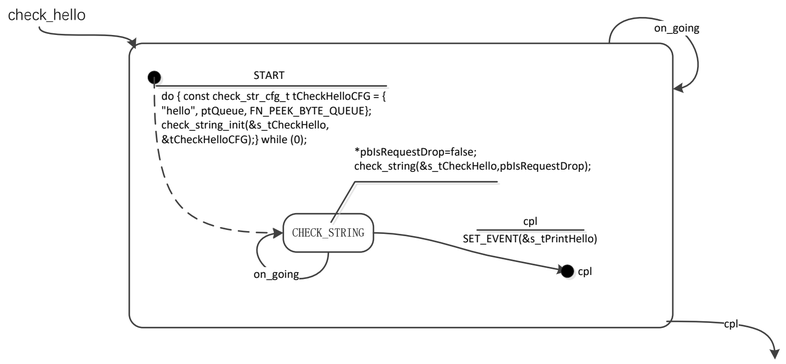
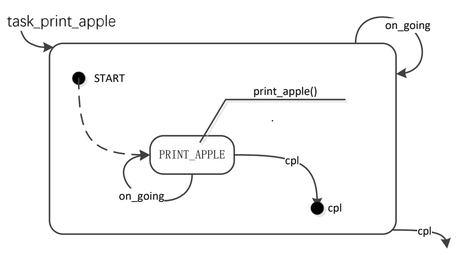
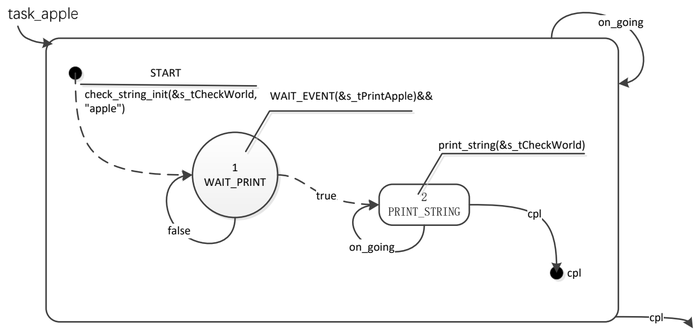

task_check_use_peek
on_going
START
s_chVoteDropCount = 0;
s_chWordsCount = 0;
bIsRequestDrop = false;
cpl
cpl
GET_ALL_PEEKED_BYTE(&s_tFIFOin);
RESET_PEEK_BYTE(&s_tFIFOin);
1
CHECK_WORDS
on_going
s_chWordsCount++;
if (bIsRequestDrop) {
s_chVoteDropCount++;}
bIsRequestDrop = false;
s_chVoteDropCount >=
WORDS_NUMBER
2
DROP
false
s_chVoteDropCount = 0;
true
DEQUEUE_BYTE(&s_tFIFOin, &chByteDrop);
RESET_PEEK_BYTE(&s_tFIFOin);
s_chVoteDropCount = 0;
s_chWordsCount >=
WORDS_NUMBER
3
CHECK_WORDS
_NUMBER
true
s_chVoteDropCount = 0;
reset
-false-
cpl

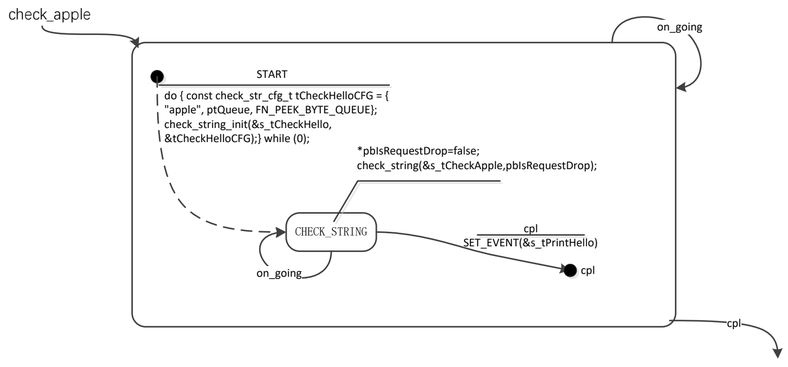

task_print_world
on_going
START
print_world()
PRINT_WORLD
cpl
on_going
cpl
cpl

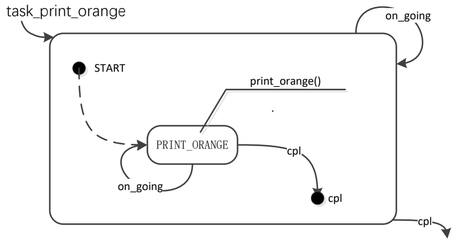

task_world
on_going
START
check_string_init(&s_tCheckWorld,
"world")
WAIT_EVENT(&s_tPrintWorld)&&
1
WAIT_PRINT
false
true
print_string(&s_tCheckWorld)
2
PRINT_STRING
on_going
cpl
cpl
cpl

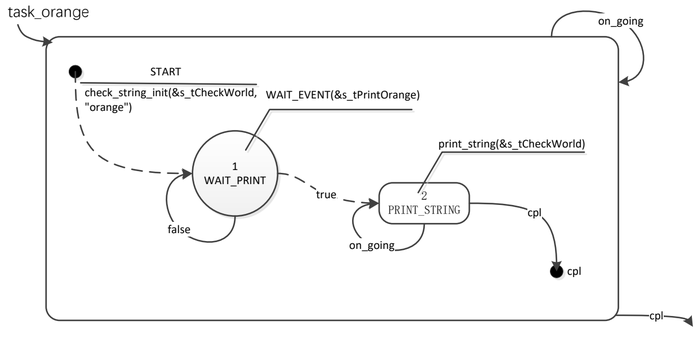

check_hello
on_going
START
do { const check_str_cfg_t tCheckHelloCFG = {
"hello", ptQueue, FN_PEEK_BYTE_QUEUE};
check_string_init(&s_tCheckHello,
&tCheckHelloCFG);;} while (0);
*pbIsRequestDrop=false;
check_string(&s_tCheckHello,pbIsRequestDrop);
CHECK_STRING
on_going
cpl
SET_EVENT(&s_tPrintHello)
cpl
cpl

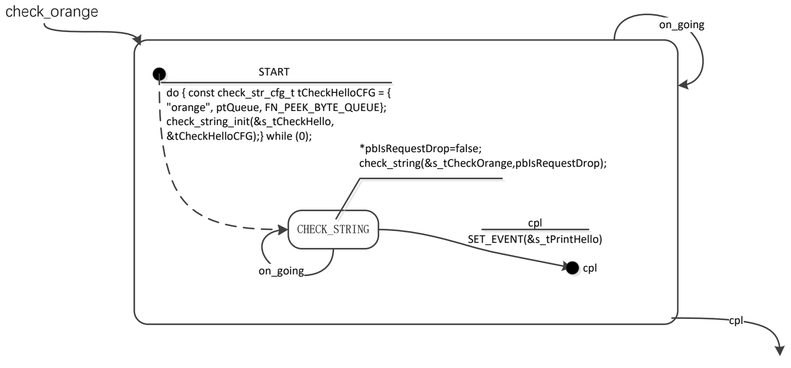

task_print_apple
on_going
START
print_apple()
PRINT_APPLE
cpl
on_going
cpl
cpl

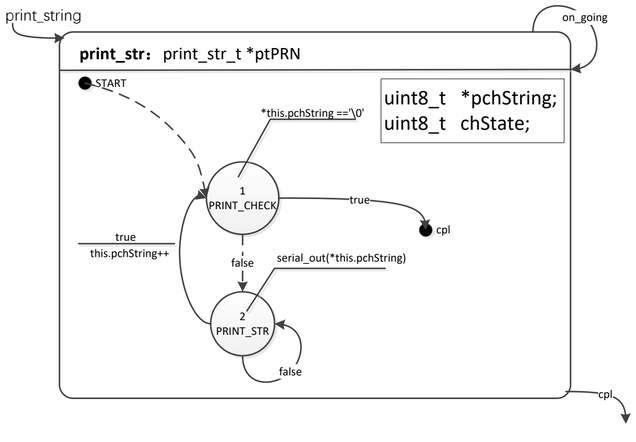

task_apple
on_going
START
check_string_init(&s_tCheckWorld,
"apple")
WAIT_EVENT(&s_tPrintApple)&&
1
WAIT_PRINT
false
true
print_string(&s_tCheckWorld)
2
PRINT_STRING
on_going
cpl
cpl
cpl

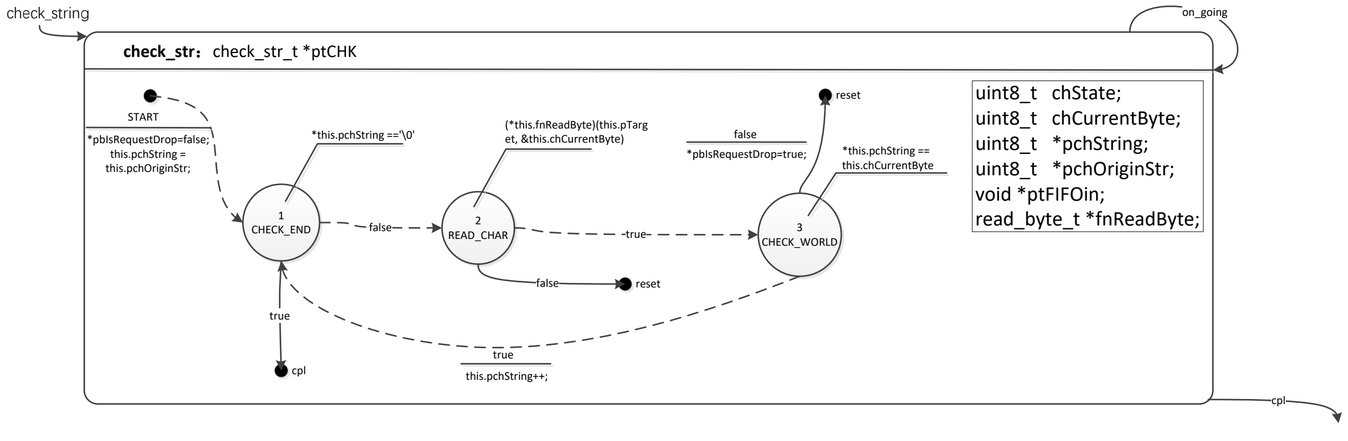

check_apple
on_going
START
do { const check_str_cfg_t tCheckHelloCFG = {
"apple", ptQueue, FN_PEEK_BYTE_QUEUE};
check_string_init(&s_tCheckHello,
&tCheckHelloCFG);;} while (0);
*pbIsRequestDrop=false;
check_string(&s_tCheckApple,pbIsRequestDrop);
CHECK_STRING
on_going
cpl
SET_EVENT(&s_tPrintHello)
cpl
cpl


task_print_orange
on_going
START
print_orange()
PRINT_ORANGE
cpl
on_going
cpl
cpl


task_orange
on_going
START
check_string_init(&s_tCheckWorld,
"orange")
WAIT_EVENT(&s_tPrintOrange)
1
WAIT_PRINT
false
true
print_string(&s_tCheckWorld)
2
PRINT_STRING
on_going
cpl
cpl
cpl


check_orange
on_going
START
do { const check_str_cfg_t tCheckHelloCFG = {
"orange", ptQueue, FN_PEEK_BYTE_QUEUE};
check_string_init(&s_tCheckHello,
&tCheckHelloCFG);;} while (0);
*pbIsRequestDrop=false;
check_string(&s_tCheckOrange,pbIsRequestDrop);
CHECK_STRING
on_going
cpl
SET_EVENT(&s_tPrintHello)
cpl
cpl


print_string
on_going
print_str: print_str_t *ptPRN
START
uint8_t *pchString;
uint8_t chState;
*this.pchString =='\0'
1
PRINT_CHECK
true
cpl
true
this.pchString++
false
serial_out(*this.pchString)
2
PRINT_STR
false
cpl


check_string
on_going
check_str: check_str_t *ptCHK
START
*pbIsRequestDrop=false;
this.pchString =
this.pchOriginStr;
*this.pchString =='\0'
1
CHECK_END
false
(*this.fnReadByte)(this.pTarg
et, &this.chCurrentByte)
2
READ_CHAR
false
reset
true
false
*pbIsRequestDrop=true;
reset
*this.pchString ==
this.chCurrentByte
3
CHECK_WORLD
true
cpl
true
this.pchString++;
uint8_t chState;
uint8_t chCurrentByte;
uint8_t *pchString;
uint8_t *pchOriginStr;
void *ptFIFOin;
read_byte_t *fnReadByte;
cpl