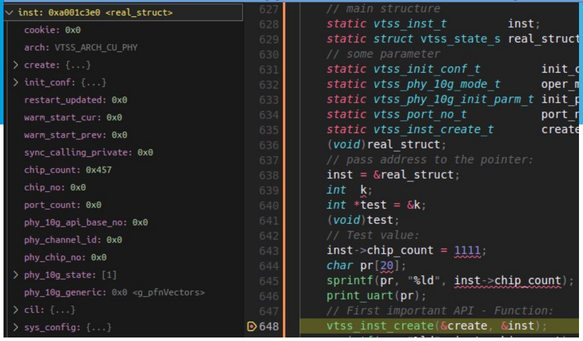
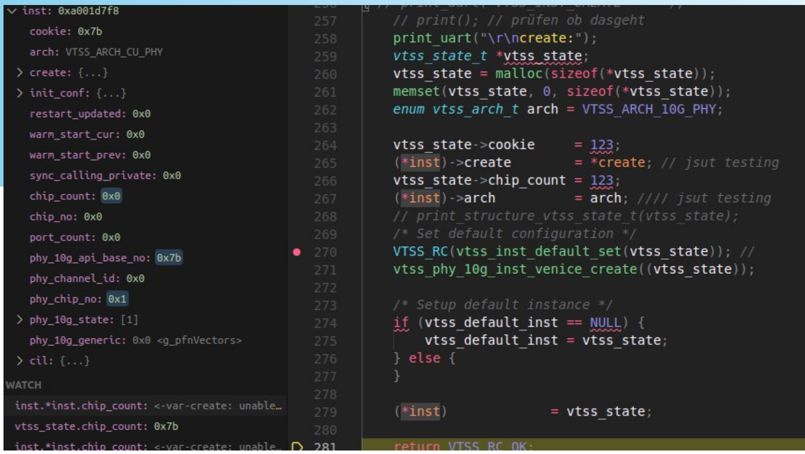
Vor vtss_inst_create():
Static habe ich nach diesem Test weggemacht aber es kommt das gleiche raus….



```
inst: 0xa001c3e0 <real_struct>
  cookie: 0x0
  arch: VTSS_ARCH_CU_PHY
> create: {...}
> init_conf: {...}
  restart_updated: 0x0
  warm_start_cur: 0x0
  warm_start_prev: 0x0
  sync_calling_private: 0x0
  chip_count: 0x457
  chip_no: 0x0
  port_count: 0x0
  phy_10g_api_base_no: 0x0
  phy_channel_id: 0x0
  phy_chip_no: 0x0
> phy_10g_state: [1]
  phy_10g_generic: 0x0 <g_pfnVectors>
> cil: {...}
> sys_config: {...}
```

```c
    // main structure
    static vtss_inst_t          inst;
    static struct vtss_state_s real_struct
    // some parameter
    static vtss_init_conf_t          init_c
    static vtss_phy_10g_mode_t       oper_m
    static vtss_phy_10g_init_parm_t init_p
    static vtss_port_no_t            port_r
    static vtss_inst_create_t        create
    (void)real_struct;
    // pass address to the pointer:
    inst = &real_struct;
    int  k;
    int *test = &k;
    (void)test;
    // Test value:
    inst->chip_count = 1111;
    char pr[20];
    sprintf(pr, "%ld", inst->chip_count);
    print_uart(pr);
    // First important API - Function:
    vtss_inst_create(&create, &inst);
```

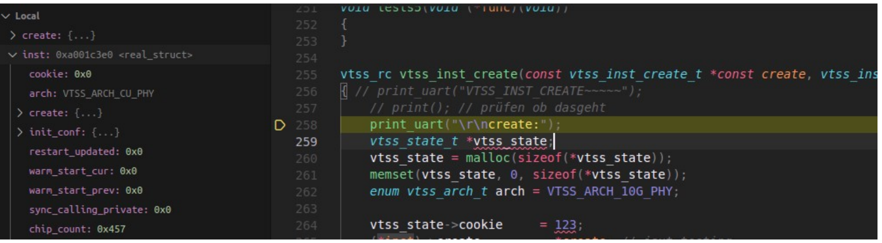
**phy_10g_api_base_no nicht korrekt übernommen = 0xfffffff**



```
inst: 0xa001d7f8
  cookie: 0x7b
  arch: VTSS_ARCH_CU_PHY
> create: {...}
> init_conf: {...}
  restart_updated: 0x0
  warm_start_cur: 0x0
  warm_start_prev: 0x0
  sync_calling_private: 0x0
  chip_count: 0x0
  chip_no: 0x0
  port_count: 0x0
  phy_10g_api_base_no: 0x7b
  phy_channel_id: 0x0
  phy_chip_no: 0x1
> phy_10g_state: [1]
  phy_10g_generic: 0x0 <g_pfnVectors>
> cil: {...}
WATCH
  inst.*inst.chip_count: <-var-create: unable…
  vtss_state.chip_count: 0x7b
  inst.*inst.chip_count: <-var-create: unable…
```

```c
    // print(); // prüfen ob dasgeht
    print_uart("\r\ncreate:");
    vtss_state_t *vtss_state;
    vtss_state = malloc(sizeof(*vtss_state));
    memset(vtss_state, 0, sizeof(*vtss_state));
    enum vtss_arch_t arch = VTSS_ARCH_10G_PHY;

    vtss_state->cookie      = 123;
    (*inst)->create         = *create; // jsut testing
    vtss_state->chip_count = 123;
    (*inst)->arch           = arch; //// jsut testing
    // print_structure_vtss_state_t(vtss_state);
    /* Set default configuration */
    VTSS_RC(vtss_inst_default_set(vtss_state)); //
    vtss_phy_10g_inst_venice_create((vtss_state));

    /* Setup default instance */
    if (vtss_default_inst == NULL) {
        vtss_default_inst = vtss_state;
    } else {
    }

    (*inst)                 = vtss_state;

    return VTSS_RC_OK;
```

**Inst hat jetzt die gleiche Adresse wie vtss_state**



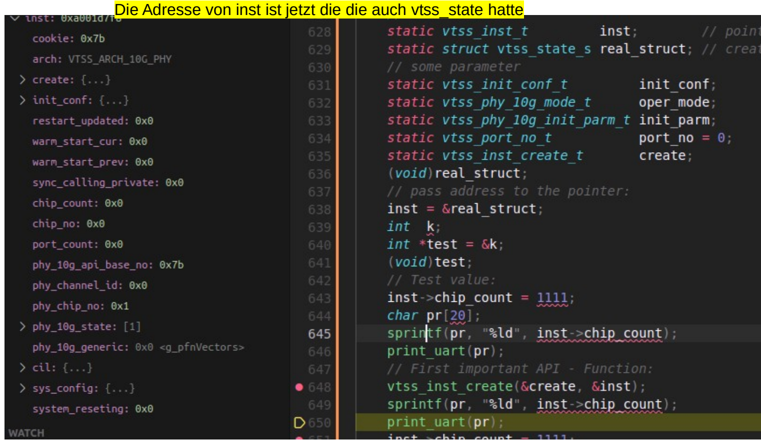
```
Local
> create: {...}
v inst: 0xa001c3e0 <real_struct>
    cookie: 0x0
    arch: VTSS_ARCH_CU_PHY
  > create: {...}
  > init_conf: {...}
    restart_updated: 0x0
    warm_start_cur: 0x0
    warm_start_prev: 0x0
    sync_calling_private: 0x0
    chip_count: 0x457
```

```c
void tests5(void (*func)(void))
{
}

vtss_rc vtss_inst_create(const vtss_inst_create_t *const create, vtss_ins
{ // print_uart("VTSS_INST_CREATE~~~~~");
    // print(); // prüfen ob dasgeht
    print_uart("\r\ncreate:");
    vtss_state_t *vtss_state;
    vtss_state = malloc(sizeof(*vtss_state));
    memset(vtss_state, 0, sizeof(*vtss_state));
    enum vtss_arch_t arch = VTSS_ARCH_10G_PHY;

    vtss_state->cookie      = 123;
```

**Am Anfang von vtss_inst_create()**
chip_count = 1111

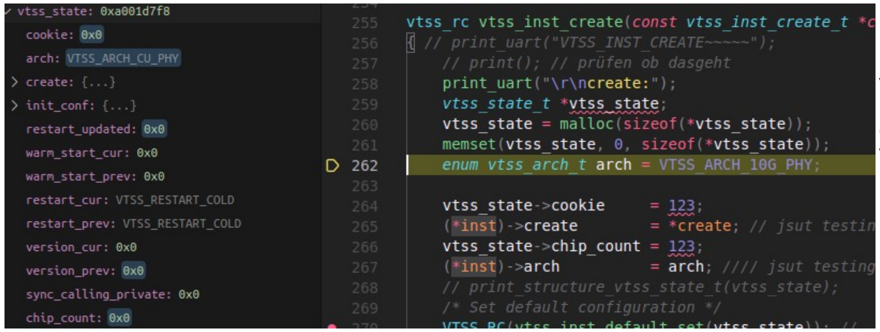
Adresse von inst
vtss_inst_create() 0xa001c3e0

In der Main hat jetzt am Ende phy_10g_api_base_no den Wert den chip_count haben sollte
Chip Count hat dann einfach 0.
Die Adresse von inst ist jetzt die die auch vtss_state hatte



```
inst: 0xa001d7f8
  cookie: 0x7b
  arch: VTSS_ARCH_10G_PHY
> create: {...}
> init_conf: {...}
  restart_updated: 0x0
  warm_start_cur: 0x0
  warm_start_prev: 0x0
  sync_calling_private: 0x0
  chip_count: 0x0
  chip_no: 0x0
  port_count: 0x0
  phy_10g_api_base_no: 0x7b
  phy_channel_id: 0x0
  phy_chip_no: 0x1
> phy_10g_state: [1]
  phy_10g_generic: 0x0 <g_pfnVectors>
> cil: {...}
> sys_config: {...}
  system_reseting: 0x0
WATCH
```

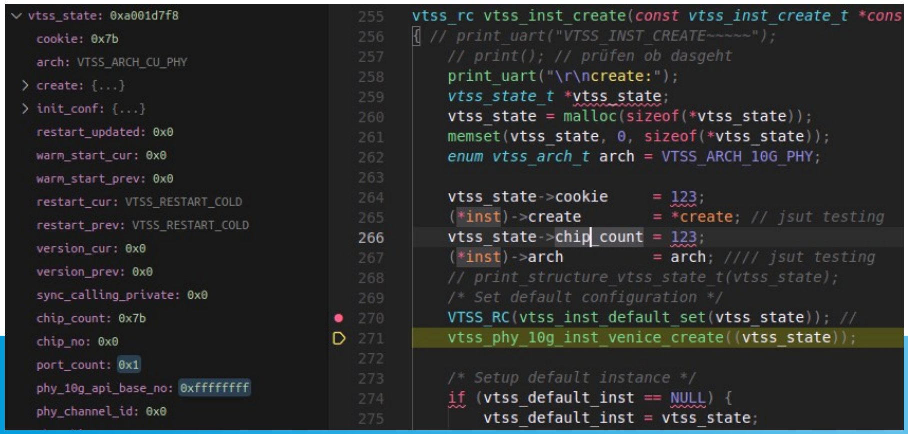
```c
    static vtss_inst_t          inst;          // point
    static struct vtss_state_s real_struct; // creat
    // some parameter
    static vtss_init_conf_t          init_conf;
    static vtss_phy_10g_mode_t       oper_mode;
    static vtss_phy_10g_init_parm_t init_parm;
    static vtss_port_no_t            port_no = 0;
    static vtss_inst_create_t        create;
    (void)real_struct;
    // pass address to the pointer:
    inst = &real_struct;
    int  k;
    int *test = &k;
    (void)test;
    // Test value:
    inst->chip_count = 1111;
    char pr[20];
    sprintf(pr, "%ld", inst->chip_count);
    print_uart(pr);
    // First important API - Function:
    vtss_inst_create(&create, &inst);
    sprintf(pr, "%ld", inst->chip_count);
    print_uart(pr);
    inst->chip_count = 1111;
```



```
vtss_state: 0xa001d7f8
  cookie: 0x0
  arch: VTSS_ARCH_CU_PHY
> create: {...}
> init_conf: {...}
  restart_updated: 0x0
  warm_start_cur: 0x0
  warm_start_prev: 0x0
  restart_cur: VTSS_RESTART_COLD
  restart_prev: VTSS_RESTART_COLD
  version_cur: 0x0
  version_prev: 0x0
  sync_calling_private: 0x0
  chip_count: 0x0
```

```c
vtss_rc vtss_inst_create(const vtss_inst_create_t *c
{ // print_uart("VTSS_INST_CREATE~~~~~");
    // print(); // prüfen ob dasgeht
    print_uart("\r\ncreate:");
    vtss_state_t *vtss_state;
    vtss_state = malloc(sizeof(*vtss_state));
    memset(vtss_state, 0, sizeof(*vtss_state));
    enum vtss_arch_t arch = VTSS_ARCH_10G_PHY;

    vtss_state->cookie      = 123;
    (*inst)->create         = *create; // jsut testin
    vtss_state->chip_count = 123;
    (*inst)->arch           = arch; //// jsut testing
    // print_structure_vtss_state_t(vtss_state);
    /* Set default configuration */
    VTSS_RC(vtss_inst_default_set(vtss_state)); //
```

vtss_state adresse:
notiere dir die **Adresse** von vtss_state.
**0xa001d7f8**
**vtss_state→cookie = 0 am Anfang**

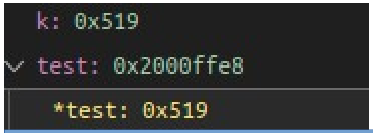


```
vtss_state: 0xa001d7f8
  cookie: 0x7b
  arch: VTSS_ARCH_CU_PHY
> create: {...}
> init_conf: {...}
  restart_updated: 0x0
  warm_start_cur: 0x0
  warm_start_prev: 0x0
  restart_cur: VTSS_RESTART_COLD
  restart_prev: VTSS_RESTART_COLD
  version_cur: 0x0
  version_prev: 0x0
  sync_calling_private: 0x0
  chip_count: 0x7b
  chip_no: 0x0
  port_count: 0x1
  phy_10g_api_base_no: 0xffffffff
  phy_channel_id: 0x0
```

```c
vtss_rc vtss_inst_create(const vtss_inst_create_t *cons
{ // print_uart("VTSS_INST_CREATE~~~~~");
    // print(); // prüfen ob dasgeht
    print_uart("\r\ncreate:");
    vtss_state_t *vtss_state;
    vtss_state = malloc(sizeof(*vtss_state));
    memset(vtss_state, 0, sizeof(*vtss_state));
    enum vtss_arch_t arch = VTSS_ARCH_10G_PHY;

    vtss_state->cookie      = 123;
    (*inst)->create         = *create; // jsut testing
    vtss_state->chip_count = 123;
    (*inst)->arch           = arch; //// jsut testing
    // print_structure_vtss_state_t(vtss_state);
    /* Set default configuration */
    VTSS_RC(vtss_inst_default_set(vtss_state)); //
    vtss_phy_10g_inst_venice_create((vtss_state));

    /* Setup default instance */
    if (vtss_default_inst == NULL) {
        vtss_default_inst = vtss_state;
```

setze einen Watchpoint
auf vtss_state→chip_count
**Ist „Breakpoint on value change“ ok?**

**vtss_state→cookie = 123 = 0x7b**
**Cookie = 0x7b**
**phy_10g_api_base_no wurde korrekt durch die Funktion gesetzt 0xfffffff**



```
k: 0x519
v test: 0x2000ffe8
    *test: 0x519
```

K ist die Adresse von der Variablen k
Test selber hat die Adresse 0x2000ffe8