Project II

Hes·so VALAIS WALLIS
Haute Ecole d'Ingénierie
Hochschule für Ingenieurwissenschaften

Software Implementation for Embedded Systems

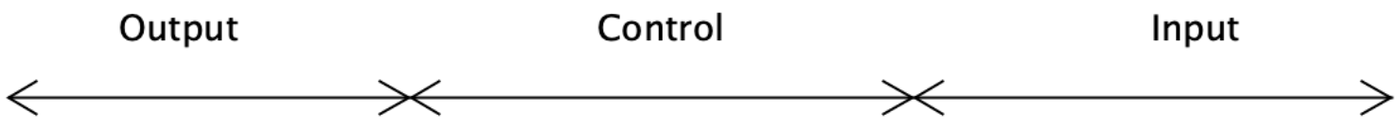
Pascal Sartoretti

Medard Rieder

HES-SO Valais, 2021

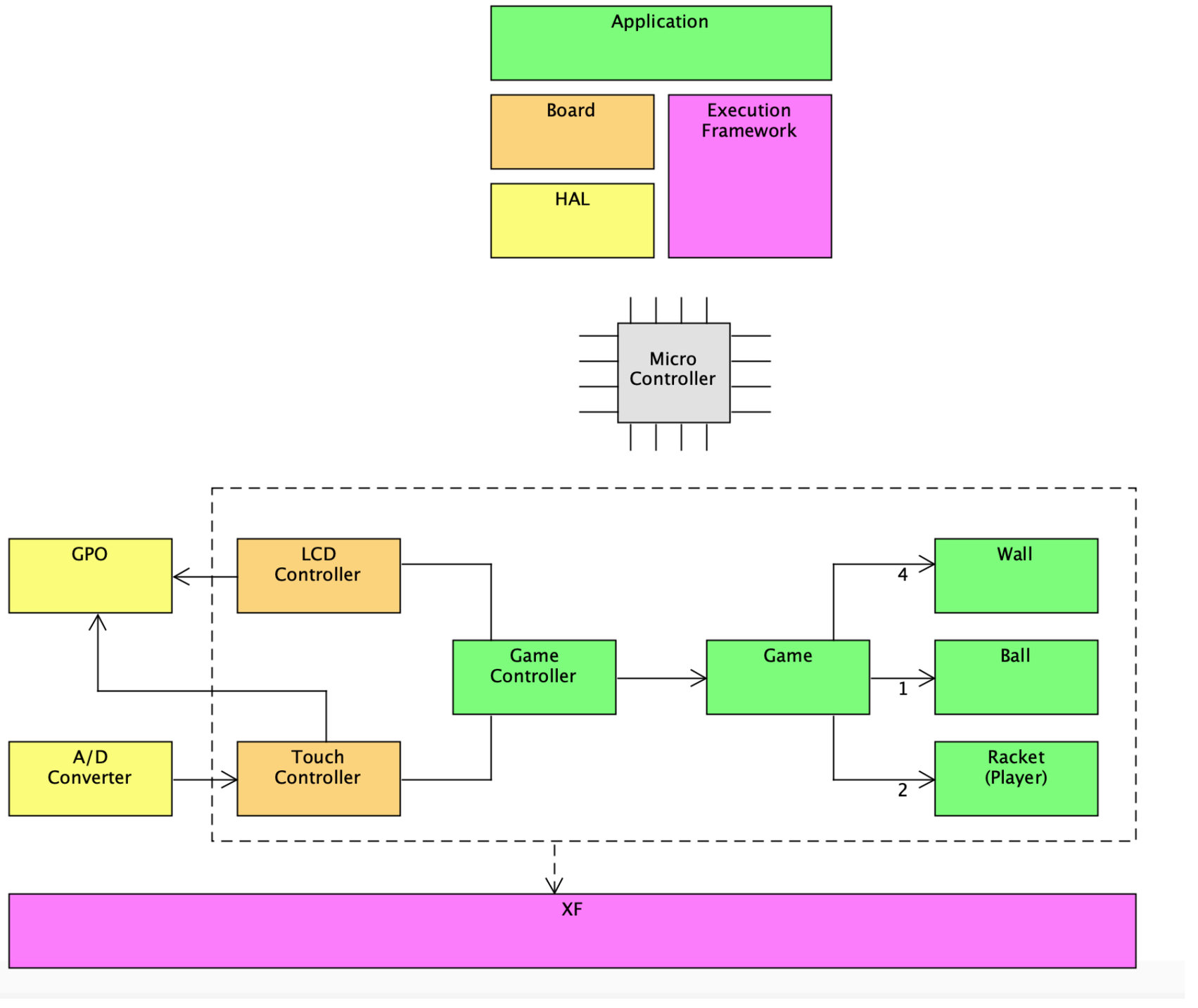
Hardware

System



Proposed Architecture

System

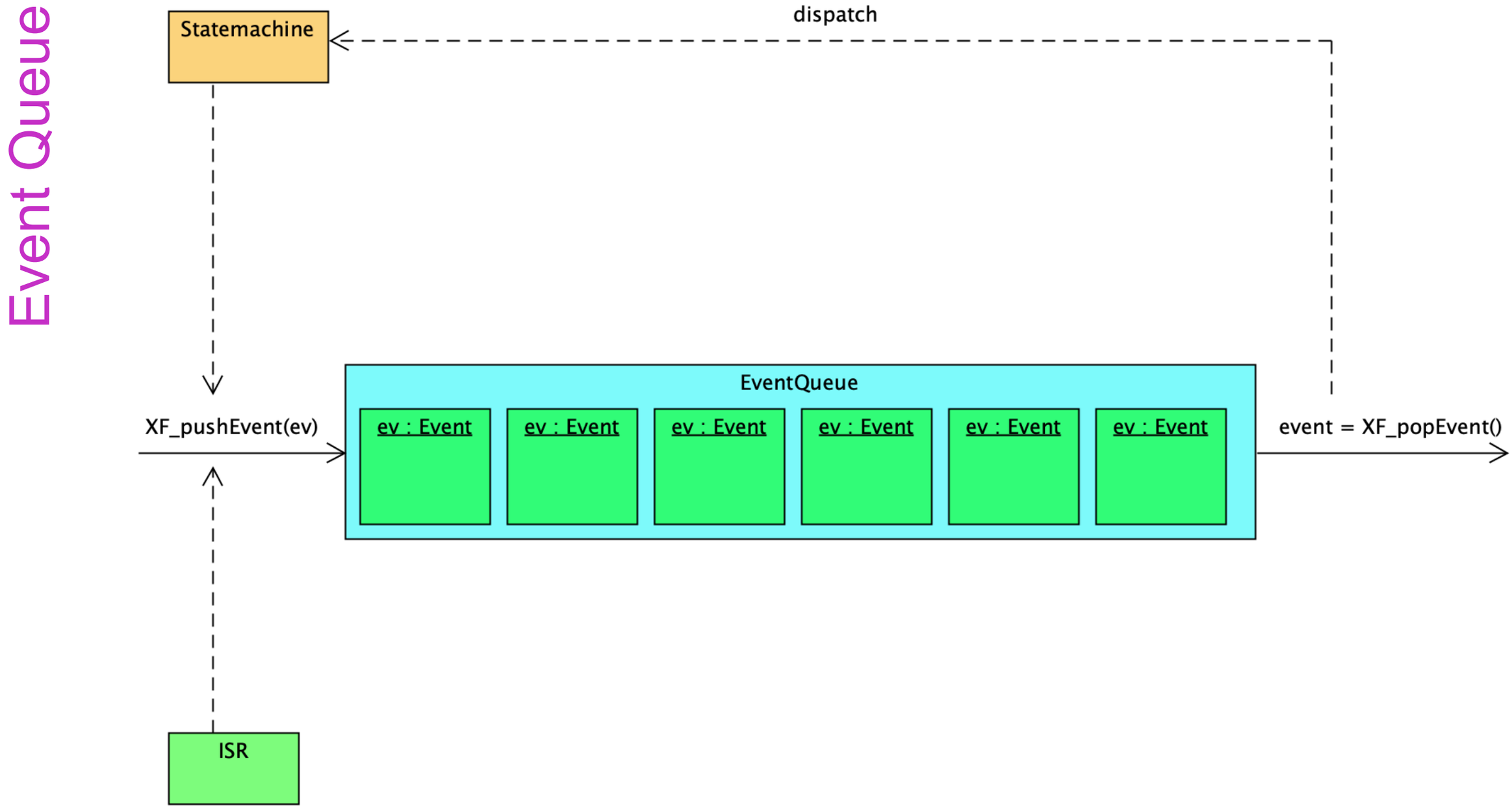


XF Services

XF

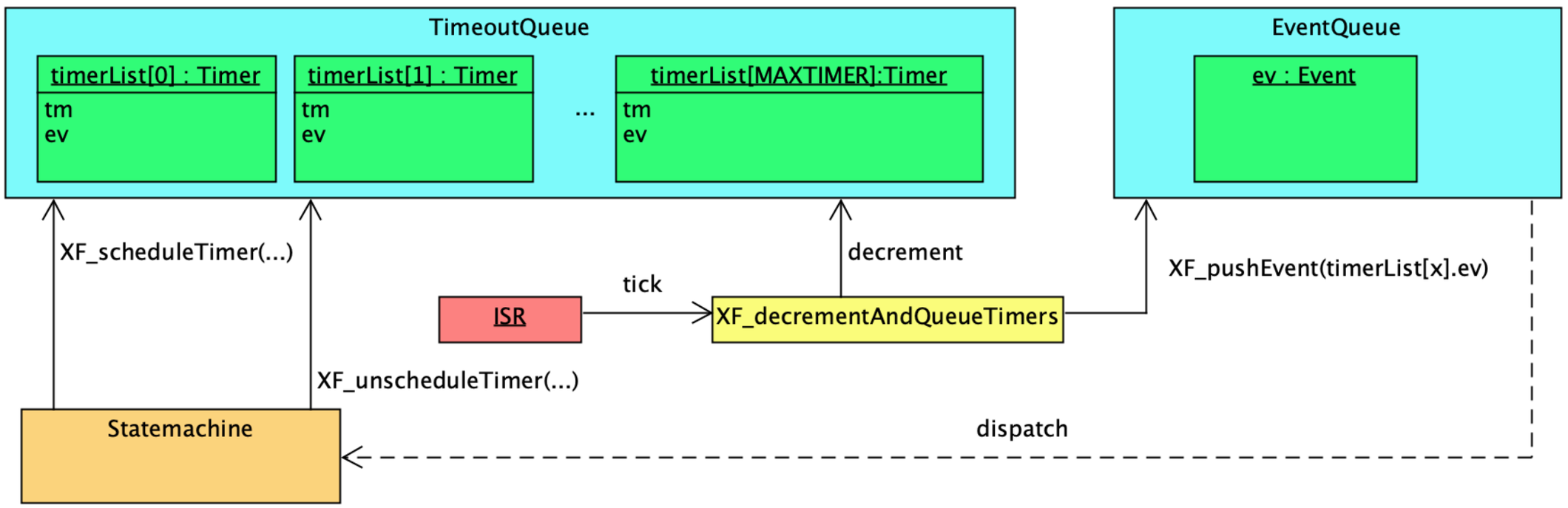
- XF is like an OS:
  - events
  - timers
  - critical sections
- Services for events
  - XF_popEvent
  - XF_pushEvent
- Services for timers
  - XF_scheduleTimer
  - XF_unscheduleTimer
- Services for protection
  - ENTERCRITICAL
  - LEAVECRITICAL

Events

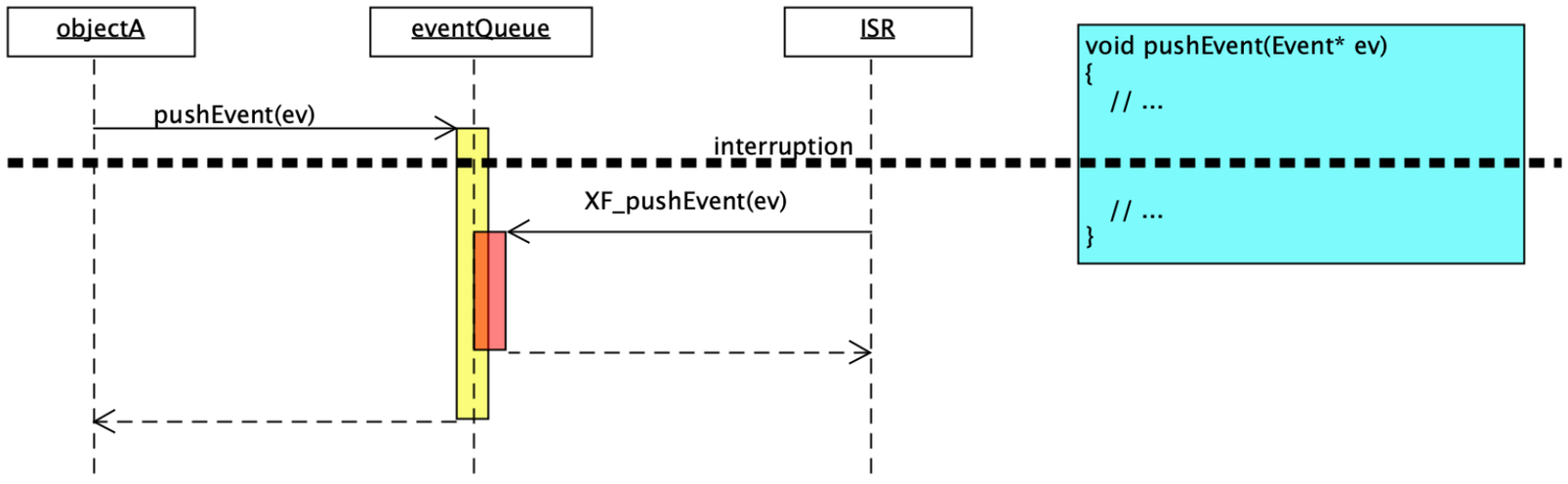


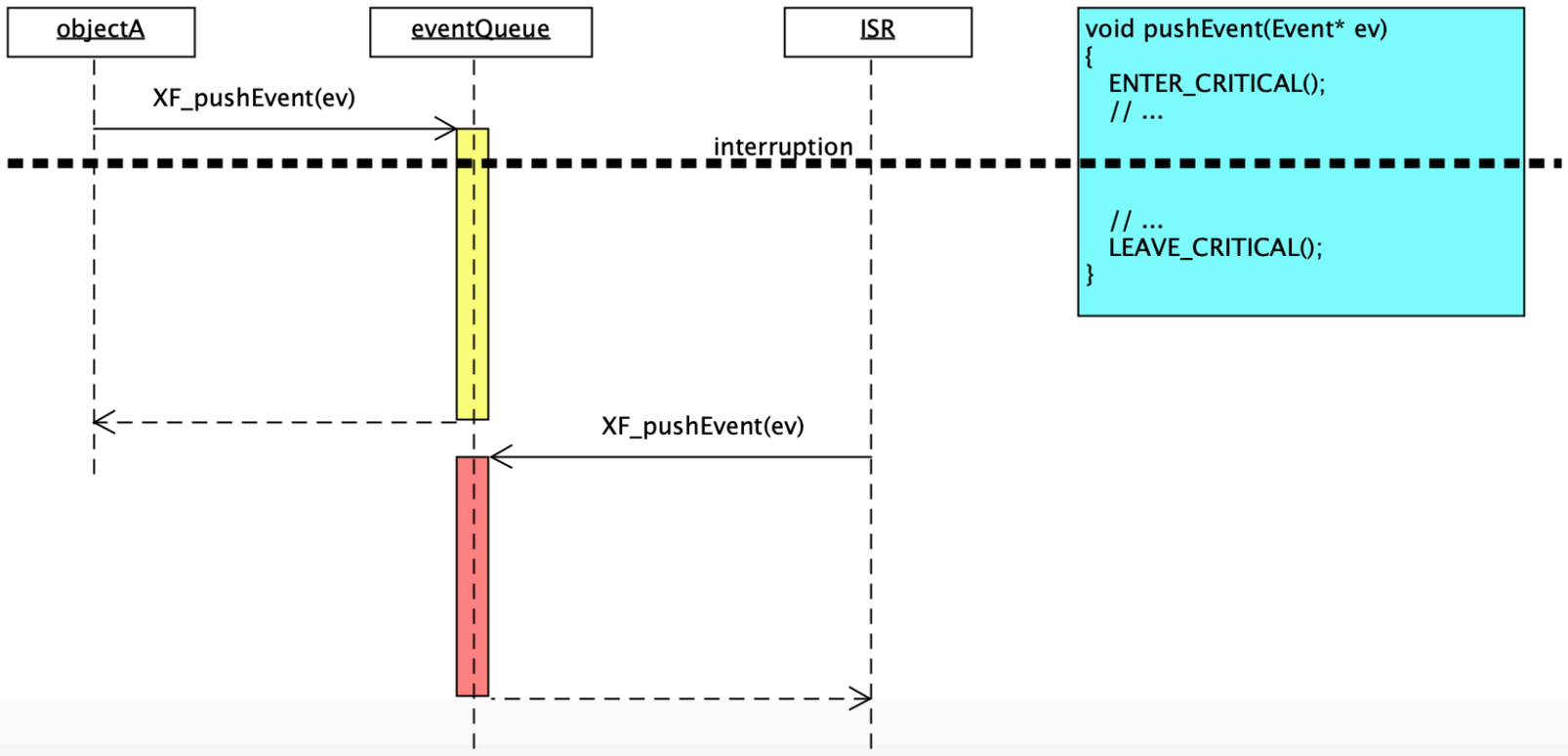
Timers

Timer Queue



Critical Sections





State Machine Pattern

Pattern for SM implementation using C

- Define enum with all states
- Define SM-variable using enum-type
- Set SM-Variable to initial state value in the INIT-section of I-C-O
- Implement SM using a switch structure in order to control SM advancement
- Implement SM using a switch structure in order to implement actions
- Implement SM using the XF interface in order to control events and timers
- Each SM is a function with one parameter of type event

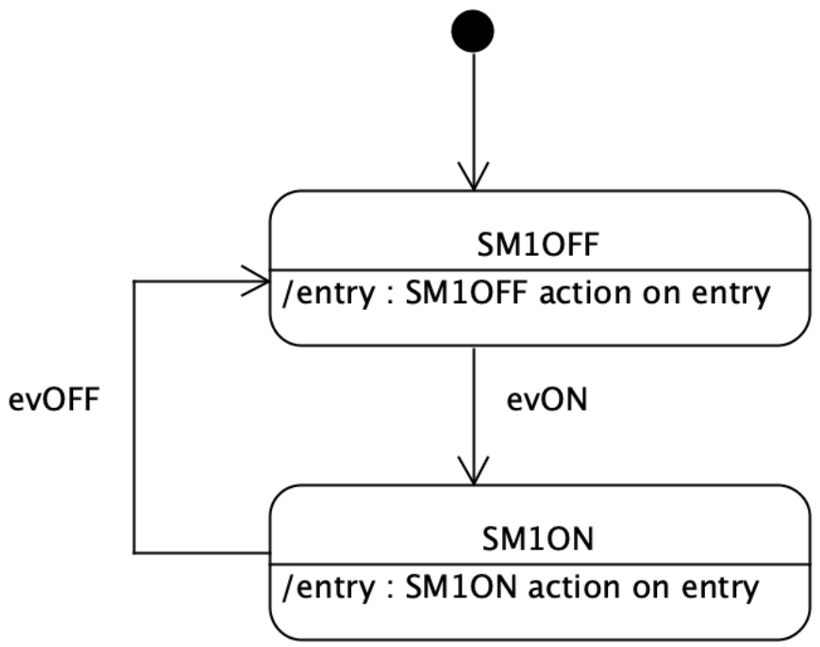
SM Example

Code

```
typedef enum SM1 {SM1ON, SM1OFF} SM1;
SM1 sm1, oldsm1;

//init section
sm1 = SM1OFF;

//function
void f_sm1(Event e) {
  //advancement control
  oldsm1 = sm1;
  switch (sm1) {
    case: SM1OFF
      if (e == evON)
        sm1 = SM1ON;
      break;
    case: SM1ON
      if (e == evOFF)
        sm1 = SM1OFF;
      break;
  }
  // action on entry control
  switch (sm1) {
    case: SM1OFF
      if (sm1old == SM1ON)
        //do SM1OFF action on entry
      break;
    case: SM1ON
      if (sm1old == SM1OFF)
        //do SM1ON action on entry
      break;
```



Classes in C

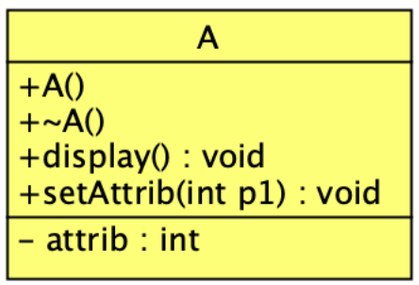
Class - Pattern

Pattern for class implementation using C

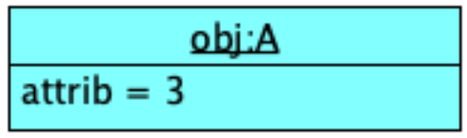
- Attributes of class = struct
- Methods of class = function with first parameter of type struct (this pointer)
- Constructor = function that dynamically creates a struct, passes it to the init-function and finally returns a pointer to the struct
- Destructor = function that passes the struct to the cleanup function and the releases the struct.
- Objects = variables of type struct
- method call = call of function and pass reference to struct type variable

Class Example


```
struct A
{
    int attrib;
}
void A_init(struct A* me)
{
  me->attrib = 0;
}
struct A* A_create()
{
  struct A* p;
  malloc(p,sizeof(struct A));
  if (p)
  {
    A_init(p);
  }
  return p;
}
```




```
void A_display(struct A* me)
{
    printf("%d", me->attrib);
}
void A_setAttrib(struct A* me,int p1)
{
  me->attrib = p1;
}
void A_destroy(struct A* me)
{
  free(me);
}
//use the class A
struct A* obj;
obj = A_create();
A_setAttrib(obj,3);
A_display(obj);
A_destroy(obj);
```



Other Stuff

Interrupts

Other Stuff

- Convert interrupts in ISR to events
- Keep ISR as small as possible
- Do not forget that ISR stop execution of "main-land" code
- Be aware: ISR may use same memory as you are already using in main-land
- ISR can be considered as a very low level HAL
- NEVER allocate dynamic memory from within ISR code
- Use static, preconfigured events to convert interrupts to events

Test Patterns

Most Different Patterns

- Patterns
  - Smoke testing
  - Exploratory Testing
  - Black box testing
  - White box testing

Smoke Test

- Quick test to see if software is operational
  - Idea comes from hardware realm – turn power on and see if smoke pours out
  - Generally simple and easy to administer
  - Makes no attempt or claim of completeness
- Good for catching catastrophic errors
  - Especially after a new build or major change
  - Exercises any built-in internal diagnosis mechanisms
- But, not usually a thorough test
  - More a check that many software components are "alive"

Exploratory Testing

- A person exercises the system, looking for unexpected result
  - Might or might not be using documented system behavior as a guide
  - Is especially looking for "strange" behaviors that are not specifically required nor prohibited by the requirements
- Advantages
  - An experienced, thoughtful tester can find many defects this way
  - Often, the defects found are ones that would have been missed by more rigid testing methods
- Disadvantages
  - Usually no documented measurement of coverage
  - Can leave big holes in coverage due to tester bias/blind spots
  - An inexperienced, non-thoughtful tester probably won't find the important bugs

Black Box Testing

- Tests designed with knowledge of behavior
  - But without knowledge of implementation
  - Often called "functional" testing
- Idea is to test what software does, but not how function is implemented
- Advantages:
  - Tests the final behavior of the software
  - Can be written independent of software design
    - Less likely to overlook same problems as design
  - Can be used to test different implementations with minimal changes
- Disadvantages:
  - Doesn't necessarily know the boundary cases
    - For example, won't know to exercise every lookup table entry
  - Can be difficult to cover all portions of software implementation

White Box Testing

- Tests designed with knowledge of software design
  - Often called "structural" testing
- Idea is to exercise software, knowing how it is designed
- Advantages:
  - Usually helps getting good coverage (tests are specifically designed for coverage)
  - Good for ensuring boundary cases and special cases get tested
- Disadvantages:
  - 100% coverage tests might not be good at assessing functionality for "surprise" behaviors and other testing goals
  - Tests based on design might miss bigger picture system problems
  - Tests need to be changed if implementation/algorithm changes

Testing

How To Test

- Test situations with applications of patterns
  - Unit test
  - Subsystem test
  - System integration test
  - Acceptance test
  - Beta test

People Involved To Test Scenarios

- Different people play roles of tester:
  - Programmer (YOU) often does own testing for
    - unit tests
  - Independent testers (OTHER GROUP) are often involved in
    - subsystem test
    - system test
    - acceptance tests
  - Customers (MR. SARTORETTI) are often involved in
    - acceptance tests
    - beta tests

Testing

- Unit test
  - smoke and white box pattern, programmer on development system
- Subsystem test
  - white box pattern, programmer on different development systems
- System integration test
  - black box pattern, programmer and test persons on development and test systems
- Acceptance test
  - smoke and black box pattern, test person on virgin system
- Beta test
  - exploratory, any person on any system

Test template

HOW TO

Test ID	Give each Test an unique name or ID
Description	Precisely describe what this test is supposed to test.
Pattern	Select a test Pattern (smoke, exploratory, black box, white box)
Test prescription	Precisely describe how this test has to be executed. Describe in which state the DUT has to be at the begin of the test(device under test) has to be at the begin of the test. Describe tools or other materials needed for the test.
Personel	Describe which persons are involved in the test and also describe the exact role of each person during this test.
Results	Describe the results of the test in a completely factual manner.
Conclusions	Make a concluding statement concerning the test and its results
Measures	Describe eventual measures to be taken. Also describe who has to take which measure. Do not forget to give a due date for each measure.

Test Example

HOW TO

Test ID	HW 03
Description	This test has to verify if the PCB has no short cuts
Pattern	white box
Test prescription	The PCB is connected to a laboratory power supply. The power supply has the voltage regulated to 3 Volts and the current s limited to 20 mA. The power leads are connected to the corresponding battery clips. Power is applied to the board. If a short circuit is shown, the power is switched off immediately. If no short circuit is seen, the current is documented. Power is switched off. This ends the test.
Personel	The test is run by Pascal
Results	No short circuit has been observed. The current consumption was 7 mA.
Conclusions	The consumption of 7 mA is maybe due to the micro controller that is not yet configured. The test shows that there is principally no problem with the PCB.
Measures	Since this test is good, testing continues to test HW 04