Exception handling presentation

EXCEPTION HANDLING PRESENTATION

NEONSEVEN

Purpose of the exception handler

- It is a debug mechanism that enables the SW designers to identify abnormal behaviours run-time.
- It is a very powerful feature in the integration phase of a development process.
- It has the capability of logging different types of debug information to track down the source of a given error situation.
- There are 3 different trap/exceptions categories:

1. Hardware generated traps
2. Software generated traps
3. Software raised exceptions

Hardware traps

- Hardware traps are issued by faults or specific system states that occur runtime
- A hardware trap results in an immediate termination of the code execution; a trap log is stored in NV-RAM.
- The E-GOLDradio is capable of detecting eight different hardware trap events, divided into two categories:

1. Class A HW traps.
2. Class B HW traps.

NEONSEVEN

Class A traps

- Share the same priority level
- Each trap has its own individual vector location (ISR)
- Code execution may continue normally (but it's not advised)
- If more Class A trap occurs at a same time, they are prioritized internally
- In order of priority:

1. NMI (Non-Maskable Interrupt)
2. STKOF (Stack Overflow)
3. STKUF (Stack Underflow)

NEONSEVEN

Class B traps

- Share the same priority level
- Share the same vector location (ISR)
- The TFR register allows trap identification and prioritization
- The interrupted instruction flow cannot be restored

1. UNDOPC (Undefined Opcode)
2. PRTFLT (Protected Instruction Fault)
3. ILLOPA (Illegal Word Operand Access)
4. ILLINA (Illegal Instruction Access)
5. ILLBUS (Illegal External Bus Access)

NEONSEVEN

SW traps

- Identified by the SW designer on an implementation time basis, where unrecoverable errors could occur.
- Result in an immediate termination of the code execution: a unique identification number is stored in NV-RAM

Example:

```
while(!HW_REGISTER && counter) {
  do_something();
  if(--counter == 0)
    TRAP_envoke_sw_trap(UNIQUE_IDENTIFICATION);
}
```

NEONSEVEN

SW raised exceptions

- Identified by the software designer on an implementation time basis where unforeseen but recoverable errors could occur.
- Do not terminate the code execution: the unique identification number is merely stored in NV-RAM.

Example:

```
switch(event) {
case x: execute_X(); break;
case y: execute_Y(); break;
default: raise_exception(UNIQUE_ID); break;
}
```

NEONSEVEN

Trap storage

1. The trap type: 0xAAAA for HW class A, 0xBBBB for HW class B, 0xCCCC for SW traps, 0xDDDD for SW exceptions
2. The TRAP Flag Register TRF
3. The Unique Identification number
4. The system stack (PSW, CSP and IP)
5. The date and time of the occurrence of the trap
6. 20 bytes of user specified data (the array *TRAP_log_data)*

- TFR is only meaningful for HW traps, Id and TRAP_log_data for SW exceptions only

NEONSEVEN

Silent reset

- handles an unrecoverable error (trap) where the only alternative is to switch off the phone, which is not suitable for production version
- Performs a seamless restart of the MS
- The SIM is not reset, so the current access level will remain
- Requires proper implementation in the MMI

NEONSEVEN

Trap interface

1. TRAP_class_a_handling: Class A ISR
2. TRAP_class_b_handling: Class B ISR
3. TRAP_envoke_sw_trap: forces a trap by setting NMI
4. TRAP_get_exception_store: retrieves the logged exceptions
5. TRAP_store_exception: saves an exception in RAM (stored in non-volatile memory at power down)
6. TRAP_check_for_silent_reset: returns TRUE if the power up was caused by a Silent Reset