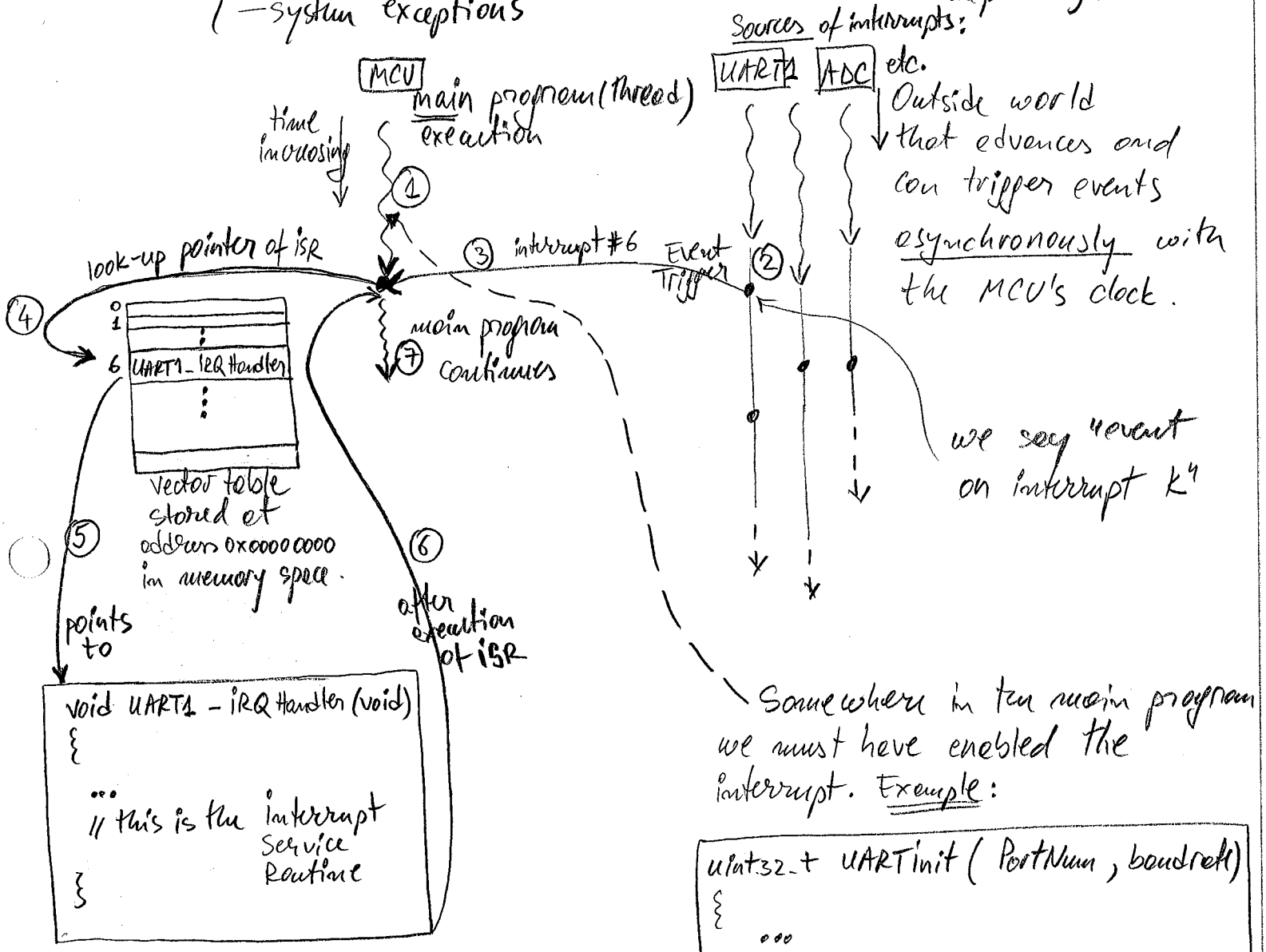
Exceptions taxonomy:

- external **interrupts** (i.e., interrupts are a special class of exceptions).
- system exceptions




```
void UART1 - IRQ Handler (void)
{
  ...
  // this is the interrupt Service Routine
}
```

```
uint32_t UARTinit ( PortNum , baudrate)
{
   ...
   NVIC_EnableIRQ (UART1_IRQn);
   LPC_UART1->IER = IER_THRE;
   ...
}
```

Concurrency is the keyword!

Example analogy/parallel:

You sitting in your chair and your mom asking you to take the garbage out. What is the ISR?

Example 1: UART1

main program:

```
while(1) {  // executed indefinitely!
    if count ≠ 0 {
        // Disable receiving
        Transmit back to PC
        // Enable receiving
    }
}
```

time →

Init

initially count=0

modified by ISR and send function!

count = 5

H	e	l	l	o

Buffer of 40 elements

ISR

UART1 RDA (receive data available) RLS and RDR (receive data ready)

event: we typed "Hello" on PC. This triggered the interrupt.

Reset by main

count = 0

H	e	l	l	o

send back to PC one by one!

UARTSend()

THRE ISR → UART1TxEmpty = 1;

RX

PC terminal

while(!(UART1TxEmpty & 0x01)); // do nothing

{LPC_UART1->THR = *BufferPtr; // send

UART1TxEmpty = 0;}

also modified by ISR & send function!

count = 4

"left over"

C	R	I	S	

event

Type "CRIS" on PC.

set to 0 and then stays like this to block placing another character until UART1TxEmpty = 1 by UART1_IRQHandler()! (ISR)

Notes:

- we enable UART1 as "source of interrupts" by: NVIC_EnableIRQ(UART1_IRQn);
- LPC_UART1->IER = (IER_RBR | IER_RLS | IER_THRE);

Used to disable/enable Receive Data Available Interrupt for UART

bit 0x01 Bit 0x04 Bit 0x02

Individual interrupt enable bits for the 7 potential UART interrupts!

Interrupt Enable Register (IER)

Receiver Buffer Register (RBR) page 302

Receive Line Status (RLS) page 302

Transmit Holding Register Empty (THRE) page 302

void UART1_IRQHandler(void) { // this is the interrupt service routine (ISR) // or interrupt handler!

LPC_UART1 → IIR — Interrupt ID Register ≡ identifies which interrupt(s) are pending! (waiting to be handled or serviced)

if - case: RLS ≡ 0x03 (receive line status)

- subcase: LPC_UART1 → LSR (one of: OE, PE, FE, RxFE, BI) : // not good! errors!

  LSR — Line Status Register

- subcase: LPC_UART1 → LSR is RDR // good! received data is ready place data in buffer!

- case: RDA ≡ 0x02 (receive data available) // good! place data in buffer!

- case: CTI ≡ 0x06 (character time out indicator) // minimum of one character in Rx FIFO and no character input or removed during a time period depending on how many characters are in FIFO and what trigger level is set! (see page 304) // not good!

- case: THRE ≡ 0x01 // good! means Tx buffer is empty; it could be loaded with another character to be transmitted!

}