## Description

I believe the following behavior is incorrect:

```ruby
ruby -r timeout -e 'r = Ractor.new { Timeout.timeout(0.1) { sleep(1) } rescue :timeout }; p r.take'
# => :timeout (ok)

ruby -r timeout -e 'r = Ractor.new { Timeout.timeout(0.1) { sleep(1) } rescue :timeout }; sleep(0.2); p r.take'
# => :timeout (ok)

ruby -r timeout -e 'r = Ractor.new { Timeout.timeout(0.1) { Ractor.receive } rescue :timeout }; p r.take'
# => :timeout (ok)

ruby -r timeout -e 'r = Ractor.new { Timeout.timeout(0.1) { Ractor.receive } rescue :timeout }; sleep(0.2); p r.take'
<internal:ractor>:130:in `take': The outgoing-port is already closed (Ractor::ClosedError) # => not ok
```

## Associated revisions

**Revision c2fa024e - 12/07/2020 07:01 AM - ko1 (Koichi Sasada)**

Fix Thread's interrupt and Ractor#take issue

Thread's interrupt set Ractor's wakeup_status as interrupted, but the status remains next Ractor communication API. This patch makes to ignore the previous interrupt state.

[Bug #17366]

Also this patch solves the Thread#kill and Ractor#take issues.

## History

**#1 - 12/07/2020 12:22 AM - ko1 (Koichi Sasada)**

I can confirm with it:

```ruby
r = Ractor.new {
  begin
    pth = Thread.current
    Thread.new{ pth.kill }.join
    rescue => e
      p e
    end
  end
}
p r.take
```

**#2 - 12/18/2020 08:25 PM - ko1 (Koichi Sasada)**

- Status changed from Open to Closed