```fortran
module cpu_timer_class                              ! time quantum is a

  use constants_and_parameters
  implicit none

  ! derived data type
  type, public            :: cpu_timer          ! name to instantiate

    private
    real ( wp )           :: saved_time         ! saved time in second
    real ( wp )           :: cum_time           ! cumulative time in se

    contains  ! bound procedures

      ! sequence:  grab  pause, resume, ... pause, resume  stop
      ! stop is equivalent to pause
      procedure, public   :: cpu_timer_grab     => cpu_timer_grab_sub
      procedure, public   :: cpu_timer_pause    => cpu_timer_pause_sub
      procedure, public   :: cpu_timer_resume   => cpu_timer_resume_su

      procedure, public   :: cpu_timer_stop     => cpu_timer_stop_fcn
      procedure, public   :: cpu_timer_cum_read => cpu_timer_cum_read_

  end type cpu_timer
  ! end derived data type

  private                 :: cpu_timer_grab_sub, cpu_timer_pause_sub,
  private                 :: cpu_timer_stop_fcn, cpu_timer_cum_read_fc

  ! methods
  contains                                      ! subroutines and func

!   ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

    subroutine cpu_timer_grab_sub ( self )      ! start a timer

      implicit none

      class ( cpu_timer ) :: self                ! timer object

      real ( wp )         :: cpu_time_t0         ! saved time in second

      ! system call for current cpu_time
      call cpu_time ( cpu_time_t0 )

      self % saved_time = cpu_time_t0            ! unique start time fo
      self % cum_time   = zero

    end subroutine cpu_timer_grab_sub

!   ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

    subroutine cpu_timer_pause_sub ( self )      ! update the cumulativ

      implicit none

      class ( cpu_timer ) :: self                ! timer object

      real ( wp )         :: cpu_time_now

      ! system call for current cpu_time
      call cpu_time ( cpu_time_now )

      ! guard against the erroneous sequence of  pause, pause, ..., pa
      self % cum_time   = self % cum_time + ( cpu_time_now - self % sa
      self % saved_time = cpu_time_now           ! unique start time fo

    end subroutine cpu_timer_pause_sub

!   ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

    subroutine cpu_timer_resume_sub ( self )     ! restart the time

      implicit none

      class ( cpu_timer ) :: self                 ! timer object

      real ( wp )         :: cpu_time_now         ! saved time in second

      ! system call for current cpu_time
      call cpu_time ( cpu_time_now )

      self % saved_time = cpu_time_now            ! unique start time fo

    end subroutine cpu_timer_resume_sub

!   ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

    function cpu_timer_stop_fcn ( self ) result ( cpu_time_cum )  ! re

      implicit none

      class ( cpu_timer ) :: self                 ! timer object

      real ( wp )         :: cpu_time_now         ! saved time in second
      real ( wp )         :: cpu_time_cum         ! accumulated time

      ! system call for current cpu_time
      call cpu_time ( cpu_time_now )

      ! guard against the erroneous sequence of  pause, pause, ...,  pa
      cpu_time_cum      = self % cum_time + ( cpu_time_now - self % sa
      self % cum_time   = cpu_time_cum
      self % saved_time = cpu_time_now            ! unique start time fo

    end function cpu_timer_stop_fcn

!   ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

    function cpu_timer_cum_read_fcn ( self ) result ( cpu_cum_time )

```

```fortran
      implicit none

      class ( cpu_timer ) :: self                 ! timer object

      real ( wp )         :: cpu_cum_time         ! cumulative time buff

      cpu_cum_time = self % cum_time              ! read cumulative time

    end function cpu_timer_cum_read_fcn

end module cpu_timer_class

!     ######################################################################
!     #
!     ######################################################################

module clock_timer_class                          ! time quantum is proc

  use constants_and_parameters
  implicit none

  ! derived data type
  type, public              :: clock_timer      ! name to instantiate

    private
    integer ( lint )        :: saved_time       ! saved time in second

    contains  ! bound procedures

      procedure, public     :: clock_start_timer  => clock_start_timer
      procedure, public     :: clock_elapsed_time => clock_elapsed_tim

  end type clock_timer
  ! end derived data type

  private :: clock_start_timer_sub, clock_elapsed_time_fcn

  ! methods
  contains                                           ! subroutines and func

!   ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

    subroutine clock_start_timer_sub ( self )

      implicit none

      class ( clock_timer ) :: self                  ! timer object

      integer ( lint )      :: clock_count_start! saved time in ticks 

      ! system call for current clock_time
      call system_clock ( clock_count_start )

      self % saved_time = clock_count_start          ! unique start time fo

    end subroutine clock_start_timer_sub

!   ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

    real function clock_elapsed_time_fcn ( self )

      implicit none

      class ( clock_timer ) :: self                  ! clock timer object
      integer ( lint )      :: clock_count_stop, clock_count_rate, clo

      ! system call for current clock_time
      call system_clock ( clock_count_stop, clock_count_rate, clock_co

      ! units are compiler dependent
      clock_count_delta = clock_count_stop - self % saved_time
      ! convert to seconds
      clock_elapsed_time_fcn = dble ( clock_count_delta ) / clock_coun

    end function clock_elapsed_time_fcn

end module clock_timer_class
```