```fortran
! CGV/solution/parts/cgvp02.F90

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!                                                              !
! This file has been written as a sample solution to an        !
! exercise in a course given at the High Performance           !
! Computing Centre Stuttgart (HLRS).                           !
! It is made freely available with the understanding that      !
! every copy of this file must include this header and that    !
! HLRS takes no responsibility for the use of the enclosed     !
! teaching material.                                           !
!                                                              !
! Authors: Dr. Rolf Rabenseifner                               !
!          Stephan Scheiderer                                  !
!                                                              !
! Contact: rabenseifner@hlrs.de                                !
!                                                              !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
                                                            !EOH
!>>>>>>>>>>>>>>>>>>>>>> begin of task 02 <<<<<<<<<<<<<<<<<<<<<<

!*********************** halo routines *************************

#ifndef serial
!------------------------------
!Init_Halo_Struct
!
! TASK:
!   Initialize the halo structure.
!
! GLOBAL DATA:
!   IN INTEGER rowsize : number of data entries in a row of the local chunk
!   IN INTEGER colsize : number of data entries in a column of the local chunk
!   IN/OUT halo_structure halo : the halo
!   OUT REAL(0:) recv_scratch_buff_north : receive buffer for north halo
!   OUT REAL(0:) recv_scratch_buff_east  : receive buffer for east halo
!   OUT REAL(0:) recv_scratch_buff_south : receive buffer for south halo
!   OUT REAL(0:) recv_scratch_buff_west  : receive buffer for west halo
! USED FUNCTIONS:
!   APPL:
!     --
!   MPI:
!     MPI_CART_SHIFT
SUBROUTINE Init_Halo_Struct(v1)
    REAL, DIMENSION(0:), TARGET :: v1
    INTEGER                     :: maxScratchBuffSize

    ! allocate sufficient memory for the send scratch buffer
    IF (rowsize > colsize) THEN
        maxScratchBuffSize = rowsize
    ELSE
        maxScratchBuffSize = colsize
    END IF
    ! allocate sufficient memory for the receive scratch buffer for the halo
    ALLOCATE(recv_scratch_buff_north(0:maxScratchBuffSize-1))
    ALLOCATE(recv_scratch_buff_east(0:maxScratchBuffSize-1))
    ALLOCATE(recv_scratch_buff_south(0:maxScratchBuffSize-1))
    ALLOCATE(recv_scratch_buff_west(0:maxScratchBuffSize-1))

    ! determine neighbors (1-dim and 2-dims)
    halo%north => v1(1:rowsize)
    halo%north_size = rowsize
    halo%south => v1((rowsize + 2) * (colsize + 1) + 1:(rowsize + 2) * (colsize + 1) + rowsize)
    halo%south_size = rowsize
    ! compute north_rank and south_rank
    CALL MPI_CART_SHIFT(comm_cart, 0, 1, halo%north_rank, halo%south_rank, ierror)

    halo%west => v1((rowsize + 2):(colsize * (rowsize + 2)):(rowsize + 2))
    halo%west_size = colsize
    halo%east => v1((rowsize + 2) * 2 - 1:((colsize + 1) * (rowsize + 2) - 1):(rowsize + 2))
    halo%east_size = colsize
    ! compute west_rank and east_rank
    CALL MPI_CART_SHIFT(comm_cart, 1, 1, halo%west_rank, halo%east_rank, ierror)
END SUBROUTINE Init_Halo_Struct

! halo communication: e.g. FROM_WEST_TO_EAST
!
! halo is included in the local vector chunk
!
!   **************************      **************************
!   *  -----NORTH HALO-----  *      *  -----NORTH HALO-----  *
!   *W ******************** E*      *W ******************** E*
!   *E *                *|* A*      *E *                  * A*
!   *S * WEST PROCESS   *------------> *   EAST PROCESS   * S*
!   *T *                *|* T*      *T *                  * T*
!   *  *                *|*  *      *  *                  *  *
!   *H ******************** H*      *H ******************** H*
!   *  -----SOUTH HALO-----  *      *  -----SOUTH HALO-----  *
!   **************************      **************************

```

```
!----------------------------
! comm_vec_halo
!
! IN  REAL(0:) v1: the vector with the data to send to other processes
!
! TASK:
!   Send neccessary data to the neighbors
!   and receive the needed data from the neighbors into
!   the halo area.
!
! USED GLOBAL DATA:
!   IN/OUT halo_structure halo : the halo
!   IN     INTEGER rowsize : number of data entries in a row of the local chunk
!   IN     INTEGER colsize : number of data entries in a column of the local chunk
!   OUT REAL(0:) recv_scratch_buff_north : receive buffer for north halo
!   OUT REAL(0:) recv_scratch_buff_east  : receive buffer for east halo
!   OUT REAL(0:) recv_scratch_buff_south : receive buffer for south halo
!   OUT REAL(0:) recv_scratch_buff_west  : receive buffer for west halo
!
! USED FUNCTIONS:
!   MPI:
!     MPI_IRECV, MPI_SEND, MPI_WAITALL
!
! message tags for the different communication directions
SUBROUTINE comm_vec_halo(v1)
    REAL, DIMENSION(0:), INTENT(IN), TARGET :: v1
    INTEGER :: i

    INTEGER :: req_array(4)
    INTEGER :: status_array(MPI_STATUS_SIZE, 4)
    INTEGER :: TAG_FROM_NORTH_TO_SOUTH = 101
    INTEGER :: TAG_FROM_EAST_TO_WEST   = 102
    INTEGER :: TAG_FROM_SOUTH_TO_NORTH = 103
    INTEGER :: TAG_FROM_WEST_TO_EAST   = 104

    REAL, DIMENSION(:), POINTER :: send_to_east_values
    REAL, DIMENSION(:), POINTER :: send_to_west_values
    REAL, DIMENSION(:), POINTER :: send_to_north_values
    REAL, DIMENSION(:), POINTER :: send_to_south_values

    ! make a non−blocking receive

    ! Receive halo (non blocking). If current rank has no neighbors,
    ! the neighbor rank is MPI_PROC_NULL and nothing is sent.

    ! ATTENTION:
    ! It is NOT allowed to use pointers or strided arrays in NON−blocking send/receive routines,
    ! because the memory MAY NOT be contiguous. One must use a contiguous scratch buffer
    ! for sending and receiving, if the memory is not contiguous or a pointer to an array.

    ! receive north halo
    CALL MPI_IRECV(recv_scratch_buff_north, halo%north_size, MPI_REAL, halo%north_rank, &
                & TAG_FROM_NORTH_TO_SOUTH, comm_cart, req_array(1), ierror)
    ! receive east halo
    CALL MPI_IRECV(recv_scratch_buff_east, halo%east_size, MPI_REAL, halo%east_rank, &
                 & TAG_FROM_EAST_TO_WEST, comm_cart, req_array(2), ierror)
    ! receive south halo
    CALL MPI_IRECV(recv_scratch_buff_south, halo%south_size, MPI_REAL, halo%south_rank, &
                 & TAG_FROM_SOUTH_TO_NORTH, comm_cart, req_array(3), ierror)
    ! receive west halo
    CALL MPI_IRECV(recv_scratch_buff_west, halo%west_size, MPI_REAL, halo%west_rank, &
                & TAG_FROM_WEST_TO_EAST, comm_cart, req_array(4), ierror)

    ! send to north neighbor (blocking)
    send_to_north_values => v1(SIDX(1, 1):SIDX(colsize, 1):1)
    CALL MPI_SEND(send_to_north_values, halo%north_size, MPI_REAL, halo%north_rank, &
                & TAG_FROM_SOUTH_TO_NORTH, comm_cart, ierror)

    ! send to east neighbor (blocking)
    send_to_east_values => v1(SIDX(1, rowsize):SIDX(colsize, rowsize):rowsize+2)
    CALL MPI_SEND(send_to_east_values, halo%east_size, MPI_REAL, halo%east_rank, &
                & TAG_FROM_WEST_TO_EAST, comm_cart, ierror)

    ! send to south neighbor (blocking)
    send_to_south_values => v1(SIDX(colsize, 1):SIDX(colsize, rowsize):1)
    CALL MPI_SEND(send_to_south_values, halo%south_size, MPI_REAL, halo%south_rank, &
                & TAG_FROM_NORTH_TO_SOUTH, comm_cart, ierror)

    ! send to west neighbor (blocking)
    send_to_west_values => v1(SIDX(1, 1):SIDX(colsize, 1):rowsize+2)
    CALL MPI_SEND(send_to_west_values, halo%west_size, MPI_REAL, halo%west_rank, &
                & TAG_FROM_EAST_TO_WEST, comm_cart, ierror)

    ! wait for all halo info
    CALL MPI_WAITALL(4, req_array, status_array, ierror)

    ! copy the received halo from the scratch buff into the real halo
    ! ATTENTION: pointers always start with the index 1!!!
    IF (halo%north_rank /= MPI_PROC_NULL) THEN
        DO i = 0, halo%north_size − 1
            halo%north(i+1) = recv_scratch_buff_north(i)
        END DO
    END IF
    IF (halo%east_rank /= MPI_PROC_NULL) THEN
        DO i = 0, halo%east_size − 1
            halo%east(i+1) = recv_scratch_buff_east(i)
        END DO
    END IF
    IF (halo%south_rank /= MPI_PROC_NULL) THEN
        DO i = 0, halo%south_size - 1
            halo%south(i+1) = recv_scratch_buff_south(i)
        END DO
    END IF
    IF (halo%west_rank /= MPI_PROC_NULL) THEN
        DO i = 0, halo%west_size - 1
            halo%west(i+1) = recv_scratch_buff_west(i)
        END DO
    END IF
END SUBROUTINE comm_vec_halo
#endif

!>>>>>>>>>>>>>>>>>>>>>>> -end- of task 02 <<<<<<<<<<<<<<<<<<<<<<
```