Project Report

Jensen Davies

June 8, 2020

1 PCAM Process

1.1 Partitioning the Data

I did a 1D column-wise decomposition of a 2D array.

1.2 Communication

The primary communication functions used were MPI_ISend, MPI_RECV, and MPI_Gather, all in the global communication channel. The communication structure was a ring, where we define a rank dependence via the modulus the number of processors. The main communication of the program is communicating necessary columns from differing processors to construct appropriate ghost cells.

1.3 Agglomeration

Since I didn't decompose the problem to a finer granularity, there wasn't a portion of my program that required any helpful agglomeration of tasks (that I could notice, anyway). In terms of replication, I replicated columns for ghost cells to send to other processors (on each processor). We were able to achieve less communication by defining the submatrix each processor uses without any communication. However, this made it so I had to use MPI_Gather to print locally off of one processor to avoid the race-condition issue.

1.4 Mapping

Each processor received an equal number of columns of a randomly initialized grid. The number of columns each processor receives depends on the amount of processors, where load-balancing is assumed (each processor has num_procs/N, where N is dimension of the square grid).

2 Documentation

2.1 Commands to Compile and Run

In order for the program to run, the user needs the source file:

```
life.f90
```

Command necessary to compile:

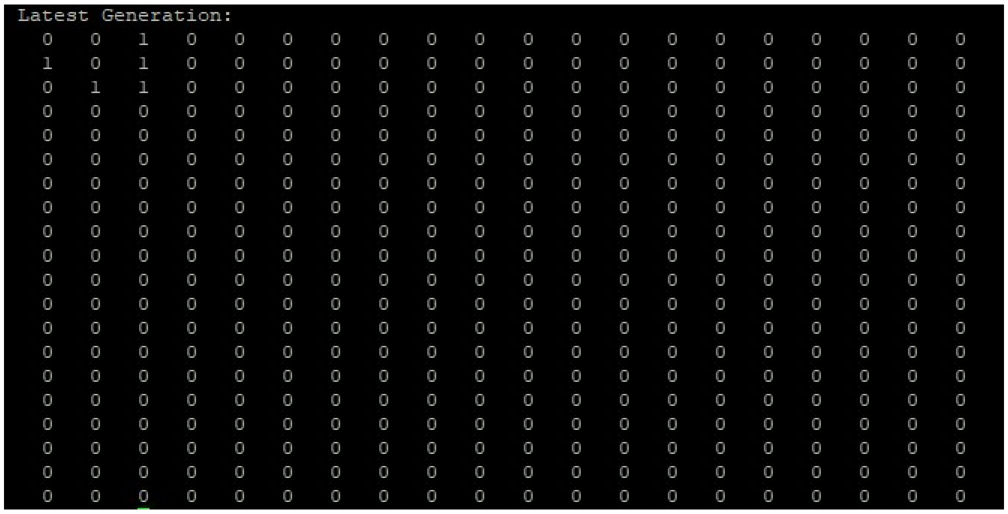
```
mpif90 -o life life.f90
```

Command necessary to run (here N represents the number of processors) :

```
mpirun -np N -stdin all life
```

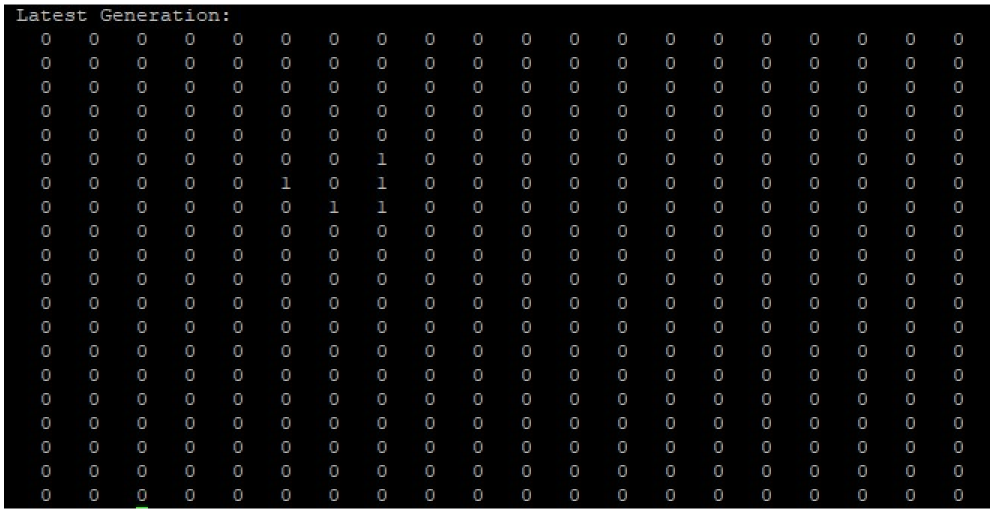
2.2 Test Samples for Glider

Step 0



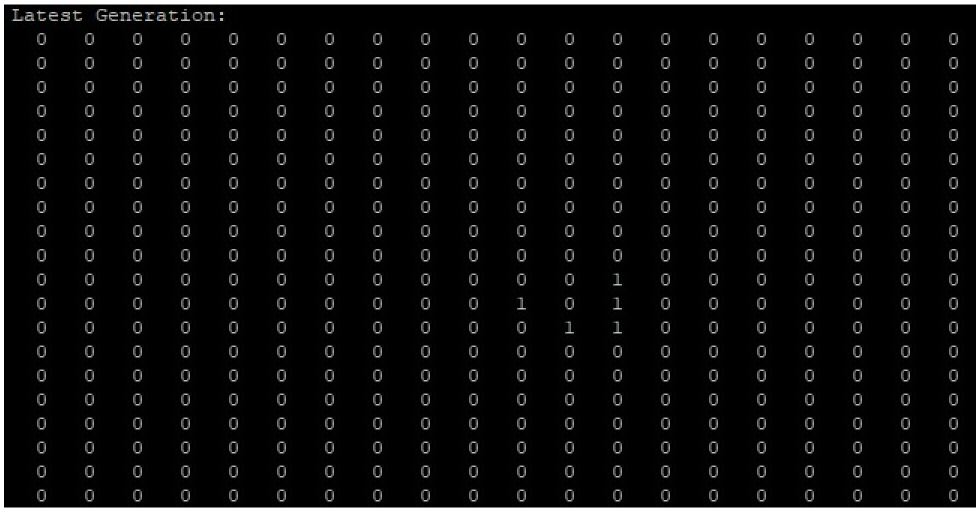
```
Latest Generation:
    0    0    1    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    1    0    1    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    1    1    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
```

Step 20



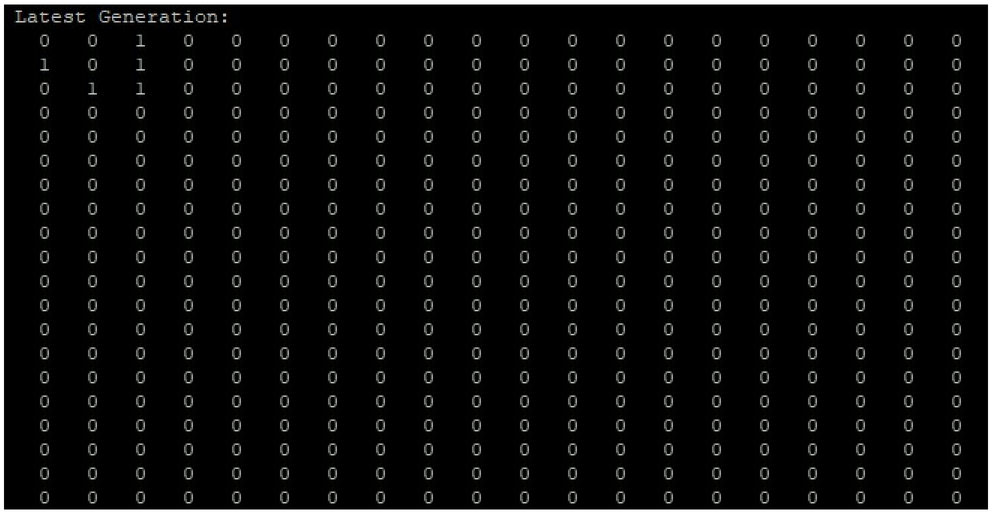
```
Latest Generation:
    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    1    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    1    0    1    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    1    1    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
```

Step 40



```
Latest Generation:
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   1   0   1   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   1   1   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
```

Step 80



```
Latest Generation:
  0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  1   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   1   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
```

3 Performance Model

Our problem involves a 2D $N \times N$ grid, hence we have that

$$T_{comp} = t_c N^2 \implies t_c = \frac{T_{comp}}{N^2}.$$

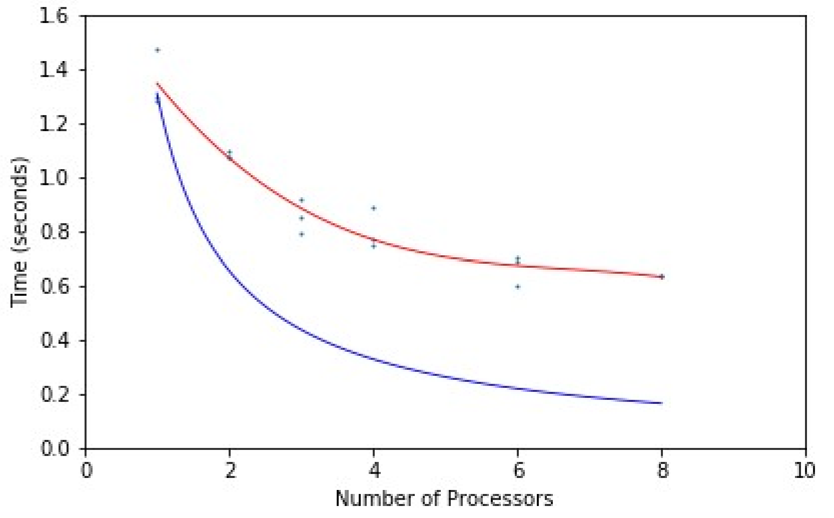
As for T_{comm}, since each processor has two ghost columns and two ghost rows, we have that

$$T_{comm} = P\big[2(t_s + t_w N_y) + 2(t_s + t_w N_x)\big] = P\big[4t_s + 2(t_w N_y + t_w \frac{N}{P})\big],$$

Since we assume load balancing (i.e. $P \mid N$), each processor receives $N_x = \frac{N}{P}$ columns. Hence, our performance model is expressed as

$$T_{game\_of\_life} = \frac{T_{comp} + T_{comm}}{P} = 4t_s + \ 2(t_w N_y + t_w \frac{N}{P}) + \frac{t_c N^2}{P}$$

Below is a graph modeling the derived performance model above. Blue is the derived model, and red is a least squares fitted polynomial to a small sample of actual data points.



4 Appendix

Below is the source code for the entire program.

```
    SUBROUTINE init_random_seed(rank)
    INTEGER :: i, n, clock
    INTEGER, DIMENSION(:), ALLOCATABLE :: seed
    INTEGER, INTENT(IN) :: rank
    CALL RANDOM_SEED(size = n)
    ALLOCATE(seed(n))

    CALL SYSTEM_CLOCK(COUNT=clock)

    seed = clock + 37 * (/ (i - 1, i = 1, n) /)
    seed = seed*(rank+1)
    CALL RANDOM_SEED(PUT = seed)

    DEALLOCATE(seed)
END SUBROUTINE

subroutine play_game(A,B,Z,size)
    implicit none
    integer :: i,j,size
    integer, intent(in) :: Z
    integer, intent(in) :: A(Z+2, (Z/size)+2)
    integer, intent(inout) :: B(Z+2,(Z/size)+2)

    do j = 2, (Z/size)+1
        do i = 2, Z+1

            if (A(i-1,j-1) + A(i-1,j) + A(i-1,j+1) + A(i,j-1) + &
                A(i+1,j-1) + A(i+1,j) + A(i+1,j+1) + A(i,j+1) == 3) then
                B(i,j) = 1

            elseif(A(i-1,j-1) + A(i-1,j) + A(i-1,j+1) + A(i,j-1) + &
                A(i+1,j-1) + A(i+1,j) + A(i+1,j+1) + A(i,j+1) == 2 .AND. A(i,j) == 1) then
                B(i,j) = 1

            elseif(A(i-1,j-1) + A(i-1,j) + A(i-1,j+1) + A(i,j-1) + &
                A(i+1,j-1) + A(i+1,j) + A(i+1,j+1) + A(i,j+1) == 2 .AND. A(i,j) == 0) then
                B(i,j) = 0
            else
                B(i,j) = 0
            end if

        end do
    end do

end subroutine play_game

program send_columns

    implicit none
    include 'mpif.h'

    integer::ierr,rank,size,N,i,j,k,next,prev,exit_stat,G
    integer,allocatable,dimension(:,:):: A(:,:),B(:,:), C(:,:)
    integer::tag
    integer::stat(MPI_STATUS_SIZE)
    integer::request
    real :: x

    call MPI_INIT(ierr)
    call MPI_COMM_RANK(MPI_COMM_WORLD, rank, ierr)
    call MPI_COMM_SIZE(MPI_COMM_WORLD, size, ierr)

    if (rank == 0) then
        print*, 'Input N:'
    end if

    read(*,*) N

        !Gracefully exits the program if the user does not load-balance
    if (MODULO(N,size) .NE. 0) then
        exit_stat = 0
        if (exit_stat == 0) then
            if (rank == 0) then
                print*, 'ERROR: Number of processors does not evenly divide input N, aborting...'
            end if
            call MPI_FINALIZE(ierr)
            call exit(exit_stat)
        end if
    end if

    if (rank == 0) then
        print*, 'Input number of generations to evolve: '
    end if

    read(*,*) G

    allocate(A(N+2,(N/size)+2))
    allocate(B(N+2,(N/size)+2))

    call init_random_seed(rank)

    !assign random 1's and 0's to each processors portion of A
    do j=2, (N/size)+1
        do i=2, N+1
            call random_number(x)
            if (x > .5) then
            A(i, j) = 1
            else
            A(i,j) = 0
          end if
        end do
    end do

    A(1,:) = 0
    A(N+2,:) = 0
    A(:,1) = 0
    A(:,(N/size)+2) = 0

    B = 0

    next = MODULO(rank + 1, size)
    prev = MODULO(rank - 1, size)

    tag = 42069

    if (rank == 0) then
        allocate(C(N,N))
        C = 0
    end if

        !TEST CASES -- note, if you choose G = 4,20,40, or 80, it will default to the test case for simplicity
    if (G == 4 .OR. G == 40 .OR. G == 80 .OR. G == 20) then
            A = 0
        if (rank == 0) then
            A(3,2) = 1
            A(4,3) = 1
            A(4,4) = 1
            A(3,4) = 1
            A(2,4) = 1
        end if
        call MPI_Gather(A(2:N+1, 2:(N/size)+1), (N**2)/size, MPI_INTEGER, C, (N**2)/size, &
        MPI_INTEGER, 0, MPI_COMM_WORLD, ierr)

        if (rank == 0) then
                print*, 'Initial Grid: '
                do j= 1,N
                    do i= 1,N
                        write(6,'(I4)',advance='no') C(j,i)
                    end do
                        print*,''
                end do
            end if
    else
        call MPI_Gather(A(2:N+1, 2:(N/size)+1), (N**2)/size, MPI_INTEGER, C, (N**2)/size, &
        MPI_INTEGER, 0, MPI_COMM_WORLD, ierr)

        if (rank == 0) then
            print*, 'Initial Grid: '
            do j= 1,N
                do i= 1,N
                    write(6,'(I4)',advance='no') C(j,i)
                end do
                    print*,''
            end do
        end if
    end if


    !construct ghost matrices for periodic boundaries
    do i = 1,G
        A(:,(N/size)+2) = A(:,2)
        A(:,1) = A(:,(N/size)+1)
        call MPI_ISEND(A(:,(N/size)+2), N+2, MPI_INTEGER, prev, tag, MPI_COMM_WORLD, request, ierr)
        call MPI_RECV(A(:,(N/size)+2), N+2, MPI_INTEGER, next, tag, MPI_COMM_WORLD, stat, ierr)
        call MPI_ISEND(A(:,1), N+2, MPI_INTEGER, next, tag, MPI_COMM_WORLD, request, ierr)
        call MPI_RECV(A(:,1), N+2, MPI_INTEGER, prev, tag, MPI_COMM_WORLD, stat, ierr)
        call MPI_Wait(request,stat,ierr)

        !assigning proper values of ghost cells to B
        A(N+2,:) = A(2,:)
        A(1,:) = A(N+1,:)

        call play_game(A,B,N,size)
        A = B
    end do

    if (rank == 0) then
        C = 0
    end if

    call MPI_Gather(A(2:N+1, 2:(N/size)+1), (N**2)/size, MPI_INTEGER, C, (N**2)/size, &
                    MPI_INTEGER, 0, MPI_COMM_WORLD, ierr)


    if (rank == 0) then
        print*, 'Latest Generation: '
        do j= 1,N
            do i= 1,N
                write(6,'(I4)',advance='no') C(j,i)
                end do
                    print*,''
            end do
        end if

        call MPI_FINALIZE(ierr)

    stop
    end program send_columns
```

Thanks Youngjun for all your help!