APPENDIX

Listing of Programs in the Order that They are Mentioned in the Text

1 MAKE_WIN.AML : Automatically creates windows and displays the the vicinity of outlet cells

2 FLOW_LENGTH.AML : Determines the flowlengths from each cell in a subwatershed to the outlet of that subwatershed

3 SLOPE.AML : If a time-index grid is being computed, this program is called by flow_length.aml to compute slope to the power b.

4 TIME_WEIGTH.AML : Computes a time-weight value when called by flow_length.aml.

5 MSWORKING2.AML : Called by flow_length.aml to provide messages to the user.

6 GENHRAP.F : Writes a file of coordinate values used to create a polygon coverage of NEXRAD cells in geographic coordinates given a user specified geographic extent.

7 GENHRAP.AML : Creates a polygon coverage of HRAP cells in geographic coordinates given the output from genhrap.f, projects these cells into Albers, and attaches the appropriate HRAP-ID values.

8 HRAP_INT.AML : Intersects a coverage of HRAP cells with a subwatershed coverage creating a number of sectors; computes mean flow length from each of these sectors to the appropriate subwatershed outlet.

9 MOUTPUT.F : Reformats the statistics file generated by hrap_int.aml.

** Codes are listed in order of their use in the procedure.
** Note on AMLs: in their current form, all output grids, coverages, and files will be killed if the procedure is run a second time without changing their names.

1 MAKE_WIN.AML

```
/***********************************************************************
/***********************************************************************
/* Name:  make_win.aml
/*
/*** Purpose:  This AML paints the vicinity of outlet locations in a point
/*** coverage so that the user can select the outlet cell from the
/*** streamlink grid which is closest to that point as a watershed outlet.
/*** Several new graphics windows are created.  The number of outlet
/*** locations that can be selected in one execution is influenced by the
/*** number of new windows that can fit on the screen.

&args linkgrid outlets
&type running make_win.aml
&messages &off &all
&if [iteminfo %outlets% -point X-COORD -exists] = .FALSE. &then
  &do
  &sys arc addxy %outlets% point
  &end
/*grid

&if [extract 1 [show display]] ne 9999 &then
  &do
   display 9999
  &end
mape %linkgrid%
describe %linkgrid%
&sv cellsize = %grd$dx%
units map
&sv mapxmin = [extract 1 [ show mape ] ]
&sv mapymin = [extract 2 [ show mape ] ]
&sv mapxmax = [extract 3 [ show mape ] ]
&sv pagxmin = [extract 1 [ show mape page ] ]
&sv pagxmax = [extract 3 [ show mape page ] ]
&sv mapfactor = ( %pagxmax% - %pagxmin% ) / ( %mapxmax% - %mapxmin% )
&sv mapxoffset = %mapxmin%
&sv mapyoffset = %mapymin%
&sv cellrange = 20.0

&sv end_of_points = .FALSE.
cursor out_cur declare %outlets%.pat info ro
cursor out_cur open

&sv count = 0
/*** Processing loop ***

&do &until %end_of_points% = .TRUE.
   &sv count = %count% + 1

   &sv x = %:out_cur.X-COORD%
   &sv y = %:out_cur.Y-COORD%
/*   &type %x%
/*   &type %y%
   &sv xmin = ( %x% - %cellrange% * %cellsize% - %mapxoffset% ) * %mapfactor%
   &sv xmax = ( %x% + %cellrange% * %cellsize% - %mapxoffset% ) * %mapfactor%
   &sv ymin = ( %y% - %cellrange% * %cellsize% - %mapyoffset% ) * %mapfactor%
   &sv ymax = ( %y% + %cellrange% * %cellsize% - %mapyoffset% ) * %mapfactor%
/*   &type %xmin% %ymin% %xmax% %ymax%
   &if %count% eq 1 &then
     windows create win%count% %xmin% %ymin% %xmax% %ymax% ~
        SIZE 350 350 POS UL DISPLAY UR
   &if %count% eq 2 &then
     windows create win%count% %xmin% %ymin% %xmax% %ymax% ~
```

```
        SIZE 350 350 POS UL WINDOW win1 LL
   &if %count% eq 3 &then
     windows create win%count% %xmin% %ymin% %xmax% %ymax% ~
        SIZE 350 350 POS UL DISPLAY LL
   &if %count% eq 4 &then
      windows create win%count% %xmin% %ymin% %xmax% %ymax% ~
        SIZE 350 350 POS UL WINDOW win3 UR

cursor out_cur next
&if %:out_cur.AML$NEXT% = .FALSE. &then
  &do
    &sv end_of_points = .TRUE.
    cursor out_cur remove
/*    &type %end_of_points%
  &end
&end   /*End of Main Processing Loop
&messages &on
/*&type quitting grid
/*q

&return
```

2 FLOW_LENGTH.AML

```
/***********************************************************************
/***********************************************************************
/* Name:  flow_length.aml
/*
/* Purpose:  Determine the flowlengths from each cell in a subwatershed to the
/*           outlet of that subwatershed.
/*
/*          Can also be used to compute the flowaccumulation for each cell in
/*            each subwatershed based only on flow originating in that
/*          subwatershed or to compute an integrated time-index parameter
/*          (requiring a call to time_weight.aml).
/*
/*          In its current form, this program also calls two "canned" programs
/*          described in "Arc/Info - HEC-1 Interface : Working Papers" by Mark
/*          Beavers (msworking2.aml and msworking2.menu).  These programs only
/*          supply information to the user and do not affect grid processing:
/*          the relevant lines can be commented out if desired.
/*
/* Inputs:   Two grids: (1) a projected grid of the subwatershed masks and
/*          (2) a grid of flowdirection for these subwatersheds.  Names of
/*            these input grids are supplied as arguments at the command line.
/*
/* Outputs:  The grid flmerge_grid contains the flowlengths from each cell in
/*           a subwatershed to the outlet of that sub-watershed.  If computed,
/*           the grid ftmerge_grid contains the time index value for each
/*           cell in a subwatershed based on flow originating in that
/*           subwatershed.
/*
/***********************************************************************
/***********************************************************************

/* Read in the names of the watershed grid and the direction grid as
/* global variables.
&args .subshed_grid .dir_grid .dem_grid

/* Initialize control variables.
&sv  first_time_thru = .TRUE.
&sv  end_of_subsheds = .FALSE.
&sv  mergelist1      = ' '
&sv  count           =  1
&sv  temp_count      =  1                  /* TEMP
&sv  first_wshed = .TRUE.
```

```
/* Enter the grid module where processing will occur.
grid
&if [extract 1 [show display]] ne 9999 &then
   display 9999
ap gridnodatasymbol transparent
mape %.subshed_grid%

gridshades %.subshed_grid%

/*  Declare a cursor for the subshed grid, and open it.
/*  Also, check to make sure that there is something
/*  in the file to read.  If not, set a flag.
/*
cursor subshed_cur declare %.subshed_grid%.vat info ro
cursor subshed_cur open
&if %:subshed_cur.AML$NEXT% = .FALSE. &then
  &sv end_of_subsheds = .TRUE.


/******* Main processing loop. ***************************************
/*
&do &while %end_of_subsheds% = .FALSE.

  &type loop begins [date -time]
  &if [exists temp_l%:subshed_cur.value% -grid] &then
    kill temp_l%:subshed_cur.value% all
  &if [exists temp_fa%:subshed_cur.value% -grid] &then
    kill temp_fa%:subshed_cur.value% all
  &if [exists temp_t%:subshed_cur.value% -grid] &then
    kill temp_t%:subshed_cur.value% all

  &if [exists length_grid -grid] &then
    kill length_grid all
  &if [exists time_grid -grid] &then
    kill time_grid all


  length_grid = flowlength (con (%.subshed_grid% == %:subshed_cur.value%, ~
             %.dir_grid%), #, downstream)

  /*  At the time this AML was first written,
  /*  the flowlength function returned zero values
  /*  instead of NODATA values at all points outside a watershed but
  /*  inside the mapextent. That is the reason for the inclusion of the
  /*   next line.  This problem may have been fixed in a later version.
```

```
 temp_l%:subshed_cur.value% = con (length_grid ne 0, length_grid)

/* <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
/* If it is desired to compute a time-index parameter, run time_weight.aml
/* followed by a weighted flowlength command.
/*
/*temp_fa%:subshed_cur.value% = flowaccumulation (con (%.subshed_grid% ~
/*                               == %:subshed_cur.value%, %.dir_grid%))
/*&if %first_wshed% = .TRUE. &then
/*   &do
/*     &r slope %.dem_grid%
/*     &sv first_wshed = .FALSE.
/*   &end
/*&r time_weight temp_fa%:subshed_cur.value%

/*time_grid = flowlength (con (%.subshed_grid% == %:subshed_cur.value%, ~
/*               %.dir_grid%), tweight, downstream)
/*temp_t%:subshed_cur.value% = con(time_grid ne 0, time_grid)
/*<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

/*******************************************************************
 /*  The next section creates a long variable which contains the NAMES
 /*  of all the temporary "fl" and "ft" grids created above.  These will be
 /*  used in a MERGE statement after loop completes.  This
 /*  is done to minimize MERGE executions - this way it will only happen
 /*  once, at the very end.

 &if [length mergelist%count%] ge 1000 &then
  &do
   &sv count = %count% + 1
   &sv first_time_thru = .TRUE.
  &end
 &if %first_time_thru% = .TRUE. &then
  &do
   &sv mergelist%count% = temp_l%:subshed_cur.value%
/*      &sv mlistft%count% = temp_t%:subshed_cur.value%
   &sv first_time_thru = .FALSE.
  &end
 &else
  &do
   &sv mergelist%count% = [value mergelist%count%], temp_l%:subshed_cur.value%
/*      &sv mlistft%count% = [value mlistft%count%],temp_t%:subshed_cur.value%
  &end

 /*  Display the current subshed, so the user will know something
 /*  is happening.
```

```
  gridpaint temp_l%:subshed_cur.value% value linear nowrap gray

  cursor subshed_cur next
  &if %:subshed_cur.AML$NEXT% = .FALSE. &then
    &do
      &sv end_of_subsheds = .TRUE.
      cursor subshed_cur remove
    &end
  &else
    &do
      &sv msg1 = 'Longest Streamlength Determination'
      &sv msg2 = Processing Subwatershed %:subshed_cur.value%
      &r msworking2 update %msg1% [quote %msg2%]
    &end
&end
/******* END OF MAIN PROCESSING LOOP ***************************

/*  This kill was moved down here so that if the user bailed out of the program
/*  early, the merge_grid would still be intact (if it existed from a previous run).

&if [exists flmerge_grid -grid] &then
  kill flmerge_grid all
&if [exists ftmerge_grid -grid] &then
  kill ftmerge_grid all

/*  Merges all the flowlists and musklists created above.  Only three lists are
/*  coded for here, but any number is possible - three should be sufficient.
&sv msg1 = 'Longest Streamlength Determination'
&sv msg2 = Creating longest streamlength grid MERGE_GRID...
&r msworking2 update %msg1% [quote %msg2%]
&if %count% = 1 &then
  &do
    flmerge_grid  = merge ( %mergelist1% )
/*    ftmerge_grid = merge ( %mlistft1% )
  &end
&if %count% = 2 &then
  &do
    flmerge_grid  = merge ( %mergelist1%, %mergelist2% )
/*    ftmerge_grid = merge ( %mlistft1%, %mlistft2% )
  &end
&if %count% = 3 &then
  &do
    flmerge_grid  = merge ( %mergelist1%, %mergelist2%, %mergelist3% )
/*    ftmerge_grid = merge ( %mlistft1%, %mlistft2%, %mlistft3% )
  &end
```

```
/******** FILE CLEANUP: REMOVE ALL TEMPORARY GRIDS CREATED
&if [exists length_grid -grid] &then
  kill length_grid all
&if [exists time_grid -grid] &then
  kill time_grid all
&sv end_loop = .FALSE.

cursor subshed_cur declare %.subshed_grid%.vat info ro
cursor subshed_cur open
&do &until %end_loop% = .TRUE.

    &if [exists temp_l%:subshed_cur.value% -grid] &then
     kill temp_l%:subshed_cur.value% all
/*   &if [exists temp_ft%:subshed_cur.value% -grid] &then
/*     kill temp_ft%:subshed_cur.value% all
/*    &if [exists temp_fa%:subshed_cur.value% -grid] &then
/*     kill temp_fa%:subshed_cur.value% all
     cursor subshed_cur next
     &if %:subshed_cur.AML$NEXT% = .FALSE. &then
       &do
          &sv end_loop = .TRUE.
          cursor subshed_cur remove
       &end

&end               /* End of loop

quit  /* quit Grid subprogram
&return
```

3 SLOPE.AML

```
/********************************************************************
/********************************************************************
/* Name: slope.aml
/*
/* Purpose:  If desired, called by flowlength.aml to compute slope to the power b.
/*
/* Read in the names of the dem_grid and the flowaccumulation grid
&args .dem_grid
&if [exists slope1 -grid] &then
   kill slope1 all
&if [exists slope_grid -grid] &then
   kill slope_grid all
slope1 = slope( %.dem_grid%, percentrise )
slope_grid = slope1 div 100

/*** Compute S^b
&sv b = 0.5

&if [exists sb -grid] &then
   kill sb all
&if [exists slope_plus -grid] &then
   kill slope_plus all
/* Adjust the slope value by 0.0001 to avoid dividing by zero.
slope_plus = slope_grid + 0.0001
sb = pow( slope_plus, %b% )
&return
```

4 TIME_WEIGHT.AML

```
/*****************************************************************************
/*****************************************************************************
/* Name: time_weight.aml
/*
/* Purpose:  Generates a grid in which the value (1/S^bA^c ) is computed for
/* each cell in a watershed.  This grid can be used as a weight grid to
/* compute an integrated time index value using the flowlength function.
/* Called by flow_length.aml.  Assumes b = c = 0.5.
/*
/* Inputs:   Two grids: (1) a projected grid of the DEM used to compute the
/* slope and (2) a grid that contains the flowaccumulation values.  Depending
/* on how the flow routing is to be done, the flowaccumulation values might be /*
computed on a per-subwatershed basis or on a basin basis
/* -- in these two cases, flowaccumulation values would only differ along the
/* main stream stem. Both cases could be easily implemented within the
/* framework of this procedure. In its current form flowaccumulation is
/* computed on a basin basis.  The name of the projected DEM grid and the
/* flowaccumulation grid are passed as arguments at the command line.
/*
/* Output:   A grid named tweight.
/*
/*****************************************************************************
/*****************************************************************************

/* Read in the names of the dem_grid and the flowaccumulation grid
&args .fa_grid

/*** Compute S^bA^c

&sv c = 0.5

&if [exists ac -grid] &then
  kill ac all
&if [exists sbac -grid] &then
  kill sbac all
&if [exists fa_plus -grid] &then
  kill fa_plus all

/* Adjust the flowaccumulation value by 0.5 to avoid dividing by zero.
fa_plus = %.fa_grid% + 0.5
ac = pow( fa_plus, %c% )

sbac = sb * ac
/** Creating an index of travel time to the outlet
```

```
&if [exists tweight -grid] &then
   kill tweight all
tweight = 1 / sbac
/*quit /* Do not quit out of grid if called from fl_arg.aml.
&return
```

5 MSWORKING2.AML

```
/*-----------------------------------------------------------------------
/*          Environmental Systems Research Institute
/*-----------------------------------------------------------------------
/*   Program: MSWORKING2.AML
/*   Purpose: Display a menu with information that an action is taking
/*          place  (let the user know that something is happening).
/*          The message can be updated by using the UPDATE routine.
/*
/*-----------------------------------------------------------------------
/*     Usage: msworking {INIT} <'message_1'> {'message_2'} {'position'} {'stripe'}
/*     Usage: msworking <routine_name>
/*
/* Arguments: routine - routine to be run
/*
/*          message_1 - The first line of the message to be displayed
/*          message_2 - The second line of the message to be displayed
/*          position  - (quoted string) menu position
/*          stripe    - (quoted string) menu stripe
/*
/*   Globals:
/*-----------------------------------------------------------------------
/*     Calls: MSWORKING.MENU
/*-----------------------------------------------------------------------
/*     Notes: All arguments must be quoted, and each of the message
/*          arguments should contain no more than 80 characters.
/*-----------------------------------------------------------------------
/*     Input:
/*    Output:
/*-----------------------------------------------------------------------
/*   History: Matt McGrath - 02/14/92 - Modified INFORM tool
/*          bernie szukalski - 09/16/92 - added UPDATE routine, changed
/*                                variable naming.
/*          bernie szukalski - 01/21/93 - added position & stripe args
/*          mark beavers     - 08/04/93 - added icon_name variable
/*=============================================================
===========
/*
&args routine message_1 message_2 position stripe icon_name

&severity &error &routine bailout

/* Check arguments
&if [NULL %routine%] &then
  &call usage
```

```
/* Default to the init routine if no routine has been specified
/*
&set routinelist = INIT UPDATE EXIT CLOSE USAGE
&if [KEYWORD %routine% %routinelist%] > 0 &then
  /* A routine has been specified
  &do
    &if [LOCASE %routine%] = init &then
      &do
        &set .msworking$message1 = [UNQUOTE %message_1%]
        &set .msworking$message2 = [UNQUOTE %message_2%]
      &end
  &end
&else
  /* A routine has not been specified, default to init
  &do
    &set stripe        = %position%
    &set position        = %message_2%
    &set .msworking$message2 = [UNQUOTE %message_1%]
    &set .msworking$message1 = [UNQUOTE %routine%]
    &set routine = INIT
  &end
/*
&call %routine%
/*
&return

/*------------
&routine UPDATE
/*------------
&set .msworking$message1 = [UNQUOTE %message_1%]
&set .msworking$message2 = [UNQUOTE %message_2%]

&thread &synchronize tool$msworking
&return

/*------------------
&routine USAGE
/*------------------
/* &type Usage: msworking <routine_name>
&type Usage: msworking2 INIT   <'"msg_1"'> {'"msg_2"'} {'"position"'} {'"stripe"'}
{icon-filename}
&type Usage: msworking2 UPDATE <'"msg_1"'> {'"msg_2"'}
&type Usage: msworking2 EXIT
&return &warning
```

```
*------------------
&routine INIT
/*------------------
/*
/* Check arguments
&if [NULL [VALUE .msworking$message1]] &then
  &call usage
/*
&if [NULL %.msworking$message2%] OR ~
    [QUOTE [UNQUOTE %.msworking$message2%_]] = [QUOTE #_] &then
  &set .msworking$message2
/*
&if [NULL %position%] OR %position%_ = #_ &then
  &set position = &cc &screen &cc
&if [NULL %stripe%] or %stripe%_ = #_ &then
  &set stripe = Working...
/*
/* Set the icon to be displayed in the menu
/* &set iconname = hourgls32.icon   /* Replaced with variable
&set iconname = %icon_name%
/*
/* Size the message menu based on the message string length
/*&set xsize = [LENGTH [QUOTE %message%]] * 10 + 60
/*&if %xsize% lt 250 &then &set xsize = 250
/*&set size = %xsize% 125
/*
&if not [SHOW &thread &exists tool$msworking] &then
  &thread &create tool$msworking ~
    &menu msworking2 ~
    &position [UNQUOTE %position%] ~
    &stripe [QUOTE [UNQUOTE %stripe%]]  ~
    &pinaction '&run msworking exit'

&thread &synchronize tool$msworking
/*
&return

/*------------------
&routine EXIT
/*------------------
/* Clean up
/*
&dv .msworking$*
&if [SHOW &thread &exists tool$msworking] &then
  &thread &delete tool$msworking
/*
```

```
&return

/*-------------------
&routine CLOSE
/*-------------------
/* Clean up
&call exit
/*
&return

/*------------------
&routine BAILOUT
/*------------------
&severity &error &ignore
&severity &warning &ignore
/*&call exit
&return &warning An error has occurred in routine: %routine% (MSWORKING.AML)

/*-----------------
&routine SAFETY_NET
/*-----------------
&return
```

```
WORKING2.MENU
7
/*---------------------------------------------------------------------------
/*            Environmental Systems Research Institute
/*---------------------------------------------------------------------------
/*    Menu: WORKING2.MENU
/* Purpose: Display a message while some action is executing.
/*---------------------------------------------------------------------------
/* Globals:
/*---------------------------------------------------------------------------
/*   Calls:
/*---------------------------------------------------------------------------
/*   Notes:
/*---------------------------------------------------------------------------
/* History: Matt McGrath - 02/10/92 - Mofified from the inform tool.
/*==========================================================================
=============

 %icn      %msg1
          %msg2
%icn display iconname 8 ICON
%msg1 display .msworking$message1 65
%msg2 display .msworking$message2 65
```

6 GENHRAP.F

```
c**********************************************************************
c Name and Location:  /export/home1/seann/hrapamls/crhrap/genhrap.f
c
c Purpose:  Write the HRAP coordinates for a selected region of cells to a
c file and create a subsequent file in geographic coordinates in a suitable
c format to serve as input to the GENERATE (polygon) command in ARC/INFO.
c This program is designed to be followed by genhrap.aml.
c
c Two options are available for defining the region of cells to be created --
c (1) Define the latitude and longitude extent of the region to be mapped, or
c (2) Specify the SW corner of the grid to be created and the number of colums
c and rows of cells to be created.  With either option, the program computes
c the HRAP coordinates of the SW corner (if necessary) and generates grid
c cells starting with the bottom row, moving from left to right, and then
c moving to the next row up and repeating.
c
c Comments:  Only the output files hrap.COD.dat and inputgc.COD are required
c as input to genhrap.aml.  Intermediate files and optional files that were
c created in an earlier version are also listed below.
c
c  Calls subroutines:  wll, topoly, crdat(numx,numy,xstart,ystart)
c
c Inputs: none
c Output: "COD" is a user defined suffix
c     hrap.COD = file of hrap coordinates  /*temporary
c     geoc.COD = file of geocentric coordinates  /*temporary
c     hrap.COD.dat = file containing HRAP coordinates in a format that can
c        be attached to the polygon attribute table
c     *pster.COD = file of polar stereographic coordinates
c     inputgc.COD = input file of geoc. coordinates to make a polgon
c        coverage
c     *inpster.COD = input file of polar stereographic coordinates to make
c     a polygon coverage
c     *inhrap.COD = input file of HRAP coordinates to make a polygon coverage
c
c A * denotes optional files -- the relevant lines have been commented out
c in this version.
c**********************************************************************

      program genhrap

c     <<< Variable Declaration >>>
      parameter (maxcol = 336, maxrow = 160)
c    *** maxcol and maxrow are limited to the extent of HRAP cells for which
```

```
c    *** data is available in the Arkans.-Red River Basin

      integer xstart,ystart,numx,numy,numpts,numx1,numy1
      double precision xhrap(maxcol), yhrap(maxrow)
      integer count,bool,rfunit,wfunit
c    *** rfunit and wfunit store the readfile unit number and the
c    *** writefile unit number to be passed to the subroutine topoly.
      character suff*3,file1*8,file2*8,file3*12,file4*9,file5*11
      character file6*11,file7*10
c    ***
c    <<< End of variable declaration >>>

c    *** Allow two options for defining the study region.
      print*, 'Enter 1 if you wish to specify the region by latitudes and
     1 longitudes of the corners of the study region.  Enter 2 if you \
     2 would like to specify region by hrap coordinates and number of \
     3 columns and rows.'

      read*, bool

      if (bool.eq.1) then
         call llinput(xstart,ystart,numx1,numy1)
        else

        print*, 'Enter the hrap(x,y) for the lower left hand corner of
     1the region of interest:'

        read*, xstart,ystart
        print*, 'Enter the number of grid columns and rows to be
     1created:'
        read*, numx1,numy1
      endif

c    *** Number of points to write is one greater than the number of
c    *** columns or rows.  The name numx1 can be thought of as number of
c    *** x coordinates - 1.

      numx = numx1 + 1
      numy = numy1 + 1
      print*, 'Enter a 3 character suffix to uniquely identify \
     1your grid:'
      read*, suff

c    ***Create names for all of the output files.
c    *** file1 = file of hrap coordinates
c    *** file2 = file of geocentric coordinates
```

```
c    *** file3 = file containing HRAP coordinates in a format that can be
c               attached to the polygon attribute table
c    *** file4 = file of polar stereographic coordinates
c    *** file5 = input file of geoc. coordinates to make a polgon coverage
c    *** file6 = input file of p. stereographic coordinates to make a polgon
c               coverage
c    *** file7 = input file of hrap coordinates to make a polgon coverage

      file1 = 'hrap.'//suff
      file2 = 'geoc.'//suff
      file3 = 'hrap.'//suff//'.dat'
c      file4 = 'pster.'//suff
      file5 = 'inputgc.'//suff
c      file6 = 'inpster.'//suff
c      file7 = 'inhrap.'//suff


      open(unit = 10, file = file1, status = 'unknown')
      open (unit = 20, file = file2, status = 'unknown')
      open (unit = 30, file = file3, status = 'unknown')
c       open (unit = 40, file = file4, status = 'unknown')
      open (unit = 50, file = file5, status = 'unknown')
c       open (unit = 60, file = file6, status = 'unknown')
c       open (unit = 70, file = file7, status = 'unknown')

c    *** Compute the total number of cell corners
      numpts = numx*numy

      xnew = xstart

      do 100 i=1,numx
        xhrap(i) = xnew
        xtemp = xnew + 1.0
        xnew = xtemp
 100  continue

      ynew = ystart
      do 200 j=1,numy
        yhrap(j) = ynew
        ytemp = ynew + 1.0
        ynew = ytemp
 200  continue

      count = 1
      do 300 j=1,numy
        do 400 i=1,numx
```

```
c          ** with "count" in the file, this file can be used as input
c          ** for creating an ARC/INFO point coverage
           write(10,*) count,xhrap(i),yhrap(j)
           count = count + 1
 400     continue
 300     continue
c    write(*,*) ystart

     call wll(numpts)
     rfunit = 20
     wfunit = 50
     call topoly(numx,numy,rfunit,wfunit)
c     rfunit = 40
c     wfunit = 60
c     call topoly(numx,numy,rfunit,wfunit)
c     rfunit = 10
c     wfunit = 70
c     call topoly(numx,numy,rfunit,wfunit)
     call crdat(numx,numy,xstart,ystart)
     close(10)
     close(20)
     close(30)
     close(50)
     stop
     end

c<<<<<<<<<<<<<<<<<<<<<<<SUBROUTINES>>>>>>>>>>>>>>>>>>>>>>>>>
c **************************************************************************
c
c Purpose: Convert HRAP coordinates contained in file "hrap.COD" to lat-long
c coordinates based on a spherical earth and write them to an output file
c that can be used to generate a point coverage.  The parameter "numpts"
c stores the number of entries that will be expected from "hrap.COD"
c
c Inputs: file hrap.COD
c Outputs: file geoc.COD
c
c**************************************************************************
     subroutine wll(numpts)
     double precision xhrap, yhrap, x, y
     double precision bigr, arg, latd, lond, ang
     double precision stlatd,earthr,mesh,stlond
     integer rec,numpts
c
c***    Define constants
     stlond = -105.0
```

```
      stlatd = 60.0
c*** earthr,mesh, x, and y are in meters.
      earthr = 6371200.0
      mesh = 4762.5

      rewind(unit=10)

      do 100 i=1,numpts

        read(10,*) rec,xhrap, yhrap
        x = (xhrap - 401.0)*mesh
        y = (yhrap - 1601.0)*mesh

        bigr = (x*x + y*y)**0.5
        arg = bigr/(earthr*(1 + dsind(stlatd)))
        latd = 90.0 - 2*datand(arg)

        ang = datan2d(y,x)

        if (y.gt.0) then
          ang = 270.0-stlond-ang
        else
          ang = -90.0-stlond-ang
        endif
        if (ang.lt.180) then
          lond = -1 * ang
        else
          lond = 360.0 - ang
        endif
c***    Write polar stereographic coordinates and geocentric
c***    coordinates to a file.
c         write(40,*) i,x,y
        write(20,*) i,lond, latd
 100  continue
      return
      end

c************************************************************************
c Purpose: Given a list of corner points for a grid (can be (ID,x,y) or
c          (ID, lon,lat) in which the coordinates for the bottom row are
c          listed one per line followed by the coordinates for the next row
c          up, create a file that can be used to generate a polygon coverage
c          of the grid cells.
c
c Input: File of corner points (ID,x,y),
c Ouput: File with lines: "poly-id, ll,lr,ur,ul,ll,end" -- repeated for
```

```
c        each polygon.  ll = lower left, lr = lower right, ur = upper right,
c        ul = upper left
c
c**********************************************************************
      subroutine topoly(numx,numy,rfunit,wfunit)

c      <<< Variable Declaration >>>
c      parameter (numx = 20, numy = 20)
c***    The old number of x-coordinates was 336.
c***    The old number of y-coordinates was 160.

      double precision xrowa(336),yrowa(336),xrowb(336),yrowb(336)
c        ** xrowa, yrowa are x and y coordinates of points in row a
      character*3 end
      integer i,l,rcount,r,polynum,numx,numy
      integer rfunit,wfunit
c     <<< End of Variable Declaration >>>

      end = 'end'

      rewind(unit=rfunit)
      rcount = 1
      polynum = 1

      do 200 i=1,numx
        read(rfunit,*) rec,xrowa(i),yrowa(i)
 200  continue

 100  if (rcount.lt.numy) then

      do 250 i=1,numx
        read(rfunit,*) rec,xrowb(i),yrowb(i)
 250  continue

      l = 1
 300  if (l.lt.numx) then
        r = l + 1
        write(wfunit,*) polynum, xrowa(l), yrowa(l)
        write(wfunit,*) xrowa(l),yrowa(l)
        write(wfunit,*) xrowa(r),yrowa(r)
        write(wfunit,*) xrowb(r),yrowb(r)
        write(wfunit,*) xrowb(l),yrowb(l)
        write(wfunit,*) xrowa(l),yrowa(l)
        write(wfunit,*) end
        l = l + 1
        polynum = polynum + 1
```

```
         goto 300
         endif

         rcount = rcount + 1
         do 350 i=1,numx
           xrowa(i) = xrowb(i)
           yrowa(i) = yrowb(i)
 350     continue
        goto 100
       endif
       write(wfunit,*) end

       return
       end

c ***********************************************************************
c Purpose: This subprogram will create a data file that can be joined to the
c projected "hrap" polygon coverage so that "hrap" coordinates of the lower
c left hand corner of each polygon will be added to the appropriate line in
c the PAT.
c
c Note: The only difference between "hrap.COD.dat" produced by this
c subroutine and "hrap.COD" produced by the main program is that
c hrap.COD.dat does not contain entries for the last column and last
c row of points.
c ***********************************************************************

       subroutine crdat(numx,numy,xstart,ystart)
c***    Old value of numx was 336
c***    Old value of numy was 160
       double precision xhrap(336), yhrap(160)
       integer count,numx,numy,xstart,ystart,numx1,numy1

       numx1 = numx - 1
       numy1 = numy - 1
       xnew = xstart

       do 100 i=1,numx1
         xhrap(i) = xnew
         xtemp = xnew + 1.0
         xnew = xtemp
 100  continue

       ynew = ystart
       do 200 j=1,numy1
         yhrap(j) = ynew
         ytemp = ynew + 1.0
         ynew = ytemp
 200     continue

      count = 1
      do 300 j=1,numy1
         do 400 i=1,numx1
            write(30,*) count,xhrap(i),yhrap(j)
            count = count + 1
 400        continue
 300     continue
      return
      end

c*********************************************************************
c     At user's request, allow the user to input the latitude and
c     longitude of the four corners that are of interest in the
c     study.
c
c     Note: The user should input geodetic coordinates.  These
c     geodetic coordinates will be interpreted as geocentric coordinates
c     to be consistent with methodology used by the
c     National Weather Service.
c*********************************************************************
      subroutine llinput(xstart,ystart,numx1,numy1)

c     <<< Variable Declaration >>>
      parameter (stlat = 60.0)
c*** clon is a constant used to account for the standard longitude
c*** see eqn. in "Geographic Positioning of the HRAP"
      parameter (clon = 15.0)
      parameter (rad = 6371.2)

      integer xstart,ystart,numx1,numy1
      real lon(4), lat(4)
      real sfactor,R,x,y,hrapx(4),hrapy(4)
c*** Declare variables llhrapx and llhrapy to pick the hrap coordinates of
c*** the lower left hand coordinates desired.
      real minhx,minhy,maxhx,maxhy
c     <<< End Variable Declaration >>>

      print*, 'Enter the latitudes and longitudes of four corners of a
     1 rectangle that encloses the study region (in decimal degrees). \
     2 Enter a longitude value and then a space and then a latitude \
     3 value. Hit return after each coordinate.  Remember to input West \
     4 longitude values as negative numbers.'
```

```
      do 100 i = 1,4

        read*, lon(i),lat(i)
        sfactor = (1+sind(stlat))/(1+sind(lat(i)))
c** x and y are in km
        R = rad*cosd(lat(i))*sfactor
        x = R*cosd(lon(i)+clon)
        y = R*sind(lon(i)+clon)
        hrapx(i) = x/4.7625 + 401
        hrapy(i) = y/4.7625 + 1601
        write(*,*) 'hrapx, hrapy:', hrapx(i), hrapy(i)
100   continue
        minhx = hrapx(1)
        minhy = hrapy(1)
        maxhx = hrapx(1)
        maxhy = hrapy(1)

      do 200 j = 2,4

         if (hrapx(j).lt.minhx) then
            minhx = hrapx(j)
         endif
         if (hrapy(j).lt.minhy) then
            minhy = hrapy(j)
         endif
         if (hrapx(j).gt.maxhx) then
            maxhx = hrapx(j)
         endif
         if (hrapy(j).gt.maxhy) then
            maxhy = hrapy(j)
         endif

 200  continue
      xstart = minhx
      ystart = minhy

      numx1 = maxhx - minhx
      numy1 = maxhy - minhy
      write(*,*) 'Lower left, num rows, num columns'
      write(*,*) xstart,ystart,numx1,numy1
      return
      end


c*********************************************************************
```

7 GENHRAP.AML

```
/*********************************************************************
/*********************************************************************
/*  Name and Location: /export/home1/seann/hrapamls/crhrap/genhrap.aml
/*  Purpose:  Generate polygon coverage(s) from user specified input file(s)
/*  (i.e. inputgc.COD)
/*  generated by genhrap.f, project the polygon coverage into chosen
/*  projection.  Create an INFO file with HRAP-IDs (given hrap.COD.dat), and
/*  join this INFO file to the PAT of the projected polygon coverage.
/*********************************************************************
/*********************************************************************

&sv suff = [response 'Enter the 3 character suffix used to ID hrap files:']
&sv covgc = %suff%geocc
&sv inputgc = inputgc.%suff%

&if [exists %covgc% -cover] &then
   kill %covgc% all
generate %covgc%
&if [exists %inputgc% -file] &then
input %inputgc%
&else &type Can't find input file.
polys
/* must quit out of the GENERATE sub-program
quit

clean %covgc%
&sv covgcprj = %covgc%alb
&if [exists %covgcprj% -cover] &then
   kill %covgcprj% all
project cover %covgc% %covgcprj% albdd.prj
clean %covgcprj%

tables
&if [exists hrapxy2.dat -info] &then
   &sv delvar = [delete hrapxy2.dat -info]

/* Add data to the INFO file hrapxy2.dat from the file hrap.***.dat
/* created by the FORTRAN program create.f
&sv addfile = hrap.%suff%.dat
/*add from %addfile%

define hrapxy2.dat
%covgcprj%-id
```

```
5
5
i
hrapx
4
4
i
hrapy
4
4
i
~
add from %addfile%
quit

/* Join the newly created INFO file to the PAT, creating two new colums
/* in the HRAP polygon coverage
joinitem %covgcprj%.pat hrapxy2.dat %covgcprj%.pat %covgcprj%-id %covgcprj%-id
~
        ordered
&return

/* Listing of albdd.prj
/*input
/*projection geographic
/*units dd
/*datum wgs72
/*parameters
/*output
/*projection albers
/*units meters
/*datum wgs72
/*parameters
/*29 30 00
/*45 30 00
/*-96 00 00
/*23 00 00
/*0.0
/*0.0
/*end
```

8 HRAP_INT.AML

```
/*******************************************************************************
/*******************************************************************************
/* Name: hrap_int.aml
/*
/* Purpose:  Intersect polygons representing subwatersheds and a radar
/* rainfall grid.  For the resulting coverage, determine
/* the mean, max, and min and median Flowlengths to the outlet from each of
/* the polygons and record this in the PAT of that coverage. Also compute
/* mean, max, and min values of the time_index parameter if desired.
/*
/* Execution: &r hrap_int <wshed_cov> <hrap_cov> <value_grid> <wshed_grid>
/*  <outfile>
/*
/* Inputs:   (1) a projected polygon coverage of subwatersheds, (2) a polygon
/*              coverage of an HRAP grid to intersect with the subwatershed
/*              coverages, (3) a value grid, (4) and a grid of the
/*           subwatershed.
/*
/* Outputs:   An output file unloaded from sector_cov.pat containing the
/*           following information for each subbasin: hrapx, hrapy,
/*           travel length to a subbasin outlet, and area of that cell
/*           draining to that subbasin.
/*
/* Comments: Polygons in sector_cov may be smaller than the size of one grid
/*          cell.  In this case, sector_grid.vat will contain fewer entries
/*          than sector_cov.pat because these small polygons were dropped.
/*          The precipitation and flowlength values written to sector_cov.pat
/*            for these polygons is zero.
/*            Before running this program, make sure that the HRAP polygons have
/*            been cleaned and projected into the same projection as the
/*            subwatershed coverage.  Also, make sure the coverage contains
/*            hrapx and hrapy values in its PAT.
/*
/******************************************************************************
/******************************************************************************

&args .subshed_cov .hrap_cov .valu_grid .subshed_grid .outfile

&if [exists sector_cov -cover] &then
  kill sector_cov all

intersect %.subshed_cov% %.hrap_cov% sector_cov

grid
```

```
&type what
&if [exists sector_grid -grid] &then
  kill sector_grid all

/*specify the cell size below
&describe %.valu_grid%
&sv cellsize = %grd$dx%
&sv max_fl = %grd$zmax%

sector_grid = polygrid (sector_cov,#,#,#,%cellsize%)
&type what

/***
/*  Create an INFO table that contains VALUE, COUNT, MEAN, MAX, MIN, and
/*  MEDIAN.
/*  VALUE = values of zones defined by sector_grid
/*  COUNT = number of cells in zones defined by sector_grid
/*  MEAN = mean of values from flowlength grid in zone defined by VALUE
/***

&if [exists flength.stat -info] &then
   &sv delvar = [delete flength.stat -info]
&if [exists flength.med -info] &then
   &sv delvar = [delete flength.med -info]
&if [exists sbac.stat -info] &then
   &sv delvar = [delete sbac.stat -info]
&if [exists sbac.med -info] &then
   &sv delvar = [delete sbac.med -info]
&if [exists flmerge_int -grid] &then
   kill flmerge_int all
&if [exists sbac_int -grid] &then
   kill sbac_int all
&if [exists time_ind_int -grid] &then
   kill time_ind_int all
&if [exists time_ind.stat -info] &then
   &sv delvar = [delete time_ind.stat -info]
&if [exists time_ind.med -info] &then
   &sv delvar = [delete time_ind.med -info]

/*flmerge_int = int (%.valu_grid%)
/*buildvat flmerge_int

flength.stat = zonalstats(sector_grid,%.valu_grid%)

/*flength.med = zonalstats(sector_grid,flmerge_int,median)
```

```
/*sbac_int = int (sbac)
/*sbac.stat = zonalstats(sector_grid,sbac)
/*sbac.med = zonalstats(sector_grid,sbac_int,median)

/*time_ind_int = int (time_ind)
/*time_ind.stat = zonalstats(sector_grid,time_ind)
/*time_ind.med = zonalstats(sector_grid,time_ind_int,median)

/* quit out of grid: joinitem cannot be used at the grid prompt
quit

/**********************************************************************
/* Join the info files created by zonalstats so that only one relate between
/* the PAT and the INFO files needs to be created.
/**********************************************************************

/*joinitem flength.stat flength.med flength.stat value max ordered

/*joinitem sbac.stat sbac.med sbac.stat value max ordered

/*joinitem time_ind.stat time_ind.med time_ind.stat value max ordered

/** Combining the three statistics tables was considered so that only one
/** "relate" would have to be established for the purpose of unloading data;
/** however, this would have required changing at least four of the item names.

/*joinitem flength.stat sbac.stat all.stat value median ordered

/*"Cursor" is not a valid command in TABLES


/* Declare a cursor named basin_cur on .subshed_grid.vat
/*&messages &off &all
&sv end_of_subsheds = .FALSE.
&sv count = 0
/*  &sv temp = 0

cursor subshed_cur declare %.subshed_grid%.vat info ro

cursor subshed_cur open

/*** Use a loop to count the number of subwatersheds.  Store the value
/*   in the variable 'count.'  The variable count will be used to control
/*   the loop that unloads data to an ASCII file.

&if %:subshed_cur.AML$NEXT% = .FALSE. &then
```

```
  &sv end_of_subsheds = .TRUE.
/* Make sure that 'temp' is the same item type as ':subshed_cur.value.'
&sv temp = %:subshed_cur.value% - 1
&do &while %end_of_subsheds% = .FALSE.
   &if %temp% ne %:subshed_cur.value% &then
      &do
         &sv count = %count% + 1
         &sv basin%count% = %:subshed_cur.value%
         &sv temp = %:subshed_cur.value%
         /* The variable 'temp' is used so that a subwatershed will not be
         /* counted more than once if it is listed twice in the VAT.
      &end

/*  Read next record from .subshed_grid.vat
  cursor subshed_cur next
  &if %:subshed_cur.AML$NEXT% = .FALSE. &then
    &do
      &sv end_of_subsheds = .TRUE.
      cursor subshed_cur remove
    &end
&end

/************************************************************************************
/*  Enter tables to perform two tasks: (1) establish two relations: (a) between
/*  sector_cov.pat and flength.stat - call it "relfl" (b) between sector_cov.pat
/*  and time_ind.stat - callit "relti"; (2)  Using a loop and simple relates
/*  unload desired output for each of the sub-watersheds from the tables
/*  sector_cov.pat, flength.stat, and time_ind.stat.
/*************************************************************************************

tables
sel sector_cov.pat
relate add
relfl
flength.stat
info
sector_cov#
value
ordered
ro
/*relsb
/*sbac.stat
/*info
/*sector_cov#
/*value
/*ordered
```

```
/*ro
/*relti
/*time_ind.stat
/*info
/*sector_cov#
/*value
/*ordered
/*ro
~

&if [exists %.outfile% -file] &then
   &sv delvar = [delete %.outfile% -file]

/*** Open the output file for writing.

&sv wfunit = [open %.outfile% openstat -append]
  &if %openstat% ne 0 &then
     &do
        &type openstat = %openstat%
        &stop Cannot open the output file %.outfile%
     &end
  &else &type File %.outfile% opened succussfully for writing.

/** Write the number of sub-watersheds being processed to the output file.
   &if [write %wfunit% %count%] ne 0 &then
      &do
         &type Error in writing to output file.  Exiting AML.
         &return
      &end

/***   Do not need to leave the output file open if using the "unload"
/*   function in tables because this function automatically opens and
/*   closes the file to which it writes.

&if [close %wfunit%] = 0 &then
   &type %.outfile% closed successfully

&sv loops = 1
&do &while %loops% le %count%
   /*  Update user on status.
   &type Processing watershed [value basin%loops%]

   select sector_cov.pat
   /** Reduce the selection to all of the polygons that are larger than one
   /** one grid cell.
   reselect sector_cov# = relfl//value
   reselect grid-code = [value basin%loops%]
   /** The unload command closes the file "hec.out."

   unload %.outfile% grid-code hrapx hrapy relfl//mean area ~
               delimited

/*   unload %.outfile% grid-code hrapx hrapy area relfl//mean relfl//max ~
/*              relfl//min relti//mean relti//max relti//min delimited

/*   unload %.outfile% grid-code hrapx hrapy area relfl//mean relfl//max ~
/*              relfl//min relfl//median relti//mean relti//max relti//min ~
/*              relti//median delimited

   &sv loops = %loops% + 1
&end   /*End of unloading real data.

/** Unload a list of the polygons (and their respective areas) that were
/** dropped during polygrid due to the fact that they had an area smaller
/** than the size of one grid cell.  To file "dropped.out"
   select sector_cov.pat
   reselect sector_cov# = relfl//value
   nselect
   unload dropped.out hrapx hrapy area

/*****************
/* Drop any "relates" before ending.
/*****************

relate drop
relfl
~
/*relate drop
/*relsb
/*~
/*relate drop
/*relti
/*~

/*&messages &on
/** Exit tables
quit
&return
```

9 MOUTPUT.F

```
/*******************************************************************
/*******************************************************************
/* Name: moutput.f
/*
/* Purpose: Translate data into the input file format for modClark
      program moutput
c     ** Modifies the ascii file created by hrap_int.aml to the form
c     ** requested by HEC.

      character sb*10, gc*10, e*4
c     ** dat(3) stores the first three data items for the current gridcell
c     ** dat(9) stores the last nine data items for the current gridcell
      integer count,dati(3)
c     ** with the median, the length of the datr array will  be 9 instead
c     ** of 7
      real datr(2)

      sb = 'SUBBASIN:'
      gc = 'GRIDCELL:'
      e = 'END:'


      open (unit = 20, file = 'tk3file.out', status = 'unknown')
      open (unit = 40, file = 'tk3modc.in', status = 'unknown')
      read(20,*) count
      read(20,*) dati, datr
      do 100 i = 1,count
        write(40,110) sb,dati(1)
 110    format (A,2x,I3)
        temp = dati(1)
c     ** If the first entry in the current row is different from the
c     ** first entry in the previous row, then the current cell
c     ** is in the same watershed and write the characteristics for the
c     ** grid-cell.
 115    if (dati(1).eq.temp) then

c***  Convert flowlength from meters to kilometers and
c***  area from meters^2 to km^2
        datr(1) = datr(1) / 1000.0
        datr(2) = datr(2) / 1000000.0
c     ** with median k = 1,9
          write(40,120) gc, (dati(j), j=2,3), (datr(k), k=1,2)
 120      format(A,1x,I3,1x,I3,1x,f8.4,1x,f7.4)
```

```
c          ** next line is the format to be used with median
c 120          format(I3,1x,I3,1x,f10.1,1x,3f9.1,1x,f8.0,1x,3f8.1,1x,f7.0)
               read(20,*,END=200) dati,datr

             goto 115
             endif
             write(40,*) e
 100       continue
 200       write(40,*) e

           end
```