

# ENVI Tutorial: Introduction to User Functions

---

Introduction to User Functions.....	2
Files Used in this Tutorial.....	2
Background.....	2
Band Math.....	3
Open TM Data.....	3
Explore a Band Math User Function.....	3
Compile the Band Math Function.....	4
Run the Band Math Function.....	5
Event Handlers.....	6
Examine Event Handler Code.....	6
Add the Event Handler to the ENVI Menu System.....	8
Compile the Event Handler.....	9
Run the Event Handler.....	9
Tiling Routines.....	11
Compile the Tile Processing Routine.....	14
Run the Tile Processing Routine.....	15

# Introduction to User Functions

This tutorial shows you how to compile and run a simple Band Math user function from within ENVI. You will then explore a modified version of the user function that includes an error handler and widgets that prompt for user input. Finally, you will see how a tiling routine is incorporated into the user function to process tiles of data.

This tutorial assumes that you are familiar with the Interactive Data Language (IDL) and that you understand how to write functions and procedures in IDL. You need ENVI+IDL for this tutorial.

## Files Used in this Tutorial

ENVI Resource DVD: Data\programming

File	Description
bm_divz2.pro	Band Math user function
tp_divz1.pro	User function (based on bm_divz2.pro) with event handler
tp_divz2.pro	User function (based on tp_divz1.pro) with event handler + tiling routine

ENVI Resource DVD: Data\bldr\_reg

File	Description
bldr_tm.img	Landsat TM image of Boulder, CO

## Background

One of ENVI's best features is that its functionality is not limited to what you find on the menu. ENVI developers designed the software so that it can be easily customized. Common ENVI extensions include user Band Math and Spectral Math functions, custom spatial, spectral, or region of interest (ROI) processing, user functions, custom file input routines, batch processing, and other report and plotting tools. Many ENVI library routines are available to help you write custom routines while maintaining the same look-and-feel as ENVI.

You can enter most Band Math and Spectral Math expressions directly in ENVI's Band Math and Spectral Math dialogs, respectively. Or, you can write user functions to handle the data input, output, and user interfaces. You can use Band Math and Spectral Math successfully with only a limited knowledge of IDL.

You can write user functions in IDL, C, Fortran, or other high-level languages, integrate them into ENVI, and execute them from the ENVI menus. User functions get input data from ENVI and enter results directly into ENVI. Also, ENVI provides a library of routines and programming tools written in IDL to handle input, output, plotting, reports, and file management. You can use many of the ENVI library routines (such as classification) in user functions or batch routines.

Please refer to the *ENVI Programmer's Guide* and *ENVI Reference Guide* (both available through ENVI Help) for more details.

## Band Math

This exercise shows you how to compile a .pro file (an IDL program) containing a simple Band Math user function, and how to call the user function from within ENVI. This simple user function performs a mathematical expression using two bands of Landsat TM data.

### Open TM Data

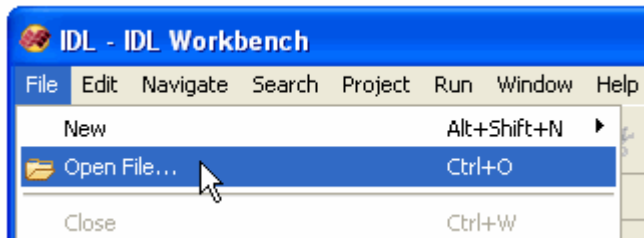
Before attempting to start the program, ensure that ENVI is properly installed as described in the *Installation and Licensing Guide* that shipped with your software.

1. From the ENVI main menu bar, select **File > Open Image File**. The Enter Data Filenames dialog appears.
2. Navigate to `Data\bldr_reg` and select `bldr_tm.img`. Click **Open**. An RGB composite appears in a new display group.

### Explore a Band Math User Function

Normally, you would open an Editor window in the IDL Workbench that starts when ENVI starts (assuming you have ENVI+IDL) to write a Band Math user function. For this exercise, however, you will use a pre-generated user function.

1. From the IDL menu bar, select **File > Open File**. The Open File dialog appears.



2. Navigate to `Data\programming` and select `bm_divz2.pro`. Click **Open**. The following code appears in an Editor window:

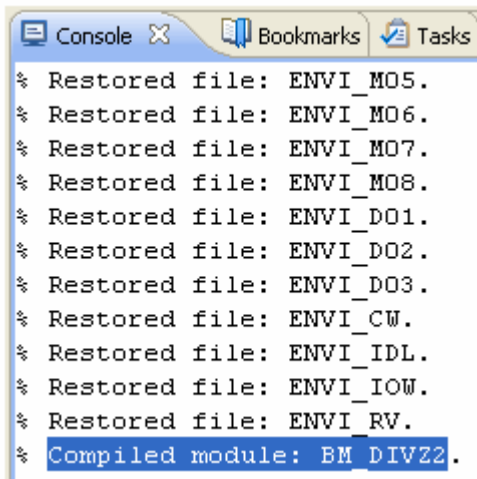
```
function bm_divz2, b1, b2, check=check, div_zero=div_zero
  if (keyword_set(check)) then begin
    ; If div_zero is set then use it otherwise use zero
    if (n_elements(div_zero) gt 0) then $
      temp_value = div_zero $
    else $
      temp_value = 0.0
    ; Find all the locations where the band is zero
    temp = float(b1) - b2
    ptr = where(temp eq 0, count)
    ; Temporarily set the divide by zero cases to divide by 1
    if (count gt 0) then $
      temp(ptr) = 1
    ; Perform the ratio
    result = (float(b1) + b2) / temp
    ; If any divide by zeros then set the output
    if (count gt 0) then $
      result(ptr) = temp_value
  endif else begin
    ; Just do the ratio and ignore divide by zeros
    result = (float(b1) + b2) / (float(b1) - b2)
  endelse
  return, result
end
```

This user function performs a simple ratio that divides the addition of two Landsat TM bands (represented by the b1 and b2 variables) by their difference. The optional CHECK keyword enables divide-by-zero checking. If any such errors are found, the DIV\_ZERO keyword sets them to a different value. See ENVI Help for a detailed discussion of writing proper Band Math expressions.

## Compile the Band Math Function

1. From the ENVI main menu bar, select **File > Compile IDL Module**. The Enter Module Filename dialog appears.
2. Navigate to Data\programming and select bm\_divz2.pro. Click **Open**.

3. To make sure the function properly compiled, look for a line that says "Compiled module BM\_DIVZ2." On a Windows platform, this line appears in the IDL Console (see following figure). On a Unix platform, the line appears in the shell window from which you started ENVI.



```
Console X Bookmarks Tasks
* Restored file: ENVI_M05.
* Restored file: ENVI_M06.
* Restored file: ENVI_M07.
* Restored file: ENVI_M08.
* Restored file: ENVI_D01.
* Restored file: ENVI_D02.
* Restored file: ENVI_D03.
* Restored file: ENVI_CW.
* Restored file: ENVI_IDL.
* Restored file: ENVI_IOW.
* Restored file: ENVI_RV.
* Compiled module: BM_DIVZ2.
```

## Run the Band Math Function

1. From the ENVI main menu bar, select **Basic Tools > Band Math**. The Band Math dialog appears.
2. In the "Enter an expression" field, you can enter a simple mathematical expression or call a user function that involves a more complex Band Math operation; you will perform the latter. Enter the following line in the Enter an expression field:

```
bm_divz2(b1,b2, /check, div_zero=1.0)
```

3. Click **OK**. The Variables to Band Pairings dialog appears.
4. Select **B1 – [undefined]** in the **Variables used in expression** field.
5. In the Available Bands List field of the Variables to Band Pairings dialog, select **Band 2**.
6. Select **B1 – [undefined]** in the Variables used in expression field.
7. In the Available Bands List field of the Variables to Band Pairings dialog, select **Band 3**. The **Variables used in expression** field should contain the following:

```
B1 - Band 2 (0.5600):bldr_tm.img
```

```
B2 - Band 3 (0.6600):bldr_tm.img
```

8. Enter an output filename and click **OK**.
9. In the Available Bands List, click **Display #1** and select **New Display**.
10. Select the **Gray Scale** radio button, select the band named **Band Math**, and click **Load Band**.
11. Explore the image. Any pixels that caused a divide-by-zero error now have values of 1.0.

## Event Handlers

ENVI user functions allow you to add new routines to interactive ENVI. You can add your user function to the ENVI menu system where it becomes semi-permanent, meaning it will remain there until you choose to remove it. When you run your user function from the menu system, you are essentially running an IDL program that calls ENVI routines.

When designing user functions, you can choose to use no interface, use ENVI's compound widgets to simplify interface design and give your functions the same look-and-feel as ENVI, or create your own interface using IDL widgets. If you choose to use ENVI's compound widgets, you can let ENVI automatically manage input from your interface to your user function.

ENVI is an IDL widget program. Because user functions are additions to an IDL widget program, they are technically *event handlers*, a special class of IDL routines that are executed in response to a widget event. A widget event occurs when you select the user function from the ENVI menu system. The distinction between an ordinary IDL procedure and an event handler is purely semantic; the only practical difference is that the procedure definition statement for a user function must include a positional parameter to receive the event structure variable.

You define all aspects of the user function, including the level of user interaction (from none to extensive). You can write user functions in IDL, C, Fortran, or other high-level languages and save them as .pro or .sav files in the save\_add directory of your ENVI installation, where they are automatically compiled or restored into the ENVI session's memory when ENVI starts. Once you have added the user function to ENVI, you can modify its code any time, recompile it within the current ENVI session, and use it in its modified form without having to restart ENVI.

In this exercise, you will look at a pre-generated program called `tp_divz1.pro`, which is a user function that performs the same mathematical operation as `bm_divz2.pro`. However, `tp_divz1.pro` includes an event handler and widgets.

### Examine Event Handler Code

1. From the IDL menu bar, select **File > Open File**. The Open File dialog appears.
2. Navigate to `Data\programming` and select `tp_divz1.pro`. Click **Open**.
3. Take a moment to examine the code in this file. Notice the variety of “widget” and “envi” routines. These are ENVI library routines used to extend ENVI functionality. They are extensively documented in the *ENVI Reference Guide* (available through ENVI Help).
4. This code shows how an event handler works. When you select the user function from the ENVI menu system, a widget event occurs. When you add the user function to the ENVI menu system (by editing the file `envi.men`) later in this tutorial, you will assign the menu item a user value (`uvalue`), which programmatically determines what type of event occurred.

The user function first extracts a `uvalue`, and compares it to the `uvalue` assigned to the menu item.

```
widget_control, ev.id, get_uvalue=uvalue
  if (uvalue eq 'user ratio') then begin
```

5. After receiving the `uvalue`, the program prompts you to select an input file. The library routine `ENVI_SELECT` is used for file selection and spatial and spectral subsetting. Returned from

ENVI\_SELECT are the input file ID (FID), the band dimensions (DIMS), and the selected bands (POS). If you click **Cancel** in the widget, the returned FID value is -1, and the event handler must exit.

```
envi_select, title='Ratio Input File', fid=fid, dims=dims, $
  pos=pos
  if (fid eq -1) then return
```

6. Since the user function performs the same mathematical expression as the earlier Band Math example, it must ensure that a user selects at least two bands. If a user does not select at least two bands, the library routine ENVI\_ERROR issues a warning message and the event handler exits.

```
if (n_elements(pos) lt 2) then begin
  mstr = 'You must select two bands to ratio.'
  envi_error, mstr, /warning
  return
endif
```

7. The remaining code creates compound widgets for you to enter Band Math parameters. See the *ENVI Reference Guide* (available through ENVI Help) for further details.

WIDGET\_AUTO\_BASE creates a widget base that allows widget events to be automatically managed by ENVI.

Row and column bases are used to position the compound widgets WIDGET\_MENU and WIDGET\_PARAM.

WIDGET\_MENU checks for divide-by-zero errors, using an exclusive list to make a Yes/No button.

WIDGET\_PARAM accepts an input value that is used as the replacement for divide-by-zero errors.

WIDGET\_PARAM accepts a floating point number (DT=4) with three decimal places (FIELDS=3) and a default value of 0 (DEFAULT=0.0).

WIDGET\_OUTFM is a widget that includes an output filename or memory option.

Each of the compound widgets sets the keyword /AUTO, giving AUTO\_WID\_MNG the responsibility of managing the compound widgets.

AUTO\_WID\_MNG returns a structure with the tag names defined by the uvalue of each compound widget. The structure contains the tag accept, which is set to 1 if you click **OK** and 0 if you click **Cancel**.

```

; Create a compound widget for the input parameters
base = widget_auto_base(title='Ratio Parameters')
  sb = widget_base(base, /column, /frame)
  sb1 = widget_base(sb, /row)
  mw = widget_menu(sb1, prompt='Check for divide by 0 ? ', $
list=['Yes', 'No'], /excl, default_ptr=0, rows=0, $
  uvalue='check', /auto)
  sb1 = widget_base(sb, /row)
  wp = widget_param(sb1, prompt='Divide by zero value', $
dt=4, field=3, xs=6, uvalue='div_zero', default=0.0, /auto)
  sb = widget_base(base, /column, /frame)
  ofw = widget_outfm(sb, func='envi_out_check', $
uvalue='outf', /auto)
; Automanage the widget
result = auto_wid_mng(base)
if (result.accept eq 0) then return
check = (result.check eq 0)
div_zero = result.div_zero

```

8. Finally, the code adds items to the widget that allow you to specify whether or not you want divide-by-zero checking, and if so, the replacement data value for divide-by-zero errors.

## Add the Event Handler to the ENVI Menu System

The ENVI menu system is comprised of the ENVI main menu bar that appears when you start ENVI, and the Display group menu bar, which is accessed only from display group windows. These are defined by two ASCII files located in the menu directory of the ENVI installation:

The `envi.men` file defines the ENVI main menu bar, and the `display.men` file defines the Display group menu bar. Each time a new ENVI session is started, ENVI reads the two menu files and constructs the menus based on the content of the files.

Each ENVI menu item is defined by a one-line entry in one of the files. The item's definition statement includes a number that defines the level of the menu item (how deeply it is nested within other pull-down menu items), the text that appears on the item, a widget user value for the item, and the name of the routine that is executed when the item is selected (i.e., the name of the user function in the `save_add` directory).

Because both menu files are editable, you can change the entire ENVI menu system. You can rename, move, duplicate, or completely remove menu items. Similarly, you can add a user function's menu item to any location. For example, if the user function is a new filtering routine, you may choose to add it to the **Filter** menu.

1. Using a text editor, open the `envi.men` file located in the menu directory of the ENVI installation (`xx` represents the software version).

Windows: C:\Program Files\ITT\IDLxx\products\envi`xx`\menu

Unix: `usr/local/itt/idlxx/products/envixx/menu`

2. Scroll to the end of `envi.men` and add the following lines immediately before the line that says `0 {Help}`:

```
0 {User Functions}
1{User Band Ratio1} {user ratio} {tp_divz1}
1{User Band Ratio2} {user ratio} {tp_divz2}
```

The number at the beginning of the line defines the hierarchy of the menu item (0 is a main item that opens a pull-down menu, 1 is the first level of items beneath the main menu, 2 is a nested menu item beneath a first-level item, etc.) Following are descriptions for the items in the first line beginning with 1.

{User Band Ratio1} — The text for the menu item

{user ratio} — The user value assigned to the menu item

{tp\_divz1} — The name of the user function to execute when the menu item is selected. You will notice that none of the event handler names include the .pro or .sav extensions. The name of the user function should be listed here, not the name of the file that contains the procedure.

3. Save and close `envi.men`.

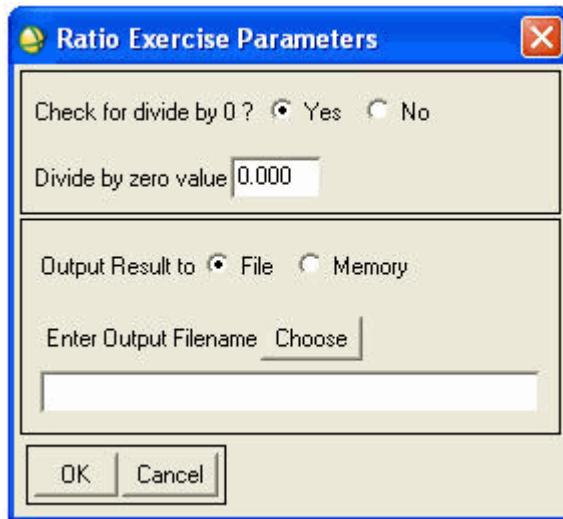
## Compile the Event Handler

ENVI can automatically compile `tp_divz1.pro` if you place it in the `save_add` directory of the ENVI installation and restart ENVI. For this exercise, however, you will learn how to compile the routine within ENVI.

1. From the ENVI main menu bar, select **File > Compile IDL Module**. The Enter Module Filename dialog appears.
2. Navigate to `Data\programming` and select `tp_divz1.pro`. Click **Open**.
3. To make sure the function properly compiled, look for a line that says "Compiled module TP\_DIVZ1." On a Windows platform, this line appears in the IDLDE Command Log. On a Unix platform, the line appears in the shell window from which you started ENVI.

## Run the Event Handler

1. From the ENVI main menu bar, select **User Functions > User Band Ratio1**. The Ratio Input File dialog appears.
2. Navigate to `Data\bldr_reg` and select `bldr_tm.img`. Click **OK**. The Ratio Exercise Parameters dialog appears.



3. Select the **No** radio button for **Check for divide by 0 ?**
4. Enter an output filename of `tp_divz1.img` and click **OK**.
5. Look in the IDL Console window to see the values printed at the bottom of the event handler.

```
Compiled module: TP_DIVZ1.  
*** Structure <47d1000>, 3 tags, length=20, data length=18, refs=2:  
NAME      STRING  'tp_divz1.img'  
IN_MEMORY LONG    0  
COMPRESSION INT     0  
0  
0.00000000
```

## Tiling Routines

Remote sensing datasets are often very large. In fact, you may encounter images as large as several gigabytes, especially if the data are calibrated. To avoid problems associated with limited RAM, ENVI automatically breaks large images into smaller pieces for processing. The size of the pieces, or tiles, are picked small enough that they will easily fit into your available RAM. The tile size (in bytes) is defined in the ENVI configuration file (from the ENVI main menu bar, select **File > Preferences > Miscellaneous**), and its default setting is 1 MB. In this exercise, you will convert the Band Math user function into a tiling routine that uses the event handler from the previous exercise.

1. From the IDL menu bar, select **File > Open File**. The Open File dialog appears.
2. Navigate to `Data\programming` and select `tp_divz2.pro`. Click **Open**. The following steps describe the code in more detail.
3. This user function first establishes I/O error handling. Next, `ENVI_FILE_QUERY` is used to get the filename (`FNAME`) and X and Y starting pixel (`XSTART` and `YSTART`). The filename is used for the processing status report, and `XSTART` and `YSTART` are used in the output image header.

```
pro tp_divz_doit, fid=fid, pos=pos, dims=dims,check=check,$
  out_name=out_name, in_memory=in_memory, $
  div_zero=div_zero, r_fid=r_fid
  ; Set up the error catching and initialize optional keywords
  !error = 0
  on_ioerror, trouble
  in_memory = keyword_set(in_memory)
  ; Get the file xstart and ystart and calculate ns and nl
  envi_file_query, fid, fname=fname, xstart=xstart, $
  ystart=ystart
  ns = dims(2) - dims(1) + 1
  nl = dims(4) - dims(3) + 1
```

4. The tile processing routine allows both output to file and memory. File operations open the output file for writing, and memory output allocates a float array.

```
; Either allocate a memory array or open the output file
get_lun, unit
if (in_memory) then $
  mem_res = fltarr(ns, nl) $
else $
openw, unit, out_name
```

5. Next, the ENVI tiles are initialized using `ENVI_INIT_TILE`. Since the processing routine uses two bands simultaneously, the `MATCH_ID` is used on the second `ENVI_INIT_TILE`, forcing processing tiles from the two bands to be the same size. `ENVI_REPORT_INIT` and `ENVI_REPORT_INC` set up a processing status widget and the report increment, respectively.

```
; Initialize the data tiles
tile_id1 = envi_init_tile(fid, pos(0), $
  num_tiles=num_tiles, xs=dims(1), xe=dims(2), $
  ys=dims(3), ye=dims(4), interleave=0)
tile_id2 = envi_init_tile(fid, pos(1), match_id=tile_id1)
; Setup the processing status report
if (in_memory) then tstr = 'Output to Memory' $
else tstr = 'Output File: ' + out_name
envi_report_init, ['Input File: ' + fname, tstr], $
  title='Ratio Processing', base=rbase, /interrupt
envi_report_inc, rbase, num_tiles
```

6. Now that everything is initialized, the processing routine can just loop over the number of tiles. At the start of each loop, the processing status is updated and the code checks to see if the Cancel button was selected. The calls to ENVI\_GET\_TILE with the TILE\_IDS return the data to process. Each tile is processed with the mathematical expression from the Band Math exercise. After processing, the data memory items are written to the variable `mem_res`. Otherwise, the result is written to a file.

```

; Loop over each processing tile
for i=0, num_tiles-1 do begin
    envi_report_stat, rbase, i, num_tiles, cancel=cancel
    if (cancel) then begin
!error = envi_cancel_val()
goto, trouble
    endif
    ; Retrieve the tile data
    data1 = envi_get_tile(tile_id1, i, ys=ys, ye=ye)
    data2 = envi_get_tile(tile_id2, i)
    ; Perform the ratio
    if (keyword_set(check)) then begin
; Find all the locations where the band is zero
temp = float(data1) - data2
ptr = where(temp eq 0.0, count)
; Temporarily set the divide by zero cases to divide
; by 1, do the ratio, and then set the divide by zero
; to div_zero
if (count gt 0) then $
temp(ptr) = 1
result = (float(data1) + data2) / temp
if (count gt 0) then $
    result(ptr) = div_zero
    endif else begin
; Just do the ratio and ignore divide by zeros
result = (float(data1) + data2) / $
    (float(data1) - data2)
    endelse
    if (in_memory) then $
mem_res(0,ys-dims(3)) = result $
    else $
writeu, unit, result
    endif
endfor
; Process error messages
!error = 0
trouble: if (!error ne 0) then $
    envi_io_error, 'Ratio Processing', unit=unit
    free_lun, unit
    if (!error eq 0) then begin
        descrip = 'Ratio Processing'
        if (in_memory) then $

```

- Now that the processing is complete, ENVI\_ENTER\_DATA or ENVI\_SETUP\_HEAD enters the new image into ENVI. Memory items use ENVI\_ENTER\_DATA while output to disk uses ENVI\_SETUP\_HEAD to open the file and to write the ENVI header file (.hdr).

```

envi_enter_data, mem_res, descrip=descrip, $
    xstart=xstart+dims(1), ystart=ystart+dims(3), $
    r_fid=r_fid $
    else $
envi_setup_head, fname=out_name, ns=ns, nl=nl, nb=1, $
data_type=4,interleave=0, xstart=xstart+dims(1), $
    ystart=ystart+dims(3), /write, /open, r_fid=r_fid, $
    descrip=descrip
endif

```

8. Finally, the tile pointers and report are cleaned up using ENVI\_TILE\_DONE and ENVI\_REPORT\_INIT.

```
; Clean up the tile pointer and the status report
envi_tile_done, tile_id1
envi_tile_done, tile_id2
envi_report_init, base=rbase, /finish
end
```

9. The rest of the code is the same as the previous tp\_divz1.pro example:

```
pro tp_divz2, ev
  widget_control, ev.id, get_uvalue=uvalue
  if (uvalue eq 'user ratio') then begin
    envi_select, title='Ratio Input File', fid=fid, dims=dims, $
    pos=pos
    if (fid eq -1) then return
    ; We will just do a ratio of the first two band
    ; from the pos array so make sure there are at
    ; least two bands are selected
    if (n_elements(pos) lt 2) then begin
      mstr = 'You must select two bands to ratio.'
      envi_error, mstr, /warning
    return
    endif
    ; Create a compound widget for the input parameters
    base = widget_auto_base(title='Ratio Parameters')
    sb = widget_base(base, /column, /frame)
    sb1 = widget_base(sb, /row)
    mw = widget_menu(sb1, prompt='Check for divide by 0 ? ', $
    list=['Yes', 'No'], /excl, default_ptr=0, rows=0, $
    uvalue='check', /auto)
    sb1 = widget_base(sb, /row)
    wp = widget_param(sb1, prompt='Divide by zero value', $
    dt=4, field=3, xs=6, uvalue='div_zero', default=0.0, /auto)
    sb = widget_base(base, /column, /frame)
    ofw = widget_outfm(sb, func='envi_out_check', $
    uvalue='outf', /auto)
    ; Automanage the widget
    result = auto_wid_mng(base)
    if (result.accept eq 0) then return
    check = (result.check eq 0)
    div_zero = result.div_zero
    tp_divz_doit, fid=fid, pos=pos, dims=dims, check=check, $
    out_name=result.outf.name, div_zero=div_zero, $
    in_memory=result.outf.in_memory
  endif
end
```

## Compile the Tile Processing Routine

ENVI can automatically compile tp\_divz2.pro if you place it in the save\_add directory of the ENVI installation and restart ENVI. For this exercise, however, you will learn how to compile the routine within ENVI.

1. From the ENVI main menu bar, select **File > Compile IDL Module**. The Enter Module Filename dialog appears.
2. Navigate to `Data\programming` and select `tp_divz2.pro`. Click **Open**.
3. To make sure the function properly compiled, look for a line that says "Compiled module TP\_DIVZ2." On a Windows platform, this line appears in the IDLDE Command Log. On a Unix platform, the line appears in the shell window from which you started ENVI.

## Run the Tile Processing Routine

1. From the ENVI main menu bar, select **User Functions > User Band Ratio2**. The Ratio Input File selection dialog appears.
2. Select `bldr_tm.img`.
3. Click **Spectral Subset**. The File Spectral Subset dialog appears.
4. Select **Band 2**, hold down the **Ctrl** key, and select **Band 3**. Click **OK**.
5. In the Ratio Input File selection dialog, click **OK**. The Ratio Exercise Parameters dialog appears.
6. Select the **Yes** radio button for **Check for divide by 0 ?**
7. Enter a **Divide by zero value** of **1.0**.
8. Enter an output filename of `tp_divz2.img` and click **OK**. The resulting image is added to the Available Bands List.
9. Evaluate the new image. It should be the same as the Band Math example that checked for divide-by-zero errors.
10. When you are finished, select **File > Exit** from the ENVI main menu bar.