



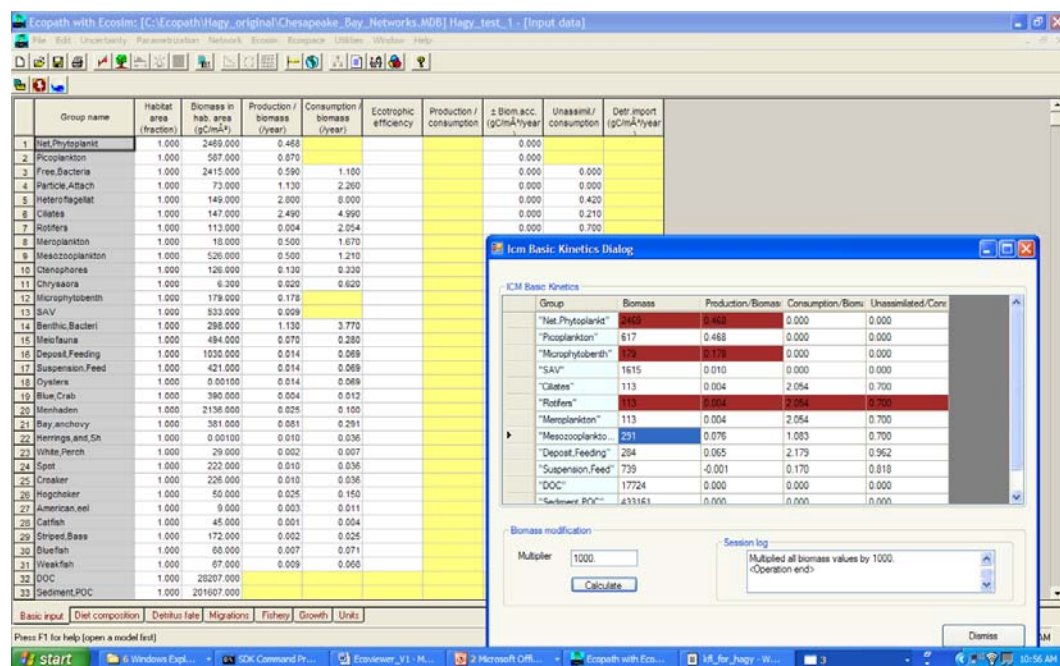
**US Army Corps
of Engineers®**
Engineer Research and
Development Center

System-Wide Water Resources Program

User's Guide to Linking the CE-QUAL-ICM and Ecopath Models

Carl F. Cerco, Dorothy H. Tillman, and Terry K. Gerald

August 2009



User's Guide to Linking the CE-QUAL-ICM and Ecopath Models

Carl F. Cerco, Dorothy H. Tillman, and Terry K. Gerald

*Environmental Laboratory
U.S. Army Engineer Research and Development Center
3909 Halls Ferry Road
Vicksburg MS 39180-6199*

Final report

Approved for public release; distribution is unlimited.

Prepared for U.S. Army Corps of Engineers
Washington, DC 20314-1000

Under System-Wide Water Resources Program

Abstract: The present report is one of a series that documents research relating the coupling of spatially and temporally detailed eutrophication models with ecosystem models that lack spatial and temporal resolution. Specifically, the Corps of Engineers Integrated Compartment Water Quality Model (CE-QUAL-ICM) is coupled to the Ecopath with Ecosim (EWE) fisheries model. Previous reports in this series introduced the concepts necessary for communication between the two models and detailed the linkage. The previous linkage relied on a “human interface” between the two models. That is, information from CE-QUAL-ICM was printed and entered into the EWE input screen by hand. This process has been replaced by a graphical user interface (GUI), which is documented herein.

DISCLAIMER: The contents of this report are not to be used for advertising, publication, or promotional purposes. Citation of trade names does not constitute an official endorsement or approval of the use of such commercial products. All product names and trademarks cited are the property of their respective owners. The findings of this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

DESTROY THIS REPORT WHEN NO LONGER NEEDED. DO NOT RETURN IT TO THE ORIGINATOR.

Contents

Figures and Tables	iv
Preface	v
1 Introduction.....	1
Linkage schematic	1
Code versions	2
2 Linking the Ecopath Model of ICM.....	3
Background	3
Step-by-step instructions	3
Transferring basic kinetics	4
Transferring diet composition	6
Transferring detrital fate	8
Important reminder	9
3 Linking the Ecopath Model of Chesapeake Bay.....	10
Introduction	10
Step-by-step instructions	10
Transferring basic kinetics	11
Transferring diet composition	13
Transferring detrital fate	14
References.....	16
Appendix A: The .ecm File for Ecopath Model of ICM	17
Appendix B: The .ecm File for Ecopath Model of Chesapeake Bay	18
Appendix C: The KFL Postprocessor.....	19
Appendix D: The forEcopathGui.f90 File	42
Appendix E: Module File for 4000-Cell KFL Postprocessor	63
Report Documentation Page	

Figures and Tables

Figures

Figure 1. Flow chart for exchanging information between ICM and Ecopath.....	2
Figure 2. Linux shell to compile the KFL postprocessor.....	4
Figure 3. Opening the .eco and .eii files in the IcmEcoViewer GUI.....	4
Figure 4. Selecting “Basic Kinetics” information to be transferred from ICM to Ecopath.....	5
Figure 5. Ecopath after importing “Basic Kinetics” information from the “ICM Basic Kinetics Dialog.”	6
Figure 6. Selecting “Diet Composition” information to be transferred from ICM to Ecopath.	7
Figure 7. Ecopath after importing “Diet Composition” information from the “ICM Diet Composition Dialog.”	7
Figure 8. Selecting “Detritus Fate” information to be transferred from ICM to Ecopath.....	8
Figure 9. Ecopath after importing “Detritus Fate” information from the “ICM Detritus Fate Dialog.”	9
Figure 10. The “ICM Basic Kinetics Dialog” including a biomass conversion.	12
Figure 11. Ecopath model of Chesapeake Bay after importing “Basic Kinetics” information from the “ICM Basic Kinetics Dialog.”	12
Figure 12. Selecting “Diet Composition” information to be transferred from ICM to the Ecopath model of Chesapeake Bay.....	13
Figure 13. Ecopath model of Chesapeake Bay after importing “Diet Composition” information from the “ICM Diet Composition Dialog.”	14
Figure 14. “Detritus Fate” information to be transferred from ICM to Ecopath model of Chesapeake Bay.....	15
Figure 15. Ecopath model of Chesapeake Bay after importing “Detritus Fate” information from the “ICM Detritus Fate Dialog.”	15

Preface

This work was conducted under funding from the System Wide Water Resources Program (SWWRP). Dr. Steven L. Ashby is Program Manager of SWWRP. The work was conducted under the direction supervision of Dr. Barry W. Bunch, Chief, Water Quality and Contaminant Modeling Branch, Environmental Laboratory (EL), U. S. Army Engineer Research and Development Center (ERDC), Vicksburg, MS.

This report was prepared by Carl F. Cerco, Dorothy H. Tillman, and Terry K. Gerald of the Water Quality and Contaminant Modeling Branch, EL, ERDC. At the time of publication of this report, Dr. Beth Fleming was Director of EL.

COL Gary E. Johnston was Commander and Executive Director of ERDC. Dr. James R. Houston was Director.

1 Introduction

The present report is one of a series that documents research relating the coupling of spatially and temporally detailed eutrophication models with ecosystem models that lack spatial and temporal resolution. Specifically, the Corps of Engineers Integrated Compartment Water Quality Model (CE-QUAL-ICM, Cerco and Meyers 2000) is coupled to the Ecopath with Ecosim (EWE) fisheries model (Christensen et al. 2000). Previous reports in this series introduced the concepts necessary for communication between the two models (Tillman et al. 2006) and detailed the linkage (Cerco and Tillman 2008). The previous linkage relied on a “human interface” between the two models. That is, information from CE-QUAL-ICM was printed and entered into the EWE input screen by hand. This process has been replaced by a graphical user interface (GUI), which is documented herein.

Linkage schematic

The linkage (Figure 1) relies on a number of computer executable, input, and output files. The **ICM Executable** is the compiled version of CE-QUAL-ICM as applied to the subject water body. **Ecopath** is the basic steady-state, spatially averaged foundation of EWE. The ICM Executable produces a KFL Output file. The **KFL Output** file is a binary file that contains the information to be passed to EWE. The KFL Output file is read by a **KFL Postprocessor**, which is assembled from several components including the main routines, routines which relate specific versions of ICM and EWE, and modules that define variables and array sizes. The **.ecm** file is an ASCII file, prepared by the user, which associates variable names between ICM and Ecopath. The **.eco file** is produced by the KFL post-processor and conveys information from ICM into the graphical user interface. The **.eii** file is an Ecopath file used for importing and exporting information. The .eco and .eii files are input to the **GUI**, which manages the exchange of information between the two models. The GUI creates a new .eii file, which contains selected information from ICM and is read back into Ecopath.

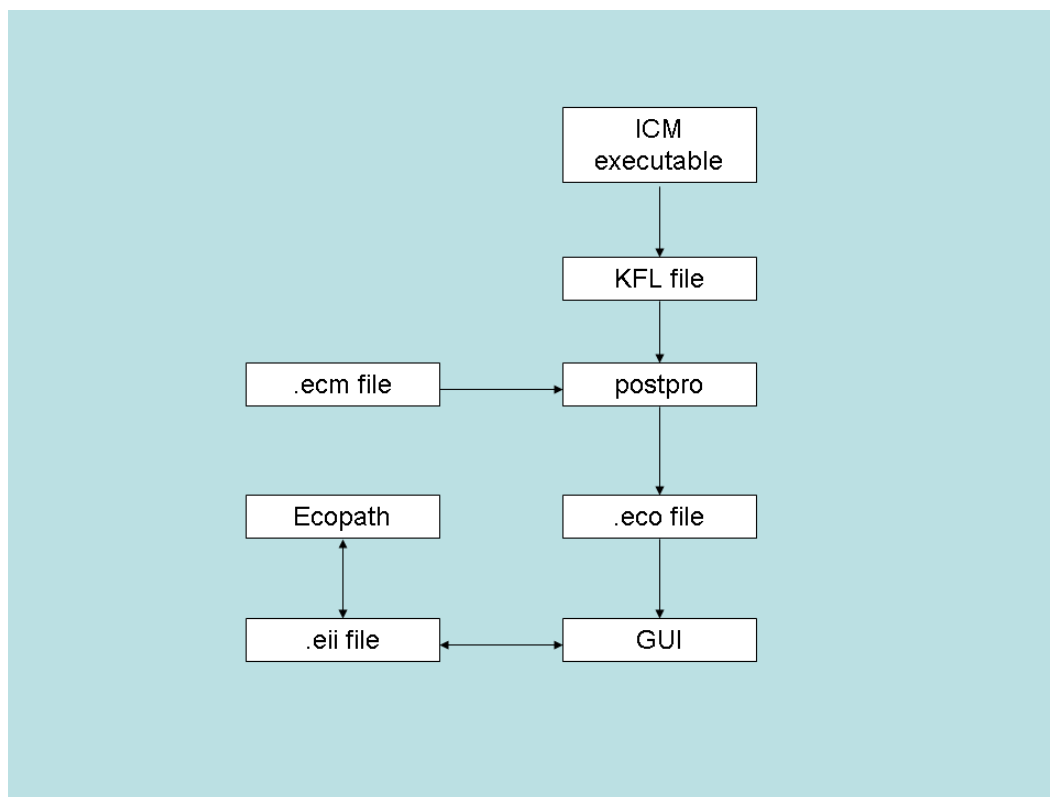


Figure 1. Flow chart for exchanging information between ICM and Ecopath.

Code versions

The version of ICM described here contains the features developed for the 2002 application to Chesapeake Bay (Cerco and Noel 2004). For development purposes, the code is applied on the 4,000-cell grid employed in the original Chesapeake Bay application (Cerco and Cole 1994). The KFL file is a binary file that must be compatible (format, array dimensions) with the KFL postprocessor, which is coded in FORTRAN 90. EWE is version 5.1 (Christensen et al. 2000), downloaded as a Windows PC executable from the Ecopath with Ecosim web site (www.ecopath.org). The Ecopath application to Chesapeake Bay was developed by Hagy (2002) and was provided by the author. The GUI is coded in C# and operates in the Windows PC environment.

2 Linking the Ecopath Model of ICM

Background

For development and debugging purposes, an Ecopath model of the ICM carbon cycle was created. This simplified Ecopath model also provides a good introduction to the GUI.

Step-by-step instructions

1. Execute the ICM model and create a KFL file.
2. Start EWE and export an .eii file based on the model of the ICM carbon cycle.
3. The names of the .ecm and .eco files are hardwired in the KFL post-processor. Edit file `kfl_cfcs_4000cell.f` and ensure the correct files are specified (`ecm_input_file = 'fort.gui_4000V2.ecm'`, `eco_output_file = 'fort.gui_4000V2.eco'`).
4. Compile the postprocessor (Figure 2).
5. The postprocessor opens file `'wqm_kfl.opt'`. Link the KFL output file to `wqm_kfl.opt`. (`ln -s wqm_kfl.sav_fix wqm_kfl.opt`).
6. Execute the postprocessor (`./postpro_4000`). The postprocessor uses two auxiliary input files. File `KFL_postpro_area.npt` lists the surface cells in the ICM grid that are to be averaged into a single Ecopath domain. File `sbox_col.dat` lists the cells that underlie the surface cells listed in `KFL_postpro_area.npt`. The postprocessor creates two output files. File `KFL_postpro_area_4000.opt` is an ASCII listing of postprocessed information. This material was previously entered into Ecopath by hand. File `fort.gui_4000V2.eco` is the information input directly to the GUI.
7. Postprocessing is conducted on the same machine on which ICM is executed. If this machine is not the PC on which Ecopath is operated, the .eco file should be transferred to the PC.
8. Start the GUI by double-clicking on the `IcmEcoViewer` icon. Go to the "File" heading and open the ICM file (Figure 3). Go to the "File" heading again and open the `EcoPath eii` file.

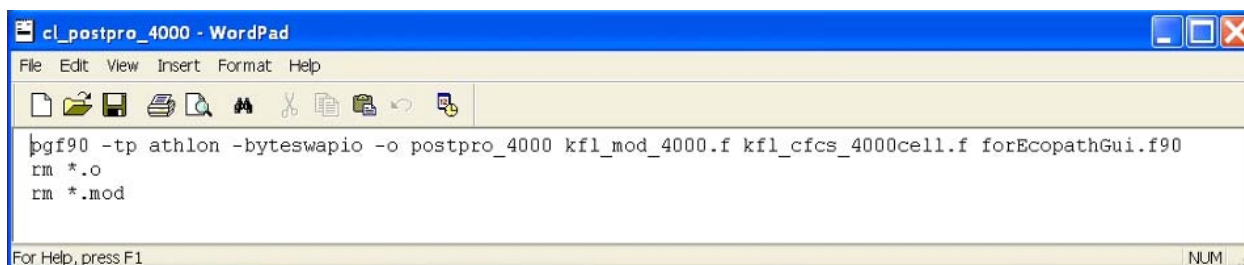


Figure 2. Linux shell to compile the KFL postprocessor.

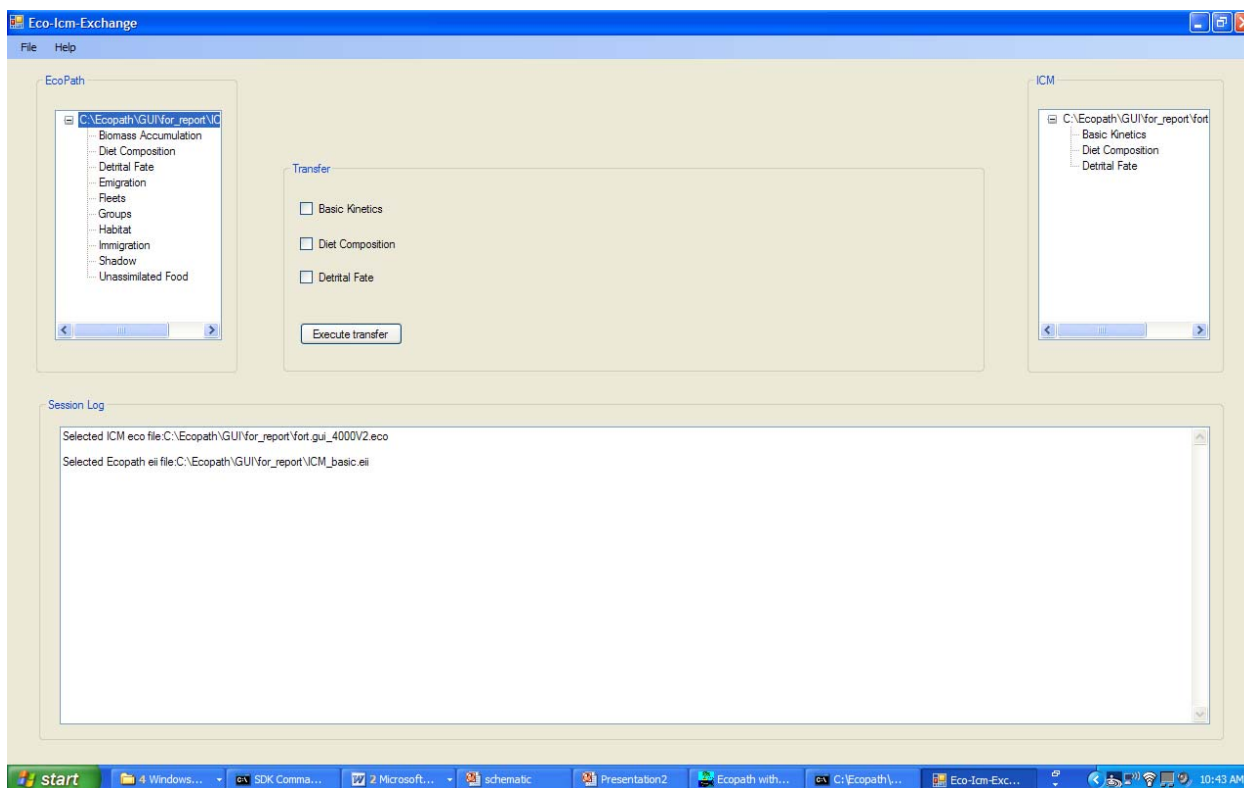


Figure 3. Opening the .eco and .eii files in the IcmEcoViewer GUI.

Transferring basic kinetics

Information may be transferred from ICM into three Ecopath screens, the Basic Kinetics screen, the Diet Composition screen, and the Detrital Fate screen. Proceed as follows to transfer information to the Ecopath Basic Kinetics screen.

1. Double-click on “Basic Kinetics” in the upper right-hand list. A screen entitled “ICM Basic Kinetics Dialog” will appear. (If not visible, look under the “Eco-Icm Exchange” window.) Double-click on the information to transfer to Ecopath. These entries will be highlighted in red (Figure 4).

2. Click in the “Basic Kinetics” box under the “Transfer” heading.
3. Click the “Execute Transfer” bar. A “Transfer Summary Report” will appear. This report can be dismissed.
4. Click the file header and then click the “Save EcoPath eii file” option. Provide a name (such as Test_1) and click the “Save” bar.
5. Activate Ecopath and click the “File” header. Click on the “Import Text (.eii)” option and navigate to the folder where the output from the GUI is stored.
6. Open the .eii file produced by the GUI. The items selected in Step 1 should appear in the Ecopath “Basic Kinetics” screen (Figure 5).

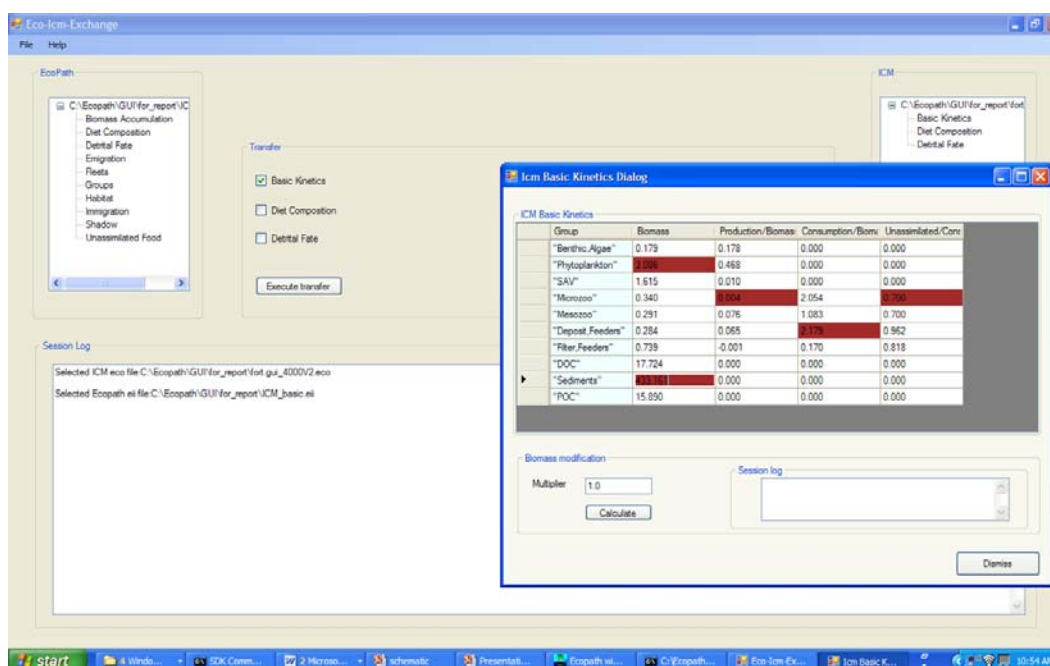


Figure 4. Selecting “Basic Kinetics” information to be transferred from ICM to Ecopath.

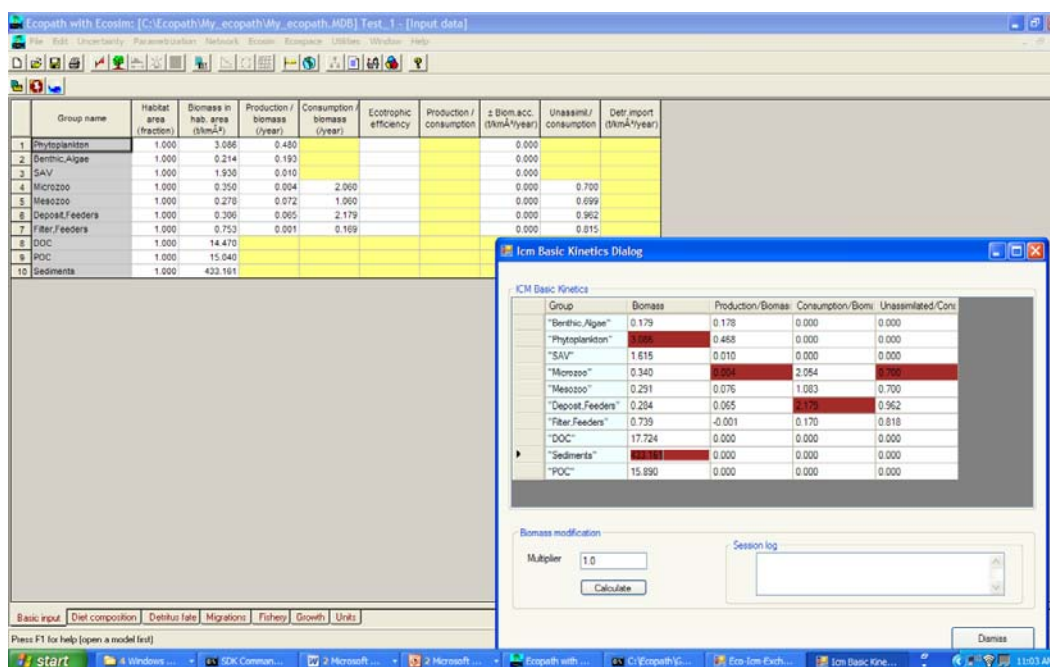


Figure 5. Ecopath after importing "Basic Kinetics" information from the "ICM Basic Kinetics Dialog."

Transferring diet composition

1. Double-click on "Diet Composition" in the upper right-hand list. A screen entitled "ICM Diet Composition Dialog" will appear. (If not visible, look under the "Eco-Icm Exchange" window.) Double-click on the information to transfer to Ecopath. These entries will be highlighted in red (Figure 6).
2. Click in the "Diet Composition" box under the "Transfer" heading.
3. Click the "Execute Transfer" bar. A "Transfer Summary Report" will appear. This report can be dismissed.
4. Click the file header and then click the "Save EcoPath eii file" option. Provide a name (such as Test_2) and click the "Save" bar.
5. Activate Ecopath and click the "File" header. Click on the "Import Text (.eii)" option and navigate to the folder where the output from the GUI is stored.
6. Open the .eii file produced by the GUI. The items selected in Step 1 should appear in the Ecopath "Diet Composition" screen (Figure 7). (Note that the screens are transposed between the GUI and Ecopath.)

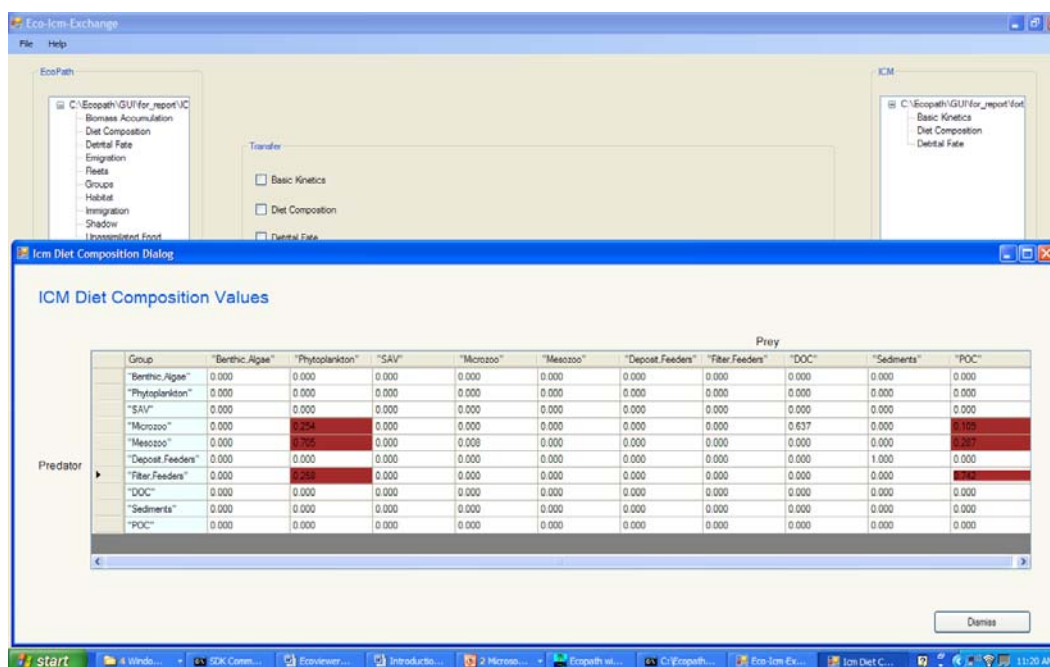


Figure 6. Selecting "Diet Composition" information to be transferred from ICM to Ecopath.

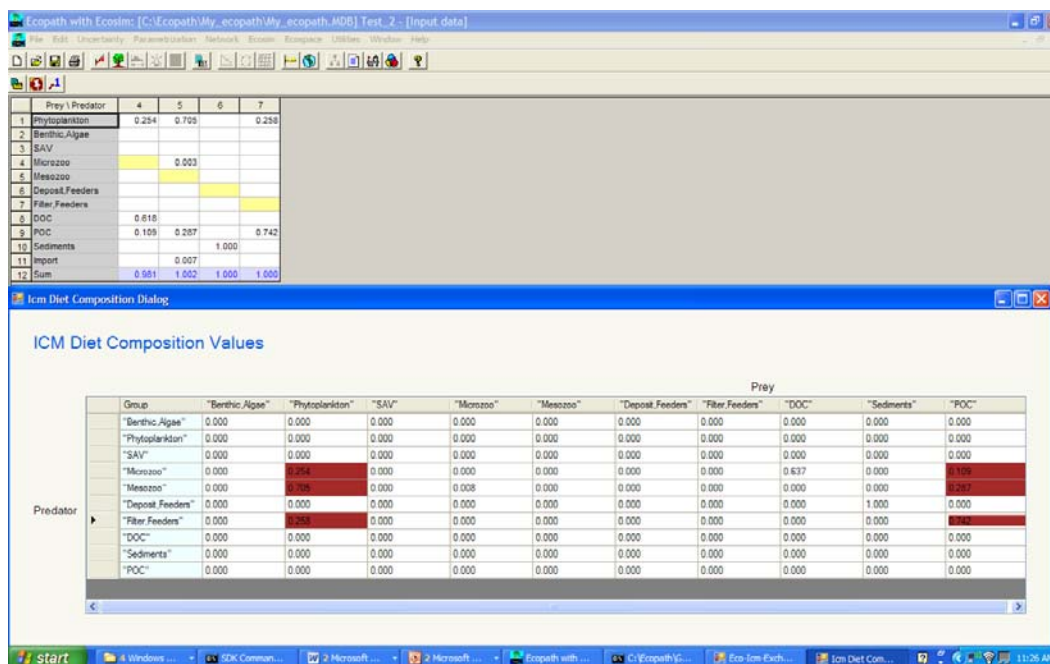


Figure 7. Ecopath after importing "Diet Composition" information from the "ICM Diet Composition Dialog."

Transferring detrital fate

1. Double-click on “Detrital Fate” in the upper right-hand list. A screen entitled “ICM Detrital Fate Dialog” will appear. (If not visible, look under the “Eco-Icm Exchange” window.) Double-click on the information to transfer to Ecopath. These entries will be highlighted in red (Figure 8).
2. Click in the “Detrital Fate” box under the “Transfer” heading.
3. Click the “Execute Transfer” bar. A “Transfer Summary Report” will appear. This report can be dismissed.
4. Click the file header and then click the “Save EcoPath eii file” option. Provide a name (such as Test_3) and click the “Save” bar.
5. Activate Ecopath and click the “File” header. Click on the “Import Text (.eii)” option and navigate to the folder where the output from the GUI is stored.
6. Open the .eii file produced by the GUI. The items selected in Step 1 should appear in the Ecopath “Detritus Fate” screen (Figure 9). (Note that the columns are ordered differently in the GUI and Ecopath.)

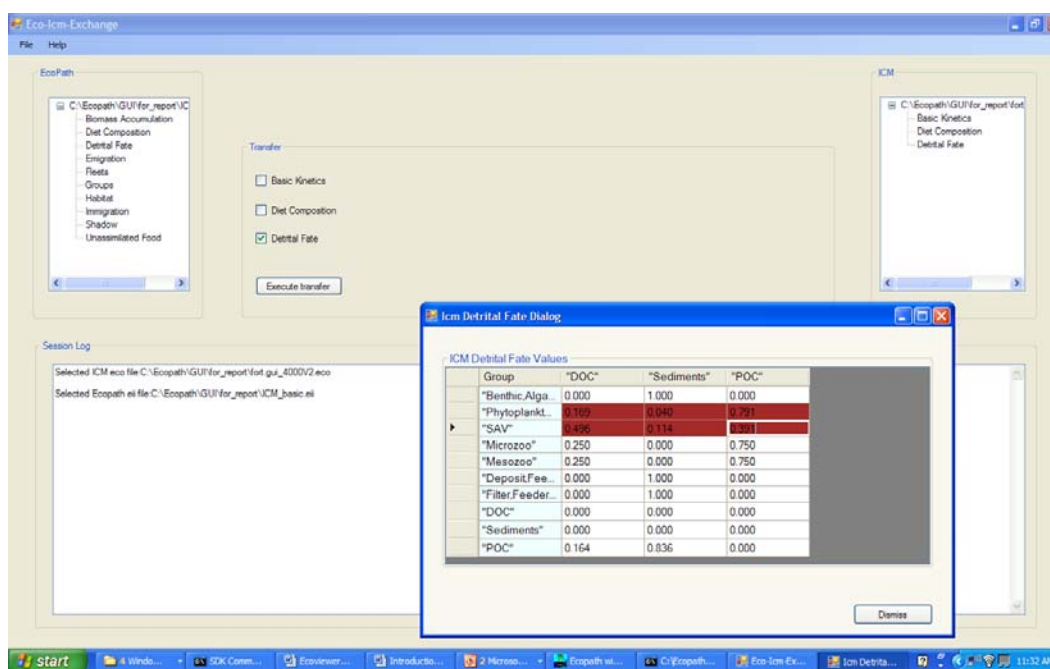


Figure 8. Selecting “Detritus Fate” information to be transferred from ICM to Ecopath.

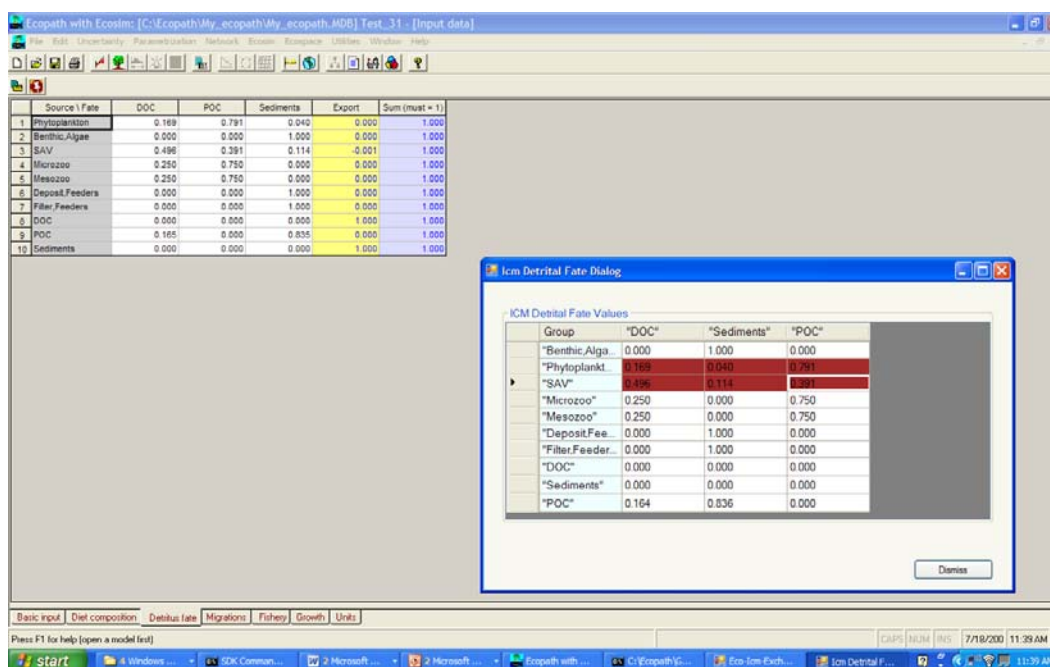


Figure 9. Ecopath after importing "Detritus Fate" information from the "ICM Detritus Fate Dialog."

Important reminder

Application of the Ecopath model largely amounts to balancing diet composition and other parameters such that a balance is attained and all "Ecotrophic Efficiencies" remain less than unity. If the Ecopath application is balanced prior to importing information from ICM, the application will likely have to be re-balanced after new information is imported. When an .eii file is saved, an "Important Reminder" message appears which cautions the user to check mass balances when returning to Ecopath. Clicking the "OK" bar will dismiss the reminder.

3 Linking the Ecopath Model of Chesapeake Bay

Introduction

For operational purposes, ICM is linked to an Ecopath model of Chesapeake Bay (Hagy 2002). Transferring information largely follows the procedure outlined in Chapter 2 with a few exceptions necessitated by differences in state variables and units between ICM and Ecopath. ICM represents the summer phytoplankton population in Chesapeake Bay as a single group. Ecopath employs two groups: Picoplankton and Net Phytoplankton. During the transfer process the ICM phytoplankton biomass is split 80 percent Net Phytoplankton and 20 percent Picoplankton. Production-to-biomass ratios and other parameters from the single ICM group are used for both Ecopath groups. ICM employs a single microzooplankton group, while Ecopath has three: Ciliates, Rotifers, and Merozooplankton. The biomass of the single ICM group is split equally into the three Ecopath groups. Production-to-biomass ratios and other parameters from the single ICM group are used for all three Ecopath groups. These splits are specified in the .ecm file (kfl_for_hagy.ecm). The ICM biomass unit is g C while Ecopath employs mg C. The units conversion is specified in the GUI as explained below.

Step-by-step instructions

1. Execute the ICM model and create a KFL file.
2. Start EWE and export an .eii file based on the model of Chesapeake Bay.
3. The names of the .ecm and .eco files are hardwired in the KFL postprocessor. Edit file kfl_cfcs_4000cell.f and ensure the correct files are specified (ecm_input_file = 'hagyV2.ecm', eco_output_file = 'hagyV2.eco')
4. Compile the postprocessor (Figure 2).
5. The postprocessor opens file 'wqm_kfl.opt'. Link the KFL output file to wqm_kfl.opt. (ln -s wqm_kfl.sav_fix wqm_kfl.opt).
6. Execute the postprocessor (./postpro_4000). The postprocessor uses two auxiliary input files. File KFL_postpro_area.npt lists the surface cells in the ICM grid that are to be averaged into a single Ecopath domain. File sbbox_col.dat lists the cells that underlie the surface cells

listed in KFL_postpro_area.npt. The postprocessor creates two output files. File KFL_postpro_area_4000.opt is an ASCII listing of post-processed information. This material was previously entered into Ecopath by hand. File hagyV2.eco is the information input directly to the GUI.

7. Postprocessing is conducted on the same machine on which ICM is executed. If this machine is not the PC on which Ecopath is operated, the .eco file should be transferred to the PC.
8. Start the GUI by double-clicking on the IcmEcoViewer icon. Go to the “File” heading and open the ICM file (Figure 3). Go to the “File” heading again and open the EcoPath eii file.

Transferring basic kinetics

Proceed as follows to transfer information to the Ecopath Basic Kinetics screen.

1. Double-click on “Basic Kinetics” in the upper right-hand list (Figure 4). A screen entitled “ICM Basic Kinetics Dialog” will appear. (If not visible, look under the “Eco-Icm Exchange” window.) Double-click on the information to transfer to Ecopath. These entries will be highlighted in red.
2. A box entitled “Biomass Multiplier” is situated in the lower left of the “ICM Basic Kinetics Dialog.” Enter 1000 to convert g C to mg C and click the “Calculate” button (Figure 10).
3. Click in the “Basic Kinetics” box under the “Transfer” heading.
4. Click the “Execute Transfer” bar. A “Transfer Summary Report” will appear. This report can be dismissed.
5. Click the file header and then click the “Save EcoPath eii file” option. Provide a name (such as Test_1) and click the “Save” bar.
6. Activate Ecopath and click the “File” header. Click on the “Import Text (.eii)” option and navigate to the folder where the output from the GUI is stored.
7. Open the .eii file produced by the GUI. The items selected in Step 1 should appear in the Ecopath “Basic Kinetics” screen (Figure 11).

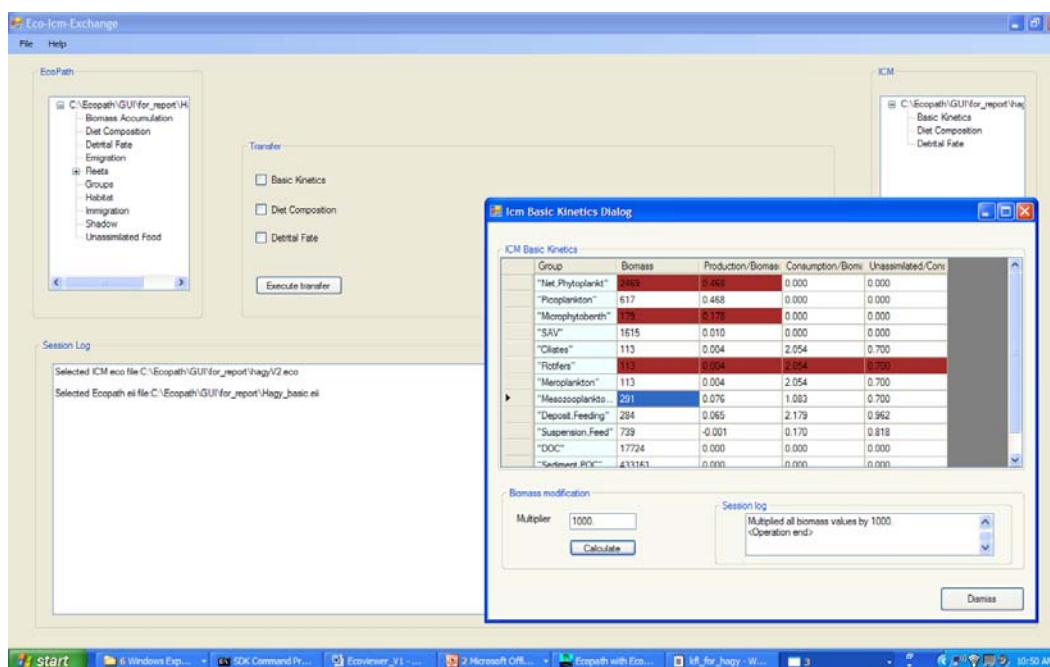


Figure 10. The "ICM Basic Kinetics Dialog" including a biomass conversion.

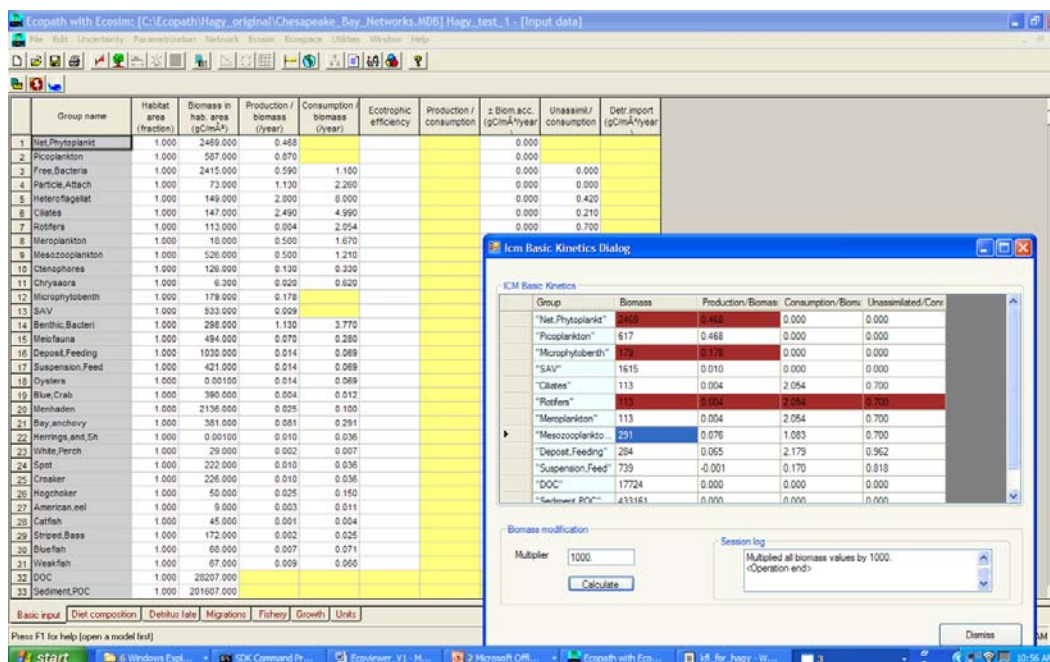


Figure 11. Ecopath model of Chesapeake Bay after importing "Basic Kinetics" information from the "ICM Basic Kinetics Dialog."

Transferring diet composition

1. Double-click on “Diet Composition” in the upper right-hand list. A screen entitled “ICM Diet Composition Dialog” will appear. (If not visible, look under the “Eco-Icm Exchange” window.) Double-click on the information to transfer to Ecopath. These entries will be highlighted in red (Figure 12).
2. Click in the “Diet Composition” box under the “Transfer” heading.
3. Click the “Execute Transfer” bar. A “Transfer Summary Report” will appear. This report can be dismissed.
4. Click the file header and then click the “Save EcoPath eii file” option. Provide a name (such as Test_2) and click the “Save” bar.
5. Activate Ecopath and click the “File” header. Click on the “Import Text (.eii)” option and navigate to the folder where the output from the GUI is stored.
6. Open the .eii file produced by the GUI. The items selected in Step 1 should appear in the Ecopath “Diet Composition” screen (Figure 13). (Note that the screens are transposed between the GUI and Ecopath.)

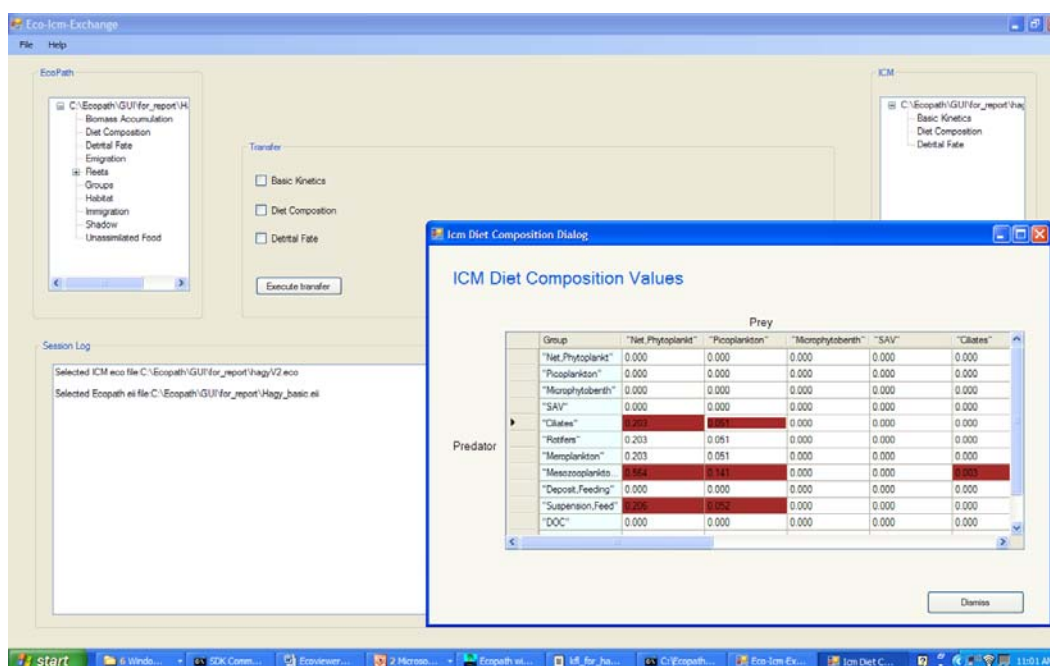


Figure 12. Selecting “Diet Composition” information to be transferred from ICM to the Ecopath model of Chesapeake Bay.

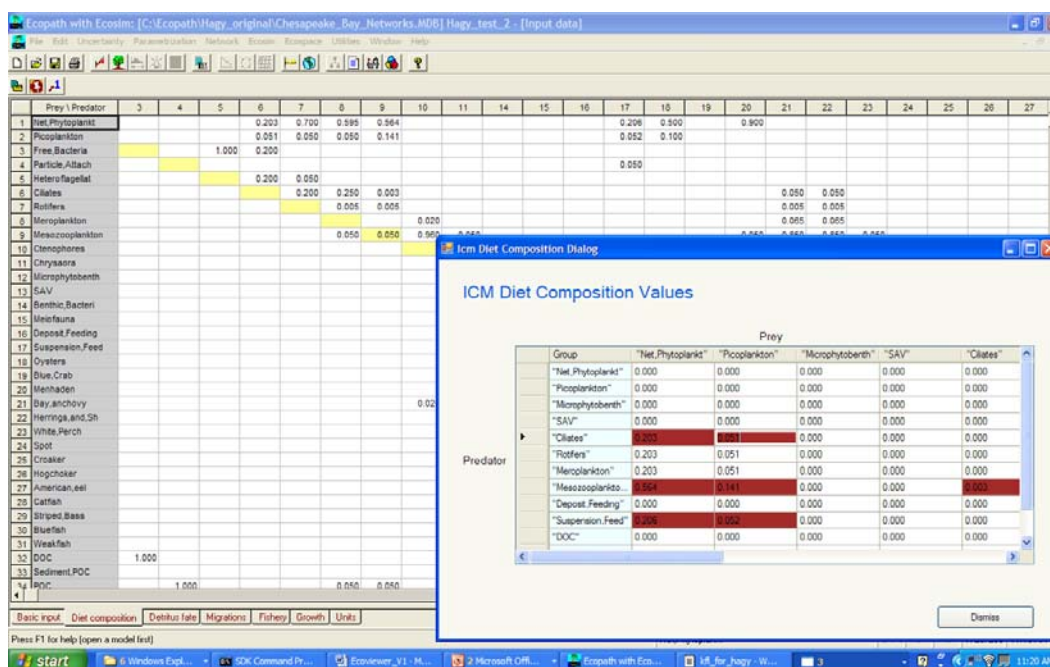


Figure 13. Ecopath model of Chesapeake Bay after importing “Diet Composition” information from the “ICM Diet Composition Dialog.”

Transferring detrital fate

1. Double-click on “Detrital Fate” in the upper right-hand list. A screen entitled “ICM Detrital Fate Dialog” will appear. (If not visible, look under the “Eco-Icm Exchange” window.) Double-click on the information to transfer to Ecopath. These entries will be highlighted in red (Figure 14).
2. Click in the “Detrital Fate” box under the “Transfer” heading.
3. Click the “Execute Transfer” bar. A “Transfer Summary Report” will appear. This report can be dismissed.
4. Click the file header and then click the “Save EcoPath eii file” option. Provide a name (such as Test_3) and click the “Save” bar.
5. Activate Ecopath and click the “File” header. Click on the “Import Text (.eii)” option and navigate to the folder where the output from the GUI is stored.
6. Open the .eii file produced by the GUI. The items selected in Step 1 should appear in the Ecopath “Detritus Fate” screen (Figure 15). (Note that the columns are ordered differently in the GUI and Ecopath.)

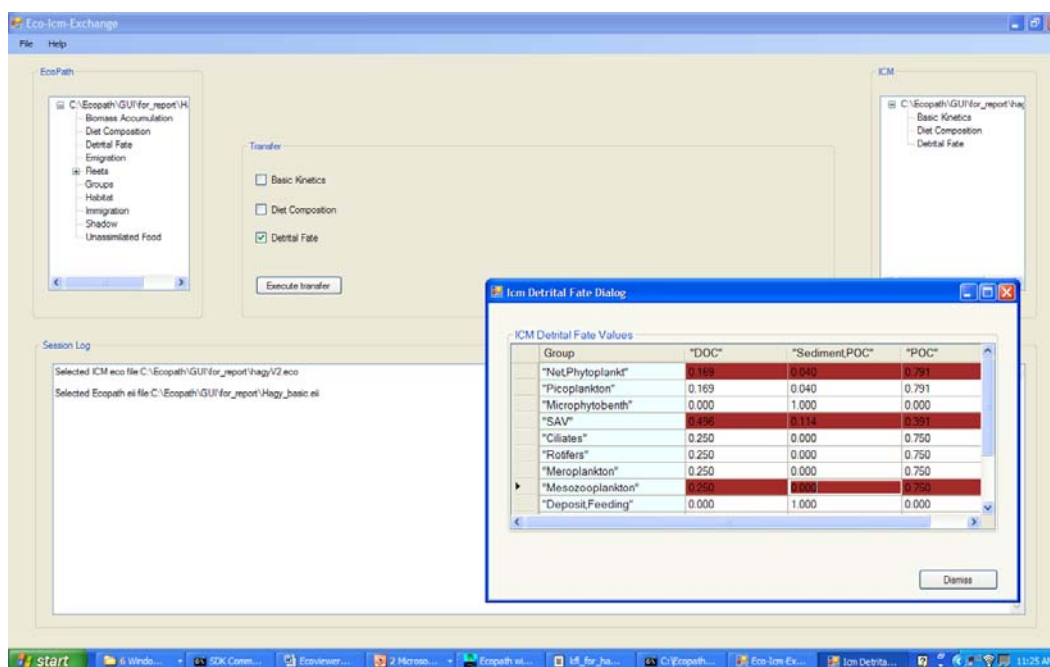


Figure 14. "Detritus Fate" information to be transferred from ICM to Ecopath model of Chesapeake Bay.

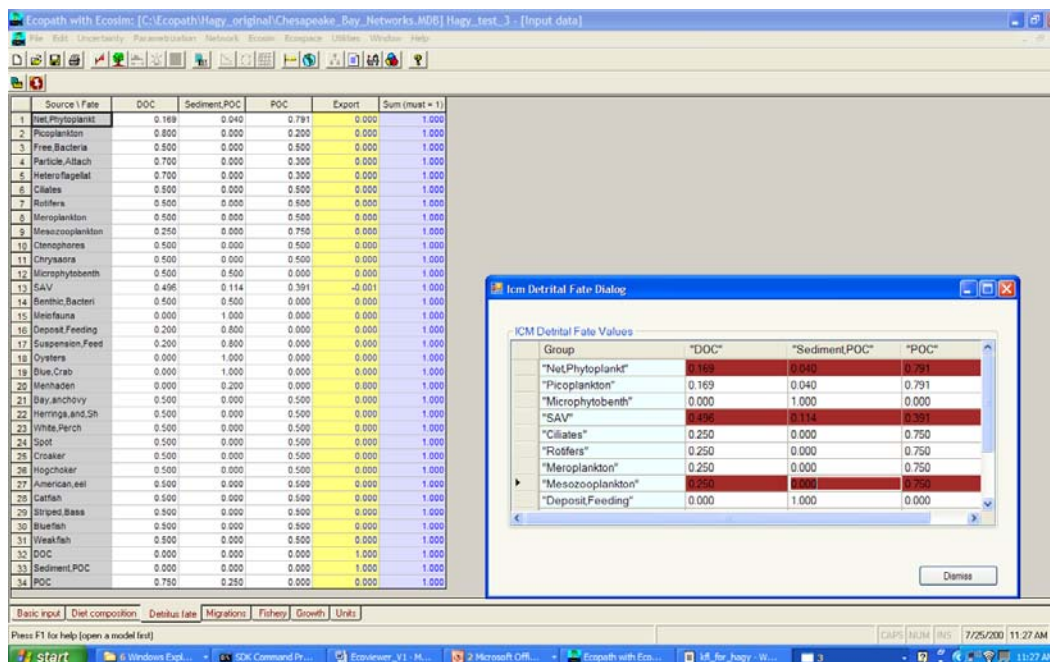


Figure 15. Ecopath model of Chesapeake Bay after importing "Detritus Fate" information from the "ICM Detritus Fate Dialog."

References

- Cerco, C., and T. Cole. 1994. *Three-dimensional eutrophication model of Chesapeake Bay*. Technical Report EL-94-4. Vicksburg, MS: U.S. Army Engineer Waterways Experiment Station.
- Cerco, C., and M. Meyers. 2000. Tributary refinements to the Chesapeake Bay model. *Journal of Environmental Engineering* 126(2): 164-174.
- Cerco, C., and M. Noel. 2004. *The 2002 Chesapeake Bay eutrophication model*. EPA 903-R-04-004. Annapolis, MD: Chesapeake Bay Program Office, U.S. Environmental Protection Agency.
- Cerco, C., and D. Tillman. 2008. *Use of coupled eutrophication and network models for examination of fisheries and eutrophication processes*. ERDC/EL TR-08-10. Vicksburg, MS: U.S. Army Engineer Research and Development Center.
- Christensen, V., C. Walters, and D. Pauly. 2000. *Ecopath with Ecosim: A user's guide*. Fisheries Centre, University of British Columbia.
- Hagy, J. 2002. *Eutrophication, hypoxia and trophic transfer efficiency in Chesapeake Bay*. PhD diss., University of Maryland Center for Environmental Science, Horn Point.
- Tillman, D., C. Cerco, and M. Noel. 2006. *Conceptual processes for linking eutrophication and network models*. TN-SWWRP-0905. Vicksburg MS: U.S. Army Engineer Research and Development Center.

Appendix A: The .ecm File for Ecopath Model of ICM

Cheasapeake Bay 4K grid, Cerco's Ecopath model of ICM

PRODUCER	COUNT	ECO_TYPE
	3	2

PRODUCER NAMES	ICM ALIAS	RATIO
"Benthic Algae"	"BALG"	1.000
"Phytoplankton"	"ALG"	1.000
"SAV"	"SAV"	1.000

CONSUMER	COUNT	ECO_TYPE
	4	3

CONSUMER NAMES	ICM ALIAS	RATIO
"Microzoo"	"Z1"	1.000
"Mesozoo"	"Z2"	1.000
"Deposit Feeders"	"DF"	1.000
"Filter Feeders"	"SF"	1.000

DETRITUS	COUNT	ECO_TYPE
	3	4

DETRITUS NAMES	ICM ALIAS	RATIO
"DOC"	"DOC"	1.000
"Sediments"	"SEDPOC"	1.000
"POC"	"POC"	1.000

Appendix B: The .ecm File for Ecopath Model of Chesapeake Bay

Chesapeake Bay 4K grid, Hagy's model with spelling error fixed.

PRODUCER	COUNT	ECO_TYPE
	4	2

PRODUCER NAMES	ICM ALIAS	RATIO
"Net, Phytoplankt"	"ALG"	0.8
"Picoplankton"	"ALG"	0.2
"Microphytobenth"	"BALG"	1.000
"SAV"	"SAV"	1.000

CONSUMER	COUNT	ECO_TYPE
	6	3

CONSUMER NAMES	ICM ALIAS	RATIO
"Ciliates"	"Z1"	0.333
"Rotifers"	"Z1"	0.333
"Meroplankton"	"Z1"	0.333
"Mesozooplankton"	"Z2"	1.000
"Deposit, Feeding"	"DF"	1.000
"Suspension, Feed"	"SF"	1.000

DETRITUS	COUNT	ECO_TYPE
	3	4

DETRITUS NAMES	ICM ALIAS	RATIO
"DOC"	"DOC"	1.000
"Sediment POC"	"SEDPOC"	1.000
"POC"	"POC"	1.000

Appendix C: The KFL Postprocessor

```
c A rudimentary KFL processor for checking purposes.
c Revised Feb 14, 2006 to go with new ecopath postprocessor
c Revised Jul 19, 2007 to go calculate Diet Compositions and De-
  tritus Fate
c Revised Jan 18, 2008 to include specification of ecm, eco files
```

```
***** Parameter declarations
```

```
!<TKG Moved to module>
```

```
use kfl_mod
use data_mod
```

```
implicit none
```

```
integer i, j, k, idum, jdum, jj
```

```
integer ncell, nread, JREG_DAY
```

```
real  acount
real  REG_AREA    ! Regional area
```

```
real  BAExport, BATotal
real  Export
```

```
character(len=132) :: ecm_input_file
character(len=132) :: eco_output_file
```

```
! TKG: Specify the ecopath input & output files
```

```
! CFC's 8 component model
ecm_input_file = 'fort.gui_4000V2.ecm'
eco_output_file = 'fort.gui_4000V2.eco'
```

```
! Hagy's 13 component model
!ecm_input_file = 'hagyV2.ecm'
!eco_output_file = 'hagyV2.eco'
```

```
C      OPEN(21, FILE='wqm_kfl.10YR_SENS153_new_grid_SEDFIX',
C      .      STATUS='UNKNOWN', FORM='UNFORMATTED')
```

```
      OPEN(21, FILE='wqm_kfl.opt',
      .      STATUS='UNKNOWN', FORM='UNFORMATTED')
```

```

OPEN(22, FILE='KFL_postpro_area.npt', STATUS='UNKNOWN')

OPEN(23, FILE='KFL_postpro_area_4000.opt', STATUS='UNKNOWN')

OPEN(24, FILE='sbox_col.dat', STATUS='OLD')

C READ FILE THAT MAPS SURFACE BOXES TO REST OF COLUMN
DO I=1, 729
C   DO I=1, 3162
      READ(24, *) NBOXCOL(I), (BOX(I, J), J=1, NBOXCOL(I))
C      READ(24, *, END=50) idum, jdum, NBOXCOL(I), (BOX(I, J),
C      .          J=1, NBOXCOL(I))
C      PRINT*, idum, jdum, NBOXCOL(I), (BOX(I, J),
C      .          J=1, NBOXCOL(I))
END DO

50  Continue
C ZERO OUT AVERAGE REGIONAL SUMS

      AREG_JDAY=0.0
      AREG_ALGC=0.0
      AREG_ANPP=0.0
      AREG_AGPP=0.0
      AREG_APRED=0.0
      AREG_ADOC=0.0
      AREG_APOC=0.0

      AREG_DOC=0.0
      AREG_POC=0.0
      AREG_DETC=0.0
      AREG_CRESP =0.0
      AREG_POC2DOC =0.0

      AREG_MICRZ =0.0
      AREG_MICRZR =0.0
      AREG_MICRZNP =0.0
      AREG_MICRZDOC =0.0
      AREG_MICRZPOC =0.0
      AREG_MICRZPR =0.0
      AREG_MICRZALG =0.0
      AREG_TCONSZ =0.0
      AREG_UADOC SZ =0.0
      AREG_UAPOC SZ =0.0

      AREG_MES0Z =0.0
      AREG_MES0ZR =0.0
      AREG_MES0ZNP =0.0
      AREG_MES0ZPOC =0.0
      AREG_MES0ZPR =0.0
      AREG_MES0ZALG =0.0
      AREG_MIC2MES =0.0
      AREG_TCONLZ =0.0
      AREG_UADOC LZ =0.0
      AREG_UAPOC LZ =0.0

      AREG_BURIAL = 0.0
      AREG_CFLUX = 0.0

```

```

    AREG_SEDR = 0.0
    AREG_ALG2SED = 0.0
    AREG_BALG = 0.0
    AREG_BALGR = 0.0
    AREG_BALGPR = 0.0
    AREG_BALGC = 0.0
    AREG_BNPP = 0.0

    AREG_SAV = 0.0
    AREG_SAVNP = 0.0
    AREG_SAVR = 0.0
    AREG_SAV2SED = 0.0
    AREG_SAV2POC = 0.0
    AREG_SAV2DOC = 0.0

    AREG_SFEED = 0.0
    AREG_SFNP = 0.0
    AREG_SFR = 0.0
    AREG_SFTCON = 0.0
    AREG_SFACON = 0.0
    AREG_SFPCCON = 0.0
    AREG_SFUAC = 0.0
    AREG_DFEED = 0.0
    AREG_DFNP = 0.0
    AREG_DFR = 0.0
    AREG_DFTCON = 0.0
    AREG_DFUAC = 0.0
    AREG_SEDPOC = 0.0

    ACOUNT = 0.
1    READ (KFL) (TITLE(I), I=1, 6), NB, NSB, (SBN(B), B=1, NSB),
      . (BBN(B), B=1, NSB),
      . (V1(B), B=0, NB), (SFA(B), B=1, NSB), SAV_CALC, BALGAE_CALC
      Write(*, *) (TITLE(I), I=1, 6)

    READ(22, *, END=3) NCELL
    READ(22, *) (REG_CELL(I), I=1, NCELL)
    Write(*, *) (REG_CELL(I), I=1, NCELL)

C GET REGIONAL AREA

    REG_AREA = 0.0
    DO I=1, NCELL
        CELL = REG_CELL(I)
        REG_AREA = REG_AREA + SFA(CELL)
    END DO

C ZERO OUT REGIONAL SUMS

    DO I=1, 10000
        REG_JDAY(I) = 0.0
        REG_ALGC(I) = 0.0
        REG_ANPP(I) = 0.0
        REG_AGPP(I) = 0.0
        REG_APRED(I) = 0.0
        REG_ADOC(I) = 0.0
        REG_APOC(I) = 0.0

```

REG_DOC(I) =0.0
REG_POC(I) =0.0
REG_DETC(I) =0.0
REG_CRESP(I) =0.0
REG_POC2DOC(I) =0.0

REG_MICRZ(I) =0.0
REG_MICRZR(I) =0.0
REG_MICRZNP(I) =0.0
REG_MICRZDOC(I) =0.0
REG_MICRZPOC(I) =0.0
REG_MICRZPR(I) =0.0
REG_MICRZALG(I) =0.0
REG_TCONSZ(I) =0.0
REG_UADOC SZ(I) =0.0
REG_UAPOC SZ(I) =0.0

REG_MES0Z(I) =0.0
REG_MES0ZR(I) =0.0
REG_MES0ZNP(I) =0.0
REG_MES0ZPOC(I) =0.0
REG_MES0ZPR(I) =0.0
REG_MES0ZALG(I) =0.0
REG_MIC2MES(I) =0.0
REG_TCONLZ(I) =0.0
REG_UADOCLZ(I) =0.0
REG_UAPOCLZ(I) =0.0

REG_BURIAL(I) = 0.0
REG_CFLUX(I) = 0.0
REG_SEDR(I) = 0.0
REG_ALG2SED(I) = 0.0
REG_BALG(I) = 0.0
REG_BALGR(I) = 0.0
REG_BALGPR(I) = 0.0
REG_BALGC(I) = 0.0
REG_BNPP(I) = 0.0

REG_SAV(I) = 0.0
REG_SAVNP(I) = 0.0
REG_SAVR(I) = 0.0
REG_SAV2SED(I) = 0.0
REG_SAV2POC(I) = 0.0
REG_SAV2DOC(I) = 0.0

REG_SFEED(I) = 0.0
REG_SFNP(I) = 0.0
REG_SFR(I) = 0.0
REG_SFTCON(I) = 0.0
REG_SFACON(I) = 0.0
REG_SFPCCON(I) = 0.0
REG_SFUAC(I) = 0.0
REG_DFEEED(I) = 0.0
REG_DFNP(I) = 0.0
REG_DFR(I) = 0.0
REG_DFTCON(I) = 0.0

```

      REG_DFUAC(I) = 0.0
      REG_SEDPOC(I) = 0.0

      END DO

      NREAD=0
      DO I=1,10000
        READ(KFL,END=2) JDAY
        NREAD = NREAD+1
        REG_JDAY(I) = JDAY
        READ(KFL) (E_ALGC(B),B=1,NB), (E_ANPP(B),B=1,NB),
          .      (E_AGPP(B),B=1,NB), (E_APRED(B),B=1,NB),
          .      (E_ADOC(B),B=1,NB), (E_APOC(B),B=1,NB)
        READ(KFL) (E_DOC(B),B=1,NB), (E_POC(B),B=1,NB),
          .      (E_DETC(B),B=1,NB), (E_CRESP(B),B=1,NB),
          .      (E_POC2DOC(B),B=1,NB)
        READ(KFL) (E_MICRZ(B),B=1,NB), (E_MICRZR(B),B=1,NB),
          .      (E_MICRZNP(B),B=1,NB),
          .
        (E_MICRZDOC(B),B=1,NB), (E_MICRZPOC(B),B=1,NB),
          .      (E_MICRZPR(B),B=1,NB),
        (E_MICRZALG(B),B=1,NB),
          .      (E_TCONSZ(B),B=1,NB),
        (E_UADOC SZ(B),B=1,NB),
          .      (E_UAPOC SZ(B),B=1,NB)
        READ(KFL) (E_MES0Z(B),B=1,NB), (E_MES0ZR(B),B=1,NB),
          .
        (E_MES0ZNP(B),B=1,NB), (E_MES0ZPOC(B),B=1,NB),
          .
        (E_MES0ZALG(B),B=1,NB), (E_MIC2MES(B),B=1,NB),
          .      (E_MES0ZPR(B),B=1,NB), (E_TCONLZ(B),B=1,NB),
          .      (E_UAD0CLZ(B),B=1,NB), (E_UAPOCLZ(B),B=1,NB)

        READ(KFL) (E_SEDPOC(B),B=1,NSB), (E_BURIAL(B),B=1,NSB),
          .      (E_CFLUX(B),B=1,NSB),
        (E_ALG2SED(B),B=1,NSB),
          .      (E_SEDR(B),B=1,NSB)
        READ(KFL) (E_BALG(B),B=1,NSB), (E_BNPP(B),B=1,NSB),
          .      (E_BALGR(B),B=1,NSB),
          .      (E_BALGPR(B),B=1,NSB), (E_BALGC(B),B=1,NSB)
        READ(KFL) (E_SAV(B),B=1,NSB), (E_SAVNP(B),B=1,NSB),
          .
        (E_SAV2SED(B),B=1,NSB), (E_SAV2POC(B),B=1,NSB),
          .      (E_SAV2DOC(B),B=1,NSB), (E_SAVR(B),B=1,NSB)
        READ(KFL) (E_SFEED(B),B=1,NSB), (E_SFNP(B),B=1,NSB),
          .      (E_SFTCON(B),B=1,NSB), (E_SFACON(B),B=1,NSB),
          .      (E_SFPCCON(B),B=1,NSB), (E_SFUAC(B),B=1,NSB),
          .      (E_SFR(B),B=1,NSB)
        READ(KFL) (E_DFEED(B),B=1,NSB), (E_DFNP(B),B=1,NSB),
          .      (E_DFTCON(B),B=1,NSB), (E_DFUAC(B),B=1,NSB),
          .      (E_DFR(B),B=1,NSB)

      C SUM THESE OVER ALL COLUMNS IN THE REGION

      DO JJ=1,NCELL

      C ZERO OUT COLUMN SUMS

```

```
COL_ALGC =0.0
COL_ANPP =0.0
COL_AGPP =0.0
COL_APRED=0.0
COL_ADOC =0.0
COL_APOC =0.0

COL_DOC =0.0
COL_POC =0.0
COL_DETC =0.0
COL_CRESP =0.0
COL_POC2DOC =0.0

COL_MICRZ =0.0
COL_MICRZR =0.0
COL_MICRZNP =0.0
COL_MICRZDOC =0.0
COL_MICRZPOC =0.0
COL_MICRZPR =0.0
COL_MICRZALG =0.0
COL_TCONSZ =0.0
COL_UADOC SZ =0.0
COL_UAPOC SZ =0.0

COL_MES0Z =0.0
COL_MES0ZR =0.0
COL_MES0ZNP =0.0
COL_MES0ZPOC =0.0
COL_MES0ZPR =0.0
COL_MES0ZALG =0.0
COL_MIC2MES =0.0
COL_TCONLZ =0.0
COL_UAD0CLZ =0.0
COL_UAPOCLZ =0.0

CELL = REG_CELL(JJ)

DO J=1,NBOXCOL(CELL)
  K=BOX(CELL,J)
  COL_ALGC=COL_ALGC+E_ALGC(K)
  COL_ANPP=COL_ANPP+E_ANPP(K)
  COL_AGPP=COL_AGPP+E_AGPP(K)
  COL_APRED=COL_APRED+E_APRED(K)
  COL_ADOC=COL_ADOC+E_ADOC(K)
  COL_APOC=COL_APOC+E_APOC(K)

  COL_DOC=COL_DOC+E_DOC(K)
  COL_POC=COL_POC+E_POC(K)
  COL_DETC=COL_DETC+E_DETC(K)
  COL_CRESP=COL_CRESP+E_CRESP(K)
  COL_POC2DOC=COL_POC2DOC+E_POC2DOC(K)

  COL_MICRZ=COL_MICRZ+E_MICRZ(K)
  COL_MICRZR=COL_MICRZR+E_MICRZR(K)
  COL_MICRZNP=COL_MICRZNP+E_MICRZNP(K)
  COL_MICRZDOC=COL_MICRZDOC+E_MICRZDOC(K)
  COL_MICRZPOC=COL_MICRZPOC+E_MICRZPOC(K)
```

```

COL_MICRZPR=COL_MICRZPR+E_MICRZPR(K)
COL_MICRZALG=COL_MICRZALG+E_MICRZALG(K)
COL_TCONSZ=COL_TCONSZ+E_TCONSZ(K)
COL_UADOC SZ=COL_UADOC SZ+E_UADOC SZ(K)
COL_UAPOC SZ=COL_UAPOC SZ+E_UAPOC SZ(K)

COL_MES0Z=COL_MES0Z+E_MES0Z(K)
COL_MES0ZR=COL_MES0ZR+E_MES0ZR(K)
COL_MES0ZNP=COL_MES0ZNP+E_MES0ZNP(K)
COL_MES0ZPOC=COL_MES0ZPOC+E_MES0ZPOC(K)
COL_MES0ZPR=COL_MES0ZPR+E_MES0ZPR(K)
COL_MES0ZALG=COL_MES0ZALG+E_MES0ZALG(K)
COL_MIC2MES=COL_MIC2MES+E_MIC2MES(K)
COL_TCONLZ=COL_TCONLZ+E_TCONLZ(K)
COL_UADOC LZ=COL_UADOC LZ+E_UADOC LZ(K)
COL_UAPOC LZ=COL_UAPOC LZ+E_UAPOC LZ(K)
END DO

```

C SAVE THE VARIABLES THAT ONLY EXIST AT THE BOTTOM

```

COL_SEDPOC = E_SEDPOC(CELL)
COL_BURIAL = E_BURIAL(CELL)
COL_CFLUX = E_CFLUX(CELL)
COL_ALG2SED = E_ALG2SED(CELL)
COL_SEDR = E_SEDR(CELL)

```

```

COL_BALG = E_BALG(CELL)
COL_BALGR = E_BALGR(CELL)
COL_BALGPR = E_BALGPR(CELL)
COL_BALGC = E_BALGC(CELL)
COL_BNPP = E_BNPP(CELL)

```

```

COL_SAV = E_SAV(CELL)
COL_SAVNP = E_SAVNP(CELL)
COL_SAVR = E_SAVR(CELL)
COL_SAV2SED = E_SAV2SED(CELL)
COL_SAV2POC = E_SAV2POC(CELL)
COL_SAV2DOC = E_SAV2DOC(CELL)

```

```

COL_SFEEED = E_SFEEED(CELL)
COL_SFNP = E_SFNP(CELL)
COL_SFR = E_SFR(CELL)
COL_SFTCON = E_SFTCON(CELL)
COL_SFACON = E_SFACON(CELL)
COL_SFPCCON = E_SFPCCON(CELL)
COL_SFUAC = E_SFUAC(CELL)

```

```

COL_DFEEED = E_DFEEED(CELL)
COL_DFNP = E_DFNP(CELL)
COL_DFR = E_DFR(CELL)
COL_DFTCON = E_DFTCON(CELL)
COL_DFUAC = E_DFUAC(CELL)

```

C SUM CELLS OVER REGION

```

REG_ALGC(I)=REG_ALGC(I)+COL_ALGC*SFA(CELL)
REG_ANPP(I)=REG_ANPP(I)+COL_ANPP*SFA(CELL)

```

REG_AGPP(I)=REG_AGPP(I)+COL_AGPP*SFA(CELL)
REG_APRED(I)=REG_APRED(I)+COL_APRED*SFA(CELL)
REG_ADOC(I)=REG_ADOC(I)+COL_ADOC*SFA(CELL)
REG_APOC(I)=REG_APOC(I)+COL_APOC*SFA(CELL)

REG_DOC(I)=REG_DOC(I)+COL_DOC*SFA(CELL)
REG_POC(I)=REG_POC(I)+COL_POC*SFA(CELL)
REG_DETC(I)=REG_DETC(I)+COL_DETC*SFA(CELL)
REG_CRESP(I)=REG_CRESP(I)+COL_CRESP*SFA(CELL)
REG_POC2DOC(I)=REG_POC2DOC(I)+COL_POC2DOC*SFA(CELL)

REG_MICRZ(I)=REG_MICRZ(I)+COL_MICRZ*SFA(CELL)
REG_MICZR(I)=REG_MICZR(I)+COL_MICZR*SFA(CELL)
REG_MICRZNP(I)=REG_MICRZNP(I)+COL_MICRZNP*SFA(CELL)
REG_MICRZDOC(I)=REG_MICRZDOC(I)+COL_MICRZDOC*SFA(CELL)
REG_MICRZPOC(I)=REG_MICRZPOC(I)+COL_MICRZPOC*SFA(CELL)
REG_MICRZPR(I)=REG_MICRZPR(I)+COL_MICRZPR*SFA(CELL)
REG_MICRZALG(I)=REG_MICRZALG(I)+COL_MICRZALG*SFA(CELL)
REG_TCONSZ(I)=REG_TCONSZ(I)+COL_TCONSZ*SFA(CELL)
REG_UADOC SZ(I)=REG_UADOC SZ(I)+COL_UADOC SZ*SFA(CELL)
REG_UAPOC SZ(I)=REG_UAPOC SZ(I)+COL_UAPOC SZ*SFA(CELL)

REG_MES0Z(I)=REG_MES0Z(I)+COL_MES0Z*SFA(CELL)
REG_MES0ZR(I)=REG_MES0ZR(I)+COL_MES0ZR*SFA(CELL)
REG_MES0ZNP(I)=REG_MES0ZNP(I)+COL_MES0ZNP*SFA(CELL)
REG_MES0ZPOC(I)=REG_MES0ZPOC(I)+COL_MES0ZPOC*SFA(CELL)
REG_MES0ZPR(I)=REG_MES0ZPR(I)+COL_MES0ZPR*SFA(CELL)
REG_MES0ZALG(I)=REG_MES0ZALG(I)+COL_MES0ZALG*SFA(CELL)
REG_MIC2MES(I)=REG_MIC2MES(I)+COL_MIC2MES*SFA(CELL)
REG_TCONLZ(I)=REG_TCONLZ(I)+COL_TCONLZ*SFA(CELL)
REG_UADOC LZ(I)=REG_UADOC LZ(I)+COL_UADOC LZ*SFA(CELL)
REG_UAPOC LZ(I)=REG_UAPOC LZ(I)+COL_UAPOC LZ*SFA(CELL)

REG_SEDPOC(I) = REG_SEDPOC(I)+COL_SEDPOC*SFA(CELL)
REG_SEDR(I) = REG_SEDR(I)+COL_SEDR*SFA(CELL)
REG_BURIAL(I) = REG_BURIAL(I)+COL_BURIAL*SFA(CELL)
REG_CFLUX(I) = REG_CFLUX(I)+COL_CFLUX*SFA(CELL)
REG_ALG2SED(I) = REG_ALG2SED(I)+COL_ALG2SED*SFA(CELL)

REG_BALG(I) = REG_BALG(I)+COL_BALG*SFA(CELL)
REG_BALGR(I) = REG_BALGR(I)+COL_BALGR*SFA(CELL)
REG_BALGPR(I) = REG_BALGPR(I)+COL_BALGPR*SFA(CELL)
REG_BALGC(I) = REG_BALGC(I)+COL_BALGC*SFA(CELL)
REG_BNPP(I) = REG_BNPP(I)+COL_BNPP*SFA(CELL)

REG_SAV(I) = REG_SAV(I)+COL_SAV*SFA(CELL)
REG_SAVNP(I) = REG_SAVNP(I)+COL_SAVNP*SFA(CELL)
REG_SAVR(I) = REG_SAVR(I)+COL_SAVR*SFA(CELL)
REG_SAV2SED(I) = REG_SAV2SED(I)+COL_SAV2SED*SFA(CELL)
REG_SAV2POC(I) = REG_SAV2POC(I)+COL_SAV2POC*SFA(CELL)
REG_SAV2DOC(I) = REG_SAV2DOC(I)+COL_SAV2DOC*SFA(CELL)

REG_SFEED(I) = REG_SFEED(I)+COL_SFEED*SFA(CELL)
REG_SFNP(I) = REG_SFNP(I)+COL_SFNP*SFA(CELL)
REG_SFR(I) = REG_SFR(I)+COL_SFR*SFA(CELL)
REG_SFTCON(I) = REG_SFTCON(I)+COL_SFTCON*SFA(CELL)
REG_SFACON(I) = REG_SFACON(I)+COL_SFACON*SFA(CELL)

```
REG_SFPCCON(I) = REG_SFPCCON(I)+COL_SFPCCON*SFA(CELL)
REG_SFUAC(I) = REG_SFUAC(I)+COL_SFUAC*SFA(CELL)
```

```
REG_DFEED(I) = REG_DFEED(I)+COL_DFEED*SFA(CELL)
REG_DFNP(I) = REG_DFNP(I)+COL_DFNP*SFA(CELL)
REG_DFR(I) = REG_DFR(I)+COL_DFR*SFA(CELL)
REG_DFTCON(I) = REG_DFTCON(I)+COL_DFTCON*SFA(CELL)
REG_DFUAC(I) = REG_DFUAC(I)+COL_DFUAC*SFA(CELL)
```

END DO

C DIVIDE REGIONAL SUMS BY SURFACE AREA

```
REG_ALGC(I)=REG_ALGC(I)/REG_AREA
REG_ANPP(I)=REG_ANPP(I)/REG_AREA
REG_AGPP(I)=REG_AGPP(I)/REG_AREA
REG_APRED(I)=REG_APRED(I)/REG_AREA
REG_ADOC(I)=REG_ADOC(I)/REG_AREA
REG_APOC(I)=REG_APOC(I)/REG_AREA
```

```
REG_DOC(I)=REG_DOC(I)/REG_AREA
REG_POC(I)=REG_POC(I)/REG_AREA
REG_DETC(I)=REG_DETC(I)/REG_AREA
REG_CRESP(I)=REG_CRESP(I)/REG_AREA
REG_POC2DOC(I)=REG_POC2DOC(I)/REG_AREA
```

```
REG_MICRZ(I)=REG_MICRZ(I)/REG_AREA
REG_MICRZR(I)=REG_MICRZR(I)/REG_AREA
REG_MICRZNP(I)=REG_MICRZNP(I)/REG_AREA
REG_MICRZDOC(I)=REG_MICRZDOC(I)/REG_AREA
REG_MICRZPOC(I)=REG_MICRZPOC(I)/REG_AREA
REG_MICRZPR(I)=REG_MICRZPR(I)/REG_AREA
REG_MICRZALG(I)=REG_MICRZALG(I)/REG_AREA
REG_TCONSZ(I)=REG_TCONSZ(I)/REG_AREA
REG_UADOC SZ(I)=REG_UADOC SZ(I)/REG_AREA
REG_UAPOC SZ(I)=REG_UAPOC SZ(I)/REG_AREA
```

```
REG_MESoz(I)=REG_MESoz(I)/REG_AREA
REG_MESozR(I)=REG_MESozR(I)/REG_AREA
REG_MESozNP(I)=REG_MESozNP(I)/REG_AREA
REG_MESozPOC(I)=REG_MESozPOC(I)/REG_AREA
REG_MESozPR(I)=REG_MESozPR(I)/REG_AREA
REG_MESozALG(I)=REG_MESozALG(I)/REG_AREA
REG_MIC2MES(I)=REG_MIC2MES(I)/REG_AREA
REG_TCONLZ(I)=REG_TCONLZ(I)/REG_AREA
REG_UADOC LZ(I)=REG_UADOC LZ(I)/REG_AREA
REG_UAPOC LZ(I)=REG_UAPOC LZ(I)/REG_AREA
```

```
REG_SEDPOC(I) = REG_SEDPOC(I)/REG_AREA
REG_BURIAL(I) = REG_BURIAL(I)/REG_AREA
REG_CFLUX(I) = REG_CFLUX(I)/REG_AREA
REG_SEDR(I) = REG_SEDR(I)/REG_AREA
REG_ALG2SED(I) = REG_ALG2SED(I)/REG_AREA
```

```
REG_BALG(I) = REG_BALG(I)/REG_AREA
REG_BALGR(I) = REG_BALGR(I)/REG_AREA
REG_BALGPR(I) = REG_BALGPR(I)/REG_AREA
```

```
REG_BALGC(I) = REG_BALGC(I)/REG_AREA
REG_BNPP(I) = REG_BNPP(I)/REG_AREA
```

```
REG_SAV(I) = REG_SAV(I)/REG_AREA
REG_SAVNP(I) = REG_SAVNP(I)/REG_AREA
REG_SAVR(I) = REG_SAVR(I)/REG_AREA
REG_SAV2SED(I) = REG_SAV2SED(I)/REG_AREA
REG_SAV2POC(I) = REG_SAV2POC(I)/REG_AREA
REG_SAV2DOC(I) = REG_SAV2DOC(I)/REG_AREA
```

```
REG_SFEED(I) = REG_SFEED(I)/REG_AREA
REG_SFNP(I) = REG_SFNP(I)/REG_AREA
REG_SFR(I) = REG_SFR(I)/REG_AREA
REG_SFTCON(I) = REG_SFTCON(I)/REG_AREA
REG_SFACON(I) = REG_SFACON(I)/REG_AREA
REG_SFPCCON(I) = REG_SFPCCON(I)/REG_AREA
REG_SFUAC(I) = REG_SFUAC(I)/REG_AREA
```

```
REG_DFEED(I) = REG_DFEED(I)/REG_AREA
REG_DFNP(I) = REG_DFNP(I)/REG_AREA
REG_DFR(I) = REG_DFR(I)/REG_AREA
REG_DFTCON(I) = REG_DFTCON(I)/REG_AREA
REG_DFUAC(I) = REG_DFUAC(I)/REG_AREA
```

C

C Find Average Regional values over seasons

C

```
JREG_DAY=REG_JDAY(I)
WRITE(*,*)'JDAY = ',JREG_DAY
IF(JREG_DAY .eq. 243. .or. JREG_DAY .eq. 608.
*   .or. JREG_DAY .eq. 973.)then
    ACOUNT =ACOUNT + 1.
    WRITE(*,*)'JDAY = ',REG_JDAY(I)
    AREG_ALGC=REG_ALGC(I)+AREG_ALGC
    AREG_ANPP=REG_ANPP(I)+AREG_ANPP
    AREG_AGPP=REG_AGPP(I)+AREG_AGPP
    AREG_APRED=REG_APRED(I)+AREG_APRED
    AREG_ADOC=REG_ADOC(I)+AREG_ADOC
    AREG_APOC=REG_APOC(I)+AREG_APOC
```

```
AREG_DOC=REG_DOC(I)+AREG_DOC
AREG_POC=REG_POC(I)+AREG_POC
AREG_DETC=REG_DETC(I)+AREG_DETC
AREG_CRESP=REG_CRESP(I)+AREG_CRESP
AREG_POC2DOC=REG_POC2DOC(I)+AREG_POC2DOC
```

```
AREG_MICRZ=REG_MICRZ(I)+AREG_MICRZ
AREG_MICRZR=REG_MICRZR(I)+AREG_MICRZR
AREG_MICRZNP=REG_MICRZNP(I)+AREG_MICRZNP
AREG_MICRZDOC=REG_MICRZDOC(I)+AREG_MICRZDOC
AREG_MICRZPOC=REG_MICRZPOC(I)+AREG_MICRZPOC
AREG_MICRZPR=REG_MICRZPR(I)+AREG_MICRZPR
AREG_MICRZALG=REG_MICRZALG(I)+AREG_MICRZALG
AREG_TCONSZ=REG_TCONSZ(I)+AREG_TCONSZ
AREG_UADOC SZ=REG_UADOC SZ(I)+AREG_UADOC SZ
AREG_UAPOC SZ=REG_UAPOC SZ(I)+AREG_UAPOC SZ
```

```
AREG_MES0Z=REG_MES0Z(I)+AREG_MES0Z
```

```

    AREG_MESOZR=REG_MESOZR(I)+AREG_MESOZR
    AREG_MESOZNP=REG_MESOZNP(I)+AREG_MESOZNP
    AREG_MESOZPOC=REG_MESOZPOC(I)+AREG_MESOZPOC
    AREG_MESOZPR=REG_MESOZPR(I)+AREG_MESOZPR
    AREG_MESOZALG=REG_MESOZALG(I)+AREG_MESOZALG
    AREG_MIC2MES=REG_MIC2MES(I)+AREG_MIC2MES
    AREG_TCONLZ=REG_TCONLZ(I)+AREG_TCONLZ
    AREG_UADOCLZ=REG_UADOCLZ(I)+AREG_UADOCLZ
    AREG_UAPOCLZ=REG_UAPOCLZ(I)+AREG_UAPOCLZ

    AREG_SEDPOC = REG_SEDPOC(I)+AREG_SEDPOC
    AREG_BURIAL = REG_BURIAL(I)+AREG_BURIAL
    AREG_CFLUX = REG_CFLUX(I)+AREG_CFLUX
    AREG_SEDR = REG_SEDR(I)+AREG_SEDR
    AREG_ALG2SED = REG_ALG2SED(I)+AREG_ALG2SED

    AREG_BALG = REG_BALG(I)+AREG_BALG
    AREG_BALGR = REG_BALGR(I)+AREG_BALGR
    AREG_BALGPR = REG_BALGPR(I)+AREG_BALGPR
    AREG_BALGC = REG_BALGC(I)+AREG_BALGC
    AREG_BNPP = REG_BNPP(I)+AREG_BNPP

    AREG_SAV = REG_SAV(I)+AREG_SAV
    AREG_SAVNP = REG_SAVNP(I)+AREG_SAVNP
    AREG_SAVR = REG_SAVR(I)+AREG_SAVR
    AREG_SAV2SED = REG_SAV2SED(I)+AREG_SAV2SED
    AREG_SAV2POC = REG_SAV2POC(I)+AREG_SAV2POC
    AREG_SAV2DOC = REG_SAV2DOC(I)+AREG_SAV2DOC

    AREG_SFEED = REG_SFEED(I)+AREG_SFEED
    AREG_SFNP = REG_SFNP(I)+AREG_SFNP
    AREG_SFR = REG_SFR(I)+AREG_SFR
    AREG_SFTCON = REG_SFTCON(I)+AREG_SFTCON
    AREG_SFACON = REG_SFACON(I)+AREG_SFACON
    AREG_SFPCCON = REG_SFPCCON(I)+AREG_SFPCCON
    AREG_SFUAC = REG_SFUAC(I)+AREG_SFUAC

    AREG_DFEED = REG_DFEED(I)+AREG_DFEED
    AREG_DFNP = REG_DFNP(I)+AREG_DFNP
    AREG_DFR = REG_DFR(I)+AREG_DFR
    AREG_DFTCON = REG_DFTCON(I)+AREG_DFTCON
    AREG_DFUAC = REG_DFUAC(I)+AREG_DFUAC
  ENDIF
  Write(*,*)' I = ',I
END DO

C WRITE THESE OUT
2   CONTINUE

C NOW TRY TO WRITE OUT THINGS THAT ECOPATH NEEDS
666  CONTINUE
    DO I=1,NREAD
      WRITE(23,40) REG_JDAY(I), REG_AREA
40    FORMAT('//' DAY ',F10.1,' AREA ',E14.6,' SQ M')

C SEDIMENTS

```

```

        WRITE(23,20)
20    FORMAT(/' SEDIMENTS', '      B      ', ' FR ALGAE', ' FR
DETR',
$ '      FR SAV ', ' DEP FEED ', ' SUS FEED ', ' BENTH ALG ',
$ '      BURIAL      RESP')
        WRITE(23,21) REG_SEDPOC(I), REG_ALG2SED(I), REG_CFLUX(I),
$ REG_SAV2SED(I),
$ REG_DFUAC(I), REG_SFUAC(I), REG_BALGC(I), REG_BURIAL(I),
$ REG_SEDR(I)
21    FORMAT(10X, 9F10.3)

C WATER COLUMN POC
        WRITE(23,22)
22    FORMAT(/'      POC      ', '      B      ', ' FROM ALG ', ' FR MIZOO
',
$ ' FR MEZOO ', ' FROM SAV ', ' TO MIZOO ', ' TO MEZOO ',
$ ' TO SEDS ', ' TO DOC ')
        WRITE(23,23) REG_POC(I), REG_APOC(I), REG_UAPOCSZ(I),
$
REG_UAPOCLZ(I), REG_SAV2POC(I), REG_MICRZPOC(I), REG_MESOZPOC(I),
$ REG_CFLUX(I), REG_POC2DOC(I)
23    FORMAT(10X, 9F10.3)

C WATER COLUMN DOC
        WRITE(23,24)
24    FORMAT(/'      DOC      ', '      B      ', ' FR ALGAE', ' FR MIZOO
',
$ ' FR MEZOO ', ' FR SAV ', ' TO MIZOO      CRESP')
        WRITE(23,25) REG_DOC(I), REG_ADOC(I), REG_UADOCsz(I),
$
REG_UADOCCLZ(I), REG_SAV2DOC(I), REG_MICRZDOC(I), REG_CRESP(I)
25    FORMAT(10X, 7F10.3)

C PHYTOPLANKTON
        WRITE(23,26)
26    FORMAT(/' ALGAE      ', '      B      ', '      NPP ', ' TO DOC
',
$ ' TO POC ', ' TO MIZOO ', ' TO MEZOO ', ' TO SEDS
RESP')
        WRITE(23,27)
REG_ALGC(I), REG_ANPP(I), REG_ADOC(I), REG_APOC(I),
$ REG_MICRZALG(I), REG_MESOZALG(I), REG_ALG2SED(I),
$ REG_AGPP(I) - REG_ANPP(I)
27    FORMAT(10X, 8F10.3)

C MICROZOOPLANKTON
        WRITE(23,28)
28    FORMAT(/' MICRO Z ', '      B      ', ' PROD ', ' TCON
',
$ ' UCON ', ' DOC IN ', ' POC IN ', ' ALGAE IN ',
$ ' DOC OUT ', ' POC OUT ', ' TO MEZOO      RESP')
        WRITE(23,29) REG_MICRZ(I), REG_MICRZNP(I), REG_TCONSZ(I),
$ REG_UADOCsz(I) + REG_UAPOCSZ(I), REG_MICRZDOC(I),
$ REG_MICRZPOC(I), REG_MICRZALG(I), REG_UADOCsz(I),
$ REG_UAPOCSZ(I), REG_MIC2MES(I), REG_MICRZR(I)
29    FORMAT(10X, 11F10.3)

```

```

C MESOZOOPLANKTON
  WRITE(23,30)
30  FORMAT(/' MESO Z ', '      B ', '      PROD ', '      TCON
',
$ '      UCON ', '      POC IN ', '      ALGAE IN ', '      MICRO IN ',
$ '      DOC OUT ', '      POC OUT      RESP')
  WRITE(23,31) REG_MES0Z(I), REG_MES0ZNP(I), REG_TCONLZ(I),
$ REG_UAD0CLZ(I)+REG_UAPOCLZ(I),
$ REG_MES0ZPOC(I), REG_MES0ZALG(I), REG_MIC2MES(I),
$ REG_UAD0CLZ(I), REG_UAPOCLZ(I), REG_MES0ZR(I)
31  FORMAT(10X,10F10.3)

```

```

C SAV
  WRITE(23,32)
32  FORMAT(/'      SAV ', '      B ', '      NPP ', '      TO DOC
',
$ '      TO POC ', '      TO SEDS      RESP')
  WRITE(23,33) REG_SAV(I), REG_SAVNP(I), REG_SAV2DOC(I),
$ REG_SAV2POC(I), REG_SAV2SED(I), REG_SAVR(I)
33  FORMAT(10X,6F10.3)

```

```

C BENTHIC ALGAE
  WRITE(23,34)
34  FORMAT(/'BENTHIC ALG', '      B ', '      NPP ', '      TO SEDS
',
$ '      RESP')
  WRITE(23,35)
REG_BALG(I), REG_BNPP(I), REG_BALGC(I), REG_BALGR(I)
35  FORMAT(10X,4F10.3)

```

```

C DEPOSIT FEEDERS
  WRITE(23,36)
36  FORMAT(/' DEP FEED ', '      B ', '      PROD ', '      TCON
',
$ '      UCON ', '      FROM SED ', '      TO SED ', '      RESP')
  WRITE(23,37) REG_DFEE(I), REG_DFNP(I), REG_DFTCON(I),
$ REG_DFUAC(I), REG_DFTCON(I), REG_DFUAC(I), REG_DFR(I)
37  FORMAT(10X,7F10.4)

```

```

C FILTER FEEDERS
  WRITE(23,38)
38  FORMAT(/' SUS FEED ', '      B ', '      PROD ', '      TCON
',
$ '      UCON ', '      FROM ALG ', '      FROM POC ', '      TO SED ',
$ '      RESP')
  WRITE(23,39) REG_SFEE(I), REG_SFNP(I), REG_SFTCON(I),
$ REG_SFUAC(I), REG_SFACON(I), REG_SFPCCON(I), REG_SFUAC(I),
$ REG_SFR(I)
39  FORMAT(10X,8F10.3)

```

```

C
C Production/Biomass ratio
C

```

```

PB_BALGRatio = REG_BNPP(I)/REG_BALG(I)
PB_ALGRatio = REG_ANPP(I)/REG_ALGC(I)
PB_Z1Ratio = REG_MICRZNP(I)/REG_MICRZ(I)

```

```

        PB_Z2Ratio = REG_MESOZNP(I)/REG_MESOZ(I)
        PB_SAVRatio = REG_SAVNP(I)/REG_SAV(I)
        PB_DFRatio = REG_DFNP(I)/REG_DFEED(I)
        PB_SFRatio = REG_SFNP(I)/REG_SFEED(I)

        WRITE(23,67)
67    FORMAT(/' P/B BALG ', ' P/B ALG ', ' P/B Z1 ', ' P/B Z2
',
        *      ' P/B SAV ', ' P/B DF ', ' P/B SF ')
        WRITE(23,68)
PB_BALGRatio,PB_ALGRatio,PB_Z1Ratio,PB_Z2Ratio,
        *      PB_SAVRatio,
        *      PB_DFRatio,PB_SFRatio
68    FORMAT(7F10.3)

```

```

C
C Consumption/Biomass
C

```

```

        QB_Z1Ratio = REG_TCONSZ(I)/REG_MICRZ(I)
        QB_Z2Ratio = REG_TCONLZ(I)/REG_MESOZ(I)
        QB_DFRatio = REG_DFTCON(I)/REG_DFEED(I)
        QB_SFRatio = REG_SFTCON(I)/REG_SFEED(I)

        WRITE(23,71)
71    FORMAT(/' Q/B Z1 ', ' Q/B Z2 ',
        *      ' Q/B DF ', ' Q/B SF ')
        WRITE(23,72) QB_Z1Ratio,QB_Z2Ratio,
        *      QB_DFRatio,QB_SFRatio
72    FORMAT(7F10.3)

```

```

C
C Uassimilated/Consumption
C

```

```

        UATC_Z1Ratio =
(REG_UADOC SZ(I)+REG_UAPOC SZ(I))/REG_TCONSZ(I)
        UATC_Z2Ratio =
(REG_UADOCLZ(I)+REG_UAPOCLZ(I))/REG_TCONLZ(I)
        UATC_DFRatio = REG_DFUAC(I)/REG_DFTCON(I)
        UATC_SFRatio = REG_SFUAC(I)/REG_SFTCON(I)

        WRITE(23,73)
73    FORMAT(/' UA/Q Z1 ', ' UA/Q Z2 ',
        *      ' UA/Q DF ', ' UA/Q SF ')
        WRITE(23,74) UATC_Z1Ratio,UATC_Z2Ratio,
        *      UATC_DFRatio,UATC_SFRatio
74    FORMAT(7F10.3)

```

```

C
C Diet Compostion
C

```

```

C Z1 Diet Compostion
        Z1DCDOC = REG_MICRZDOC(I)/(REG_MICRZDOC(I)+REG_MICRZPOC(I)+
        *      REG_MICRZALG(I))
        Z1DCPOC = REG_MICRZPOC(I)/(REG_MICRZDOC(I)+REG_MICRZPOC(I)+
        *      REG_MICRZALG(I))
        Z1DCALG = REG_MICRZALG(I)/(REG_MICRZDOC(I)+REG_MICRZPOC(I)+
        *      REG_MICRZALG(I))

```

```

        WRITE(23,69)
69    FORMAT(/' Z1 DC From DOC ', 'Z1 DC From POC ',
*        'Z1 DC From ALG ')
        WRITE(23,70) Z1DCDOC, Z1DCPOC, Z1DCALG
70    FORMAT(3(5X, F10.3))

C Z2 Diet Compostion
    Z2DCPOC = REG_MESOZPOC(I)/(REG_MIC2MES(I)+REG_MESOZPOC(I)+
*    REG_MESOZALG(I))
    Z2DCZ1 = REG_MIC2MES(I)/(REG_MIC2MES(I)+REG_MESOZPOC(I)+
*    REG_MESOZALG(I))
    Z2DCALG = REG_MESOZALG(I)/(REG_MIC2MES(I)+REG_MESOZPOC(I)+
*    REG_MESOZALG(I))
    WRITE(23,42)
42    FORMAT(/' Z2 DC From POC ', ' Z2 DC From Z1 ',
*    ' Z2 DC From ALG ')
    WRITE(23,70) Z2DCPOC, Z2DCZ1, Z2DCALG
41    FORMAT(10X, 3F10.3)

C Deposit Feeders (DF) Diet Compostion
    DFDCSedPOC = REG_DFUAC(I)/REG_DFUAC(I)
    WRITE(23,43)
43    FORMAT(/' DF DC From SedPOC ')
    WRITE(23,70) DFDCSedPOC
44    FORMAT(10X, 3F10.3)

C Filter Feeders (SF) Diet Compostion
    SFDCPOC = REG_SFPCCON(I)/(REG_SFPCCON(I)+REG_SFACON(I))
    SFDCALG = REG_SFACON(I)/(REG_SFPCCON(I)+REG_SFACON(I))
    WRITE(23,45)
45    FORMAT(/' SF DC From POC ', ' SF DC From ALG ')
    WRITE(23,70) SFDCPOC, SFDCALG
46    FORMAT(10X, 3F10.3)

C
C Detrital Fate
C

C Microphytobenthos Detrital Fate
    MICRBENALG_SedPOC = REG_BALGC(I)/REG_BALGC(I)
    MICRBENALG_POC = 0.
    MICRBENALG_DOC = 0.
    BAExport = 0.
    BATotal =
MICRBENALG_SedPOC+MICRBENALG_POC+MICRBENALG_DOC+Export
    WRITE(23,47)
47    FORMAT(/' BALG DF to DOC ', ' BALG DF to SedPOC ',
*    ' BALG DF to POC ', ' BA Export', ' Total Sum')
    WRITE(23,48)
MICRBENALG_DOC, MICRBENALG_SedPOC, MICRBENALG_POC,
*    BAExport, BATotal
48    FORMAT(f10.3, 9x, f10.3, 10x, f10.3, 3x, f10.3, 4x, f10.3)

C Phytoplankton Detrital Fate
    ALG_SedPOC = REG_ALG2SED(I)/(REG_ADOC(I)+REG_APOC(I)+
*    REG_ALG2SED(I))

```

```

      ALG_POC =
REG_APOC(I)/(REG_ADOC(I)+REG_APOC(I)+REG_ALG2SED(I))
      ALG_DOC =
REG_ADOC(I)/(REG_ADOC(I)+REG_APOC(I)+REG_ALG2SED(I))
      AlgExport = 0.
      AlgTotal = ALG_SedPOC+ALG_POC+ALG_DOC+AlgExport
      WRITE(23,49)
49  FORMAT(/'  ALG DF to DOC ','  ALG DF to SedPOC ',
*        '  ALG DF to POC ','  ALG  Export','  Total Sum')
      WRITE(23,48) ALG_DOC,ALG_SedPOC,ALG_POC,
*                AlgExport,AlgTotal

```

C Microzooplankton Detrital Fate

```

      Z1_SedPOC = 0.
      Z1_POC = REG_MICRZPOC(I)/(REG_MICRZDOC(I)+REG_MICRZPOC(I))
      Z1_DOC = REG_MICRZDOC(I)/(REG_MICRZDOC(I)+REG_MICRZPOC(I))
      Z1_POC = REG_UAPOCSZ(I)/(REG_UAPOCSZ(I)+REG_UADOC SZ(I))
      Z1_DOC = REG_UADOC SZ(I)/(REG_UAPOCSZ(I)+REG_UADOC SZ(I))
      Z1Export = 0.
      Z1Total = Z1_SedPOC +Z1_POC +Z1_DOC +Z1Export
      WRITE(23,51)
51  FORMAT(/'  Z1 DF to DOC ','  Z1 DF to SedPOC ',
*        '  Z1 DF to POC ','  Z1 Export ','  Total Sum')
      WRITE(23,48) Z1_DOC,Z1_SedPOC,Z1_POC,
*                Z1Export,Z1Total

```

C Mesozooplankton Detrital Fate

```

      Z2_SedPOC = 0.
      Z2_POC = REG_UAPOCLZ(I)/(REG_UAPOCLZ(I)+REG_UADOC LZ(I))
      Z2_DOC = REG_UADOC LZ(I)/(REG_UAPOCLZ(I)+REG_UADOC LZ(I))
      Z2Export = 0.
      Z2Total = Z2_SedPOC +Z2_POC +Z2_DOC +Z2Export
      WRITE(23,53)
53  FORMAT(/'  Z2 DF to DOC ','  Z2 DF to SedPOC ',
*        '  Z2 DF to POC ','  Z2 Export ','  Total Sum')
      WRITE(23,48) Z2_DOC,Z2_SedPOC,Z2_POC,
*                Z2Export,Z2Total

```

C SAV Detrital Fate

```

      SAV_SedPOC = REG_SAV2SED(I)/(REG_SAV2DOC(I)+REG_SAV2POC(I)+
*        REG_SAV2SED(I))
      SAV_POC = REG_SAV2POC(I)/(REG_SAV2DOC(I)+REG_SAV2POC(I)+
*        REG_SAV2SED(I))
      SAV_DOC = REG_SAV2DOC(I)/(REG_SAV2DOC(I)+REG_SAV2POC(I)+
*        REG_SAV2SED(I))
      SAVExport = 0.
      SAVTotal = SAV_SedPOC +SAV_POC +SAV_DOC +SAVExport
      WRITE(23,55)
55  FORMAT(/'  SAV DF to DOC ','  SAV DF to SedPOC ',
*        '  SAV DF to POC ','  SAV Export ','  Total Sum')
      WRITE(23,48) SAV_DOC,SAV_SedPOC,SAV_POC,
*                SAVExport,SAVTotal

```

C Deposit Feeders Detrital Fate

```

      DF_SedPOC = REG_DFUAC(I)/REG_DFUAC(I)
      DF_POC = 0.
      DF_DOC = 0.

```

```

        DFExport = 0.
        DFTotal = DF_SedPOC+DF_POC+DF_DOC+DFExport
        WRITE(23,57)
57    FORMAT(/' DF DF to DOC ', ' DF DF to SedPOC ',
*          ' DF DF to POC ', ' DF Export ', ' Total Sum')
        WRITE(23,48) DF_DOC, DF_SedPOC, DF_POC,
*          DFExport, DFTotal

C Suspension Feeders Detrital Fate
        SF_SedPOC = REG_SFUAC(I)/REG_SFUAC(I)
        SF_POC = 0.
        SF_DOC = 0.
        SFExport = 0.
        SFTotal = SF_SedPOC+SF_POC+SF_DOC+SFExport
        WRITE(23,59)
59    FORMAT(/' SF DF to DOC ', ' SF DF to SedPOC ',
*          ' SF DF to POC ', ' SF Export ', ' Total Sum')
        WRITE(23,48) SF_DOC, SF_SedPOC, SF_POC,
*          SFExport, SFTotal

C DOC Detrital Fate
        DOC_SedPOC = 0.
        DOC_POC = 0.
        DOC_DOC = 0.
        DOCExport = 1.
        DOCTotal = DOC_SedPOC +DOC_POC +DOC_DOC +DOCExport
        WRITE(23,61)
61    FORMAT(/' DOC DF to DOC ', ' DOC DF to SedPOC ',
*          ' DOC DF to POC ', ' DOC Export ', ' Total Sum')
        WRITE(23,48) DOC_DOC, DOC_SedPOC, DOC_POC,
*          DOCExport, DOCTotal

C Sed POC Detrital Fate
        SedPOC_SedPOC = 0.
        SedPOC_POC = 0.
        SedPOC_DOC = 0.
        SedPOCExport = 1.
        SedPOCTotal = Sed-
POC_SedPOC+SedPOC_POC+SedPOC_DOC+SedPOCExport
        WRITE(23,63)
63    FORMAT(/' SedPOC to DOC ', ' SedPOC to SedPOC ',
*          ' SedPOC to POC ', ' SedPOC Export', ' Total
Sum')
        WRITE(23,48) SedPOC_DOC, SedPOC_SedPOC, SedPOC_POC,
*          SedPOCExport, SedPOCTotal

C POC Detrital Fate
        POC_SedPOC = REG_CFLUX(I)/(REG_CFLUX(I)+REG_POC2DOC(I))
        POC_POC = 0.
        POC_DOC = REG_POC2DOC(I)/(REG_CFLUX(I)+REG_POC2DOC(I))
        POCExport = 0.
        POCTotal = SF_SedPOC+SF_POC+SF_DOC+SFExport
        WRITE(23,65)
65    FORMAT(/' POC DF to DOC ', ' POC DF to SedPOC ',
*          ' POC DF to POC ', ' POC Export ', ' Total Sum')
        WRITE(23,48) POC_DOC, POC_SedPOC, POC_POC,
*          POCExport, POCTotal

```

```
END DO
C
C Caculate Average Seasonal Values
C

  AREG_ALGC=AREG_ALGC/ACOUNT
  AREG_ANPP=AREG_ANPP/ACount
  AREG_AGPP=AREG_AGPP/ACOUNT
  AREG_APRED=AREG_APRED/ACOUNT
  AREG_ADOC=AREG_ADOC/ACOUNT
  AREG_APOC=AREG_APOC/ACOUNT

  AREG_DOC=AREG_DOC/ACOUNT
  AREG_POC=AREG_POC/ACOUNT
  AREG_DETC=AREG_DETC/ACOUNT
  AREG_CRESP=AREG_CRESP/ACOUNT
  AREG_POC2DOC=AREG_POC2DOC/ACOUNT

  AREG_MICRZ=AREG_MICRZ/ACOUNT
  AREG_MICRZR=AREG_MICRZR/ACOUNT
  AREG_MICRZNP=AREG_MICRZNP/ACOUNT
  AREG_MICRZDOC=AREG_MICRZDOC/ACOUNT
  AREG_MICRZPOC=AREG_MICRZPOC/ACOUNT
  AREG_MICRZPR=AREG_MICRZPR/ACOUNT
  AREG_MICRZALG=AREG_MICRZALG/ACOUNT
  AREG_TCONSZ=AREG_TCONSZ/ACOUNT
  AREG_UADOCsz=AREG_UADOCsz/ACOUNT
  AREG_UAPOCSZ=AREG_UAPOCSZ/ACOUNT

  AREG_MESoz=AREG_MESoz/ACOUNT
  AREG_MESozR=AREG_MESozR/ACOUNT
  AREG_MESozNP=AREG_MESozNP/ACOUNT
  AREG_MESozPOC=AREG_MESozPOC/ACOUNT
  AREG_MESozPR=AREG_MESozPR/ACOUNT
  AREG_MESozALG=AREG_MESozALG/ACOUNT
  AREG_MIC2MES=AREG_MIC2MES/ACOUNT
  AREG_TCONLZ=AREG_TCONLZ/ACOUNT
  AREG_UADOCCLZ=AREG_UADOCCLZ/ACOUNT
  AREG_UAPOCLZ=AREG_UAPOCLZ/ACOUNT

  AREG_SEDPOC = AREG_SEDPOC/ACOUNT
  AREG_BURIAL = AREG_BURIAL/ACOUNT
  AREG_CFLUX = AREG_CFLUX/ACOUNT
  AREG_SEDR = AREG_SEDR/ACOUNT
  AREG_ALG2SED = AREG_ALG2SED/ACOUNT

  AREG_BALG =AREG_BALG/ACOUNT
  AREG_BALGR =AREG_BALGR/ACOUNT
  AREG_BALGPR = AREG_BALGPR/ACOUNT
  AREG_BALGC =AREG_BALGC/ACOUNT
  AREG_BNPP = AREG_BNPP/ACOUNT

  AREG_SAV = AREG_SAV/ACOUNT
  AREG_SAVNP = AREG_SAVNP/ACOUNT
  AREG_SAVR = AREG_SAVR/ACOUNT
```

```

    AREG_SAV2SED = AREG_SAV2SED/ACOUNT
    AREG_SAV2POC = AREG_SAV2POC/ACOUNT
    AREG_SAV2DOC = AREG_SAV2DOC/ACOUNT

    AREG_SFED = AREG_SFED/ACOUNT
    AREG_SFNP = AREG_SFNP/ACOUNT
    AREG_SFR = AREG_SFR/ACOUNT
    AREG_SFTCON = AREG_SFTCON/ACOUNT
    AREG_SFACON = AREG_SFACON/ACOUNT
    AREG_SFPCCON = AREG_SFPCCON/ACOUNT
    AREG_SFUAC = AREG_SFUAC/ACOUNT

    AREG_DFEED = AREG_DFEED/ACOUNT
    AREG_DFNP = AREG_DFNP/ACOUNT
    AREG_DFR = AREG_DFR/ACOUNT
    AREG_DFTCON = AREG_DFTCON/ACOUNT
    AREG_DFUAC = AREG_DFUAC /ACOUNT
    WRITE(23,80)
80    FORMAT('// ' Average Seasonal Values ')

C SEDIMENTS
    WRITE(23,20)
    WRITE(23,21) AREG_SEDPOC, AREG_ALG2SED, AREG_CFLUX,
$    AREG_SAV2SED,
$    AREG_DFUAC, AREG_SFUAC, AREG_BALGC, AREG_BURIAL,
$    AREG_SEDR

C WATER COLUMN POC
    WRITE(23,22)
    WRITE(23,23) AREG_POC, AREG_APOC, AREG_UAPOCSZ,
$    AREG_UAPOCLZ, AREG_SAV2POC, AREG_MICRZPOC, AREG_MESOPZPOC,
$    AREG_CFLUX, AREG_POC2DOC

C WATER COLUMN DOC
    WRITE(23,24)
    WRITE(23,25) AREG_DOC, AREG_ADOC, AREG_UADOC SZ,
$    AREG_UADOC LZ, AREG_SAV2DOC, AREG_MICRZDOC, AREG_CRESP

C PHYTOPLANKTON
    WRITE(23,26)
    WRITE(23,27) AREG_ALGC, AREG_ANPP, AREG_ADOC, AREG_APOC,
$    AREG_MICRZALG, AREG_MESOZALG, AREG_ALG2SED,
$    AREG_AGPP - AREG_ANPP

C MICROZOOPLANKTON
    WRITE(23,28)
    WRITE(23,29) AREG_MICRZ, AREG_MICRZNP, AREG_TCONSZ,
$    AREG_UADOC SZ + AREG_UAPOC SZ, AREG_MICRZDOC,
$    AREG_MICRZPOC, AREG_MICRZALG, AREG_UADOC SZ,
$    AREG_UAPOC SZ, AREG_MIC2MES, AREG_MICRZR

C MESOZOOPLANKTON
    WRITE(23,30)
    WRITE(23,31) AREG_MESOPZ, AREG_MESOPZNP, AREG_TCONLZ,
$    AREG_UADOC LZ + AREG_UAPOC LZ,
$    AREG_MESOPZPOC, AREG_MESOZALG, AREG_MIC2MES,
$    AREG_UADOC LZ, AREG_UAPOC LZ, AREG_MESOPZR

```

```

C SAV
    WRITE(23,32)
    WRITE(23,33) AREG_SAV,AREG_SAVNP,AREG_SAV2DOC,
$   AREG_SAV2POC,AREG_SAV2SED,AREG_SAVR

C BENTHIC ALGAE
    WRITE(23,34)
    WRITE(23,35) AREG_BALG,AREG_BNPP,AREG_BALGC,AREG_BALGR

C DEPOSIT FEEDERS
    WRITE(23,36)
    WRITE(23,37) AREG_DFEED,AREG_DFNP,AREG_DFTCON,
$   AREG_DFUAC,AREG_DFTCON,AREG_DFUAC,AREG_DFR

C FILTER FEEDERS
    WRITE(23,38)
    WRITE(23,39) AREG_SFEED,AREG_SFNP,AREG_SFTCON,
$   AREG_SFUAC,AREG_SFACON,AREG_SFPCCON,AREG_SFUAC,
$   AREG_SFR

C
C Production/Biomass ratio
C

    APB_BALGRatio = AREG_BNPP/AREG_BALG
    APB_ALGRatio = AREG_ANPP/AREG_ALGC
    APB_Z1Ratio = AREG_MICRZNP/AREG_MICRZ
    APB_Z2Ratio = AREG_MES0ZNP/AREG_MES0Z
    APB_SAVRatio = AREG_SAVNP/AREG_SAV
    APB_DFRatio = AREG_DFNP/AREG_DFEED
    APB_SFRatio = AREG_SFNP/AREG_SFEED

    WRITE(23,67)
    WRITE(23,68)
APB_BALGRatio,APB_ALGRatio,APB_Z1Ratio,APB_Z2Ratio,
*
*   APB_SAVRatio,
*   APB_DFRatio,APB_SFRatio

C
C Consumption/Biomass
C

    AQB_Z1Ratio = AREG_TCONSZ/AREG_MICRZ
    AQB_Z2Ratio = AREG_TCONLZ/AREG_MES0Z
    AQB_DFRatio = AREG_DFTCON/AREG_DFEED
    AQB_SFRatio = AREG_SFTCON/AREG_SFEED

    WRITE(23,71)
    WRITE(23,72) AQB_Z1Ratio,AQB_Z2Ratio,
*
*   AQB_DFRatio,AQB_SFRatio

C
C Uassimilated/Consumption
C

    AUATC_Z1Ratio = (AREG_UAD0CSZ+AREG_UAPOCSZ)/AREG_TCONSZ
    AUATC_Z2Ratio = (AREG_UAD0CLZ+AREG_UAPOCLZ)/AREG_TCONLZ
    AUATC_DFRatio = AREG_DFUAC/AREG_DFTCON

```

```

    AUATC_SFRatio = AREG_SFUAC/AREG_SFTCON

    WRITE(23,73)
    WRITE(23,74) AUATC_Z1Ratio,AUATC_Z2Ratio,
    *           AUATC_DFRatio,AUATC_SFRatio
C
C Diet Compostion
C
C Z1 Diet Compostion
    AZ1DCDOC = AREG_MICRZDOC/(AREG_MICRZDOC+AREG_MICRZPOC+
    *           AREG_MICRZALG)
    AZ1DCPOC = AREG_MICRZPOC/(AREG_MICRZDOC+AREG_MICRZPOC+
    *           AREG_MICRZALG)
    AZ1DCALG = AREG_MICRZALG/(AREG_MICRZDOC+AREG_MICRZPOC+
    *           AREG_MICRZALG)
    WRITE(23,69)
    WRITE(23,70) AZ1DCDOC,AZ1DCPOC,AZ1DCALG

C Z2 Diet Compostion
    AZ2DCPOC = AREG_MESOZPOC/(AREG_MIC2MES+AREG_MESOZPOC+
    *           AREG_MESOZALG)
    AZ2DCZ1 = AREG_MIC2MES/(AREG_MIC2MES+AREG_MESOZPOC+
    *           AREG_MESOZALG)
    AZ2DCALG = AREG_MESOZALG/(AREG_MIC2MES+AREG_MESOZPOC+
    *           AREG_MESOZALG)
    WRITE(23,42)
    WRITE(23,70) AZ2DCPOC,AZ2DCZ1,AZ2DCALG

C Deposit Feeders (DF) Diet Compostion
    ADFDCSedPOC = AREG_DFUAC/AREG_DFUAC
    WRITE(23,43)
    WRITE(23,70) ADFDCSedPOC

C Filter Feeders (SF) Diet Compostion
    ASFDCPOC = AREG_SFPCCON/(AREG_SFPCCON+AREG_SFACON)
    ASFDCALG = AREG_SFACON/(AREG_SFPCCON+AREG_SFACON)
    WRITE(23,45)
    WRITE(23,70) ASFDCPOC,ASFDCALG

C
C Detrital Fate
C
C Microphytobenthos Detrital Fate
    AMICRBENALG_SedPOC = AREG_BALGC/AREG_BALGC
    AMICRBENALG_POC = 0.
    AMICRBENALG_DOC = 0.
    ABAExport = 0.
    ABATotal =
    &
    AMICRBENALG_SedPOC+AMICRBENALG_POC+AMICRBENALG_DOC+AExport
    WRITE(23,47)
    WRITE(23,48)
    AMICRBENALG_DOC,AMICRBENALG_SedPOC,AMICRBENALG_POC,
    *           ABAExport,ABATotal

```

C Phytoplankton Detrital Fate

```

AALG_SedPOC = AREG_ALG2SED/(AREG_ADOC+AREG_APOC+
*      AREG_ALG2SED)
AALG_POC = AREG_APOC/(AREG_ADOC+AREG_APOC+AREG_ALG2SED)
AALG_DOC = AREG_ADOC/(AREG_ADOC+AREG_APOC+AREG_ALG2SED)
AAlgExport = 0.
AAlgTotal = AALG_SedPOC+AALG_POC+AALG_DOC+AAlgExport
WRITE(23,49)
WRITE(23,48) AALG_DOC,AALG_SedPOC,AALG_POC,
*      AAlgExport,AAlgTotal

```

C Microzooplankton Detrital Fate

```

AZ1_SedPOC = 0.
AZ1_POC = AREG_UAPOCSZ/(AREG_UAPOCSZ+AREG_UADOC SZ)
AZ1_DOC = AREG_UADOC SZ/(AREG_UAPOCSZ+AREG_UADOC SZ)
AZ1Export = 0.
AZ1Total = AZ1_SedPOC +AZ1_POC +AZ1_DOC +AZ1Export
WRITE(23,51)
WRITE(23,48) AZ1_DOC,AZ1_SedPOC,AZ1_POC,
*      AZ1Export,AZ1Total

```

C Mesozooplankton Detrital Fate

```

AZ2_SedPOC = 0.
AZ2_POC = AREG_UAPOCLZ/(AREG_UAPOCLZ+AREG_UADOC LZ)
AZ2_DOC = AREG_UADOC LZ/(AREG_UAPOCLZ+AREG_UADOC LZ)
AZ2Export = 0.
AZ2Total = AZ2_SedPOC +AZ2_POC +AZ2_DOC +AZ2Export
WRITE(23,53)
WRITE(23,48) AZ2_DOC,AZ2_SedPOC,AZ2_POC,
*      AZ2Export,AZ2Total

```

C SAV Detrital Fate

```

ASAV_SedPOC = AREG_SAV2SED/(AREG_SAV2DOC+AREG_SAV2POC+
*      AREG_SAV2SED)
ASAV_POC = AREG_SAV2POC/(AREG_SAV2DOC+AREG_SAV2POC+
*      AREG_SAV2SED)
ASAV_DOC = AREG_SAV2DOC/(AREG_SAV2DOC+AREG_SAV2POC+
*      AREG_SAV2SED)
ASAVExport = 0.
ASAVTotal = ASAV_SedPOC +ASAV_POC +ASAV_DOC +ASAVExport
WRITE(23,55)
WRITE(23,48) ASAV_DOC,ASAV_SedPOC,ASAV_POC,
*      ASAVExport,ASAVTotal

```

C Deposit Feeders Detrital Fate

```

ADF_SedPOC = AREG_DFUAC/AREG_DFUAC
ADF_POC = 0.
ADF_DOC = 0.
ADFExport = 0.
ADFTotal = ADF_SedPOC+ADF_POC+ADF_DOC+ADFExport
WRITE(23,57)
WRITE(23,48) ADF_DOC,ADF_SedPOC,ADF_POC,
*      ADFExport,ADFTotal

```

C Suspension Feeders Detrital Fate

```

ASF_SedPOC = AREG_SFUAC/AREG_SFUAC
ASF_POC = 0.

```

```

    ASF_DOC = 0.
    ASFExport = 0.
    ASFTotal = ASF_SedPOC+ASF_POC+ASF_DOC+ASFExport
    WRITE(23,59)
    WRITE(23,48) ASF_DOC, ASF_SedPOC, ASF_POC,
*               ASFExport, ASFTotal

C DOC Detrital Fate
    ADOC_SedPOC = 0.
    ADOC_POC = 0.
    ADOC_DOC = 0.
    ADOCExport = 1.
    ADOCTotal = ADOC_SedPOC +ADOC_POC +ADOC_DOC +ADOCExport
    WRITE(23,61)
    WRITE(23,48) ADOC_DOC, ADOC_SedPOC, ADOC_POC,
*               ADOCExport, ADOCTotal

C Sed POC Detrital Fate
    ASedPOC_SedPOC = 0.
    ASedPOC_POC = 0.
    ASedPOC_DOC = 0.
    ASedPOCExport = 1.
    ASedPOCTotal =
& ASedPOC_SedPOC+ASedPOC_POC+ASedPOC_DOC+ASedPOCExport
    WRITE(23,63)
    WRITE(23,48) ASedPOC_DOC, ASedPOC_SedPOC, ASedPOC_POC,
*               ASedPOCExport, ASedPOCTotal

C POC Detrital Fate
    APOC_SedPOC = AREG_CFLUX/(AREG_CFLUX+AREG_POC2DOC)
    APOC_POC = 0.
    APOC_DOC = AREG_POC2DOC/(AREG_CFLUX+AREG_POC2DOC)
    APOCExport = 0.
    APOCTotal = ASF_SedPOC+ASF_POC+ASF_DOC+ASFExport
    WRITE(23,65)
    WRITE(23,48) APOC_DOC, APOC_SedPOC, APOC_POC,
*               APOCExport, APOCTotal

C
C
    REWIND (KFL)
    GO TO 1

!3    STOP
3     continue

    ! Write out data for the ECOPATH GUI

    call write_ecopath_gui_file(ecm_input_file,
eco_output_file)

    END

```

Appendix D: The forEcopathGui.f90 File

```
! d:\xp\work\cerco\eco\from_dottie\kfl_post_processor\
! NOTE:  integer :: funit = 101      ! TEMPORARY File unit number
used by enclosed routines
```

```
module predator_prey_module

! CONCEPTUAL MAP: {PREY => ROW, PREDATOR => COLUMN}

type predator_prey_type

    integer :: size

    character(len=20), pointer :: names(:)

    real, pointer :: values(:, :)

endtype predator_prey_type

contains

    function predator_prey_create_table(size, names) re-
    sult(predator_prey_table)

        implicit none

        integer :: size

        character(len=*) :: names(:)

        type(predator_prey_type), pointer :: preda-
        tor_prey_table

        integer :: n

        allocate(predator_prey_table)

        predator_prey_table%size = size

        allocate(predator_prey_table%names(size))

        do n=1, size
            predator_prey_table%names(n) = names(n)
        enddo
```

```

        allocate(predator_prey_table%values(size, size))

        predator_prey_table%values = 0.0

    end function predator_prey_create_table


    function predator_prey_get_table_value(predator_prey_table, prey, predator) result(value)

        implicit none

        type(predator_prey_type) :: predator_prey_table
        character(len=*) :: prey, predator
        integer :: prey_index, predator_index
        real :: value

        prey_index = predator_prey_find_index(predator_prey_table, prey)

        predator_index = predator_prey_find_index(predator_prey_table, predator)

        value = predator_prey_table%values(prey_index, predator_index)

    end function predator_prey_get_table_value


    function predator_prey_set_table_value(predator_prey_table, prey, predator,
    newValue) result(oldValue)

        implicit none

        type(predator_prey_type) :: predator_prey_table
        character(len=*) :: prey, predator
        integer :: prey_index, predator_index
        real :: newValue

        real :: oldValue

        prey_index = predator_prey_find_index(predator_prey_table, prey)

        predator_index = predator_prey_find_index(predator_prey_table, predator)

        oldValue = predator_prey_table%values(prey_index, predator_index)

```

```

        predator_prey_table%values(prej_index, predator_index) = newValue
    end function predator_prey_set_table_value

    function predator_prey_find_index(predator_prey_table,
name) result (index)

        implicit none

        type(predator_prey_type) :: predator_prey_table
        character(len=*) :: name
        integer :: index
        integer :: n
        index = -1

        do n=1,predator_prey_table%size
            if(predator_prey_table%names(n) == name) then
                index = n
                return
            endif
        enddo

    end function predator_prey_find_index

end module predator_prey_module

```

```

module icm_constituent_module

    use kfl_mod
    use data_mod
    use predator_prey_module

    type(predator_prey_type) :: icm_diet_composition

    integer :: number_of_constituents

    character(len=20), allocatable :: names(:)

contains

```

```

subroutine initialize_constituent_module()

    implicit none

    number_of_constituents = 10

    allocate(names(number_of_constituents))

    ! Populate ICM constituent list

    names(1) = '"ALG"'           ! Algae
    names(2) = '"Z1"'           ! MicroZooplankton
    names(3) = '"Z2"'           ! MesoZooplankton
    names(4) = '"BALG"'         ! Benthic Algae
    names(5) = '"SAV"'          ! SAV
    names(6) = '"DF"'           ! Deposit Feeders
    names(7) = '"SF"'           ! Suspension Feeders
    names(8) = '"DOC"'          ! Dissolved Organic
Carbon      names(9) = '"SEDPOC"'       ! Sediment Particulate
Organic Carbon      names(10) = '"POC"'       ! Dissolved Particulate
Organic Carbon

    icm_diet_composition = preda-
tor_pre_create_table(number_of_constituents, names)

end subroutine initialize_constituent_module


subroutine build_icm_predator_prey_table()

    implicit none

    icm_diet_composition = preda-
tor_pre_create_table(number_of_constituents, names)

    call calculate_icm_diet_composition()

end subroutine build_icm_predator_prey_table


subroutine calculate_icm_diet_composition()

    implicit none

    integer :: pred_index, prey_index

    integer :: n

```

```

character(len=20), pointer :: name

! Algae consumption by Z1
prey_index = preda-
tor_prey_find_index(icm_diet_composition, "ALG")
pred_index = preda-
tor_prey_find_index(icm_diet_composition, "Z1")
icm_diet_composition%values(prey_index, pred_index) =
AZ1DCALG

! DOC consumption by Z1
prey_index = preda-
tor_prey_find_index(icm_diet_composition, "DOC")
pred_index = preda-
tor_prey_find_index(icm_diet_composition, "Z1")
icm_diet_composition%values(prey_index, pred_index) =
AZ1DCDOC

! POC consumption by Z1
prey_index = preda-
tor_prey_find_index(icm_diet_composition, "POC")
pred_index = preda-
tor_prey_find_index(icm_diet_composition, "Z1")
icm_diet_composition%values(prey_index, pred_index) =
AZ1DCPOC


! Algae consumption by Z2
prey_index = preda-
tor_prey_find_index(icm_diet_composition, "ALG")
pred_index = preda-
tor_prey_find_index(icm_diet_composition, "Z2")
icm_diet_composition%values(prey_index, pred_index) =
AZ2DCALG

! Z1 consumption by Z2
prey_index = preda-
tor_prey_find_index(icm_diet_composition, "Z1")
pred_index = preda-
tor_prey_find_index(icm_diet_composition, "Z2")
icm_diet_composition%values(prey_index, pred_index) =
AZ2DCZ1

! POC consumption by Z2
prey_index = preda-
tor_prey_find_index(icm_diet_composition, "POC")
pred_index = preda-
tor_prey_find_index(icm_diet_composition, "Z2")
icm_diet_composition%values(prey_index, pred_index) =
AZ2DCPOC


! SedPOC consumpton by DF

```

```

        prey_index = predator_prey_find_index(icm_diet_composition, "SEDPOC")
        predator_index = predator_prey_find_index(icm_diet_composition, "DF")
        icm_diet_composition%values(prey_index, predator_index) =
ADFDCSedPOC

```

```

        ! Algae consumption by SF ! ADDED 9-20-2007
        prey_index = predator_prey_find_index(icm_diet_composition, "ALG")
        predator_index = predator_prey_find_index(icm_diet_composition, "SF")
        icm_diet_composition%values(prey_index, predator_index) =
ASFDCALG

```

```

        ! POC consumption by SF
        prey_index = predator_prey_find_index(icm_diet_composition, "POC")
        predator_index = predator_prey_find_index(icm_diet_composition, "SF")
        icm_diet_composition%values(prey_index, predator_index) =
ASFDCPOC

```

```

    end subroutine calculate_icm_diet_composition

```

```

end module icm_constituent_module

```

```

module ecopath_module

```

```

    use predator_prey_module

```

```

    type(predator_prey_type) :: ecopath_diet_composition

```

```

        ! Ideally this would be an array, but post-processor
        treats them as entities.

```

```

    type detrital_fate_type

```

```

        real :: DOC

```

```

        real :: sedimentPOC

```

```

        real :: POC

```

```

    endtype detrital_fate_type

```

```

    type ecopath_group_type

```

```

        integer :: n

```

```

        character(len=20) :: name
        character(len=20) :: icm_name_alias
        real :: fraction
        real :: biomass
        real :: production_biomass_ratio
        real :: consumption_biomass_ratio
        real :: unassimilated_consumption_ratio
        type(detrital_fate_type) :: detrital_fate
    endtype  ecopath_group_type

    type ecopath_type
        type(ecopath_group_type), pointer :: groups(:)
        integer :: number_of_groups

        type(ecopath_group_type), pointer :: producer_groups(:)
        integer :: number_producer_groups
        integer :: producer_eco_type

        type(ecopath_group_type), pointer :: consumer_groups(:)
        integer :: number_consumer_groups
        integer :: consumer_eco_type

        type(ecopath_group_type), pointer :: detrital_groups(:)
        integer :: number_detrital_groups
        integer :: detrital_eco_type

    endtype ecopath_type

    type(ecopath_type) :: ecopath

    integer, parameter :: outf = 102

contains

    subroutine initialize_ecopath_module(ecm_filename,
    eco_filename)

        ! ecm_filename specifies  input file
        ! eco_filename specifies  input file

```

```

        implicit none
        character(len=*) :: ecm_filename
        character(len=*) :: eco_filename
        character(len=200) :: line
        integer, parameter :: inf = 101
        integer :: m, n
        type(ecopath_group_type), pointer :: group
        character(len=20) :: name, alias
        character(len=20), pointer :: names(:)
        real :: fraction

100      format(A80)
120      format(A20)
150      format(4x, I12, I12)
200      format(A20, 10x, A20, 10x, f5.3)

        ! Input file
        open(unit=inf, file=ecm_filename, form='FORMATTED',
status='OLD')

        ! Output file
        open(unit=outf, file=eco_filename,
form='FORMATTED', status='UNKNOWN')

File Title      read(inf, 100) line;   write(outf, 100) line  !

        ! Step 1: Process all PRODUCERS
        ! Read "NUMBER OF PRODUCER GROUPS"

Blank line      read(inf, 100) line;   write(outf, 100) line  !

PRODUCER count & type header      read(inf, 100) line;   write(outf, 100) line  !

        read(inf, 150) ecopath%number_producer_groups, eco-
path%producer_eco_type
        write(outf, 150) ecopath%number_producer_groups,
ecopath%producer_eco_type

```

```

        allo-
cate(ecopath%producer_groups(ecopath%number_producer_groups))

        do n=1,ecopath%number_producer_groups
            ecopath%producer_groups(n)%n = -1
            ecopath%producer_groups(n)%name = 'UNKNOWN'
            ecopath%producer_groups(n)%icm_name_alias =
'UNKNOWN ALIAS'
            ecopath%producer_groups(n)%fraction = 0.0
            ecopath%producer_groups(n)%biomass = 0.0
            eco-
path%producer_groups(n)%production_biomass_ratio = 0.0
            eco-
path%producer_groups(n)%consumption_biomass_ratio = 0.0
            eco-
path%producer_groups(n)%unassimilated_consumption_ratio = 0.0
            ecopath%producer_groups(n)%detrital_fate = de-
trital_fate_type(0.0, 0.0, 0.0)
        enddo

        read(inf, 100) line;    write(outf, 100) line    !
Blank line

        read(inf, 100) line;    write(outf, 100) line    !
"Producer names" header

        do n=1, ecopath%number_producer_groups
            read(inf, 200) name, alias, fraction;
write(outf, 200) name, alias, fraction

            group => ecopath%producer_groups(n)

            group%name = name

            group%n = n    ! group id

            group%icm_name_alias = alias

            group%fraction = fraction

            !write(*,*) name

        enddo

        read(inf, 100) line;    write(outf, 100) line    !
Blank line

```

! Step 2: Process all CONSUMERS

```

! Read "NUMBER OF CONSUMER GROUPS"

Blank line      read(inf, 100) line;   write(outf, 100) line   !

                read(inf, 100) line;   write(outf, 100) line   !
CONSUMER count & type header

                read(inf, 150) ecopath%number_consumer_groups, eco-
path%consumer_eco_type
                write(outf, 150) ecopath%number_consumer_groups,
ecopath%consumer_eco_type

                allo-
cate(ecopath%consumer_groups(ecopath%number_consumer_groups))

                do n=1, ecopath%number_consumer_groups
                    ecopath%consumer_groups(n)%n = -1
                    ecopath%consumer_groups(n)%name = 'UNKNOWN'
                    ecopath%consumer_groups(n)%icm_name_alias =
'UNKNOWN ALIAS'
                    ecopath%consumer_groups(n)%fraction = 0.0
                    ecopath%consumer_groups(n)%biomass = 0.0
                    eco-
path%consumer_groups(n)%production_biomass_ratio = 0.0
                    eco-
path%consumer_groups(n)%consumption_biomass_ratio = 0.0
                    eco-
path%consumer_groups(n)%unassimilated_consumption_ratio = 0.0
                    ecopath%consumer_groups(n)%detrital_fate = de-
trital_fate_type(0.0, 0.0, 0.0)
                enddo

Blank line      read(inf, 100) line;   write(outf, 100) line   !

                read(inf, 100) line;   write(outf, 100) line   !
"CONSUMER names" header

                do n=1, ecopath%number_consumer_groups

                    read(inf, 200) name, alias, fraction;
write(outf, 200) name, alias, fraction

                    group => ecopath%consumer_groups(n)

                    group%name = name

                    group%n = n      ! group id

                    group%icm_name_alias = alias

                    group%fraction = fraction

                    !write(*,*) name

```

```

                                enddo

                                read(inf, 100) line;   write(outf, 100) line      !
Blank line

                                ! Step 3: Process all DETRITUS

                                ! Read "NUMBER OF DETRITAL GROUPS"

                                read(inf, 100) line;   write(outf, 100) line      !
Blank line

                                read(inf, 100) line;   write(outf, 100) line      !
DETRITUS count & type header

                                read(inf, 150) ecopath%number_detrital_groups, eco-
path%detrital_eco_type
                                write(outf, 150) ecopath%number_detrital_groups,
ecopath%detrital_eco_type

                                allo-
cate(ecopath%detrital_groups(ecopath%number_detrital_groups))

                                do n=1, ecopath%number_detrital_groups
                                    ecopath%detrital_groups(n)%n = -1
                                    ecopath%detrital_groups(n)%name = 'UNKNOWN'
                                    ecopath%detrital_groups(n)%icm_name_alias =
'UNKNOWN ALIAS'
                                    ecopath%detrital_groups(n)%fraction = 0.0
                                    ecopath%detrital_groups(n)%biomass = 0.0
                                    eco-
path%detrital_groups(n)%production_biomass_ratio = 0.0
                                    eco-
path%detrital_groups(n)%consumption_biomass_ratio = 0.0
                                    eco-
path%detrital_groups(n)%unassimilated_consumption_ratio = 0.0
                                    ecopath%detrital_groups(n)%detrital_fate = de-
trital_fate_type(0.0, 0.0, 0.0)
                                enddo

                                read(inf, 100) line;   write(outf, 100) line      !
Blank line

                                read(inf, 100) line;   write(outf, 100) line      !
"DETRITUS names" header

                                do n=1, ecopath%number_detrital_groups

                                    read(inf, 200) name, alias, fraction;
write(outf, 200) name, alias, fraction

```

```

        group => ecopath%detrital_groups(n)
        group%name = name
        group%n = n      ! group id
        group%icm_name_alias = alias
        group%fraction = fraction
        !write(*,*) name
    enddo

    write(outf, 100)      ! Write blank line

    ! Step 4: Combine producers, consumers, and detri-
    tus into one group

    ecopath%number_of_groups = eco-
    path%number_producer_groups &
                                + eco-
    path%number_consumer_groups &
                                + eco-
    path%number_detrital_groups

    allocate(ecopath%groups(ecopath%number_of_groups))

    m = 1
    do n = 1, ecopath%number_producer_groups
        ecopath%groups(m) = ecopath%producer_groups(n)
        m = m+1
    enddo

    do n = 1, ecopath%number_consumer_groups
        ecopath%groups(m) = ecopath%consumer_groups(n)
        m = m+1
    enddo

    do n = 1, ecopath%number_detrital_groups
        ecopath%groups(m) = ecopath%detrital_groups(n)
        m = m+1
    enddo

    ! Step 5: Write out group names to screen log

    do n = 1, ecopath%number_of_groups
        write(*,*) ecopath%groups(n)%name, eco-
    path%groups(n)%icm_name_alias
    enddo

```

```

                                close(inf);
!close(outf);

                                ! L A S T   S T E P   -   B U I L D   P R E D A T O R
P R E Y   T A B L E

                                ! Build predator-prey table. (Need an array of char
variables containing group name)

                                allocate(names(ecopath%number_of_groups))

                                do n=1,ecopath%number_of_groups
                                    names(n) = ecopath%groups(n)%name
                                enddo

                                ecopath_diet_composition = preda-
tor_pre_create_table(ecopath%number_of_groups, names)

                                deallocate(names) ! No longer needed

                                end subroutine initialize_ecopath_module

!
!      function get_ecopath_group_type(group_name) re-
!      sult(target)
!
!          implicit none
!
!          character(len=*) :: group_name
!
!          type(ecopath_group_type), pointer :: target
!
!          integer :: n
!
!          nullify(target)
!
!          do n=1, ecopath%number_of_groups
!              if(ecopath%groups(n)%name .eq. group_name) then
!                  target => ecopath%groups(n)
!                  exit
!              endif
!          enddo
!
!      endfunction

```

```

!           enddo
!
!           end function get_ecopath_group_type

end module  ecopath_module

!-----
!-----

subroutine compute_ecopath_parameters()

  use kfl_mod
  use data_mod
  use ecopath_module

  implicit none

  character(len=20) :: name
  character(len=20) :: alias
  real :: fraction

  type(ecopath_group_type), pointer :: target

  integer :: n

  do n=1, ecopath%number_of_groups
    name = ecopath%groups(n)%name
    alias = ecopath%groups(n)%icm_name_alias
    fraction = ecopath%groups(n)%fraction
    select case (alias)
      case('"ALG"')
        ecopath%groups(n)%biomass =
AREG_ALGC*ecopath%groups(n)%fraction
        ecopath%groups(n)%production_biomass_ratio =
APB_ALGRatio
        ecopath%groups(n)%detrital_fate%DOC =
AALG_DOC
        ecopath%groups(n)%detrital_fate%SedimentPOC =
AALG_sedPOC
        ecopath%groups(n)%detrital_fate%POC =
AALG_POC
      case('"Z1"')

```

```

                                ecopath%groups(n)%biomass =
AREG_MICRZ*ecopath%groups(n)%fraction
                                ecopath%groups(n)%production_biomass_ratio =
APB_Z1Ratio
                                ecopath%groups(n)%consumption_biomass_ratio =
AQB_Z1Ratio
                                eco-
path%groups(n)%unassimilated_consumption_ratio = AUATC_Z1Ratio
                                ecopath%groups(n)%detrital_fate%DOC = AZ1_DOC
                                ecopath%groups(n)%detrital_fate%SedimentPOC =
AZ1_sedPOC
                                ecopath%groups(n)%detrital_fate%POC = AZ1_POC

                                case('"Z2"')
                                ecopath%groups(n)%biomass =
AREG_MES0Z*ecopath%groups(n)%fraction
                                ecopath%groups(n)%production_biomass_ratio =
APB_Z2Ratio
                                ecopath%groups(n)%consumption_biomass_ratio =
AQB_Z2Ratio
                                eco-
path%groups(n)%unassimilated_consumption_ratio = AUATC_Z2Ratio
                                ecopath%groups(n)%detrital_fate%DOC = AZ2_DOC
                                ecopath%groups(n)%detrital_fate%SedimentPOC =
AZ2_sedPOC
                                ecopath%groups(n)%detrital_fate%POC = AZ2_POC

                                case('"BALG"')
                                ecopath%groups(n)%biomass =
AREG_BALG*ecopath%groups(n)%fraction
                                ecopath%groups(n)%production_biomass_ratio =
APB_BALGRatio
                                ecopath%groups(n)%detrital_fate%DOC =
AMICRBENALG_DOC
                                ecopath%groups(n)%detrital_fate%SedimentPOC =
AMICRBENALG_sedPOC
                                ecopath%groups(n)%detrital_fate%POC =
AMICRBENALG_POC

                                case('"SAV"')
                                ecopath%groups(n)%biomass =
AREG_SAV*ecopath%groups(n)%fraction
                                ecopath%groups(n)%production_biomass_ratio =
APB_SAVRatio
                                ecopath%groups(n)%detrital_fate%DOC =
ASAV_DOC
                                ecopath%groups(n)%detrital_fate%SedimentPOC =
ASAV_sedPOC
                                ecopath%groups(n)%detrital_fate%POC =
ASAV_POC

                                case('"DF"')
                                ecopath%groups(n)%biomass =
AREG_DFEED*ecopath%groups(n)%fraction
                                ecopath%groups(n)%production_biomass_ratio =
APB_DFRatio

```

```

ecopath%groups(n)%consumption_biomass_ratio =
AQB_DFRatio
eco-
path%groups(n)%unassimilated_consumption_ratio = AUATC_DFRatio
ecopath%groups(n)%detrital_fate%DOC = ADF_DOC
ecopath%groups(n)%detrital_fate%SedimentPOC =
ADF_sedPOC
ecopath%groups(n)%detrital_fate%POC = ADF_POC

case('"SF"')
ecopath%groups(n)%biomass =
AREG_SFEED*ecopath%groups(n)%fraction
ecopath%groups(n)%production_biomass_ratio =
APB_SFRatio
ecopath%groups(n)%consumption_biomass_ratio =
AQB_SFRatio
eco-
path%groups(n)%unassimilated_consumption_ratio = AUATC_SFRatio
ecopath%groups(n)%detrital_fate%DOC = ASF_DOC
ecopath%groups(n)%detrital_fate%SedimentPOC =
ASF_sedPOC
ecopath%groups(n)%detrital_fate%POC = ASF_POC

case('"DOC"')
ecopath%groups(n)%biomass =
AREG_DOC*ecopath%groups(n)%fraction
ecopath%groups(n)%detrital_fate%DOC =
ADOC_DOC
ecopath%groups(n)%detrital_fate%SedimentPOC =
ADOC_sedPOC
ecopath%groups(n)%detrital_fate%POC =
ADOC_POC

case('"SEDPOC"')
ecopath%groups(n)%biomass =
AREG_SEDPOC*ecopath%groups(n)%fraction
ecopath%groups(n)%detrital_fate%DOC = ASed-
POC_DOC
ecopath%groups(n)%detrital_fate%SedimentPOC =
ASedPOC_sedPOC
ecopath%groups(n)%detrital_fate%POC = ASed-
POC_POC

case('"POC"')
ecopath%groups(n)%biomass =
AREG_POC*ecopath%groups(n)%fraction
!9-21-2007:ERROR HERE: eco-
path%groups(n)%detrital_fate%DOC = POC_DOC
ecopath%groups(n)%detrital_fate%DOC =
APOC_DOC
ecopath%groups(n)%detrital_fate%SedimentPOC =
APOC_sedPOC
ecopath%groups(n)%detrital_fate%POC =
APOC_POC

case default

```

```

                                write(*,*) 'Error encountered in subroutine
compute_ecopath_parameters()'
                                write(*,*) alias
                                stop
                                end select

                                enddo

end subroutine compute_ecopath_parameters

```

```

subroutine write_ecopath_basic_kinetics_parameters()

    use kfl_mod
    use data_mod
    use ecopath_module

    implicit none

    character(len=200),dimension(13) :: line
    integer :: n

    write(outf,*)
    write(outf,*) 'ECOPATH INPUT'
    write(outf,*)

    ! B A S I C   K I N E T I C S

    line(1) = "BASIC kinetics"
    write(outf,*) adjustl(trim(line(1)))

    line(1) = 'GROUP           "Biomass"           "Produc-
tion/Biomass"           "Consumption/Biomass"           "Unassimi-
lated/Consumption"'
    write(outf,120) adjustl(line(1))

    do n=1, ecopath%number_of_groups

        write(outf,130) ecopath%groups(n)%name, &
                        ecopath%groups(n)%biomass, &
                        eco-
path%groups(n)%production_biomass_ratio, &
                        eco-
path%groups(n)%consumption_biomass_ratio, &
                        eco-
path%groups(n)%unassimilated_consumption_ratio
    enddo

```

```

120  format(a200)
130  format(a17,  f8.3, 15x, f8.3, 15x, f8.3, 15x, f8.3)

    end subroutine write_ecopath_basic_kinetics_parameters

subroutine write_ecopath_detrital_fate_parameters()

    use kfl_mod
    use data_mod
    use ecopath_module

    implicit none

    character(len=200),dimension(13) :: line

    integer :: n

    ! D E T R I T A L   F A T E

    write(outf,*)

    line(1) = "DETRITAL FATE (from-->to)"
    write(outf,*) adjustl(trim(line(1)))

    line(1) = 'GROUP                                "DOC"                "Sediment
POC"          "POC"'
    write(outf,120) adjustl(line(1))

    do n=1, ecopath%number_of_groups

        write(outf,130) ecopath%groups(n)%name, &
                        ecopath%groups(n)%detrital_fate%DOC,
&
                        eco-
path%groups(n)%detrital_fate%SedimentPOC, &
                        ecopath%groups(n)%detrital_fate%POC
    enddo

120  format(a200)
130  format(a17,  f8.3, 15x, f8.3, 15x, f8.3, 15x, f8.3)

    end subroutine write_ecopath_detrital_fate_parameters

```

```

subroutine build_ecopath_predator_prey_table()

  use predator_prey_module
  use icm_constituent_module
  use ecopath_module

  implicit none

  type( ecopath_group_type), pointer :: prey, predator

  character(len=20) :: alias_prey_name,
alias_predator_name

  integer :: icm_prey_index, icm_predator_index

  integer :: ecopath_prey_index, ecopath_predator_index

  integer m, n

  real :: icm_value, ecopath_value

  do m=1, ecopath%number_of_groups

    prey => ecopath%groups(m)

    do n=1, ecopath%number_of_groups

      predator => ecopath%groups(n)

      icm_value = preda-
tor_prey_get_table_value(icm_diet_composition,
prey%icm_name_alias, predator%icm_name_alias)

      !Per Carl, these are ratios - dont multiply by
group fraction
      !ecopath_value = icm_value * prey%fraction *
predator%fraction
      ecopath_value = icm_value * prey%fraction

      ecopath_value = preda-
tor_prey_set_table_value(ecopath_diet_composition, prey%name,
predator%name, ecopath_value)

      ! Current value of "ecopath_value" is now the
previous old value

    enddo

  enddo

end subroutine build_ecopath_predator_prey_table

```

```

subroutine write_ecopath_predator_preym_table()

  use predator_preym_module
  use icm_constituent_module
  use ecopath_module

  implicit none
  character(len=132),dimension(13) :: line
  character(len=20) :: prey_name
  integer :: m, n
  integer :: table_size

  ! D I E T    C O M P O S I T I O N

  write(outf,*)

  line(1) = "DIET COMPOSITION (from-->to)"
  write(outf,*) adjustl(trim(line(1)))

  table_size = ecopath_diet_composition%size

  write(outf,150) (ad-
  justr(ecopath_diet_composition%names(m)), m=1,table_size)

  do m=1,  ecopath_diet_composition%size
    prey_name = ecopath_diet_composition%names(m)

    write(outf,155) prey_name, (eco-
  path_diet_composition%values(m,n), n=1,table_size)

  enddo

100      format(50(A20))
120      format(A20,50(10x,f10.4))
150      format("group", 15(a20))
155      FORMAT(A20,13(12X,F8.3))

  end subroutine write_ecopath_predator_preym_table

```

```

!-----
-----

```

```

      subroutine write_ecopath_gui_file(input_filename, out-
put_filename)

      ! input_filename: the "ecm" file to read
      ! output_filename: the "eco" file to create

      use ecopath_module

      use icm_constituent_module

      implicit none

      character(len=*) :: input_filename

      character(len=*) :: output_filename


      ! I N I T I A L I Z E   I C M

      ! GET DATA FROM ICM

      call initialize_constituent_module()

      call build_icm_predator_preym_table()           ! Thats the
icm_diet_composition table


      ! I N I T I A L I Z E   E C O P A T H

      call initialize_ecopath_module(input_filename, out-
put_filename)

      call compute_ecopath_parameters()

      call build_ecopath_predator_preym_table()       ! Thats the
ecopath_diet_composition table


      ! W R I T E   O U T P U T

      call write_ecopath_basic_kinetics_parameters()

      call write_ecopath_detrital_fate_parameters()

      call write_ecopath_predator_preym_table()

      close(outf)

      end subroutine write_ecopath_gui_file

```

Appendix E: Module File for 4000-Cell KFL Postprocessor

```

module kfl_mod

  INTEGER NCP, NBP, NQFP, NHQP, NSBP, NLP, NS1P, NS2P, NS3P,
  .       NBCP, NMP, NDP, NSAVP, NFLP, NOIP, NSSFP, NPES

  PARAMETER (NCP=24)

c Chesapeake Bay ( for 1 PE run ) 4000 cells
  PARAMETER (NBP=4073, NQFP=9874, NHQP=6530, NSBP=729, NLP=15,
!CHESAPEAKE
  .       NS1P=600, NS2P=600,
!CHESAPEAKE
  .       NS3P=2961, NBCP=120, NMP=30, NDP=500, NSAVP=5,
!CHESAPEAKE
  .       NFLP=100, NOIP=10, NSSFP=3, NPES=1)
!CHESAPEAKE

c ! Chesapeake Bay ( for 1 PE run ) 12000 cells
c   PARAMETER
(NBP=12920, NQFP=30835, NHQP=20876, NSBP=2961, NLP=19, !CHESAPEAKE
c   .       NS1P=4000, NS2P=4000,
!CHESAPEAKE
c   .       NS3P=4000, NBCP=496, NMP=30, NDP=500, NSAVP=5,
!CHESAPEAKE
c   .       NFLP=100, NOIP=10, NSSFP=3, NPES=1)
!CHESAPEAKE

  REAL E_BALG(NSBP), E_BNPP(NSBP), E_DFEED(NSBP),
E_SAV(NSBP),
  .       E_CFLUX(NSBP), E_SAVNP(NSBP), E_BALGR(NSBP),
  .       E_BALGPR(NSBP), E_BALGC(NSBP), E_SFEED(NSBP),
E_BURIAL(NSBP),
  .       E_SAV2SED(NSBP), E_SAV2POC(NSBP), E_SAV2DOC(NSBP),
  .       E_DFNP(NSBP), E_DFTCON(NSBP), E_DFUAC(NSBP),
E_SFNP(NSBP),
  .
E_SFTCON(NSBP), E_SFACON(NSBP), E_SFPCCON(NSBP), E_SFUAC(NSBP),
  .       E_ALG2SED(NSBP), E_SEDPOC(NSBP), E_SEDR(NSBP),
E_DFR(NSBP),
  .       E_SFR(NSBP), E_SAVR(NSBP)

  REAL E_ALGC(NBP), E_ANPP(NBP), E_AGPP(NBP),
E_MICRZ(NBP),
  .       E_MESoz(NBP), E_DOC(NBP), E_POC(NBP),
E_DETC(NBP),
  .       E_APRED(NBP), E_ADOC(NBP), E_APOC(NBP),
E_CRESP(NBP),

```

```

      .      E_MICRZR(NBP), E_MESOZR(NBP),
E_MIC2MES(NBP), E_MICRZNP(NBP),
      .      E_MESOZNP(NBP), E_MICRZDOC(NBP),
      .      E_MICRZPOC(NBP), E_MESOZPOC(NBP), E_MICRZPR(NBP),
      .      E_MESOZPR(NBP), E_MICRZALG(NBP), E_MESOZALG(NBP)

      REAL E_UADOC SZ(NBP), E_UAPOC SZ(NBP), E_UAPOCLZ(NBP),
      .      E_UADOC LZ(NBP), E_POC2DOC(NBP), E_TCONLZ(NBP),
      .      E_TCONS Z(NBP)

      REAL COL_JDAY,          COL_ALGC,          COL_ANPP,
      .      COL_AGPP,          COL_APRED,          COL_ADOC,
      .      COL_APOC,
      .      COL_DOC,          COL_POC,          COL_DETC,
      .      COL_CRESP,        COL_POC2DOC,        COL_MICRZ,
      .      COL_MICRZR,        COL_MICRZNP,        COL_MICRZDOC,
      .      COL_MICRZPOC,      COL_MICRZPR,        COL_MICRZALG,
      .      COL_TCONS Z,      COL_UADOC SZ,        COL_UAPOC SZ,
      .      COL_MESOZ,        COL_MESOZR,        COL_MESOZNP,
      .      COL_MESOZPOC,      COL_MESOZPR,
COL_MESOZALG,
      .      COL_MIC2MES,      COL_TCONLZ,        COL_UADOC LZ,
      .      COL_UAPOCLZ

      REAL COL_BURIAL,        COL_CFLUX,        COL_ALG2SED,
      .      COL_BALG,        COL_BALGR,        COL_BNPP,
      .      COL_BALGPR,      COL_BALGC,
      .      COL_SAV,        COL_SAVNP,        COL_SAV2SED,
      .      COL_SAV2POC,      COL_SAV2DOC,        COL_SFEE D,
      .      COL_SFNP,        COL_SFTCON,        COL_SFACON,
      .      COL_SFPCCON,      COL_SFUAC,        COL_DFEE D,
      .      COL_DFN P,        COL_DFTCON,        COL_DFUAC,
      .      COL_SEDPOC,      COL_SEDR,        COL_SFR,
      .      COL_DFR,        COL_SAVR

      REAL REG_JDAY(10000),  REG_ALGC(10000),  REG_ANPP(10000),
      .      REG_AGPP(10000),  REG_APRED(10000),  REG_ADOC(10000),
      .      REG_APOC(10000),
      .      REG_DOC(10000),  REG_POC(10000),  REG_DETC(10000),
      .      REG_CRESP(10000), REG_POC2DOC(10000), REG_MICRZ(10000),
      .
REG_MICRZR(10000), REG_MICRZNP(10000), REG_MICRZDOC(10000),
      .
REG_MICRZPOC(10000), REG_MICRZPR(10000), REG_MICRZALG(10000),
      .
REG_TCONS Z(10000), REG_UADOC SZ(10000), REG_UAPOC SZ(10000),
      .      REG_MESOZ(10000), REG_MESOZR(10000),
REG_MESOZNP(10000),
      .
REG_MESOZPOC(10000), REG_MESOZPR(10000), REG_MESOZALG(10000),
      .
REG_MIC2MES(10000), REG_TCONLZ(10000), REG_UADOC LZ(10000),

```

```

        .      REG_UAPOCLZ(10000)

        REAL REG_BURIAL(10000), REG_CFLUX(10000),
REG_ALG2SED(10000),
        .      REG_BALG(10000),    REG_BALGR(10000),
        .      REG_BALGPR(10000), REG_BALGC(10000), REG_BNPP(10000),

        .      REG_SAV(10000),    REG_SAVNP(10000),
REG_SAV2SED(10000),
        .
REG_SAV2POC(10000), REG_SAV2DOC(10000), REG_SFEED(10000),
        .      REG_SFNP(10000),
REG_SFTCON(10000), REG_SFACON(10000),
        .      REG_SFPCCON(10000), REG_SFUAC(10000), REG_DFEED(10000),

        .      REG_DFN(10000),    REG_DFTCON(10000), REG_DFUAC(10000),
        .      REG_SEDPOC(10000), REG_SEDR(10000),  REG_SFR(10000),
        .      REG_DFR(10000),    REG_SAVR(10000)

        REAL MICRBENALG_DOC, MICRBENALG_POC, MICRBENALG_SedPOC

        REAL V1(0:NBP), SFA(NSBP), JDAY

        INTEGER NB, NSB, SBN(NSBP), BBN(NSBP), CELL, B
        INTEGER NBOXCOL(NSBP), BOX(NSBP,NLP), REG_CELL(1000)

        CHARACTER*72 TITLE(6)

        LOGICAL SAV_CALC, BALGAE_CALC

        DATA KFL /21/

        end module kfl_mod

        module data_mod

        real  AREG_JDAY, AREG_ALGC, AREG_ANPP, AREG_AGPP,
&          AREG_APRED, AREG_ADOC, AREG_APOC

        real AREG_DOC, AREG_POC, AREG_DETC, AREG_CRESP,
AREG_POC2DOC

        real AREG_MICRZ, AREG_MICRZR, AREG_MICRZNP, AREG_MICRZDOC,
&          AREG_MICRZPOC, AREG_MICRZPR, AREG_MICRZALG,
AREG_TCONSZ,
&          AREG_UADOC SZ, AREG_UAPOC SZ

        real AREG_MES0Z, AREG_MES0ZR, AREG_MES0ZNP, AREG_MES0ZPOC,
&          AREG_MES0ZPR, AREG_MES0ZALG, AREG_MIC2MES,
AREG_TCONLZ,
&          AREG_UADOC LZ, AREG_UAPOC LZ

```

```
    real AREG_BURIAL, AREG_CFLUX, AREG_SEDR, AREG_ALG2SED,
&    AREG_BALG, AREG_BALGR, AREG_BALGPR, AREG_BALGC,
&    AREG_BNPP

    real AREG_SAV, AREG_SAVNP, AREG_SAVR, AREG_SAV2SED,
&    AREG_SAV2POC, AREG_SAV2DOC

    real AREG_SFEED, AREG_SFNP, AREG_SFR, AREG_SFTCON,
&    AREG_SFACON, AREG_SFPCCON, AREG_SFUAC,
&    AREG_DFEED, AREG_DFNP, AREG_DFR, AREG_DFTCON,
&    AREG_DFUAC, AREG_SEDPOC

! Production/Biomass ratio
    real PB_BALGRatio, PB_ALGRatio, PB_Z1Ratio, PB_Z2Ratio,
&    PB_SAVRatio, PB_DFRatio, PB_SFRatio

    real APB_BALGRatio, APB_ALGRatio, APB_Z1Ratio,
&    APB_Z2Ratio, APB_SAVRatio, APB_DFRatio, APB_SFRatio

! Consumption/Biomass
    real QB_Z1Ratio, QB_Z2Ratio, QB_DFRatio, QB_SFRatio
    real AQB_Z1Ratio, AQB_Z2Ratio, AQB_DFRatio, AQB_SFRatio

! Uassimilated/Consumption
    real UATC_Z1Ratio, UATC_Z2Ratio, UATC_DFRatio, UATC_SFRatio
    real
AUATC_Z1Ratio, AUATC_Z2Ratio, AUATC_DFRatio, AUATC_SFRatio

! Z1 Diet Compostion
    real Z1DCDOC, Z1DCPOC, Z1DCALG
    real AZ1DCDOC, AZ1DCPOC, AZ1DCALG

! Z2 Diet Compostion
    real Z2DCPOC, Z2DCZ1, Z2DCALG
    real AZ2DCPOC, AZ2DCZ1, AZ2DCALG

! Deposit Feeders (DF) Diet Compostion
    real DFDCSedPOC
    real ADFDCSedPOC

! Filter Feeders (SF) Diet Compostion
    real SFDCPOC, SFDCALG
    real ASFDCPOC, ASFDCALG
```

```

! Phytoplankton Detrital Fate
real ALG_SedPOC, ALG_POC, ALG_DOC, AlgExport, AlgTotal
real AALG_SedPOC, AALG_POC, AALG_DOC, AAlgExport, AAlgTotal

```

```

! Microzooplankton Detrital Fate
real Z1_SedPOC, Z1_POC, Z1_DOC, Z1Export, Z1Total
real AZ1_SedPOC, AZ1_POC, AZ1_DOC, AZ1Export, AZ1Total

```

```

! Mesozooplankton Detrital Fate
real Z2_SedPOC, Z2_POC, Z2_DOC, Z2Export, Z2Total
real AZ2_SedPOC, AZ2_POC, AZ2_DOC, AZ2Export, AZ2Total

```

```

! SAV Detrital Fate
real SAV_SedPOC, SAV_POC, SAV_DOC, SAVExport, SAVTotal
real ASAV_SedPOC, ASAV_POC, ASAV_DOC, ASAVExport, ASAVTotal

```

```

! Deposit Feeders Detrital Fate
real DF_SedPOC, DF_POC, DF_DOC, DFExport, DFTotal
real ADF_SedPOC, ADF_POC, ADF_DOC, ADFExport, ADFTotal

```

```

! Suspension Feeders Detrital Fate
real SF_SedPOC, SF_POC, SF_DOC, SFExport, SFTotal
real ASF_SedPOC, ASF_POC, ASF_DOC, ASFExport, ASFTotal

```

```

! DOC Detrital Fate
real DOC_SedPOC, DOC_POC, DOC_DOC, DOCExport, DOCTotal
real ADOC_SedPOC, ADOC_POC, ADOC_DOC, ADOCExport, ADOCTotal

```

```

! Sed POC Detrital Fate
real SedPOC_SedPOC, SedPOC_POC, SedPOC_DOC,
& SedPOCExport, SedPOCTotal

real ASedPOC_SedPOC, ASedPOC_POC, ASedPOC_DOC,
& ASedPOCExport, ASedPOCTotal

```

```

! POC Detrital Fate
real POC_SedPOC, POC_POC, POC_DOC, POCExport, POCTotal
real APOC_SedPOC, APOC_POC, APOC_DOC, APOCExport, APOCTotal

```

```

! Microphytobenthos Detrital Fate
real AMICRBENALG_SedPOC, AMICRBENALG_POC, AMICRBENALG_DOC,
& ABAExport, ABATotal, AExport

```

```

end module data_mod

```

