CENTER FOR CONSTRUCTION ENGINEERING AND MANAGEMENT

UM-CYCLONE
Discrete Event Simulation System
User's Guide

Photios G. Ioannou
Assistant Professor

UMCEE Report No. 89-12
Civil and Environmental Engineering Department
UNIVERSITY OF MICHIGAN
Ann Arbor, Michigan
March 19, 1990

UM-CYCLONE
Discrete Event Simulation System
User's Guide

(C) Photios G. Ioannou
Assistant Professor of Civil Engineering

March 19, 1990

UMCE 89-12
Department of Civil and Environmental Engineering
University of Michigan
Ann Arbor, MI 48109-2125
(313) 764-3369

INTRODUCTION

UM-CYCLONE is a discrete event simulation system made up of a collection of programs that work together to provide an integrated modeling environment.

A UM-CYCLONE simulation model is a network of interconnected nodes: Activities, Queues, and Consolidation Nodes. Before using the UM-CYCLONE programs you must first develop a network model by drawing it on paper. This can be done either by hand or by using a computer-based drawing program.

The information necessary for constructing network models can be found in the UM-CYCLONE Reference Manual. This manual describes the various modeling elements that are used to construct networks, explains how these elements interact during the simulation process, and describes the meaning of the various statistics produced by the computer program.

This User's Guide provides information on how to install and use the UM-CYCLONE programs on IBM PC, PS/2, or compatible computers. Example models are provided on the same disk as the UM-CYCLONE programs.

HARDWARE REQUIREMENTS

UM-CYCLONE runs on any IBM PC compatible computer, with at least one floppy drive, 384 Kbyte RAM, a compatible display (Monochrome, CGA, EGA, VGA, or Hercules), running PC-DOS or MS-DOS version 3.0 or higher. Even though an Intel 80x87 math coprocessor is not required, it is highly recommended for speed and accuracy. A color display is also recommended.

REQUIRED FILES

The files required to run UM-CYCLONE are:

CYCLONE.EXE	The main program.
CYC_87.EXE	The UM-CYCLONE simulator for PCs with a 80x87 coprocessor.
CYC_NO87.EXE	The UM-CYCLONE simulator for PCs without a 80x87 coprocessor.

LIST.COM A shareware program for listing text files in full-screen mode. Used for viewing the simulation output.

SIMPRINT.BAT A batch file for printing the simulation output.

INSTALLATION

You can install UM-CYCLONE by copying the above programs either to a floppy disk or to a subdirectory on a hard drive. In either case, it is recommended that you change your computer's PATH to include the disk or subdirectory that contains the UM-CYCLONE programs so that you can run the system from any other disk or subdirectory. This will allow you to keep your simulation model files organized by individual subdirectories, separate from the program files.

Running the UM-CYCLONE System

In order to run the UM-CYCLONE System you must either be in the subdirectory that contains the program files, or you must change your computer's PATH as described above. In either case you can run UM-CYCLONE by typing:

CYCLONE<cr>

where <cr> is the *Enter* or Carriage Return key.

THE UM-CYCLONE USER INTERFACE

The UM-CYCLONE user interface is provided by the executable program CYCLONE.EXE. CYCLONE.EXE is a menu-driven interface that provides facilities for entering, editing, saving, and retrieving the data for a model; performing the simulation; and viewing and printing the results.

CYCLONE.EXE is the only program that the user needs to run directly. All other programs in the UM-CYCLONE system are automatically executed from within CYCLONE.EXE as child processes, by selecting the appropriate menu items. It's data input windows and menu structure are described below.

DATA INPUT WINDOWS

When you first run UM-CYCLONE you are presented with the Activity data input window. There are two other similar data input windows: one for Queues and another for Consolidation Nodes.

You can jump from one input window to any other by holding down the *Alt* key and then typing a number from 1 to 3 using the keys in the top row of your keyboard (not the numbers on the numeric keypad):

Alt-1 To *Activity* Window

Alt-2 To *Queue* Window

Alt-3 To *Consolidation Node* Window

These windows allow you to define records for Activities, Queues, or Consolidation Nodes, by entering values in the appropriate fields. All the fields in each window correspond to a single node and are part of the same record. The fields for a new node record are shown as zeros or blanks.

The structure and operation of the three data input windows is very similar and is described in this section. Specific information about each data window type is provided separately in subsequent sections.

The top-left field is used to input the node's ID number, *NodeNo*, where *NodeNo* is one of: *ActNo*, *QueNo*, or *ConNo*. This field has a special status because it is used to distinguish between nodes, and thus between records. In order to define a record for a new node you must supply a value for this field first. CYCLONE will not let you enter data in the other fields if the *NodeNo* field is empty (*i.e.*, zero). You cannot use the same *NodeNo* to identify more than one node. If you accidentally type the number of record that already exists, UM-CYCLONE will warn you and will disregard your input.

The *NodeNo* can be any valid number and has no special significance (except for Combi Activities as explained below). For large simulation models, a useful numbering scheme is to use different groups of numbers for each of the three types of nodes. This allows the *NodeNo* to also serve as identification of the node type. Furthermore, related nodes can be logically grouped

by giving them numbers that end in the same digits. For example, a logical group of three Activities and two Queues can be numbered as follows: Activities 108, 118, 128, and Queues 308, 318.

In order to ensure the integrity of the data, the fields for each node record, shown in the data input window, are divided into two sets. The first set includes only the *NodeNo* field. The second set includes the remaining fields. You can switch from one set to the other by pressing the *Esc* key. As a reminder, the bottom row on the screen includes the message "*Esc-switch*" to indicate the switching function of the *Esc* key.

You cannot access the second set of fields unless the *NodeNo* field contains an acceptable value. After typing a new number in the *NodeNo* field you must press the *Enter* key to accept it. Assuming that this node number has not been used before, it will be accepted and the cursor will be switched to the first data field of the second set of fields.

The behavior of UM-CYCLONE when you make a new entry in the *NodeNo* field is determined by the "*Record Mode*". The *Record Mode* can be in one of two states: either "*Insert New Rec.*" or "*Edit Old RecNo.*" The current mode is shown in the field at the right bottom of the data input screen. You can toggle the *Record Mode* from "*Insert New Rec.*" to "*Edit Old RecNo.*" and vice versa, by pressing the *F9* key while the cursor is pointing to any field in the second set of data input fields.

The "*Insert New Rec.*" mode is the default. This mode allows you to define new nodes by making a new entry in the *NodeNo* field. To create a new node record press *Esc* to move from the second set of data fields to the *NodeNo* field, type the new node number, and press *Enter*. UM-CYCLONE will create a new record if the node number you type is greater then zero, and has not been used before to define another node. This way, it is impossible to define two nodes with the same *NodeNo*, or to create a new record without specifying its node number. The remaining fields of a newly created record are blank or set equal to zero. For all practical purposes, these fields are considered empty.

Records for a particular node type (*e.g.,* all Activity records) are stored as a linked chain similar to a circular *Rolodex*. You can *scroll* through the various records that you have created by pressing the *PgDn* and *PgUp* keys to move from one record to the next. *PgDn* moves forward

to next record in the record chain and *PgUp* moves backward to the previous record. Obviously, *PgUp* and *PgDn* do not work while the cursor is in *edit mode* (*i.e.,* while you are entering new data or change the value of a field). The chain is "*closed*" and it "*wraps around*" when you reach the first or the last record: pressing *PgUp* while viewing the first record will bring up the last record; pressing *PgDn* while viewing the last record moves you to first record. In either case, a short *beep* indicates that you have crossed the boundary between the first and the last record.

When you enter a new node number, the newly created record is inserted immediately *after* the one that was shown on the screen previously. Records can be entered in the record chain in any desired sequence. The position of a record within the chain has no significance except for making it easier to understand the structure of the model. As a result, nodes that are logically related are usually entered as a group one after the other.

Obviously, the "*Insert New Rec.*" mode makes it impossible to change the number of an existing node. If you try to do so, you will create a new record instead. In order to change the number of an existing node you must first change the *Record Mode* to "*Edit Old No.*" by pressing the *F9* key (while the cursor is not in *edit mode*). You can then press *Esc* to move to the *NodeNo* field and type in a new number. After you press *Enter* to accept your entry, you will see that the rest of the data fields retain their previous values (they are not reset to zero) indicating that no new record is created. You can verify this by pressing the *PgUp* and *PgDn* keys to scroll through the defined records.

The sequence of records in the record chain can be modified by *cutting* and *pasting*. To *cut* the record shown on the screen press the *Del* (Delete) key. UM-CYCLONE will prompt you to verify that you really want to do so. The data for this record are saved on the *clipboard*, an internal buffer that serves as a temporary storage area, until you either paste it back to the chain or perform another cut operation. You can *paste* the record back by pressing the *Ins* (Insert) key. Before inserting the record, UM-CYCLONE will ask you whether you want to paste the record (B)efore or (A)fter the current record. You can paste a record at any point in the record chain by using *PgUp* or *PgDn* to scroll to the appropriate location.

UM-CYCLONE has three separate clipboards, one for each node type. These clipboards are completely independent of each other. Also, only one of them can be active at any point in

time. The active clipboard is determined by which data input window is currently shown on the screen. Switching to another data input window changes the active clipboard but does not affect its contents. Thus, if you cut an Activity, a Queue, and a Consolidation Node, each will be stored in its own clipboard and no node data will be lost. Given the different record structure of each node type, it is impossible to paste one node type as another. This means, for example, that you cannot paste an Activity as a Queue, or vice versa. The following description applies to each one of the three clipboards.

After cutting a record, it is possible to create a new node, or to renumber an existing node, using the *NodeNo* of the record stored on the clipboard. In this case, you will not be able to paste the contents of the clipboard because doing so would lead to two records with the same *NodeNo*. If you attempt to do so, you will receive a warning. The record stored on the clipboard is not lost and can be pasted in by renumbering the existing record to a different *NodeNo*.

A record can be pasted only once. After pasting, UM-CYCLONE clears the clipboard in order to ensure that no two nodes have the same *NodeNo*. Also, the clipboard can only hold one record at a time. If you cut a record and then cut a second record without pasting the first, the data of the first record will be lost. Only the data for the last record are saved on the clipboard. The contents of the clipboard are not lost if you clear the model data (by selecting "*New*" in the *File* menu), or when you open a new model file from disk (by selecting "*Open*" in the *File* menu). The clipboard contents, however, are not saved on disk when you save your model file and are lost when you exit the program.

Similar to the *PgUp* and *PgDn* keys, the *Del* and *Ins* keys can be used for cutting and pasting records only if UM-CYCLONE is not currently in *edit mode* (*i.e.*, not in data entry mode). In edit mode, these keys are used for character editing like in a word-processor. The *Del* key erases the character under the cursor and the *Ins* key toggles between *insert* and *overwrite* mode for character input.

The record shown on the screen can be deleted (without affecting the contents of the clipboard) by pressing *Ctrl-BackSpace* while not in edit mode. You will be prompted to verify that you really want to delete the current record. If you press "*Y*" (Yes) the record is erased and its data are lost; any other key cancels the operation and will bring you back to the data window.

Activity Window

The Activity Window includes the following data input fields:

Act No: Activity Number; unique integer > 0.

Description: Optional descriptive text up to 72 characters long (only 50 chars are visible at once).

Duration: Probability Distribution and Parameters. (The data in these four fields cannot be input directly. They are entered through the Duration Menu as explained below).

Predecessors: The *NodeNo*'s for up to 6 direct predecessors. The number zero indicates a blank field. Should be filled from left to right with no blank fields in between.

Successors: The *NodeNo*'s for up to 6 direct successors. The number zero indicates a blank field. Should be filled from left to right with no blank fields in between.

Probabilities: The associated successor probabilities if the Activity is followed by a probabilistic fork. The value of each field must be greater than or equal to 0 and less than 1. The sum of all six fields must be either 0 or 1. The number zero indicates a blank field. If all fields are blank then the links to successors are not a probabilistic fork: all successors are activated at the end of the Activity. Otherwise, the links are a probabilistic fork: the sum of all fields should be 1, and there must be a nonzero entry for every successor.

All the above fields are numeric initially set to zero. Exceptions are the *Description*, a text field initially set to blank, and the *Distribution*, an enumerated type field initially set to "*None*".

Each Activity must have at least one direct predecessor. Even though activities can have no successors, terminal activities are typically succeeded by a Queue that acts like a "*sink*".

An Activity can either be a *Combi* or a *Normal* Activity. Its particular type is not input directly, but is rather determined by the type of predecessors it has. If all direct predecessors are Queues then it is a Combi; otherwise, it is a Normal Activity. Notice that if one of the direct Activity predecessors is a Queue, then all direct predecessors for that Activity must be Queues. A set of direct predecessors for the same Activity cannot include Queues and some other kind of node. A valid simulation model must include at least one Combi Activity which must be preceded by at least one Queue. It is important to remember that UM-CYCLONE does not test for these rules during data input. It does, however, make a test during simulation.

The actual value of *ActNo* is significant for Combi Activities because it determines their relative priority when competing for the same resources. When two or more Combi Activities are preceded by the same Queue(s), the Combi Activity with the smallest *ActNo* has the highest priority, whereas the one with the largest *ActNo* has the lowest priority. This means that if all of these Combi Activities could potentially start at the same point during the simulation, the system will start the one with the highest priority first.

UM-CYCLONE will continue starting the same Activity with the highest priority (without advancing the simulation clock) for as long as its preceding Queues are not empty. When one of these Queues does become empty, the system will attempt to start the Activity with the next highest priority, and so on. Since these Combi Activities are directly preceded by the same Queue(s), however, it is quite possible that one or more of the Activities with low priority will not be able to start, because one or more of the common Queues have become empty by starting Activities with higher priority.

It is extremely important that Combi Activities be numbered very carefully because different numbering schemes can lead to dramatically different model behavior. Obviously, the incorrect numbering of Combi Activities cannot be detected by UM-CYCLONE, and is probably the most common source of modeling errors.

The duration for each Activity is described by a probability distribution selected from the Distribution Menu. To access this menu type *Alt-D* (or *F10, D*) while the cursor is pointing to any data field other than *ActNo*.

The Distribution Menu includes 6 probability distributions. The available distributions and their parameters are described below:

- *Deterministic* (Constant duration)
- *Uniform* (Lower Bound, Upper Bound)
- *Normal* (Mean, Standard Deviation)
- *Beta* (5th percentile, Mode, 95th percentile)
  (uses Perry & Greig formulas)
- *Gamma* (k, $1/\lambda$)
  Expected Value of the *Gamma* pdf = k / λ
  Variance of the *Gamma* pdf = k / λ^2
  k = No of IID *Exponential* random variables summed to give the *Gamma* pdf
  $1/\lambda$ = Expected value of the related *Exponential* pdf
- *Triangular* (Lower Bound, Mode, Upper Bound)

The parameters for all distributions are nonnegative real numbers. After you select the desired probability distribution and its parameters, the cursor will return to the Activity Window and your selection will be shown in the *Distribution* and the three *Parameter* fields. For distributions that have less than three parameters the unused *Parameter* fields as shown as zero(s).

It is important that you explicitly specify the distribution type for every defined Activity. For this reason, the default distribution type "*None*" is not an acceptable option and will result in an error during simulation.

It must be pointed out that the Distribution Menu can also be accessed from the Queue and Consolidation Node Windows, and from the *ActNo* field of the Activity Window. In these cases, the Distribution Menu is disabled. Even though it appears to function normally, it does not make any changes to the model data.

Queue Window

The Queue Window includes the following data input fields:

Queue No: Queue Number; unique integer > 0.

Description: Optional descriptive text up to 72 characters long (only 50 chars are visible at once).

Units at Start: Number of resource units stored in the Queue at the start of simulation (before the Queue's GEN function is applied).

Units GENerated: Number of resource units GENerated and stored in the Queue every time a direct predecessor node finishes.

All the above fields are numeric, initially set to zero, except for the *Description*, a text field initially set blank.

Notice that a Queue record does not include fields for predecessor or successor nodes. Instead of being explicitly defined, the links in and out of a Queue are inferred from the predecessor and successor lists of Activities and Consolidation Nodes. As a result, the same Queue may precede or succeed any number of nodes. As a minimum, a valid simulation network must include at least one properly-connected Queue.

The definition of a Queue requires only a *QueNo*. If the fields "*Units at start*" and "*Units GENerated*" are left blank (zero), then the Queue will start empty with GEN=1. It is impossible to define a Queue with GEN=0. Thus, the field "*Units GENerated*" is typically left blank (zero) for Queues with GEN=1. There is no need to explicitly set *Units GENerated* = 1.

The initial contents of a Queue at the start of simulation are specified in terms of *incoming* units in the field "*Units at start*". Immediately after the start of simulation the GEN function of the Queue is activated and the actual number of units in the Queue becomes equals to "*Units at start*" times GEN. As a result, the "*Units at start*" field does not actually represent the initial number of resource units in the Queue, but rather the number of incoming *start* signals that the Queue receives prior to the start of simulation.

Consolidation Node Window

The Consolidation Window includes the following data input fields:

Con. Node No: Consolidation Node Number; unique integer >0.

Units CONed: Number of units consolidated. The Consolidation Node finishes once for every CON number of *start signals* it receives.

Description: Optional descriptive text up to 72 characters long (only 50 chars are visible at once).

Predecessors: The *NodeNo*'s for up to 6 direct predecessors. The number zero indicates a blank field. Should be filled from left to right with no blank fields in between.

Successors: The *NodeNo*'s for up to 6 direct successors. The number zero indicates a blank field. Should be filled from left to right with no blank fields in between.

Probabilities: The associated successor probabilities if the Consolidation Node is followed by a probabilistic fork. The value of each field must be greater than or equal to 0 and less than 1. The sum of all six fields must be either 0 or 1. The number zero indicates a blank field. If all fields are blank then the links to successors are not a probabilistic fork: all successors are activated when the Consolidation Node *finishes*. Otherwise, the links are a probabilistic fork: the sum of all fields should be 1, and there must be a non-zero entry for every successor.

All the above fields are numeric, initially set to zero, except for the Description, a text field initially set blank.

Each Consolidation Node must have at least one direct predecessor, which can be an Activity or another Consolidation Node, but not a Queue. In order to be useful, a Consolidation Node must also have at least one direct successor (Activity, Queue, or Consolidation Node). In contrast to Queues and Activities, a valid simulation model need not include any Consolidation Nodes.

It is important that you explicitly specify the number of units consolidated, CON, for every defined Consolidation Node. For this reason, the default is CON = 0, which is not an acceptable option and will result in an error during simulation.

UM-CYCLONE MENUS

CYCLONE.EXE is a menu-driven program. Its five drop-down main menus are shown at the top of the screen in the Top Menu Line: *File*, *Control*, *Duration*, *Go!*, and *Quit*. Each one of these menus contains *Items* that may lead to submenus, data entry fields, or options that can be turned *On* and *Off*. Menus are accessed via the *F2*, *F10*, and the various *Alt*-key shortcuts. A complete list of the available menu options and keyboard commands appears below.

File Menu

The *File* Menu includes commands for changing the default directory, erasing the current model, reading and saving simulation models to and from disk files, viewing (output) files on the screen, printing files, and running a DOS shell. The available options are:

Change Dir...	Changes the default directory for opening and saving model files.
New...	Clears all data and starts a new simulation model.
Open...	Prompts with a directory listing and allows the selection of a disk file which is opened and read as a simulation model.
Save	Saves the current model to a file in the current default directory using the file name shown in the *Filename* field.
Save As...	Prompts for a new filename and uses it to save the current model in the current default directory.
View...	Prompts with a directory listing, allows the selection of a file, and shows its contents on the screen using the program LIST.COM.
Print...	Prompts with a directory listing, allows the selection of a file, and prints it using the batch file SIMPRINT.BAT (See SIMPRINT.BAT for further instructions).

DOS shell — Runs a DOS shell and allows you to enter DOS commands and run other programs. Type EXIT to return to UM-CYCLONE.

The commands *Open...*, *View...*, and *Print...* prompt the user for a *File Spec*, that may include the standard DOS wildcard characters "*" and "?", and produce a file list that includes only the files that match the file spec. For example, the file spec "*.*" produces a list that includes all files, while "*.*IN*" shows only the files with extension "*.IN*".

By default, UM-CYCLONE simulation files end in "*.IN*" (for INput), and output files end in "*.OUT*" (for OUTput). As a result, the default file spec for "*Open...*" is "*.*IN*" and the default file spec for "*View...*", and "*Print...*" is "*.*OUT*". You can produce other file lists by editing the default file specs.

The command *View...* runs the program LIST.COM as a child process and allows you to view any file in full screen mode. LIST.COM has its own set of keyboard commands and is described in detail in the file LIST.DOC. Pressing the *H* key while LIST.COM is active will bring up a quick help screen.

Instead of sending files directly to the printer, the command *Print...* runs the batch file SIMPRINT.BAT. The purpose of this batch file is to allow you to customize the printing procedure depending on your printer, and the computer port to which the printer is connected. Before printing a file you must use a word-processor or text-editor to edit SIMPRINT.BAT so that it issues the correct commands (escape sequences) for your printer. Instructions for using various kinds of printers are included within SIMPRINT.BAT.

Control Menu

The *Control* Menu is used for defining the global parameters of the simulation and includes the following options:

Model Title... — Optional text, up to 72 characters long, describing the simulation model.

Sim Time Limit... — Real Number > 0 used for controlling the duration of the simulation.

Production Limit... — Integer > 0 used for controlling the duration of the simulation.

For Activity... — The *ActNo* for the *Flagged Activity*.

RNG Seed... Integer from 0 to 2,147,483,647 used as Random Number Generator (RNG) *Seed*.

Debug... Leads to a submenu that allows you to activate options for debugging a UM-CYCLONE model.

The most important of these options are the *Simulation Time Limit*, the *Production Limit*, and the *Flagged Activity*. Together, these parameters control the duration of the simulation. A UM-CYCLONE simulation stops when either the simulation clock exceeds *Sim Time Limit*, or the number of times the *Flagged Activity* has finished equals its *Production Limit*. These parameters must be explicitly specified for any model. For this reason, their default values are zero, which is not an acceptable option and will result in an error during simulation.

The Random Number Generator (RNG) *Seed* can be any integer from 0 to 2,147,483,647. If it set equal to zero (the default), then the actual *Seed* is "*internal*". It is a function of the current value of the PC's internal clock (down to hundreds of a second) at the start of simulation. As a result, repeated simulation runs of the same model produce *different* results. If the *Seed* is set to any value other than zero it is considered to be "*external*". Repeated runs of the same model with the same "*external*" *Seed* produce *identical* results.

The *Debug...* command leads to a submenu with two items: *Data Map* and *Event Log*. You can turn either of these options ON or OFF by moving the cursor and pressing *Enter*. A small triangle on the left side of each option indicates that it is turned ON. When turned ON these options produce additional output in the simulation output file. The default setting for both options is OFF. You should leave it that way unless you have a specific reason for getting a "look behind the scenes".

Data Map produces a representation of the internal UM-CYCLONE data structures. This option is primarily used for program development and for explaining how UM-CYCLONE works. The average user does not need to activate this option.

Event Log produces the *Future Event List*, *i.e.*, the list of event times EET used for advancing the simulation clock. This list is usually very long and is of interest when trying to understand the simulation clock mechanism or when *tracing* the sequence of events followed by the simulation in order to *debug* the model.

Go! Menu

This menu includes only one option: *Run Simulation*. When you select this option UM-CYCLONE goes through the following sequence of steps:

- Saves the current model to a file on disk. If the current value of the *Filename* field is blank, you will be prompted for the name of the file in which to save your model.

- Detects the presence of an Intel 80x87 math coprocessor chip in your computer.

- Runs the appropriate version of the UM-CYCLONE simulator (CYC_87.EXE or CYC_NO87.EXE) as a child process.

- Shows the simulation results on screen using LIST.COM.

The UM-CYCLONE simulator is supplied in two versions: CYC_87.EXE and CYC_NO87.EXE. These programs are identical except for the way they perform floating-point operations (*i.e.*, calculations involving Real numbers). The reason for having two separate versions is to optimize execution performance depending on whether an Intel 80x87 math coprocessor is installed in the host computer. A math coprocessor is a specialized chip that performs floating-point operations up to 100 times faster than the PC's 80x86 general-purpose microprocessor.

CYC_87.EXE performs all real number calculations by redirecting them to an Intel 80x87 math coprocessor. If you try to run this program on a computer without a math coprocessor it will abort with the message "*floating point not loaded*". CYC_87.EXE has the smallest code size, produces the most accurate results, and yields the fastest execution times. Because UM-CYCLONE is a computation-intensive program it is strongly recommended that you run it on a system with a math coprocessor.

CYC_NO87.EXE is the smallest and fastest version for computers that do not have a math coprocessor. This version ignores a math coprocessor even if one is installed. Since this program performs all floating-point operations using software, it can run on any computer. However, it is about an order of magnitude slower than CYC_87.EXE and its results as not

accurate to as many significant digits. As a result, it is recommended that you use CYC_87.EXE if possible.

When you select the *Run Simulation* option, UM-CYCLONE detects the presence of a math coprocessor (by reading the *equipment flag* in your computer) and automatically runs the appropriate version of the simulator as a child process. At the end of the simulation the child process is terminated and you will return back to the *File View* menu option for examining the simulation output.

Even though it is not necessary, you can also run both versions of the simulator program directly from DOS by typing a command such as:

ProgramName InFile [*OutFile*]

where:

ProgramName: is either CYC_87 or CYC_NO87
InFile: the simulation model input file
OutFile: optional name for the output file.
(*Outfile* is optional. If *Outfile* is not given then *InFile*.OUT is used)

Both the input and the output files are text files (plain ASCII). (The input file can also be a specially formatted Lotus 1-2-3 worksheet. This feature is not described here.)

Quit Menu

This menu is used for exiting UM-CYCLONE. It includes the following two options:

No: Returns back to UM-CYCLONE.

Yes: Exits the program. Prompts you to save your model in a file before quitting. Make sure you do so!

KEYBOARD COMMANDS

UM CYCLONE has many keyboard commands, most of which are similar to those used by other IBM PC software. Pressing the *F1* key will provide you with context-sensitive help with keyboard commands and the expected input data. In addition, the last line on the screen lists

some of the most common keys and their current actions. The information on this line is also context-sensitive. Finally, you can see the available *Alt*-key shortcuts by pressing and holding the *Alt* key down for about a second.

Many keyboard keys function differently depending on the context in which they are used. The various keys and their respective actions are described below.

Global Function Keys

F1	Help
F2	*Pull/Pop Menus*. Switches between a previously accessed menu and the Data Entry Window. Remembers previous menu and field positions (has memory).
F10	Move cursor to the Top Menu Line. Similar to *Alt* in Windows.

Global *Alt*-Key Shortcuts

Alt-1	To Activity Window
Alt-2	To Queue Window
Alt-3	To Consolidation Node Window
Alt-F	(F)ile Menu
Alt-C	(C)ontrol Menu
Alt-D	(D)uration Menu
Alt-G	(G)o! Menu
Alt-Q	(Q)uit Menu
Alt-O	(O)pen command in *File* Menu
Alt-S	(S)ave command in *File* Menu
Alt-V	(V)iew command in *File* Menu
Alt-P	(P)rint command in *File* Menu

Top Menu Line

Esc	Return to Data Entry Window.
Left/Right Arrow	Move the cursor highlight Left/Right.
Enter	Select highlighted Menu.
Letter key	Select Menu with same highlighted letter.

Drop-Down Menus/Submenus

Esc	Return to Top Menu Line or to parent Menu.
Up/Down Arrow	Move cursor Up/Down within menu.
Enter	Select highlighted menu item.
Letter key	Select menu Item with same highlighted letter.
Left/Right Arrow	Move to next Menu on Left/Right (does not work for submenus)

Directory List

Enter	Selects highlighted file from directory list. Highlight bar defaults to a file that closely matches the entry in the "FileName" field, if any.
Letter key	Find file name with same first letter.
Up/Down Arrow	Move cursor up/down.
Home/End	Move cursor to upper/lower file name.
PgUp/PgDn	Move cursor up/down a page.
^Home/^End	Move cursor to top/bottom file name.
^PgUp/^PgDn	Move cursor to top/bottom page.

Data Entry Window

Esc	Switch to/from NodeNo (ActNo, QueNo, ConNo) field and rest of fields.
F9	Toggle: Insert New Rec./Edit Old RecNo.
PgUp/PgDn	Move to Previous/Next record.
Del	Cut current record and save on clipboard.
Ins	Paste record and clear clipboard.
Ctrl-BackSpace	Delete current record (clipboard is unaffected).
Left/Right Arrows	Move cursor to field on Left/Right
Up/Down Arrows	Move cursor to nearest field Up/Down
Home/Ctrl Left Arrow	Move to First field on the same row.
End /Ctrl Right Arrow	Move to Last field on the same row.
PgUp/Ctrl-Home	Move to First field in sequence.
*PgDn/Ctrl-E*nd	Move to Last field in sequence.
Tab/Shift Tab	Move to Next field in sequence.
Enter	Enter "edit mode".
Any non-extended key	Enter edit mode and begin data entry.

Data Entry in Numeric or Text Field (Edit Mode)

Any Key	Only valid numbers and symbols can be typed.
Enter	Enter data and move to next field.
Esc	Restore original data & exit edit mode.
^R, or ^U	Restore original data & stay in edit mode.
^A,^S,^D,^F,^G,^H,^Y	WordStar-type editing keys.

Left/Right Arrow	Move cursor Left/Right by one character.
Home/End	Move cursor to beginning/end of field.
^Left/^Right Arrow	Move cursor to beginning/end of field.
Ins	Toggle insert/overwrite character mode.
Del	Delete character under the cursor.
BackSpace	Delete character to the left of cursor.

Simulation Output

The output of a UM-CYCLONE simulation consists of two parts. The first part is a complete duplicate of the model description as input by the user, so that the model may be fully reconstructed from a printout of the output file. This part includes:

- Model Title
- Sim Time Limit
- Production Limit
- Flagged Activity
- Whether an internal data map is desired
- Whether a listing of event times (EET) is desired
- Random Number Generator Seed and whether it is internal or external
- Activity Data List
- Queue Data List
- Consolidation Node Data List

The second part of the output includes the simulation results and the collected statistics.

The simulation results first report the final values of the simulation clock and the iteration counter for the *Flagged Activity*, along with their respective limits. This information indicates which limit was reached first, causing the simulation to stop. In the following description of the system statistics, the final value of the simulation clock is represented by the symbol *TotalSimTime*.

The rest of the output provides useful statistics about each node in the simulated network. The statistics section consists of three tables, one for each node type, in the following order: Queues, Activities, and Consolidation Nodes. Each line in these tables shows statistics for one node. Nodes within each table are sorted in ascending order of their ID numbers. The node's description, if any, is shown on the preceding line. The various system statistics are described below.

Queue Statistics (*)

QUEUE: The Queue Number, *QueNo*.

UNITS @START: Number of resource units in Queue at the start of simulation.

UNITS @END: Number of resource units in Queue at the end of simulation.

OUTPUT: Number of resource units that departed the Queue.

AVERAGE: Time average of the number of resource units stored in the Queue during the simulation.

STD DEV: Standard deviation of the number of resource units stored in the Queue during the simulation.

Q>q: P[$Q > q$]; fraction of *TotalSimTime* during which the number of units in the Queue exceeded the value q.

(*) Note: The number of units in the above Queue statistics are given in terms of *incoming* units and not GENerated (*outgoing*) units. *I.e.*, units to which the GEN function has not been applied yet.

Activity Statistics

ACTIVITY: The Activity Number, *ActNo*.

OUTPUT: Number of times the Activity has finished.

GROSS_RATE: *Output* / *TotalSimTime*

NET_RATE: *Output* / (*LastFinishTime* - *FirstStartTime*)

AVGDUR: Average Activity duration.

SDDUR: Standard deviation of Activity duration

START1: Time of Activity's first start.

STARTN: Time of Activity's last start.

AVGINT: Average of the time intervals between successive Activity starts.

SDINT: Standard deviation of the time intervals between successive Activity starts.

Consolidation Node Statistics

CONSOL: The Consolidation Node Number, *ConNo*.

OUTPUT: Number of times the Consolidation Node *finished* (*i.e., consolidated*).

GROSS_RATE: *Output* / *TotalSimTime*

NET_RATE: *Output* / (*LastFinishTime* - *FirstStartTime*)

AVGDELAY: Average Consolidation Node delay. (*)

SDDLY: Standard deviation of Consolidation Node delay. (*)

START1: Time the Consolidation Node received the first *start signal*.

STARTN: Time the Consolidation Node received the last *start signal*.

AVGINT: Average of time intervals between the Consolidation Node's successive finish events.

SDINT: Standard deviation of time intervals between the Consolidation Node's successive finish events.

(*) Note: The *delay* in a Consolidation Node is the time span from the time the internal *start signal* counter becomes 1 (first *start signal*) to the next time the *start signal* counter reaches the value CON (next *finish*).

Other Queue Statistics

Most of the statistics reported by UM-CYCLONE are straightforward and can be used directly. Some useful Queue statistics, however, are not part of the output and must be derived from other simulation results. This section explains how to compute the *average waiting time per visit in a Queue*, and the *average of the total waiting time that resources spend in a Queue.*

Consider a Queue with at least one incoming link and one outgoing link. Resources enter and exit the queue during simulation. At any point in time, the number of units q in the Queue might be 0, 1, 2,..., *N*. Let $t(q)$ be the simulated time during which the Queue contained exactly q units, and T be the total duration of the simulation. Obviously,

$$T = \sum_{q=0}^{N} t(q)$$

The time average of the number of units stored in the Queue, reported by UM-CYCLONE under the heading "AVERAGE", is given by:

$$\bar{q} = \sum_{q=0}^{N} [q \frac{t(q)}{T}] = \sum_{q=0}^{N} [q \cdot f(q)]$$

where $f(q)$ is the fraction of the total simulation time when the Queue had exactly q units.

Every time a resource unit enters the Queue it remains there (waiting) until it is removed by the start of a following Combi. Thus, resource units *visit* the Queue (enter and exit), and each visit is associated with a certain waiting time w_v. The sum of all the waiting times over all visits equals

$$\sum_{v=1}^{V} w_v = \sum_{q=0}^{N} q \cdot t(q) = \bar{q} \cdot T$$

where V is the number of resource units that have exited the Queue plus the number of units that remain in the Queue at the end of the simulation:

$$V = \text{OUTPUT} + \text{UNITS @END.}$$

Then, the average waiting time per visit is:

$$\overline{w} = \frac{\overline{q} \cdot T}{V} = \frac{(\text{AVERAGE queue contents})(\text{TotalSimTime})}{(\text{OUTPUT} + \text{UNITS@END})}$$

Notice that $\overline{w}$ is the average duration of a *generic* visit, and not the average waiting time for a *particular* resource that may have visited the Queue. Since several resource units may enter the Queue, and each resource unit may visit the Queue a different number of times (each having a different duration), it is impossible to compute the average waiting time for a particular resource without tracking the flow of individual resource units through the simulation network. As a result, $\overline{w}$ must be interpreted as the average waiting time per visit of the *generic* (or *average*) resource unit.

Suppose that the number of *different* resource units that have visited the Queue at various points in time is known to be N. Obviously, N must be less than or equal to the number of visits V to account for multiple visits by the same resource unit. The total waiting time (total visiting time) for the i^{th} of the N resources is W_i. Each W_i is the sum of nv_i separate waiting times, where nv_i is the number of visits to the Queue by the i^{th} resource unit. If W_i and nv_i were known, then it would be possible to compute the average waiting time per visit for the i^{th} resource unit: W_i / nv_i.

Even though the individual W_i and nv_i are not reported by the simulation, it is possible to compute their sums and averages over all N resources.

The sum of the individual number of visits over all resources equals the total number of visits:

$$\sum_{i=1}^{N} nv_i = V$$

The average number of visits per resource is:

$$\overline{nv} = \frac{1}{N} \sum_{i-1}^{N} nv_i = \frac{V}{N}$$

The sum of the individual total waiting times over all N resources equals the sum of the waiting times for all visits V:

$$\sum_{i=1}^{N} W_i = \sum_{v=1}^{V} w_v = \sum_{q=o}^{N} q \cdot t(q) = \overline{q} \cdot T$$

The average total waiting time over all N resources is:

$$\overline{W} = \frac{1}{N}\sum_{i=1}^{N} W_i = \frac{\overline{q} \cdot T}{N} = \frac{\text{(AVERAGE queue contents)(TotalSimTime)}}{\text{(Number of different resources that visited the queue)}}$$

Notice that $\overline{W}$ can also be interpreted as the total waiting time for the *average* resource. As a result, this quantity can also be derived by multiplying the average waiting time per visit $\overline{w}$ times the average number of visits per resource $\overline{nv}$:

$$\overline{W} = \overline{w} \cdot \overline{nv} = \frac{\overline{q} \cdot T}{V} \cdot \frac{V}{N} = \frac{\overline{q} \cdot T}{N}$$

Other Queue statistics can be computed by similar calculations.

Credits

The UM-CYCLONE simulator is written in FORTRAN and was originally developed by R.I. Carr for the University of Michigan MTS operating system in 1984. Since then, P.G Ioannou modified the program for the IBM-PC family of computers, wrote the random number generators, the ASCII I/O routines, the various versions of Lotus 1-2-3 interfaces, and developed the current CYCLONE.EXE user interface.

CYCLONE.EXE consists of about 7000 lines of code written and compiled with Borland International's Turbo Pascal Version 5.0. CYC_87.EXE / CYC_NO87.EXE consist of about 2000 lines of code which have been compiled with Microsoft FORTRAN 5.0 using the medium model library (data accessed with 16-bit NEAR calls) and the 80x87 (inline) / Alternate math packages respectively.

If you have any suggestions, or if you have found any bugs, please contact P.G. Ioannou at the address shown on the first page.

UM-CYCLONE vs. Other Implementations

Currently, there exist three separate implementations of CYCLONE for microcomputers. In addition to UM-CYCLONE, there is INSIGHT developed at Stanford, and Micro-CYCLONE developed at Purdue. Both INSIGHT and Micro-CYCLONE are based on D.W. Halpin's original FORTRAN implementation of CYCLONE on a mainframe. INSIGHT is written in FORTRAN. Micro-CYCLONE is written in BASIC.

We have compared UM-CYCLONE's performance against INSIGHT using the "*Queuing*" and "*Piles*" examples supplied on the UM-CYCLONE system disk. The results on a 6 MHz IBM AT with a 80287 math coprocessor are as follows (not counting the time for data input/output):

Queuing Example: Time for 100 customers to leave the system

UM-CYCLONE	6 sec
INSIGHT	85 sec

Piles Example: Time to simulate construction of X footings

UM-CYCLONE	X=50:	14 sec
INSIGHT	X=20:	100 sec (*)

(*) INSIGHT crashed after 24 footings

We have not compared UM-CYCLONE with Micro-CYCLONE. In all likelihood, however, UM-CYCLONE is the smallest and fastest implementation. It should also be the easiest system to learn and use given its unique user interface of drop-down menus and windows.

References

A description of the original CYCLONE modeling system, as well as example network models, can be found in:

Halpin, D.W., and Woodhead, R.W. (1976). "*Design of Construction and Process Operations.*" John Wiley and Sons, Inc., New York, N.Y.

A description of the UM-CYCLONE elements, network structure, and simulation logic can be found in:

Ioannou, P.G. (1989). "*UM-CYCLONE Discrete Event Simulation System, Reference Manual*", Report UMCE-89-11, Dept. of Civil Engineering, University of Michigan, Ann Arbor, MI.

Information about the Perry and Greig approximations for the *Beta* and other probability distributions based on the Mode (or the Median) and the 5% and 95% percentiles is in:

Perry, C. and I.D. Greig (1975), "Estimating the Mean and Variance of Subjective Distributions in PERT and Decision Analysis," *Management Science*, Vol. 21, No. 12, August 1975.

Information about the *Gamma* distribution and its relationship to the *Erlang* and *Exponential* distributions can be found in:

Benjamin, J.R. and C.A. Cornell (1970). *Probability, Statistics and Decision for Civil Engineers*, 2nd Ed., McGraw-Hill, New York, NY, 1970.