

EPANET MULTI-SPECIES EXTENSION

USER'S MANUAL

By

Feng Shang
James G. Uber
University of Cincinnati
Cincinnati, OH 45221

Lewis A. Rossman
Water Supply and Water Resources Division
National Risk Management Research Laboratory
Cincinnati, OH 45268

NATIONAL RISK MANAGEMENT RESEARCH LABORATORY
NATIONAL HOMELAND SECURITY RESEARCH CENTER
OFFICE OF RESEARCH AND DEVELOPMENT
U.S. ENVIRONMENTAL PROTECTION AGENCY
CINCINNATI, OH 45268

DISCLAIMER

The information in this document has been funded wholly or in part by the U.S. Environmental Protection Agency (EPA). It has been subjected to the Agency's peer and administrative review, and has been approved for publication as an EPA document. Mention of trade names or commercial products does not constitute endorsement or recommendation for use.

The computer programs described in this manual are subject to copyright. They are free software that can be redistributed and/or modified under the terms of the GNU Lesser General Public License as published by the Free Software Foundation.

These programs are distributed in the hope that they will be useful, but without any warranty; without even the implied warranty of merchantability or fitness for a particular purpose. See the GNU Lesser General Public License for more details.

The authors and the U.S. Environmental Protection Agency are not responsible and assume no liability whatsoever for any results or any use made of the results obtained from these programs, nor for any damages or litigation that result from the use of these programs for any purpose.

FOREWORD

The U.S. Environmental Protection Agency is charged by Congress with protecting the Nation's land, air, and water resources. Under a mandate of national environmental laws, the Agency strives to formulate and implement actions leading to a compatible balance between human activities and the ability of natural systems to support and nurture life. To meet this mandate, EPA's research program is providing data and technical support for solving environmental problems today and building a science knowledge base necessary to manage our ecological resources wisely, understand how pollutants affect our health, and prevent or reduce environmental risks in the future.

The National Risk Management Research Laboratory is the Agency's center for investigation of technological and management approaches for reducing risks from threats to human health and the environment. The focus of the Laboratory's research program is on methods for the prevention and control of pollution to the air, land, water, and subsurface resources; protection of water quality in public water systems; remediation of contaminated sites and ground water; and prevention and control of indoor air pollution. The goal of this research effort is to catalyze development and implementation of innovative, cost-effective environmental technologies; develop scientific and engineering information needed by EPA to support regulatory and policy decisions; and provide technical support and information transfer to ensure effective implementation of environmental regulations and strategies.

On September 24, 2002 EPA formed the National Homeland Security Research Center (NHSRC). NHSRC manages, coordinates, and supports a variety of research and technical assistance efforts to enhance the security of the nation's water infrastructure, improve the effectiveness and efficiency of decontamination activities, and to expand the capacity and capabilities of laboratories to respond in the event of a nationally significant event. NHSRC's mission and research agenda were formulated to support EPA's homeland security role.

This Users Manual for the EPANET-MSX software package has been produced as a joint effort of the NRMRL and NHSRC. It is published and made available by EPA's Office of Research and Development to assist the user community and to link researchers with their clients.

Sally C. Gutierrez, Director
National Risk Management Research Laboratory

Jonathan Herrmann, Director
National Homeland Security Research Center

ACKNOWLEDGEMENTS

Partial support for the work described in this manual was provided by the U.S. Environmental Protection Agency's National Homeland Security Research Center under Contract Number EP-C-05-056 to Pegasus Technical Services, Inc.

The authors wish to acknowledge the contributions made by Robert Janke and Regan Murray of the EPA's National Homeland Security Research Center to the development of the EPANET-MSX software.

CONTENTS

1. INTRODUCTION.....	7
2. CONCEPTUAL FRAMEWORK.....	9
3. PROGRAM USAGE.....	17
4. INPUT FILE FORMAT	33
5. EXAMPLE REACTION SYSTEMS.....	51
APPENDIX A. MSX TOOLKIT FUNCTIONS	63
APPENDIX B. BINARY OUTPUT FILE FORMAT	101
APPENDIX C. MSX ERROR CODES	103

1. INTRODUCTION

EPANET is a widely used program for modeling the hydraulic and water quality behavior of drinking water distribution systems. Its water quality component is limited to tracking the transport and fate of just a single chemical species, such as fluoride used in a tracer study or free chlorine used in a disinfectant decay study. This manual describes how the original EPANET model has been extended to handle multiple interacting chemical species and how this capability has been incorporated into both a stand-alone executable program as well as a toolkit library of functions that programmers can use to build custom applications. This set of software tools is referred to as EPANET-MSX, where MSX stands for Multi-Species Extension.

Many water quality problems in distribution systems can only be analyzed by using a multi-species approach. Consider the following descriptive examples:

- Free chlorine disinfectant is lost in bulk solution due mainly to oxidation-reduction reactions involving HOCl and OCl^- and natural organic matter (NOM). The NOM itself is a heterogeneous mixture of organic compounds (e.g., humic and fulvic acids) of varying chemical characteristics. Current single-species models, however, must model free chlorine loss under the assumption that all other reactants are in excess and thus their concentrations can be considered constant. This limitation is responsible for the widespread observation that the water-specific decay rate constant of the common first-order model is not a constant at all, but rather varies significantly with chlorine dose (a clear indication of model structure error). The formation of regulated chlorination by-products, which result from free chlorine and NOM interactions, presents yet another set of reaction mechanisms involving multiple interacting species.
- Mono-, di-, and tri-chloramine result from interactions between free chlorine species and ammonia, and are increasingly used as residual disinfectants. These chloramines also interact with NOM, though the reactions are slower than those for free chlorine. Thus chloramine decay in distribution systems involves multiple interacting chemical species, which a single-species model is forced to simplify as a quasi-first order reaction. Further, ammonia may be produced by auto-decomposition of chloramines, which is of significant practical importance for understanding nitrification episodes in distribution systems and storage tanks. Nitrification models may need to consider attached-growth nitrifying biofilms, suspended nitrifying biomass, and the electron donor (ammonia), electron acceptor (oxygen), and carbon source that supports microbial growth.
- For the relatively common situation where more than one water source supplies a distribution system, current models are not able to represent meaningful differences in source water quality, as they relate to water quality evolution in the distribution system. Modelers must try to compensate for this limitation by assigning bulk decay rate coefficients to specific pipes, according to which source supplies them. Such an approach has obvious deficiencies when attempting to model distribution system zones where sources blend together, and these zones are sometimes the focus of water quality issues.

None of these examples can be accurately modeled by using the single-species capabilities of the current EPANET program. This shortcoming provides the motivation to extend EPANET so that it can model reaction systems of any level of complexity.

The following sections of this manual describe the conceptual framework used by EPANET-MSX to model multiple reacting species within a distribution system, provide instructions on how to use both the command line and toolkit versions of EPANET-MSX, give a complete description of the format of an MSX input file, and describe several example applications in detail. The appendices describe each function in the EPANET-MSX toolkit, the format of its binary output file, and the meaning of its error codes.

2. CONCEPTUAL FRAMEWORK

From a water quality modeling perspective, two significant physical phases exist within a water distribution system: a mobile bulk water phase and a fixed pipe surface phase. Bulk phase species are chemical or biological components that exist within the bulk water phase and are transported through the system with the average water velocity. Surface phase species are components that are attached or incorporated into the pipe wall and are thus rendered immobile. Figure 2.1 shows an example of bulk phase chlorine (HOCl) reacting with bulk phase NOM (natural organic matter) to produce a bulk phase disinfectant by-product (DBP), while also oxidizing ferrous iron to ferric iron in the fixed surface phase at the pipe wall.

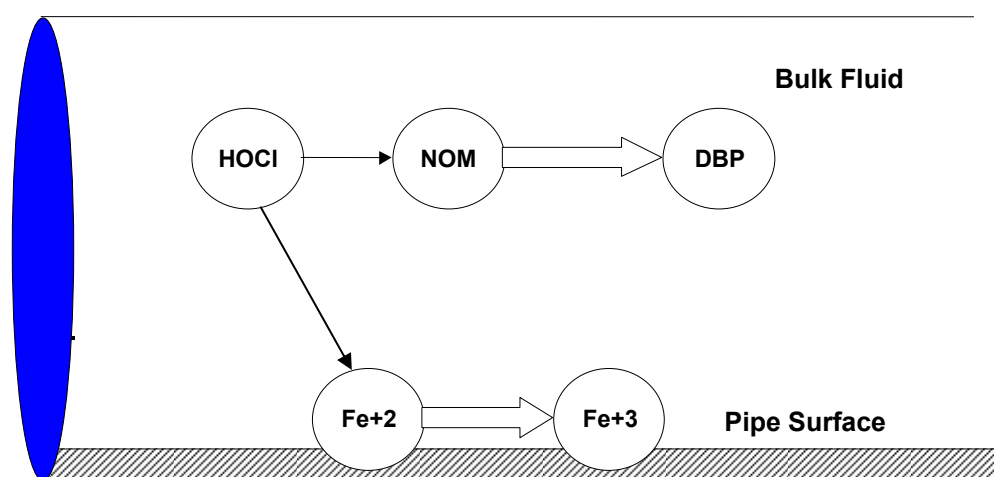


Figure 2.1 Example of reactions in the mobile bulk phase and at the fixed pipe surface phase.

Examples of bulk species include dissolved constituents (individual compounds or ions, such as HOCl and OCl^- , as well as aggregate components such as TOC), suspended constituents (such as bacterial cells and inorganic particulates), and chemicals adsorbed onto particles. Examples of surface species include bacteria incorporated within biofilm, oxidized forms of iron contained within corrosion scale, particulate material that settles out due to gravity or is attached to the pipe wall surface through ionic or molecular (i.e., van der Waal) forces, and organic compounds that can diffuse into or out of plastic pipes or be adsorbed onto or desorbed from iron oxide pipe surfaces. Some components, such as bacteria and particulates, can exist in both the bulk and surface phases and transfer from one phase to another by such mechanisms as physical attachment/detachment, chemical adsorption or molecular diffusion. In such situations, the component is modeled as two species: one bulk and the other surface.

Additional phases that might exist within a distribution system, such as a mobile bed sediment phase, an immobile water phase within the pore structure of pipe scale, or an air phase overlying the water surface in a storage tank, could also be included within this modeling framework.

Material Transport

A water distribution system consists of pipes, pumps, valves, fittings and storage facilities that convey water from source points to consumers. This physical system is modeled as a network of links connected together at nodes in some particular branched or looped arrangement. Figure 2.2 provides an example of a network representation of a very simple distribution system. Links represent pipes, pumps, and valves; nodes serve as source points, consumption points and storage facilities. The following phenomena all influence the quality of water contained in the system and can be modeled using principles of conservation of mass coupled with reaction kinetics:

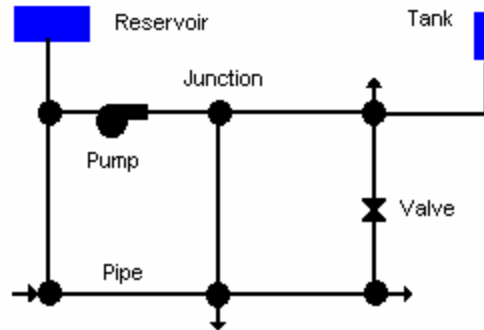


Figure 2.2 Node-link representation of a simple distribution system.

- a. *Advective transport in pipes*: bulk species are transported down the length of a pipe with the same average velocity as the carrier fluid while at the same time reacting with other bulk species and with the pipe wall surface.
- b. *Mixing at pipe junctions*: at junctions receiving inflow from two or more links the flows are assumed to undergo complete and instantaneous mixing.
- c. *Mixing in storage nodes*: all inflows to storage nodes mix completely with the existing contents in storage while these contents are subjected to possible bulk phase reactions (alternative schemes are available for modeling plug flow storage tanks).

Chemical Reactions

Reactions can be divided into two classes based on reaction rates. Some reactions are reversible and fast enough in comparison with the system's other processes so that a local equilibrium can be assumed; others are not sufficiently fast and/or irreversible and it is inappropriate to use an equilibrium formulation to represent them. Theoretically, very large backward and forward rate constants (with their ratio equaling the equilibrium constant) can be used to model fast/equilibrium reactions and therefore both fast/ equilibrium and slow/kinetic reaction dynamics can be written as a single set of ordinary differential equations (ODEs) that can be integrated over time to simulate changes in species concentrations. This approach can result in reaction rates that may range over several orders of magnitude and lead to such small integration time steps so as to make a numerical solution impractical.

In EPANET-MSX, algebraic equations are used to represent the fast/equilibrium reactions and mass conservation. Thus it is assumed that all reaction dynamics can be described by a set of differential-algebraic equations (DAEs) that is in semi-explicit format. The system of DAEs that defines the interactions between bulk species, surface species, and parameter values can be written in general terms as:

$$\frac{d\mathbf{x}_b}{dt} = \mathbf{f}(\mathbf{x}_b, \mathbf{x}_s, \mathbf{z}_b, \mathbf{z}_s, \mathbf{p}) \quad (1)$$

$$\frac{d\mathbf{x}_s}{dt} = \mathbf{g}(\mathbf{x}_b, \mathbf{x}_s, \mathbf{z}_b, \mathbf{z}_s, \mathbf{p}) \quad (2)$$

$$\mathbf{0} = \mathbf{h}(\mathbf{x}_b, \mathbf{x}_s, \mathbf{z}_b, \mathbf{z}_s, \mathbf{p}) \quad (3)$$

where the vectors of time-varying differential variables \mathbf{x}_b and \mathbf{x}_s are associated with the bulk water and pipe surface, respectively, the time-varying algebraic variables \mathbf{z}_b and \mathbf{z}_s are similarly associated, and the model parameters \mathbf{p} are time invariant. The algebraic variables are assumed to reach equilibrium in the system within a much smaller time scale compared to the numerical time step used to integrate the ODEs. The dimension of the algebraic equations \mathbf{h} must agree with that of the algebraic variables $\mathbf{z} = [\mathbf{z}_b \ \mathbf{z}_s]$, so that the total number of equations (1-3) equals the total number of time-varying species ($[\mathbf{x}_b \ \mathbf{x}_s \ \mathbf{z}_b \ \mathbf{z}_s]$).

As a simple example of a reaction/equilibrium system modeled as a set of DAE's consider the oxidation of arsenite (As-III) to arsenate (As-V) by a monochloramine disinfectant residual in the bulk flow and the subsequent adsorption of arsenate onto exposed iron on the pipe wall. This system consists of four species (arsenite, arsenate, and monochloramine in the bulk flow, and sorbed arsenate on the pipe surface). It can be modeled with three differential rate equations and one equilibrium algebraic equation:

$$\frac{d[\text{As}^{+3}]}{dt} = -k_a[\text{As}^{+3}][\text{NHCl}] \quad (4)$$

$$\frac{d[\text{As}^{+5}]}{dt} = k_a[\text{As}^{+3}][\text{NHCl}] \quad (5)$$

$$\frac{d[\text{NHCl}]}{dt} = -k_b[\text{NHCl}] \quad (6)$$

$$[\text{As}^{+5}]_s = \frac{k_s S_{\max} [\text{As}^{+5}]}{1 + k_s [\text{As}^{+5}]} \quad (7)$$

where $[\text{As}^{+3}]$ is the bulk phase concentration of arsenite, $[\text{As}^{+5}]$ is the bulk phase concentration of arsenate, $[\text{As}^{+5}]_s$ is surface phase concentration of arsenate, $[\text{NHCl}]$ is the bulk phase concentration of monochloramine, k_a is a rate coefficient for arsenite oxidation, k_b is a monochloramine decay rate coefficient, and k_s and S_{\max} are adsorption isotherm parameters. Thus in terms of the notation used in (1)-(3), $\mathbf{x}_b = \{\text{As}^{+3}, \text{As}^{+5}, \text{NHCl}\}$, $\mathbf{x}_s = \{\emptyset\}$, $\mathbf{z}_b = \{\emptyset\}$, $\mathbf{z}_s = \{\text{As}^{+5}_s\}$, $\mathbf{p} = \{k_a, k_b, k_s, S_{\max}\}$.

Full Network Solution

Dynamic models of water quality within water distribution systems can be classified spatially as either Eulerian or Lagrangian. Eulerian models divide the network into a series of fixed control elements and record the changes at the boundaries and within these elements, while Lagrangian models track changes of discrete parcels of water as they travel through the network. EPANET-MSX utilizes the same Lagrangian transport algorithm used by EPANET. This algorithm ignores axial dispersion and tracks the movement and reaction of chemicals in discrete water volumes, or segments. These segments are transported through network pipes by the bulk velocity, and completely mix at junction nodes. This method is relatively efficient because the number and size of the segments in a pipe can change as hydraulic conditions change.

In summary form, the following steps, depicted visually in Figure 2.3, are performed for each water quality time step:

1. *React*: Apply reaction dynamics within each pipe segment and storage tank over the time step to compute new concentrations throughout the network.
2. *Advect*: Within each pipe, compute the flow volume transported over the time step and transfer this amount of volume and its associated bulk species mass from the pipe's leading segments into accumulated mass and volume totals at the downstream node.
3. *Mix*: Compute new bulk species concentrations at each node based on its accumulated mass and volume inputs from the advection step as well as any external sources.
4. *Release*: Create a new segment at the upstream end of each pipe whose size equals the pipe's flow volume and whose bulk species concentrations equal that of the upstream node (or if the difference in quality between the most upstream segment and the upstream node is below some tolerance, simply increase the size of the current upstream segment).

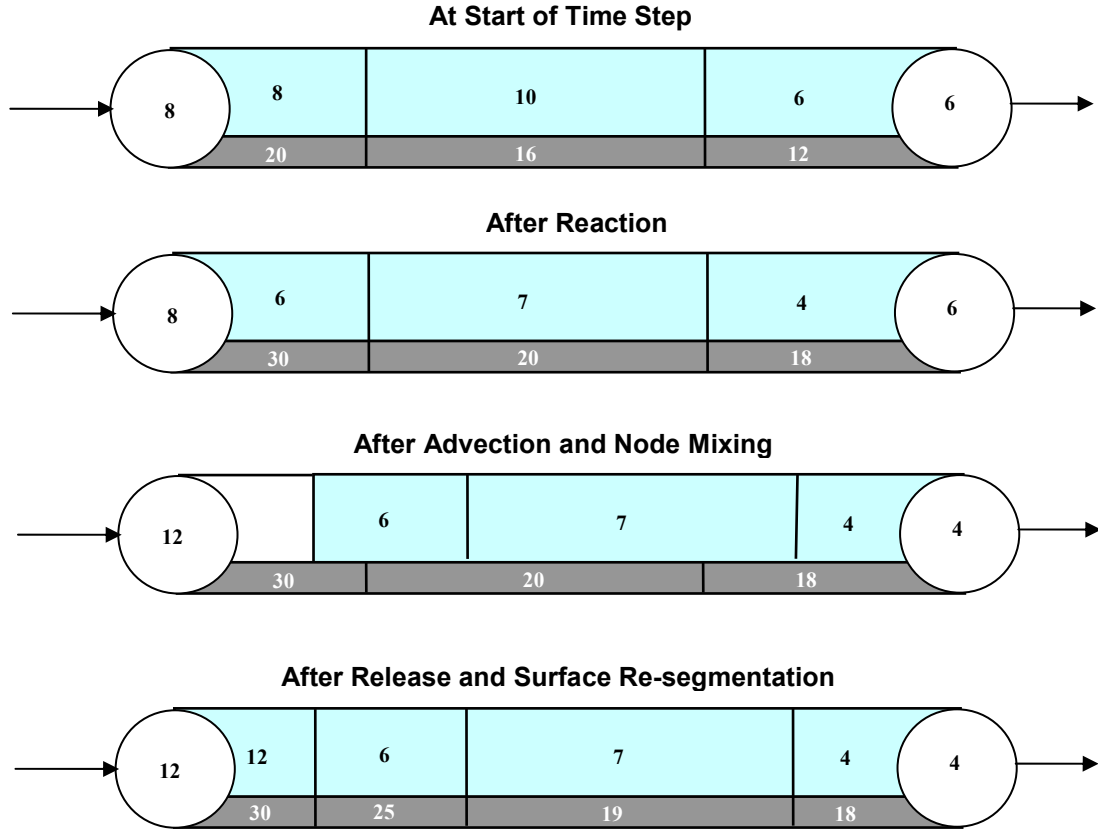


Figure 2.3 Illustration of the 4-step water quality transport method for pipe networks. The upper pipe segments contain flowing water while the lower segments are the pipe wall surface. The numbers in each segment represent hypothetical bulk and surface specie concentrations, respectively.

Reaction System Solution

The multi-species water quality algorithm modifies the *React* step (step 1) of the solution scheme described above. Within each pipe segment, specie reaction dynamics are represented by the system of DAEs (1)-(3). The same applies for storage tanks, except that the DAEs are modified to consider only bulk reactions. Although not indicated, the model parameters \mathbf{p} can possibly vary by pipe. For the equilibrium reactions it is assumed that the Jacobian matrix of \mathbf{h} with respect to \mathbf{z} , $\partial \mathbf{h} / \partial \mathbf{z}$, is unique and nonsingular for all t . In this case, the implicit functions defined by equation (3),

$$\mathbf{z}_b = \mathbf{z}_b(\mathbf{x}_b, \mathbf{x}_s, \mathbf{p}) \quad (8)$$

$$\mathbf{z}_s = \mathbf{z}_s(\mathbf{x}_b, \mathbf{x}_s, \mathbf{p}) \quad (9)$$

exist, are continuous and unique, and possess continuous partial derivatives. These properties, and in particular the resultant ability to evaluate (8)-(9) (numerically), are central to the numerical algorithms proposed for solution of (1)-(3).

Given the implicit functions (8)-(9), the solution of (1)-(3) is performed by substituting (8)-(9) into (1)-(2), thus eliminating the algebraic equations (3) and leaving a reduced system of ordinary differential equations (ODEs) that can be integrated numerically:

$$\begin{aligned}\frac{dx_b}{dt} &= \mathbf{f}(\mathbf{x}_b, \mathbf{x}_s, \mathbf{z}_b(\mathbf{x}_b, \mathbf{x}_s, \mathbf{p}), \mathbf{z}_s(\mathbf{x}_b, \mathbf{x}_s, \mathbf{p}), \mathbf{p}) \\ &= \mathbf{f}'(\mathbf{x}_b, \mathbf{x}_s, \mathbf{p})\end{aligned}\tag{10}$$

$$\begin{aligned}\frac{dx_s}{dt} &= \mathbf{g}(\mathbf{x}_b, \mathbf{x}_s, \mathbf{z}_b(\mathbf{x}_b, \mathbf{x}_s, \mathbf{p}), \mathbf{z}_s(\mathbf{x}_b, \mathbf{x}_s, \mathbf{p}), \mathbf{p}) \\ &= \mathbf{g}'(\mathbf{x}_b, \mathbf{x}_s, \mathbf{p})\end{aligned}\tag{11}$$

Note that the above “substitution” is not performed literally, since (8)-(9) are implicit, and thus so are the reduced trajectories \mathbf{f}' and \mathbf{g}' . Solving (10)-(11) numerically with an explicit method, such as any of the Runge-Kutta schemes, will require that \mathbf{f}' and \mathbf{g}' be evaluated at intermediate values of \mathbf{x}_b and \mathbf{x}_s over the integration time step. Each such evaluation will in turn require a solution of the nested set of algebraic equations (8)-(9). Alternative strategies for accomplishing these steps are discussed in the Model Implementation section below.

In addition to the *React* step, evaluation of the equilibrium equations also needs to be performed at the *Mix* phase of the overall algorithm since the blending together of multiple flow streams can result in a new equilibrium condition. This process needs to be performed at each network node, including storage tanks.

Pipe Surface Discretization

The segment bulk water state variables \mathbf{x}_b , \mathbf{z}_b have moving coordinates, due to the nature of the Lagrangian water quality model (they move with the bulk water velocity). In contrast the associated pipe surface variables \mathbf{x}_s , \mathbf{z}_s have fixed coordinates, since they are associated with the non-moving pipe. The lack of a common fixed coordinate system for the bulk and surface state variables must be reconciled, since these variables interact through the common pipe-water interface (through equations (1)-(3)). To resolve this issue a simple mass-conserving scheme is applied at every water quality time step to update the pipe surface elements to remain consistent with the (advected) water quality segments and re-distribute the surface variable mass among the updated elements.

As shown in Figure 2.3, within any single water quality time step, a moving mesh divides each pipe surface into discrete-length elements, such that each shares a common surface/water interface with the water quality segment above it. At the end of the time step the pipe elements will, however, be inconsistent with the water quality segments, due to advection of the latter (i.e., through the *Advect* step of the overall algorithm). This inconsistency is removed by updating the surface species concentrations using an interfacial area-weighted average:

$$\mathbf{x}_{si}^{new} = \left(\frac{1}{L_i^{new}} \right) \sum_{j=0}^n (L_i^{new} \cap L_j) \mathbf{x}_{sj} \quad \text{for } i = 1, \dots, n^{new} \quad (12)$$

$$\mathbf{z}_{si}^{new} = \left(\frac{1}{L_i^{new}} \right) \sum_{j=0}^n (L_i^{new} \cap L_j) \mathbf{z}_{sj} \quad \text{for } i = 1, \dots, n^{new} \quad (13)$$

where i is the water quality segment index, n is the number of water quality segments in the pipe during the most recent *React* step, L_j is the length of segment j , with corresponding vectors of surface species \mathbf{x}_{sj} and \mathbf{z}_{sj} , n^{new} is the updated number of water quality segments in the pipe after advection, L_i^{new} is the length of each updated segment, with corresponding updated surface concentrations \mathbf{x}_{si}^{new} and \mathbf{z}_{si}^{new} . The quantity $(L_i^{new} \cap L_j)$ is the length of the overlapping intersection between segment j and updated segment i .

Model Implementation

EPANET-MSX offers several choices of numerical integration methods for solving the reaction system's ODE's. These include a forward Euler method (as used in EPANET), a fifth order Runge-Kutta method with automatic time step control, and a second order Rosenbrock method with automatic time step control. These are listed in order of the numerical work required to obtain a solution. The Euler method is best applied to non-stiff, linear reaction systems, the Runge-Kutta method to non-stiff, nonlinear systems, and the Rosenbrock method to stiff systems.

The algebraic equilibrium equations (3) are solved using a standard implementation of the Newton method. This algorithm requires that the Jacobian of \mathbf{h} with respect to the algebraic variables \mathbf{z}_b and \mathbf{z}_s be used to iteratively solve an approximating linear system of equations until convergence is achieved. This can be a computationally expensive procedure since the Jacobian must be evaluated numerically and the system (3) is being solved within every pipe segment of every pipe at every time step, possibly several times over, as the ODE's are integrated. To help reduce this burden EPANET-MSX offers the following options for evaluating the nonlinear equilibrium equation system:

- The **Non-Coupled** option only evaluates the equilibrium equations at the end of the time step after a new ODE solution has been found; the algebraic variables maintain the values they had at the start of the time step while the ODEs are being numerically integrated.
- The **Fully-Coupled** option solves the algebraic equations at each stage of the ODE solution process using a fresh Jacobian for each Newton step.

The choice of coupling involves a trade-off between computational effort and level of accuracy, the degree of which will likely be very system dependent.

3. PROGRAM USAGE

EPANET-MSX is distributed in a compressed archive file named EPANETMSX.ZIP. The contents of this archive are listed in Table 3.1.

Table 3.1 Files distributed with EPANET-MSX.

File Name	Description
Readme.txt	A description of the files contained in the archive
license.txt	Licensing requirements on EPANET-MSX applications
epanetmsx.exe	Command line executable of EPANET-MSX
epanetmsx.dll	Dynamic link library of EPANET-MSX functions
epanet2.dll	Dynamic link library of standard EPANET functions
epanetmsx.h	C/C++ header file for EPANET-MSX functions
epanet2.h	C/C++ header file for standard EPANET functions
epanetmsx.lib	C/C++ library definition module for EPANET-MSX
epanet2.lib	C/C++ library definition module for standard EPANET
epanetmsx.bas	Visual Basic declarations of EPANET-MSX functions
epanet2.bas	Visual Basic declarations of standard EPANET functions
epanetmsx.pas	Delphi Pascal declarations of EPANET-MSX functions
epanet2.pas	Delphi Pascal declarations of standard EPANET functions
epanetmsx.pdf	PDF version of this manual
example.inp	EPANET input file for the example discussed in this chapter
example.msx	MSX input file for the example discussed in this chapter

An EPANET-MSX user must prepare two input files to run an analysis. One of these is a standard EPANET input file that describes the hydraulic characteristics of the network being analyzed (EPANET-MSX will ignore any water quality information that might be in this file). The format of this file is described in the EPANET Users Manual. Any network file that was created, edited and then exported from the Windows version of EPANET can serve as the EPANET input file for the multi-species extension.

The second file that must be prepared is a special EPANET-MSX file that describes the species being simulated and the chemical reaction/equilibrium model that governs their dynamics. The format of this file is described in Chapter 4 of this manual.

With these two input files EPANET-MSX can be run as a stand-alone console application or be used as a function library in custom-built applications. (At this point in time the extension has not been integrated into the Windows version of EPANET. This is expected to happen at some future date.) Examples of each type of usage will now be provided.

Example Command Line Run

In order to demonstrate the use of the command line version of EPANET-MSX we will simulate the arsenic oxidation/adsorption reaction system that was briefly described in Section 2 of this manual using the simple pipe network¹ shown in Figure 3.1. Table 3.2 lists the properties associated with the nodes of this network while Table 3.3 does the same for the pipe links.

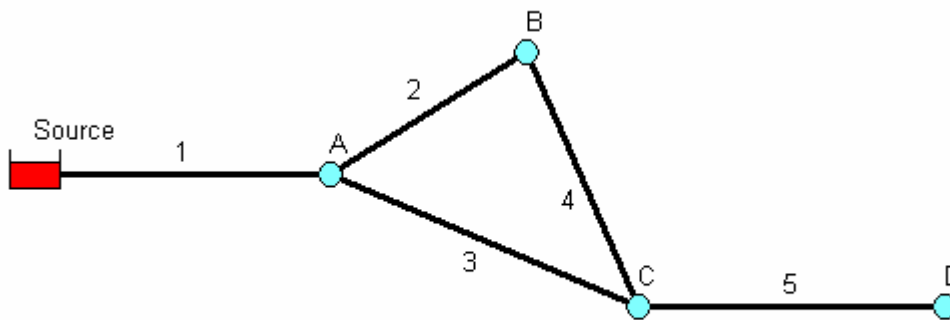


Figure 3.1 Schematic of the example pipe network.

Table 3.2 Nodal properties of the example pipe network

Node	Elevation, (m)	Demand (m ³ /hr)
Source	100	--
A	0	4.1
B	0	3.4
C	0	5.5
D	0	2.3

¹ This network is based on the example used in Zhang, W., Miller, C. T., and DiGiano, F. A., "Bacterial regrowth model for water distribution systems incorporating alternating split-operator solution technique", *Jour. Environmental Engineering*, 130 (9), 932-941, 2004.

Table 3.3 Pipe properties of the example pipe network

Pipe	Length (m)	Diameter (mm)	C-Factor
1	1000	200	100
2	800	150	100
3	1200	200	100
4	1000	150	100
5	2000	150	100

The first step in running a multi-species analysis of a water distribution system is to prepare a standard EPANET input file of the system that contains all of the information needed to perform a hydraulic analysis of the system. The Windows version of EPANET 2 was used to draw the network layout and assign node and pipe attributes using the program's graphical editing tools. A standard .INP file was then created by issuing the **File | Export | Network** command. The resulting file was named *example.inp* and is shown in Figure 3.2 (after some editing was performed to remove empty sections and default options). Note that for this simple application the water demands remain constant over time and that a 48 hour simulation period is requested.

```

[TITLE]
EPANET-MSX Example Network

[JUNCTIONS]
;ID      Elev  Demand  Pattern
A        0     4.1
B        0     3.4
C        0     5.5
D        0     2.3

[RESERVOIRS]
;ID      Head  Pattern
Source  100

[PIPES]
;ID      Node1  Node2  Length  Diameter  Roughness
1        Source A      1000    200     100
2        A      B      800     150     100
3        A      C     1200    200     100
4        B      C     1000    150     100
5        C      D     2000    150     100

[TIMES]
Duration              48
Hydraulic Timestep    1:00
Quality Timestep      0:05
Report Timestep       2
Report Start          0
Statistic             NONE

[OPTIONS]
Units                 CMH
Headloss              H-W
Quality               NONE

```

Figure 3.2 Contents of the *example.inp* input file.

The next step is to prepare the MSX input file that defines the individual water quality species of interest and the reaction expressions that govern their dynamics. This was done using a text editor, following the format described in Section 4 of this manual. The resulting MSX input file, named *example.msx*, is shown in Figure 3.3.

```

[TITLE]
Arsenic Oxidation/Adsorption Example

[OPTIONS]
AREA_UNITS M2 ;Surface concentration is mass/m2
RATE_UNITS HR ;Reaction rates are concentration/hour
SOLVER RK5 ;5-th order Runge-Kutta integrator
TIMESTEP 360 ;360 sec (5 min) solution time step
RTOL 0.001 ;Relative concentration tolerance
ATOL 0.0001 ;Absolute concentration tolerance

[SPECIES]
BULK AS3 UG ;Dissolved arsenite
BULK AS5 UG ;Dissolved arsenate
BULK AStot UG ;Total dissolved arsenic
WALL AS5s UG ;Adsorbed arsenate
BULK NH2CL MG ;Monochloramine

[COEFFICIENTS]
CONSTANT Ka 10.0 ;Arsenite oxidation rate coefficient
CONSTANT Kb 0.1 ;Monochloramine decay rate coefficient
CONSTANT Ks 5.0 ;Arsenate adsorption coefficient
CONSTANT Smax 50 ;Arsenate adsorption saturation limit

[PIPES]
RATE AS3 -Ka*AS3*NH2CL ;Arsenite oxidation
RATE AS5 Ka*AS3*NH2CL ;Arsenate production
RATE NH2CL -Kb*NH2CL ;Monochloramine decay
EQUIL AS5s Ks*Smax*AS5/(1+Ks*AS5) - AS5s ;Arsenate adsorption
FORMULA AStot AS3 + AS5 ;Total bulk arsenic

[TANKS]
RATE AS3 -Ka*AS3*NH2CL
RATE AS5 Ka*AS3*NH2CL
RATE NH2CL -Kb*NH2CL
FORMULA AStot AS3 + AS5

[QUALITY]
;Initial conditions (= 0 if not specified here)
NODE Source AS3 10.0
NODE Source NH2CL 2.5

[REPORT]
NODES C D ;Report results for nodes C and D
LINKS 5 ;Report results for pipe 5
SPECIE AStot YES ;Report results for each specie
SPECIE As5 YES
SPECIE As5s YES
SPECIE NH2CL YES

```

Figure 3.3 Contents of the *example.msx* input file.

There are several things of note in this file:

- The species have been named as follows:
 - AS3 is dissolved arsenite (As^{+3}), expressed in $\mu\text{g/L}$
 - AS5 is dissolved arsenate (As^{+5}), expressed in $\mu\text{g/L}$
 - AS_{tot} is total dissolved arsenic expressed in $\mu\text{g/L}$
 - AS5_s is adsorbed arsenate, expressed in $\mu\text{g/m}^2$
 - NH2CL is dissolved monochloramine, expressed in mg/L
- The reaction rate coefficients, K_a and K_b , and the adsorption coefficients, K_s and S_{max} , have been designated as constants. If instead they varied by pipe, then they could have been declared as parameters and their values could have been adjusted on a pipe-specific basis in the [PARAMETERS] section of the file.
- The [PIPES] section supplies the three reaction rate expressions and the single equilibrium expression for this system as was presented previously in Eqs. (4)-(7) of Section 2. For example, the rate expression for arsenite oxidation

$$\frac{d[\text{As}^{+3}]}{dt} = -k_a[\text{As}^{+3}][\text{NH}_2\text{Cl}]$$

is expressed in the file as:

```
RATE  As3  -Ka*As3*NH2CL
```

while the equilibrium expression

$$[\text{As}^{+5}]_s = \frac{k_s S_{\text{max}} [\text{As}^{+5}]}{1 + k_s [\text{As}^{+5}]}$$

is re-written so that it has an implied 0 on the left hand side:

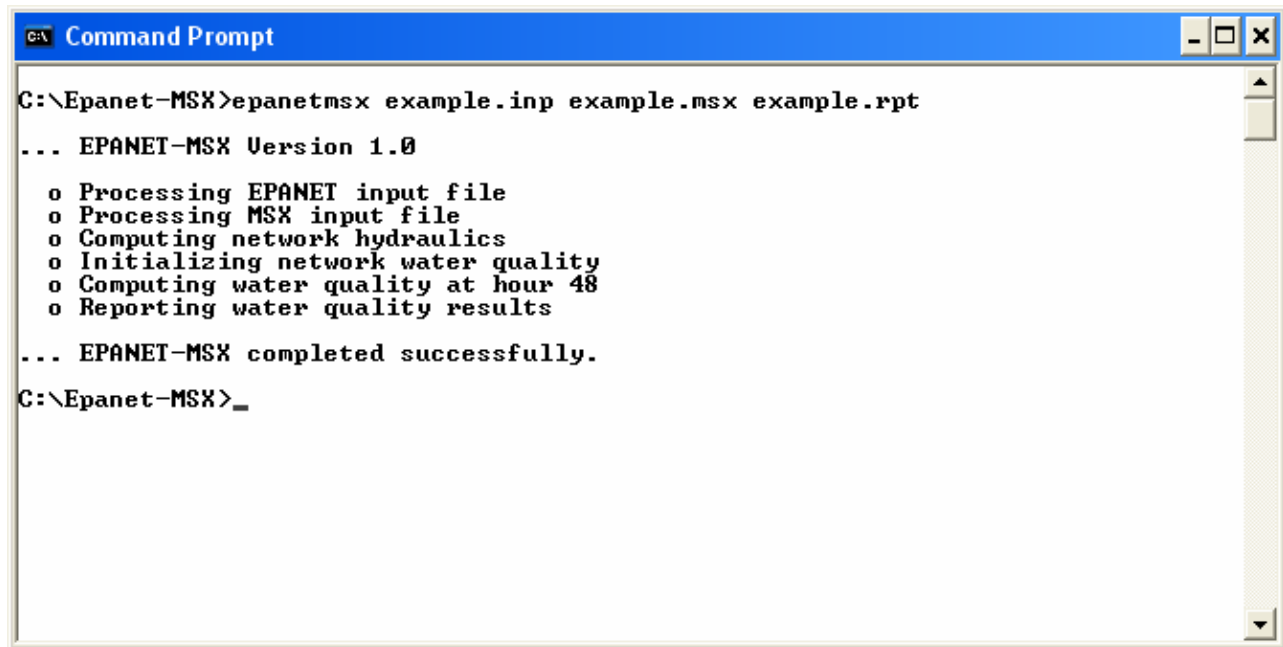
```
EQUIL  As5s  Ks*Smax*As5/(1+Ks*As5) - As5s
```

- Even though there are no tanks in this example, a [TANKS] section is still needed in the MSX file because both BULK and WALL species have been defined. If only BULK species were present then a redundant [TANKS] section would not be required.
- An initial quality is assigned to the source reservoir which remains constant over the course of the simulation. If source quality was to vary over time or there were source injections at other locations they could be described in a [SOURCES] section.
- In the [REPORT] section we ask that results for all species at nodes C and D and link 5 be written to the report file.

The final step in analyzing arsenic oxidation/adsorption for our example network is to run the EPANET-MSX command line executable. This can be done by first opening a Command Prompt window in Windows, navigating to the folder where epanetmsx.exe and the input files were saved, and issuing the following command:

```
epanetmsx example.inp example.msx example.rpt
```

where *example.rpt* is the name of the file where the results will be written.. If the executable were saved to a different folder than that of the example files, then either the full path name would have to be added to the name of the executable on the command line or the folder name would have to be added to the user's PATH environment variable. Figure 3.4 is a screen capture of what appears on the screen as the program runs.



```
C:\ Command Prompt

C:\Epanet-MSX>epanetmsx example.inp example.msx example.rpt
... EPANET-MSX Version 1.0
  o Processing EPANET input file
  o Processing MSX input file
  o Computing network hydraulics
  o Initializing network water quality
  o Computing water quality at hour 48
  o Reporting water quality results
... EPANET-MSX completed successfully.
C:\Epanet-MSX>_
```

Figure 3.4 Command line execution of Epanet-MSX.

After the program finishes the *example.rpt* file can be opened in any text editor (such as Windows Notepad) where its contents can be viewed. Excerpts from the results file are reproduced in Figure 3.5. The first page contains a summary of the standard EPANET options that were chosen for the run. Following this is a table of results for each node and each link. These tables contain the concentrations of each species at each reporting period. Note that the surface species are not listed for nodes since by definition this class of constituent is associated only with pipe surfaces.

```
*****
*                               E P A N E T                               *
*                               Hydraulic and Water Quality                 *
*                               Analysis for Pipe Networks                   *
*                               Version 2.0                                 *
*****
```

EPANET-MSX Example Network

```
Input Data File ..... example.inp
Number of Junctions..... 4
Number of Reservoirs..... 1
Number of Tanks ..... 0
Number of Pipes ..... 5
Number of Pumps ..... 0
Number of Valves ..... 0
Headloss Formula ..... Hazen-Williams
Hydraulic Timestep ..... 1.00 hrs
Hydraulic Accuracy ..... 0.001000
Maximum Trials ..... 40
Quality Analysis ..... None
Specific Gravity ..... 1.00
Relative Kinematic Viscosity ..... 1.00
Relative Chemical Diffusivity ..... 1.00
Demand Multiplier ..... 1.00
Total Duration ..... 48.00 hrs
Reporting Criteria:
  No Nodes
  No Links
```

Analysis begun Fri Feb 02 10:00:34 2007

Processing MSX input file example.msx

Figure 3.5 EPANET-MSX results for the example network (continued on next page).


```

*****
*               E P A N E T   -   M S X               *
*               Multi-Species Water Quality             *
*               Analysis for Pipe Networks              *
*               Version 1.0                             *
*****

```

Arsenic Oxidation/Adsorption Example

<<< Node C >>>

Time hr:min	As3 UG/L	As5 UG/L	NH2CL MG/L
0:00	0.00	0.00	0.00
2:00	0.00	0.00	0.00
4:00	0.00	0.00	0.00
6:00	0.00	0.00	0.00
8:00	0.00	9.14	1.10
10:00	0.00	9.14	1.10
12:00	0.00	9.14	1.10
14:00	0.00	9.14	1.10
16:00	0.00	9.14	1.10
18:00	0.00	9.14	1.10
20:00	0.00	9.14	1.10
22:00	0.00	9.14	1.10
24:00	0.00	9.14	1.10
26:00	0.00	9.14	1.10
28:00	0.00	9.14	1.10
30:00	0.00	9.14	1.10
32:00	0.00	9.61	1.10
34:00	0.00	10.00	1.11
36:00	0.00	10.00	1.11
38:00	0.00	10.00	1.11
40:00	0.00	10.00	1.11
42:00	0.00	10.00	1.11
44:00	0.00	10.00	1.11
46:00	0.00	10.00	1.11
48:00	0.00	10.00	1.11

<<< Node D >>>

Time hr:min	As3 UG/L	As5 UG/L	NH2CL MG/L
0:00	0.00	0.00	0.00

<Remaining entries not shown>

Figure 3.5 EPANET-MSX results for the example network (continued from previous page).

<<< Link 5 >>>				
Time hr:min	As3 UG/L	As5 UG/L	As5s UG/M2	NH2CL MG/L
0:00	0.00	0.00	0.00	0.00
2:00	0.00	0.00	0.00	0.00
4:00	0.00	0.00	0.00	0.00
6:00	0.00	0.00	0.00	0.00
8:00	0.00	0.39	2.15	0.05
10:00	0.00	1.58	8.51	0.17
12:00	0.00	2.77	14.88	0.27
14:00	0.00	3.96	21.25	0.35
16:00	0.00	5.15	27.62	0.42
18:00	0.00	6.34	33.99	0.47
20:00	0.00	7.53	40.36	0.52
22:00	0.00	8.72	46.72	0.55
24:00	0.00	9.14	48.93	0.56
26:00	0.00	9.14	48.93	0.56
28:00	0.00	9.14	48.93	0.56
30:00	0.00	9.14	48.93	0.56
32:00	0.00	9.15	48.93	0.56
34:00	0.00	9.26	48.94	0.56
36:00	0.00	9.37	48.95	0.57
38:00	0.00	9.48	48.96	0.57
40:00	0.00	9.59	48.98	0.57
42:00	0.00	9.70	48.99	0.57
44:00	0.00	9.82	49.00	0.57
46:00	0.00	9.93	49.01	0.57
48:00	0.00	10.00	49.02	0.57
Analysis ended Fri Feb 02 10:00:35 2007				

Figure 3.5 EPANET-MSX results for the example network (continued from previous page).

Example Toolkit Usage

Using the EPANET-MSX function library requires some programming effort to build custom applications and must be used in conjunction with the standard EPANET Programmer's Toolkit. Applications can be written in any programming language that can call external functions residing in a Windows DLL, such as C, C++, Visual Basic, and Delphi Pascal. Appendix A describes each of the functions that are included in the MSX toolkit library. The functions in the EPANET toolkit library are described in the Help file that is distributed with that toolkit (available at www.epa.gov/ORD/NRMRL/wswrd/epanet.html).

As an example of how the library might be used to construct a nontrivial application, Figure 3.6 lists the C code used to write a simple command line exposure calculator for a chemical released anywhere within a distribution system. Its command line arguments are the names of the EPANET network file and EPANET-MSX reaction file that describe the system being analyzed. The latter

file should characterize the fate of the injected chemical as it reacts with any normally occurring constituents in the water. The user is prompted for the name of the chemical being injected, and the location, duration, and mass flow rate at which it is injected. The program will then compute the fraction of the total average consumption in the service area that receives a dose of the chemical above some stipulated threshold concentration.

The entire program has been written as a single “main” function in order to simplify its presentation. As shown in Figure 3.6-a, the code begins with some “#include” statements that reference the “*epanet2.h*” and “*epanetmsx.h*” files. These files contain declarations of all of the EPANET and MSX toolkit functions, respectively, and are distributed with the EPANET-MSX download. After the program’s internal variables are declared, a check for the proper number of command line arguments (as passed into *main* by the “*argc*” variable) is made. Following this, the name of a temporary report file is obtained from the operating system by calling the C-library function *tmpnam*. This file is required to open the EPANET system, but will otherwise not be utilized by the exposure calculator.

The code listing continues in Figure 3.6-b by using a “while” statement to facilitate error checking throughout the remainder of the code. This is followed by calls to *ENopen* and *MSXopen* that open and read the EPANET and MSX systems and their input data files, respectively. The names of the data files are contained in the string array “*argv[]*” that is passed in through the command line. It is essential that the calls to these functions have their return values checked to insure that valid data sets are being used. If an error condition were to occur, then all subsequent calls to any of the EPANET or MSX toolkit functions would be ignored.

After the input files have been opened, a call to *MSXsolveH* is made to obtain a hydraulic solution for the network being analyzed. Once again, a check is made to insure that no errors have been encountered. The listing in Figure 3.6-b continues by using a series of screen prompts to obtain from the user the name of the chemical being injected, the injection location, the amount being injected, the duration of the injection, and the threshold concentration that constitutes an exposure. The remaining lines in this portion of the code define a source time pattern that lasts for 24 hours, where the pattern multipliers are 1.0 during the period of injection and are 0 otherwise. This pattern is used along with the other injection information to define an external mass injection source for the chemical of concern at the specified injection location.

The code listing continues in Figure 3.6-c by setting the initial concentration of the injected chemical to 0 throughout the network and then calling the *MSXinit* function to initialize a water quality simulation. The actual simulation is next carried out within a “do” loop that calls the *MSXstep* function at each time step until no more simulation time remains (or an error condition has been encountered). Within this loop, as new water quality conditions are computed at each time step, the concentration of the injected chemical for each network node is retrieved and is compared with the threshold exposure concentration. If it is above the threshold, then the initial quality of the chemical is set to 1. Using the initial quality to store exposures is simply a convenience, since that variable is only used to initialize the simulation at time 0 and provides a free storage area in which to record exposures after the simulation begins. Note that the *MSXstep* function automatically takes care of injecting the chemical of concern at the stipulated node of the system at the time periods recorded in source pattern that was created.

After the water quality simulation ends, the fraction of the total base water consumption that received an exposure is computed. Base consumption is being used as a surrogate for population since the latter is not part of the input data for EPANET. For each node, the EPANET toolkit function *ENgetnodevalue* is used to retrieve its base demand while the MSX toolkit function *MSXgetinitqual* retrieves the initial quality of the injected chemical for the node. Recall that the latter quantity will equal 1 if the node was exposed and be 0 otherwise. These values are used to update the total base demand and total exposed base demand, respectively. The ratio of these quantities is then computed and displayed as program output on the console.

The final portion of the source code shown in Figure 3.6-c closes up the error detection loop, displays any error messages that were generated, and calls the *MSXclose* and *ENclose* functions to shut down the MSX and EPANET processing systems, respectively. Note that the error reporting source code uses the fact that MSX error codes begin at 500 in order to determine whether the MSX or regular EPANET error message function should be called.

If this source code were saved to a file named *expcalc.c* then it could be compiled into an executable named *expcalc.exe* by using the following commands with the Microsoft C/C++ command line compiler:

```
CL /c expcalc.c  
LINK expcalc.obj epanet2.lib epanetmsx.lib /OUT:expcalc.exe
```

Note that for developing any MSX applications in C/C++, the library modules *epanet2.lib* and *epanetmsx.lib* must be linked in with the application's object modules. Versions of these files that are compatible with the Microsoft C/C++ compiler (Version 6 and higher) are supplied with the EPANET-MSX distribution.

```

// EXPCALC.C
// A console application that uses the EPANET and EPANET-MSX
// toolkits to compute the fraction of a distribution system's
// population exposed to a particular chemical release.

// The command line to launch the application is:
//   expcalc f1 f2
// where f1 is the name of the EPANET network data file,
// f2 is the name of the MSX reaction file.

#include <stdio.h>
#include "epanet2.h"
#include "epanetmsx.h"

int main(int argc, char *argv[]) {

// Define some variables
    int i; // general index
    int err = 0; // error code
    int nChems; // number of chemical species
    int nNodes; // number of nodes
    int srcChem; // source chemical index
    int srcNode; // source node index
    int srcPat; // source pattern index
    int srcStart; // injection start hour
    int srcEnd; // injection end hour
    long t; // elapsed time (sec)
    long tLeft; // time left (sec)
    char id[81]; // object id name
    char errMsg[81]; // error message text
    char *fRpt; // name of temporary report file
    double srcMass; // source inflow (mass/min)
    double cChem; // chemical concentration
    double cThresh; // exposure threshold
    double mult[24]; // pattern multipliers
    double baseDmnd; // nodal base demand
    double totDmnd; // total base demand
    double totExpDmnd; // total exposed demand
    double expFrac; // fraction of demand exposed

// Check for correct number of command line arguments
    printf("\nEPANET Exposure Calculator");
    printf("\n=====");
    if (argc < 3) {
        printf("\nToo few command line arguments.\n");
        return 0;
    }

// Create a temporary report file name
    fRpt = tmpnam(NULL);

```

Figure 3.6-a C source code for an EPANET exposure calculator (continued on next page)

```

// Start an error detection loop
while (err == 0) {

    // Open the Epanet and MSX input files
    err = ENopen(argv[1], fRpt, "");
    if (err == 0) err = MSXopen(argv[2]);
    if (err > 0) break;

    // Compute a hydraulic solution
    err = MSXsolveH();
    if (err > 0) break;

    // Identify which chemical is injected
    MSXgetcount(MSX_SPECIE, &nChems);
    printf("\nWhich chemical is being injected:");
    for (i=1; i<=nChems; i++) {
        MSXgetID(MSX_SPECIE, i, id);
        printf("\n  %2d - %s", i, id);
    }
    scanf("%d", &srcChem);
    if (srcChem <= 0 || srcChem > nChems) {
        printf("\nInvalid choice of chemicals.");
        break;
    }

    // Get index of injection node
    printf("\nWhat is the id of the injection node? ");
    scanf("%s", id);
    err = ENgetnodeindex(id, &srcNode);
    if (err > 0) {
        printf("\nThere is no such node in the network.");
        break;
    }

    // Get injection scenario
    printf("\nHow much mass per minute is injected? ");
    scanf("%f", &srcMass);
    printf("\nEnter first and last hour of injection: ");
    scanf("%d %d", &srcStart, &srcEnd);
    if (srcStart < 1) srcStart = 1;
    if (srcEnd > 24) srcEnd = 24;
    printf("\nWhat is the threshold exposure concentration? ");
    scanf("%f", &cThresh);

    // Setup the injection source
    err = MSXaddpattern("SourcePattern");
    if (err > 0) break;
    for (i=0; i<24; i++) mult[i] = 0.0;
    for (i=srcStart; i<=srcEnd; i++) mult[i-1] = 1.0;
    MSXgetindex(MSX_PATTERN, "SourcePattern", &srcPat);
    MSXsetpattern(srcPat, mult, 24);
    MSXsetsource(srcNode, srcChem, MSX_MASS, srcMass, srcPat);
}

```

Figure 3.6-b Continued from previous page.

```

// Initialize source specie concentration
ENgetcount(EN_NODE, &nNodes);
for (i=1; i<=nNodes; i++)
    MSXsetinitqual(MSX_NODE, i, srcChem, 0.0);

// Step through a water quality simulation
MSXinit(0);
do {
    // Advance one time step
    err = MSXstep(&t, &tleft);

    // Examine each network node
    for (i=1; i<=nNodes; i++) {

        // Record any exposure in the node's initial quality slot
        MSXgetqual(MSX_NODE, i, srcChem, &cChem);
        if (cChem > cThresh)
            MSXsetinitqual(MSX_NODE, i, srcChem, 1.0);
    }
} while (err == 0 && tleft > 0);
if (err > 0) break;

// Compute fraction of total base demand that was exposed
totDmnd = 0.0;
totExpDmnd = 0.0;
for (i=1; i<=nNodes; i++) {
    ENgetnodevalue(i, EN_BASEDEMAND, &baseDmnd);
    MSXgetinitqual(MSX_NODE, i, srcChem, &cChem);
    totDmnd = totDmnd + nodeDmnd;
    if (cChem > 0.0) totExpDmnd = totExpDmnd + baseDmnd;
}
expFrac = 0.0;
if (totDmnd > 0.0) expFrac = totExpDmnd / totDmnd;
printf("\nExposed Fraction = %.3f", expFrac);
break;

// End the error loop
}

// Printout any internal error condition
if (err >= 500) MSXgeterror(err, errMsg, 80);
else ENgeterror(err, errMsg, 80);
printf("\n%s\n", errMsg);

// Close the toolkits
MSXclose();
ENclose();

// Delete the temporary report file
remove(fRpt);
return err;
}

```

Figure 3.6-c Continued from previous page.

4. INPUT FILE FORMAT

The input file used by EPANET-MSX to describe the species and reaction system being modeled is organized into sections, where each section begins with a keyword enclosed in brackets. The various section keywords are listed in Table 4.1. Figure 4.1 contains a template of what the input file layout looks like.

Table 4.1 EPANET-MSX input file section keywords.

[TITLE]	adds a descriptive title to the data set
[OPTIONS]	sets the values of computational options
[SPECIES]	names the chemical species being analyzed
[COEFFICIENTS]	names the parameters and constants used in chemical rate and equilibrium expressions
[TERMS]	defines intermediate terms used in chemical rate and equilibrium expressions
[PIPES]	supplies the rate and equilibrium expressions that govern species dynamics in pipes
[TANKS]	supplies the rate and equilibrium expressions that govern species dynamics in storage tanks
[SOURCES]	identifies input sources (i.e., boundary conditions) for selected species
[QUALITY]	supplies initial conditions for selected species throughout the network
[PARAMETERS]	allows parameter values to be assigned on a pipe by pipe basis
[PATTERNS]	defines time patterns used with input sources
[REPORT]	specifies reporting options

Each section can contain any number of lines of data and appear in any order. Blank lines can appear anywhere in the file and the semicolon (;) can be used to indicate that what follows on the line is a comment, not data. A maximum of 1024 characters can appear on a line. The ID labels used to identify objects can be any combination of characters and numbers that do not contain square brackets ([]), double quotes or a semicolon.

On the pages that follow the contents and formats of each input file section are described in the order shown above. Reserved keywords are shown in bold and option choices are separated by slashes.

```

[TITLE]
  <title line>

[OPTIONS]
AREA_UNITS    FT2/M2/CM2
TIME_UNITS    SEC/MIN/HR/DAY
SOLVER        EUL/RK5/ROS2
TIMESTEP      <seconds>
ATOL          <value>
RTOL          <value>

[SPECIES]
BULK          <specieID>    <units>    (<atol> <rtol>)
WALL          <specieID>    <units>    (<atol> <rtol>)

[COEFFICIENTS]
PARAMETER     <paramID>     <value>
CONSTANT      <constID>     <value>

[TERMS]
<termID>        <expression>

[PIPES] or [TANKS]
EQUIL         <specieID>     <expression>
RATE          <specieID>     <expression>
FORMULA       <specieID>     <expression>

[SOURCES]
<type>          <nodeID>      <specieID>    <strength>    (<patternID>)

[QUALITY]
GLOBAL        <specieID>     <value>
NODE          <nodeID>       <bulkSpecieID> <value>
LINK          <linkID>       <wallSpecieID> <value>

[PARAMETERS]
PIPE          <pipeID>        <paramID>     <value>
TANK          <tankID>        <paramID>     <value>

[PATTERNS]
<patternID> <multiplier> <multiplier> ...

[REPORT]
NODES         ALL
NODES         <node1ID>      <node2ID>    ...
LINKS         ALL
LINKS         <link1ID>      <link2ID>    ...
SPECIE        <specieID>     YES/NO    (<precision>)
FILE          <filename>
PAGESIZE      <lines>

```

Figure 4.1 EPANET-MSX input file template.

[TITLE]

Purpose:

Attaches a descriptive title to the problem being analyzed.

Format:

A single line of text.

Remarks:

The [TITLE] section is optional.

[OPTIONS]

Purpose:

Defines various simulation options.

Formats:

AREA_UNITS	FT2/M2/CM2
TIME_UNITS	SEC/MIN/HR/DAY
SOLVER	EUL/RK5/ROS2
COUPLING	FULL/NONE
TIMESTEP	<i>seconds</i>
ATOL	<i>value</i>
RTOL	<i>value</i>

Definitions:

AREA_UNITS sets the units used to express pipe wall surface area where:

- FT2** = square feet
- M2** = square meters
- CM2** = square centimeters

The default is **FT2**.

TIME_UNITS is the units in which all reaction rate terms are expressed. The default units are hours (**HR**).

SOLVER is the choice of numerical integration method used to solve the reaction system where

- EUL** = standard Euler integrator
- RK5** = Runge-Kutta 5th order integrator
- ROS2** = 2nd order Rosenbrock integrator.

The default solver is **EUL**.

COUPLING determines to what degree the solution of any algebraic equilibrium equations is coupled to the integration of the reaction rate equations. If coupling is **NONE** then the solution to the algebraic equations is only updated at the end of each integration time step. With **FULL** coupling the updating is done whenever a new set of values for the differential variables in the reaction rate expressions is computed. This can occur at several intermediate times during the normal integration time step when using the **RK5** and **ROS2** integration methods. The default is **FULL** coupling.

TIMESTEP is the time step, in seconds, used to integrate the reaction system. The default time step is 300 seconds (5 minutes).

ATOL is the default absolute tolerance used to determine when two concentration levels of a species are the same. It applies to all species included in the model. Different values for individual species can be set in the **[SPECIES]** section of the input (see below). If no **ATOL** option is specified then it defaults to 0.01 (regardless of species concentration units).

RTOL is a default relative accuracy level on a specie's concentration used to adjust time steps in the **RK5** and **ROS2** integration methods. It applies to all species included in the model. Different values for individual species can be set in the **[SPECIES]** section of the input (see below). If no **RTOL** option is specified then it defaults to 0.001.

[SPECIES]

Purpose:

Defines each chemical species being simulated.

Formats:

BULK *name* *units* (*Atol* *Rtol*)

WALL *name* *units* (*Atol* *Rtol*)

Definitions:

name specie's name

units specie's mass units

Atol optional absolute tolerance that overrides the global value set in the **[OPTIONS]** section

Rtol optional relative tolerance that overrides the global value set in the **[OPTIONS]** section

Remarks:

- The first format is used to define a bulk water (i.e., dissolved) specie while the second is used for species attached (i.e., adsorbed) to the pipe wall.
- Bulk species are measured in concentration units of mass units per liter while wall species are measured in mass units per unit area.
- Any units can be used to represent species mass. The user is responsible for including any necessary unit conversion factors when specifying chemical reaction and equilibrium expressions that involve several species with different mass units.

Examples:

```
[SPECIES]
```

```
;Bulk chlorine in mg/L with default tolerances  
BULK CL2 MG
```

```
;Bulk biomass in ug/L with specific tolerances  
BULK Xb UG 0.0001 0.01
```

```
;Attached biomass in ug/area with specific tolerances  
WALL Xa UG 0.0001 0.01
```

[COEFFICIENTS]

Purpose:

Defines parameters and constants that are used in the reaction/equilibrium chemistry model.

Formats:

PARAMETER *name* *value*

CONSTANT *name* *value*

Definitions:

name coefficient's identifying name

value global value of the coefficient.

Remarks:

A **PARAMETER** is a coefficient whose value can be changed on a pipe by pipe (or tank by tank) basis (see the **[PARAMETERS]** section below) while a **CONSTANT** coefficient maintains the same value throughout the pipe network.

Examples:

```
[COEFFICIENTS]
;Kb can vary by pipe
PARAMETER Kb 0.1

;Kw is fixed for all pipes
CONSTANT Kw 1.5
```

[TERMS]

Purpose:

Defines mathematical expressions that are used as intermediate terms in the expressions for the chemical reaction/equilibrium model.

Formats:

termID *expression*

Definitions:

termID identifying name given to the term

expression any well-formed mathematical expression involving species, parameters, constants, hydraulic variables or other terms.

Remarks:

Terms can be used to simplify reaction rate or equilibrium expressions that would otherwise be unwieldy to write all on one line or have the same terms repeated in several different rate/equilibrium equations.

Hydraulic variables consist of the following reserved names:

D	pipe diameter (feet or meters)
Q	pipe flow rate (flow units)
U	pipe flow velocity (ft/sec or m/sec)
Re	flow Reynolds number
Us	pipe shear velocity (ft/sec or m/sec)
Ff	Darcy-Weisbach friction factor
Av	Area per unit volume (area units/L)

Examples:

```
[TERMS]
;A mass transfer coefficient
Kf  1.2e-4*Re^0.88/D

;A reaction term
a1  k1*HOCL*NH3
```


[PIPES]

Purpose:

Supplies the rate and equilibrium expressions that govern species dynamics in pipes.

Formats:

EQUIL	<i>specieID</i>	<i>expression</i>
RATE	<i>specieID</i>	<i>expression</i>
FORMULA	<i>specieID</i>	<i>expression</i>

Definitions:

<i>specieID</i>	a specie identifier
<i>expression</i>	any well-formed mathematical expression involving species, parameters, constants, hydraulic variables or terms.

Remarks:

- There should be one expression supplied for each species defined in the model.
- The allowable hydraulic variables were defined above in the description of the **[TERMS]** section.
- The **EQUIL** format is used for equilibrium expressions where it is assumed that the expression supplied is being equated to zero. Thus formally there is no need to supply the name of a species but doing so allows one to make sure that all species are accounted for.
- The **RATE** format is used to supply the equation that expresses the rate of change of the given specie with respect to time as a function of the other species in the model.
- The **FORMULA** format is used when the concentration of the named specie is a simple function of the remaining species.

Examples:

```
[PIPES]
;Bulk chlorine decay
RATE CL2 -Kb*CL2

;Adsorption equilibrium between Cb in bulk and Cw on wall
EQUIL Cw Cmax*k*Cb / (1 + k*Cb) - Cw

;Conversion between biomass (X) and cell numbers (N)
FORMULA N log10(X*1.0e6)
```

[TANKS]

Purpose:

Supplies the rate and equilibrium expressions that govern species dynamics in storage tanks.

Formats:

EQUIL	<i>specieID</i>	<i>expression</i>
RATE	<i>specieID</i>	<i>expression</i>
FORMULA	<i>specieID</i>	<i>expression</i>

Definitions:

<i>specieID</i>	a specie identifier
<i>expression</i>	any well-formed mathematical expression involving species, parameters, constants, or terms.

Remarks:

- There should be one expression supplied for each bulk species defined in the model. By definition, wall species are assumed to not exist on tank walls.
- Hydraulic variables are associated only with pipes and cannot appear in tank expressions.
- The **EQUIL** format is used for equilibrium expressions where it is assumed that the expression supplied is being equated to zero. Thus formally there is no need to supply the name of a species but doing so allows one to make sure that all species are accounted for.
- The **RATE** format is used to supply the equation that expresses the rate of change of the given specie with respect to time as a function of the other species in the model.
- The **FORMULA** format is used when the concentration of the named specie is a simple function of the remaining species.
- If no **[TANKS]** section is provided then the reaction/equilibrium expressions for bulk species defaults to those given in the **[PIPES]** section.

Examples:

See the examples listed for the **[PIPES]** section.

[SOURCES]

Purpose:

Defines the locations where external sources of particular species enter the pipe network.

Formats:

sourceType *nodeID* *specieID* *strength* (*patternID*)

Definitions:

<i>sourceType</i>	either MASS , CONCEN , FLOWPACED , or SETPOINT
<i>nodeID</i>	the ID label of the network node where the source is located
<i>specieID</i>	a bulk species identifier
<i>strength</i>	the baseline mass inflow rate (mass/minute) for MASS sources or concentration (mass/L) for all other source types
<i>patternID</i>	the name of an optional time pattern that is used to vary the source strength over time.

Remarks:

- Use one line for each species that has non-zero source strength.
- Only bulk species can enter the pipe network, not wall species.
- The definitions of the different source types conform to those used in the original EPANET program are as follows:
 - A **MASS** type source adds a specific mass of specie per unit of time to the total flow entering the source node from all connecting pipes.
 - A **CONCEN** type source sets the concentration of the specie in any external source inflow (i.e., a negative demand) entering the node. The external inflow must be established as part of the hydraulic specification of the network model.
 - A **FLOWPACED** type source adds a specific concentration to the concentration that results when all inflows to the source node from its connecting pipes are mixed together.
 - A **SETPOINT** type source fixes the concentration leaving the source node to a specific level as long as the mixture concentration of flows from all connecting pipes entering the node is less than the setpoint concentration.

Examples:

[SOURCES]

```
;Inject 6.5 mg/minute of chemical X into Node N1  
;over the period of time defined by pattern PAT1  
MASS  N1  X  6.5  PAT1
```

```
;Maintain a 1.0 mg/L level of chlorine at node N100  
SETPOINT  N100  CL2  1.0
```

[QUALITY]

Purpose:

Specifies the initial concentrations of species throughout the pipe network.

Formats:

GLOBAL	<i>specieID</i>	<i>concen</i>	
NODE	<i>nodeID</i>	<i>specieID</i>	<i>concen</i>
LINK	<i>linkID</i>	<i>specieID</i>	<i>concen</i>

Definitions:

<i>specieID</i>	a specie identifier
<i>nodeID</i>	a network node ID label
<i>linkID</i>	a network link ID label
<i>concen</i>	a species concentration

Remarks:

- Use as many lines as necessary to define a network's initial condition.
- Use the **GLOBAL** format to set the same initial concentration at all nodes (for bulk species) or within all pipes (for wall species).
- Use the **NODE** format to set an initial concentration of a bulk species at a particular node.
- Use the **LINK** format to set an initial concentration of a wall species within a particular pipe.
- The initial concentration of a bulk species within a pipe is assumed equal to the initial concentration at the downstream node of the pipe.
- If not specified, initial concentrations are assumed equal to zero.
- It may be necessary to provide reasonable initial conditions for chemistry models that involve equilibrium expressions in order that the equations remain solveable.

Examples:

```
[QUALITY]
;Set concentration of bulk species Cb to 1.0 at all nodes
GLOBAL  Cb  1.0

;Override above condition for node N100
NODE 100 Cb 0.5
```

[PARAMETERS]

Purpose:

Defines values for specific reaction rate parameters on a pipe by pipe or tank by tank basis.

Formats:

PIPE *pipeID* *paramID* *value*

TANK *tankID* *paramID* *value*

Definitions:

<i>pipeID</i>	the ID label of a pipe link in the network
<i>tankID</i>	the ID label of a tank node in the network
<i>paramID</i>	the name of one of the reaction rate parameters listed in the [COEFFICIENTS] section
<i>value</i>	the parameter's value used for the specified pipe or tank.

Remarks:

- Use one line for each pipe or tank whose parameter value is different than the global value.

[PATTERNS]

Purpose:

Defines time patterns used to vary external source strength over time.

Formats:

name multiplier multiplier ...

Definitions:

name an identifier assigned to the time pattern

multiplier a multiplier used to adjust a baseline value

Remarks:

- Use one or more lines for each time pattern included in the model.
- If extending the list of multipliers to another line remember to begin the line with the pattern name.
- All patterns share the same time period interval as defined in the **[TIMES]** section of the EPANET input file being used in conjunction with the EPANET-MSX input file.
- Each pattern can have a different number of time periods.
- When the simulation time exceeds the pattern length the pattern wraps around to its first period.

Examples:

```
[PATTERNS]
;A 3-hour injection pattern over a 24 hour period
;(assuming a 1-hour pattern time interval is in use)
P1 0.0 0.0 0.0 0.0 1.0 1.0
P1 1.0 0.0 0.0 0.0 0.0 0.0
P1 0.0 0.0 0.0 0.0 0.0 0.0
P1 0.0 0.0 0.0 0.0 0.0 0.0
```

[REPORT]

Purpose:

Describes the contents of the output report produced from a simulation.

Formats:

NODES	ALL
NODES	<i>node1 node2 ...</i>
LINKS	ALL
LINKS	<i>link1 link2 ...</i>
SPECIE	<i>specieID</i> YES/NO (<i>precision</i>)
STATISTIC	NONE / AVERAGE / MINIMUM / MAXIMUM / RANGE
FILE	<i>filename</i>
PAGESIZE	<i>lines</i>

Definitions:

<i>node1,</i> <i>node2, etc.</i>	a list of nodes whose results are to be reported
<i>link1,</i> <i>link2, etc.</i>	a list of links whose results are to be reported
<i>specieID</i>	the name of a specie to be reported on
<i>precision</i>	number of decimal places used to report a specie's concentration
<i>filename</i>	the name of a file to which the report will be written
<i>lines</i>	the number of lines per page to use in the report.

Remarks:

- Use as many **NODES** and **LINKS** lines as it takes to specify which locations get reported. The default is not to report results for any nodes or links.
- Use the **SPECIE** line to specify which species get reported and at what precision. The default is to report all species at two decimal places of precision.
- The **FILE** line is used to have the report written to a specific file. If not provided the report will be written to the same file used for reporting program errors and simulation status.

Examples:

```
[REPORT]
;Write results for all species at all nodes and links
;at all time periods to a specific file
NODES ALL
LINKS ALL
FILE "c:\my files\epanet-msx\myreport.txt"
```

```
[REPORT]
;Write average nodal results for species S1 and S2 using
;4 decimal places to the standard EPANET report file
SPECIE S1 YES 4
SPECIE S2 YES 4
NODES ALL
STATISTIC AVERAGE
```


5. EXAMPLE REACTION SYSTEMS

This section demonstrates how several different multi-species reaction systems of interest can be modelled with EPANET-MSX.

Multi-Source Chlorine Decay

Multi-source networks present problems when modeling a single species, such as free chlorine, when the decay rates observed in the source waters vary quite significantly. As the sources blend differently throughout the network it becomes difficult to assign a single decay coefficient that accurately reflects the decay rate observed in the blended water. Consider the distribution system shown in Figure 5.1 that is served by two different sources. The network has been color-coded to show the average fraction of water in each pipe that originates from the River (Source 1).

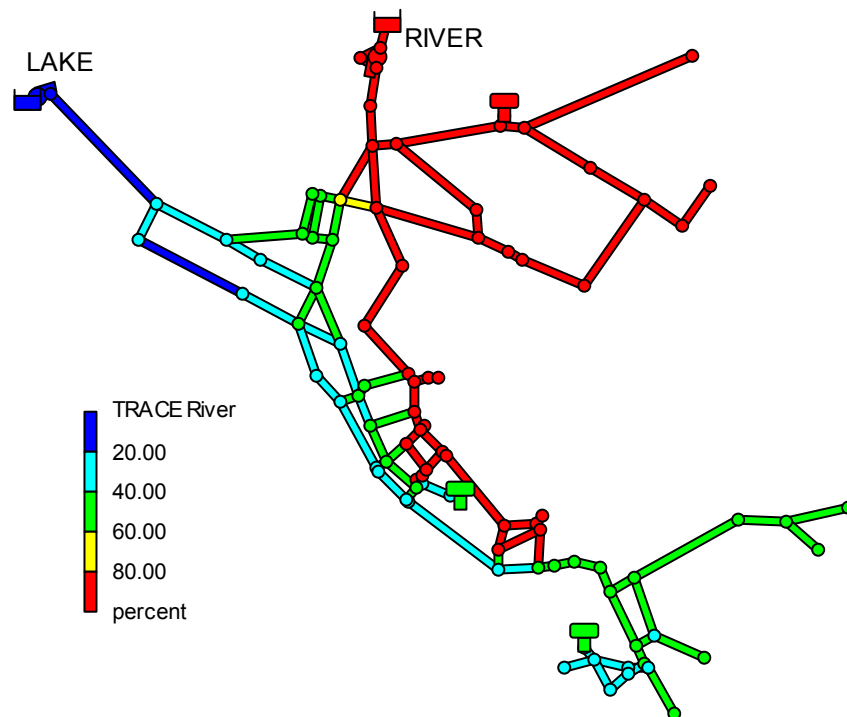


Figure 5.1 Example of a two-source water distribution system showing the average percent of water originating from the River source.

Assume that free chlorine reacts in the bulk flow along a pipe according to the following first-order rate expression:

$$\frac{dC}{dt} = -kC$$

where C is the concentration of free chlorine at time t and k is a reaction rate constant. Now suppose that when analyzed separately in bottle tests, water from Source 1 has a $k = 1.3 \text{ days}^{-1}$ while Source 2's water has $k = 17.7 \text{ days}^{-1}$. The issue becomes one of determining a k-value for each pipe of the network that will reflect the proper reactivity of the blended water from both sources.

One approach to reconciling the vastly different chlorine decay constants in this example, without introducing a more complex chlorine decay mechanism that attempts to represent the different reactivity of the TOC from the two sources, is to assume that at any instance the chlorine decay constant within a pipe is given by a weighted average of the two source values, where the weights are the fraction of each source water present in the pipe. These fractions can be deduced by introducing a fictitious tracer compound at Source 1, denoted as T1, whose concentration is fixed at a constant 1.0 mg/L. Then at any point out in the network the fraction of water from Source 1 would be the concentration of T1 while the fraction from Source 2 would be 1.0 minus that value. The resulting chlorine decay model now consists of two-species -- a tracer species T1 and a free chlorine species CL2. The first-order decay constant k for any pipe in the system would be given by

$$k = 1.3 \text{ T1} + 17.7 (1.0 - \text{T1}) \quad (14)$$

while the system dynamics would be expressed by:

$$\frac{dT1}{dt} = 0 \quad (15)$$

$$\frac{dCL2}{dt} = -(1.3 \text{ T1} + 17.7 (1.0 - \text{T1})) \text{CL2} \quad (16)$$

Figure 5.2 is the MSX input file that defines this model for a network where the two source nodes are represented as reservoirs with ID names "1" and "2", respectively. Note that it contains no surface species, no equilibrium species, and assumes that a constant chlorine concentration of 1.2 mg/L is maintained at each source.

```

[SPECIES]
BULK  T1      MG      ;Source 1 tracer
BULK  CL2     MG      ;Free chlorine

[COEFFICIENTS]
CONSTANT  k1  1.3      ;Source 1 decay coeff.
CONSTANT  k2  17.7     ;Source 2 decay coeff.

[PIPES]
;T1 is conservative
RATE  T1      0

;CL2 has first order decay
RATE  CL2  -(k1*T1 + k2*(1-T1))*CL2

[QUALITY]
;Initial conditions (= 0 if not specified here)
NODE   1      T1      1.0
NODE   1      CL2     1.2
NODE   2      CL2     1.2

```

Figure 5.2 MSX input file for modeling two-source chlorine decay.

Oxidation, Mass Transfer, and Adsorption

This example is an extension of the arsenic oxidation/adsorption model that was presented previously in Sections 2 and 3 of this manual. It models the oxidation of arsenite (As^{+3}) to arsenate (As^{+5}) by a monochloramine disinfectant residual (NH_2Cl) in the bulk flow along with the subsequent adsorption of arsenate onto exposed iron on the pipe wall. However we now introduce a mass transfer limitation to the rate at which arsenate can migrate to the pipe wall where it is adsorbed. Figure 5.3 shows a schematic of how this version of the arsenic model operates. Note that after arsenate is produced by the oxidation of arsenite in the bulk solution it diffuses through a boundary layer reach a concentration denoted as $(\text{As}^{+5})_w$ just adjacent to the pipe wall. It is this concentration that exists in equilibrium with adsorbed arsenate $(\text{As}^{+5})_s$ on the pipe wall. Thus the system contains four species (dissolved arsenite in bulk solution, dissolved arsenate in bulk solution, monochloramine in bulk solution, dissolved arsenate just adjacent to the pipe wall surface and sorbed arsenate on the pipe surface). One might argue that arsenate is really a single species that appears in three different forms (bulk dissolved, wall dissolved, and wall sorbed), but for purposes of modeling it is necessary to distinguish each form as a separate species.

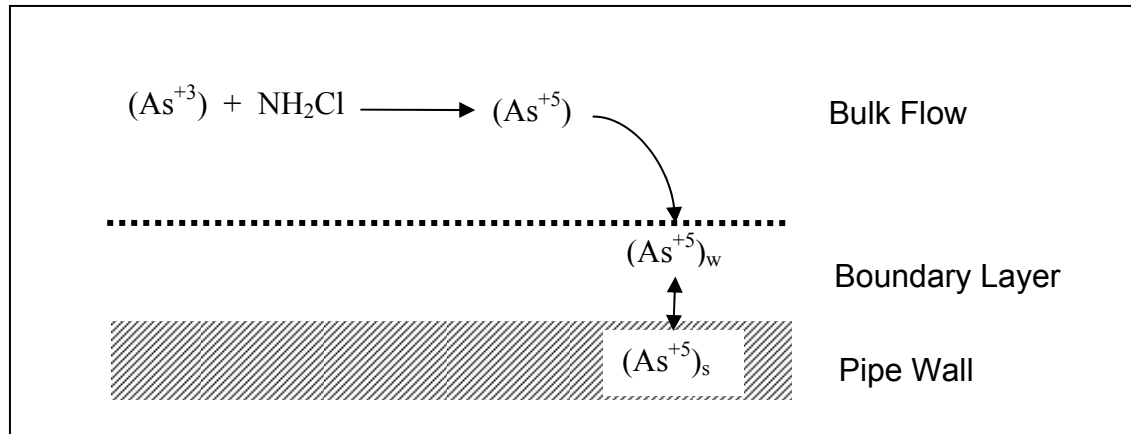


Figure 5.3 Schematic of the mass transfer limited arsenic oxidation/adsorption system.

The mathematical form of this reaction system can be modeled with four differential rate equations and one equilibrium algebraic equation:

$$\frac{d(A3)}{dt} = -K_a (A3)(NH_2CL)$$

$$\frac{d(NH_2CL)}{dt} = -K_b (NH_2CL)$$

$$\frac{d(A5)}{dt} = K_a (A3)(NH_2CL) - K_f A_v (A5 - A5_w)$$

$$\frac{d(A5_w)}{dt} = K_f A_v (A5 - A5_w)$$

$$A5_s = \frac{S_{max} K_s (A5_w)}{1 + K_s (A5_w)}$$

where the standard chemical notation for the various species have been replaced with names more suitable for computer usage as follows: A3 is the bulk phase concentration of arsenite, A5 is the bulk phase concentration of arsenate, A5w is the bulk phase concentration of arsenate adjacent to the pipe wall, A5s is the surface phase concentration of arsenate, and NH₂CL is the bulk phase concentration of monochloramine. The parameters in these equations are as follows: K_a is a rate coefficient for arsenite oxidation, K_b is a monochloramine decay rate coefficient due to reactions with all other reactants (including arsenite), K_f is a mass transfer rate coefficient, A_v is the surface area per unit volume in the pipe, and K_s and S_{max} are Langmuir adsorption isotherm constants. The mass transfer coefficient K_f will in general depend on the amount of flow turbulence as well as the diameter of the pipe. A typical empirical relation might be:

$$K_f = \frac{1.6 \times 10^{-4} \text{ Re}^{0.88}}{D}$$

where Re is the flow Reynolds number and D is the pipe diameter.

Figure 5.4 shows the MSX input file for this system. The [PIPES] section contains three kinetic rate reactions involving the three bulk species and one equilibrium reaction that includes the lone surface species. The [TANKS] section contains only the bulk specie reactions. To complete the model specification, the [QUALITY] section assumes that the network has a single source which is a reservoir node labelled “1”, and that the concentrations at this source remain constant. If this were not the case then a [SOURCES] section could be added that describes the sources in more detail.

```
[SPECIES]
  BULK  A3      UG          ;Dissolved arsenite
  BULK  A5      UG          ;Dissolved arsenate
  BULK  A5w     UG          ;Dissolved arsenate at wall
  WALL  A5s     UG          ;Adsorbed arsenate
  BULK  NH2CL   MG          ;Monochloramine

[COEFFICIENTS]
  CONSTANT  Ka  10.0        ;Arsenite oxidation rate coeff.
  CONSTANT  Kb   0.50       ;Monochloramine decay rate coeff.
  CONSTANT  Ks   5.0        ;Arsenate adsorption coeff.
  CONSTANT  Smax 50         ;Arsenate adsorption coeff.

[TERMS]
  Kf      1.6e-4*Re^0.88/D      ;Mass transfer coefficient

[PIPES]
  RATE  A3      -Ka*A3*NH2CL      ;Arsenite oxidation
  RATE  A5      Ka*A3*NH2CL - Kf*Av*(A5 - A5w) ;Arsenate production
  RATE  A5w     Kf*Av*(A5 - A5w)   ;Arsenate at pipe wall
  RATE  NH2CL   -Kb*NH2CL         ;Monochloramine oxidation
  EQUIL A5s     Smax*Ks*A5w/(1.0 + Ks*A5w) ;Arsenate adsorption

[TANKS]
  RATE  A3      -Ka*A3*NH2CL      ;Arsenite oxidation
  RATE  A5      Ka*A3*NHCL        ;Arsenate production
  RATE  NHCL    -Ka*A3*NHCL - Kb*NHCL ;Monochloramine oxidation

[QUALITY]
  ;Initial conditions (= 0 if not specified here)
  NODE  1      A3      1.0
  NODE  1      A5      0.0
  NODE  1      NHCL    2.5
```

Figure 5.4 MSX input file for the mass transfer limited arsenic oxidation/adsorption system.

Bacterial Regrowth with Chlorine Inhibition

This next example models bacterial re-growth as affected by chlorine inhibition within a distribution system. The re-growth model is taken from Zhang et al.² and includes the following processes as depicted in Figure 5.5:

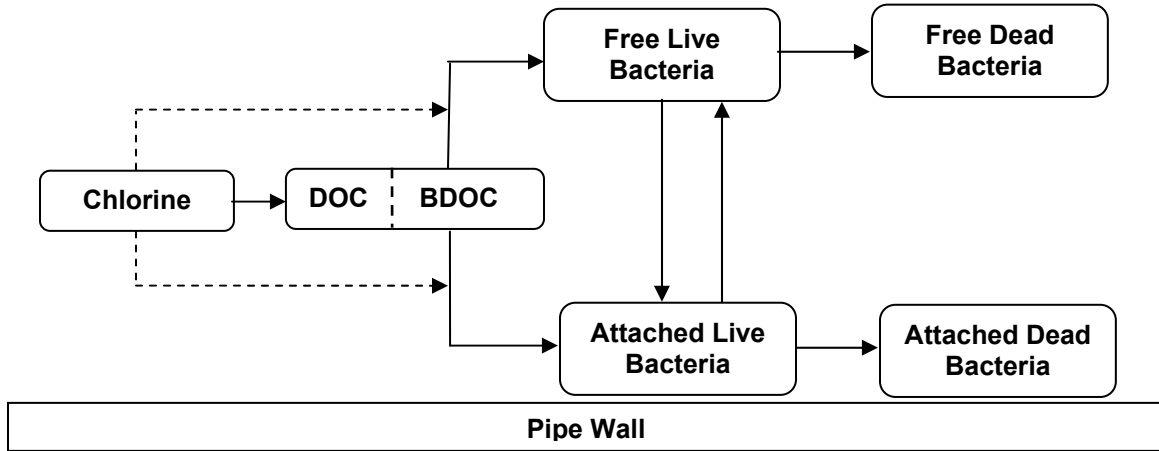


Figure 5.5 Conceptual diagram of bacterial regrowth within a pipeline.

- Both free bacteria in the bulk flow and bacteria attached to the pipe wall utilize the biodegradable fraction of dissolved organic carbon (BDOC and DOC, respectively) as a growth substrate. Monod kinetics are used to describe this growth with the following rate equations:

$$\left. \frac{dX}{dt} \right|_{\text{growth}} = \mu X$$

$$\frac{dS}{dt} = -\mu X / Y$$

where X is mass concentration of bacterial cells, S is the concentration of BDOC, Y is a yield coefficient (mass of cells produced per unit conversion of BDOC), and μ is a specific growth coefficient. The latter decreases as less BDOC remains available according to the equation

$$\mu = \frac{\mu_{\max} S}{S + K_s}$$

² Zhang, W., Miller, C. T., and DiGiano, F. A. (2004). "Bacterial regrowth model for water distribution systems incorporating alternating split-operator solution technique", *Jour. Environmental Engineering*, 130(9):932-941.

where μ_{max} is the maximum growth rate coefficient and K_s is the half-saturation constant.

- b. Both free and attached bacteria die at a first order rate according to the equation:

$$\left. \frac{dX}{dt} \right|_{\text{decay}} = -k_d X$$

where k_d is a decay rate coefficient.

- c. Deposition of free bacterial cells onto the pipe wall is modeled with the following first-order rate process:

$$\left. \frac{dX}{dt} \right|_{\text{deposition}} = -k_{\text{dep}} X$$

while detachment of attached cells into the bulk flow depends also on flow velocity:

$$\left. \frac{dX}{dt} \right|_{\text{detachment}} = k_{\text{det}} XU$$

where k_{dep} is a deposition rate constant, k_{det} is a detachment rate constant, and U is the bulk flow velocity.

- d. The effect of chlorine on limiting the number of viable bacterial cells is modeled by applying an inhibition factor to the bacterial specific growth rate which looks as follows:

$$I = \exp\left(\frac{-(C - C_t)}{C_c}\right)$$

Here C is the chlorine concentration, C_t is a threshold chlorine concentration below which no inhibition occurs, and C_c is a characteristic chlorine concentration that scales the degree of inhibition. Note at higher values of C , I becomes smaller and therefore results in smaller bacterial growth rates.

- e. Chlorine reacts with DOC in the bulk flow to decay at a first-order rate:

$$\frac{dC}{dt} = -k_b C$$

where C is chlorine concentration and k_b is a bulk decay rate coefficient.

The EPANET-MSX specification of the full model is shown in Figure 5.6. Several notes of explanation require mentioning:

```

[SPECIES]
BULK  CL2    MG           ;chlorine
BULK  S      MG           ;organic substrate
BULK  Xb     UG           ;mass of free bacteria
WALL  Xa     UG           ;mass of attached bacteria
BULK  Nb     log(N)       ;number of free bacteria
WALL  Na     log(N)       ;number of attached bacteria

[PARAMETERS]
CONSTANT  Kb      1.3      ;CL2 decay constant (1/days)
CONSTANT  CL2C    0.20     ;characteristic CL2 (mg/L)
CONSTANT  CL2Tb   0.03     ;threshold CL2 for Xb (mg/L)
CONSTANT  CL2Ta   0.10     ;threshold CL2 for Xa (mg/L)
CONSTANT  MUMAXb  0.20     ;max. growth rate for Xb (1/hr)
CONSTANT  MUMAXa  0.20     ;max. growth rate for Xa (1/hr)
CONSTANT  Ks      0.40     ;half saturation constant (mg/L)
CONSTANT  Kdet    0.03     ;detachment rate constant (1/hr)
CONSTANT  Kdep    0.08     ;deposition rate constant (1/hr/(ft/s))
CONSTANT  Kd      0.06     ;bacterial decay constant (1/hr)
CONSTANT  Yg      0.15     ;bacterial yield coefficient (mg/mg)

[TERMS]
Ib  EXP(-STEP(CL2-CL2Tb)*(CL2-CL2Tb)/CL2C) ;Xb inhibition coeff.
Ia  EXP(-STEP(CL2-CL2Ta)*(CL2-CL2Ta)/CL2C) ;Xa inhibition coeff.
MUb MUMAXb*S/(S+Ks)*Ib                      ;Xb growth rate coeff.
MUa MUMAXa*S/(S+Ks)*Ia                      ;Xa growth rate coeff.

[PIPES]
RATE  CL2      -Kb*CL2
RATE  S         -(MUa*Xa*Av + MUb*Xb)/Yg/1000
RATE  Xb        (MUb-Kd)*Xb + Kdet*Xa*U*Av - Kdep*Xb
RATE  Xa        (MUa-Kd)*Xa - Kdet*Xa*U + Kdep*Xb/Av
FORMULA Nb      LOG10(1.0e6*Xb)
FORMULA Na      LOG10(1.0e6*Xa)

[TANKS]
RATE  CL2      -Kb*CL2
RATE  S         -MUb*Xb/Yg/1000
RATE  Xb        (MUb-Kd)*Xb
FORMULA Nb      LOG10(1.0e6*Xb)
FORMULA Na      LOG10(1.0e6*Xa)

[SOURCES]
CONCEN  SrcNode  CL2  1.2
CONCEN  SrcNode  S    0.4
CONCEN  SrcNode  Xb   0.01

```

Figure 5.6 MSX input file for a bacterial regrowth model with chlorine inhibition.

1. There are six species defined for the model: bulk chlorine ($CL2$), bulk biodegradable organic carbon (S), bulk bacterial concentration (Xb), bulk bacterial cell count, attached bacterial concentration (Xa) and attached bacterial cell count (Na). $CL2$ and S are measured in milligrams. The bacterial concentrations are expressed in micrograms of equivalent carbon so that their numerical values scale more evenly. The bacterial cell counts are expressed as the logarithm of the number of cells.
2. The entries in the [PARAMETERS] section are based on values provided by Zhang et al. (2004) and are used only for illustrative purposes.
3. The [TERMS] section allows one to define intermediate mathematical terms in the model's description so that the rate equations can be expressed more clearly and compactly.
4. The chlorine inhibition threshold concentration is lower for the bulk phase than for the surface phase. This results in defining separate inhibition factors, Ib and Ia for these two phases, respectively.
5. The special EPANET-MSX function $STEP(x)$ used in the definitions of the inhibition factors Ib and Ia is internally evaluated to 1 when $x > 0$ and is 0 otherwise.
6. The variables U and Av are reserved symbols in EPANET-MSX that represent flow velocity and pipe surface area per unit volume, respectively, and their values are automatically computed by the program.
7. Whenever the surface biomass species appears in the rate expression for a bulk species it is multiplied by Av to convert from areal density to volumetric concentration. Likewise, the bulk biomass concentration is divided by Av in the rate expression for attached biomass to convert it to an areal density.
8. The kinetic rate expressions for tanks do not include any terms involving Xa since it is assumed that surface species do not exist (or have reduced significance) within storage facilities.
9. A simple FORMULA expression is used to convert from micrograms of bacterial carbon to logarithmic cell counts. It assumes that there are 10^6 cells per microgram of carbon in the cell biomass.
10. The model assumes that there is a single source node named `SrcNode` that supplies all water to the system. The [SOURCES] section specifies the concentrations of chlorine, biodegradable carbon, and bulk bacterial concentration in this water. The latter value was derived from assuming that the treated source water contained 10 cells/mL (i.e., 10^4 cells per liter).

Chloramine Decomposition

This final example illustrates a complex chemical reaction system involving both kinetic rate expressions and nonlinear equilibrium relationships. The system being studied is the auto-decomposition of monochloramine to ammonia in the presence of natural organic matter (NOM). When chloramines are used as a secondary disinfectant care must be taken to avoid producing excessive amounts of free ammonia that can contribute to biological nitrification episodes within the distribution system. The reaction model used for this system was developed by Valentine and co-workers³⁴ and is shown in Table 5.1. The principal species are hypochlorous acid (HOCl), hypochlorite ion (OCl⁻), ammonia (NH₃), ammonium ion (NH₄⁺), monochloramine (NH₂Cl), dichloramine (NHCl₂), an unidentified intermediate compound (I), and total organic carbon (TOC). Because the reactions involve acid-base dissociations and the rate coefficient of the disproportionation of NH₂Cl is a function of both pH and carbonate species, the pH-carbonate equilibrium system is also included.

Figure 5.7 shows the EPANET-MSX specification of the monochloramine decay model. There are 14 bulk species and no surface species. To save space in the figure, the entries in the [COEFFICIENTS] section were omitted since they are simply the rate coefficients k_1 through k_{12} already listed in Table 5.1. The expression for k_5 as a function of pH and carbonate species is included in the [TERMS] section, as are the rate terms contributed by the reactants of reactions 1 through 10 in Table 1. Because there are no surface species in the model, the reaction expressions listed in the [PIPES] section apply to the storage tanks as well.

The first five rate expressions apply to the various chlorinated species, ammonia, and the un-named intermediate compound. The next three rate expressions, all set equal to 0, state that pH, alkalinity, and TOC are assumed to remain constant. These are followed by two equilibrium expressions that represent the dissociation reactions of hypochlorous acid and ammonia, respectively. The final set of four equilibrium expressions model the distribution of the various carbonate species under conditions of constant alkalinity and pH. Note that in order to solve this carbonate equilibrium subsystem it is necessary to supply initial values for pH and alkalinity at all nodes of whatever network is being modeled. This is done in the [QUALITY] section, using the *GLOBAL* specifier to set values throughout the network. (The alkalinity of 0.004 moles/L is equivalent to 200 mg/L as CaCO₃ while the H⁺ value of 2.818×10^{-8} moles/L is the same as a pH of 7.75.)

³ Vikesland, P.J.; Ozekin, K.; Valentine, R.L., "Monochloramine decay in model and distribution system waters", *Water Research*, 35 (7), 1766-1776, 2001.

⁴ Duirk, S.E.; Gombert, B.; Croue, J-P.; Valentine, R.L., "Modeling monochloramine loss in the presence of natural organic matter", *Water Research*, 39, 3418-3431, 2005.

Table 5.1 Monochloramine decay model based on Vikesland et al. (2001) and Duirk et al. (2005).

	<i>Reaction Stoichiometry</i>	<i>Rate Coefficient/ Equilibrium Constant^a</i>
R.1	$\text{HOCl} + \text{NH}_3 \rightarrow \text{NH}_2\text{Cl} + \text{H}_2\text{O}$	$k_1 = 1.5 \times 10^{10} \text{ M}^{-1}\text{h}^{-1}$
R.2	$\text{NH}_2\text{Cl} + \text{H}_2\text{O} \rightarrow \text{HOCl} + \text{NH}_3$	$k_2 = 7.6 \times 10^{-2} \text{ h}^{-1}$
R.3	$\text{HOCl} + \text{NH}_2\text{Cl} \rightarrow \text{NHCl}_2 + \text{H}_2\text{O}$	$k_3 = 1.0 \times 10^6 \text{ M}^{-1}\text{h}^{-1}$
R.4	$\text{NHCl}_2 + \text{H}_2\text{O} \rightarrow \text{HOCl} + \text{NH}_2\text{Cl}$	$k_4 = 2.3 \times 10^{-3} \text{ h}^{-1}$
R.5	$\text{NH}_2\text{Cl} + \text{NH}_2\text{Cl} \rightarrow \text{NHCl}_2 + \text{NH}_3$	$k_5 = 2.5 \times 10^7 [\text{H}^+] +$ $4.0 \times 10^4 [\text{H}_2\text{CO}_3] +$ $800 [\text{HCO}_3^-] \text{ M}^{-2}\text{h}^{-1}$
R.6	$\text{NHCl}_2 + \text{NH}_3 \rightarrow \text{NH}_2\text{Cl} + \text{NH}_2\text{Cl}$	$k_6 = 2.2 \times 10^8 \text{ M}^{-2}\text{h}^{-1}$
R.7	$\text{NHCl}_2 + \text{H}_2\text{O} \rightarrow \text{I}$	$k_7 = 4.0 \times 10^5 \text{ M}^{-1}\text{h}^{-1}$
R.8	$\text{I} + \text{NHCl}_2 \rightarrow \text{HOCl} + \text{products}$	$k_8 = 1.0 \times 10^8 \text{ M}^{-1}\text{h}^{-1}$
R.9	$\text{I} + \text{NH}_2\text{Cl} \rightarrow \text{products}$	$k_9 = 3.0 \times 10^7 \text{ M}^{-1}\text{h}^{-1}$
R.10	$\text{NH}_2\text{Cl} + \text{NHCl}_2 \rightarrow \text{products}$	$k_{10} = 55.0 \text{ M}^{-1}\text{h}^{-1}$
R.11	$\text{NH}_2\text{Cl} + \text{S}_1 \times \text{TOC} \rightarrow \text{products}^b$	$k_{11} = 3.0 \times 10^4 \text{ M}^{-1}\text{h}^{-1}$ $\text{S}_1 = 0.02$
R.12	$\text{HOCl} + \text{S}_2 \times \text{TOC} \rightarrow \text{products}^c$	$k_{12} = 6.5 \times 10^5 \text{ M}^{-1}\text{h}^{-1}$ $\text{S}_2 = 0.5$
E.1	$\text{HOCl} \rightarrow \text{H}^+ + \text{OCl}^-$	$\text{pK}_a = 7.5$
E.2	$\text{NH}_4^+ \rightarrow \text{NH}_3 + \text{H}^+$	$\text{pK}_a = 9.3$
E.3	$\text{H}_2\text{CO}_3 \rightarrow \text{HCO}_3^- + \text{H}^+$	$\text{pK}_a = 6.3$
E.4	$\text{HCO}_3^- \rightarrow \text{CO}_3^{2-} + \text{H}^+$	$\text{pK}_a = 10.3$

Notes:

- All rate coefficients and equilibrium constants are for 25 degrees C.
- S_1 is the fast reactive fraction of TOC.
- S_2 is the slow reactive fraction of TOC.

```

[SPECIES]
BULK  HOCL      MOLES      ;hypochlorous acid
BULK  NH3        MOLES      ;ammonia
BULK  NH2CL      MOLES      ;monochloramine
BULK  NHCL2      MOLES      ;dichloramine
BULK  I          MOLES      ;unknown intermediate
BULK  OCL        MOLES      ;hypochlorite ion
BULK  NH4        MOLES      ;ammonium ion
BULK  ALK        MOLES      ;total alkalinity
BULK  TOC        MOLES      ;total organic carbon
BULK  H          MOLES      ;hydrogen ion
BULK  OH         MOLES      ;hydroxide ion
BULK  CO3        MOLES      ;carbonate ion
BULK  HCO3       MOLES      ;bicarbonate ion
BULK  H2CO3      MOLES      ;dissolved carbon dioxide

[COEFFICIENTS]
<See Table 5.1 for coefficients k1,..k4, k6,..k12, S1, and S2>

[TERMS]
k5    (2.5e7*H) + (4.0e4*H2CO3) + (800*HCO3)
a1    k1*HOCL*NH3
a2    k2*NH2CL
a3    k3*HOCL*NH2CL
a4    k4*NHCL2
a5    k5*NH2CL*NH2CL
a6    k6*NHCL2*NH3*H
a7    k7*NHCL2*OH
a8    k8*I*NHCL2
a9    k9*I*NH2CL
a10   k10*NH2CL*NHCL2
a11   k11*S1*TOC*NH2CL
a12   k12*S2*TOC*HOCL

[PIPES]
RATE  HOCL      -a1 + a2 - a3 + a4 + a8 - a12
RATE  NH3       -a1 + a2 + a5 - a6 + a11
RATE  NH2CL     a1 - a2 - a3 + a4 - a5 + a6 - a9 - a10 - a11
RATE  NHCL2     a3 - a4 + a5 - a6 - a7 - a8 - a10
RATE  I         a7 - a8 - a9
RATE  H         0
RATE  ALK       0
RATE  TOC       0
EQUIL OCL       H*OCL - 3.16E-8*HOCL
EQUIL NH4      H*NH3 - 5.01E-10*NH4
EQUIL CO3      H*CO3 - 5.01E-11*HCO3
EQUIL H2CO3    H*HCO3 - 5.01E-7*H2CO3
EQUIL HCO3     ALK - HCO3 - 2*CO3 - OH + H
EQUIL OH       H*OH - 1.0E-14

[QUALITY]
GLOBAL ALK      0.004
GLOBAL H        2.818E-8

```

Figure 5.7 MSX input file of the monochloramine decomposition model.

APPENDIX A. MSX TOOLKIT FUNCTIONS

The EPANET-MSX toolkit is a library of functions that programmers can use to create their own custom versions of the multi-species extension of EPANET. The MSX functions are used in conjunction with the standard EPANET toolkit functions which can also provide additional flexibility for programmers.⁵ Tables A.1 through A.3 list the name of each MSX toolkit function along with a brief description of its purpose.

These functions reside in a Windows Dynamic Link Library (DLL) named *epanetmsx.dll* and can be used in any programming language that can access DLLs, such as C/C++, Delphi Pascal, Visual Basic, and MatLab. The toolkit also includes special header files that must be included in any program modules that reference the MSX functions. These header files are named *epanetmsx.h* for C/C++ programs, *epanetmsx.pas* for Delphi programs, and *epanetmsx.bas* for Visual Basic programs.

Prior to using any of the MSX toolkit functions a standard EPANET input file must be opened using the *ENopen* function from the standard EPANET toolkit DLL, *epanet2.dll*. In addition, after all processing is completed the *ENclose* function from the standard toolkit must be called. Thus the header files for the standard toolkit (*epanet2.h*, *epanet2.pas*, or *epanet2.bas*) must also be included in the application's code. Finally, if a stand-alone command line executable is being produced from C/C++ then the LIB files *epanet2.lib* and *epanetmsx.lib* must be linked in when the compiled source files are linked together. (Note: The LIB files supplied with the EPANET-MSX distribution are compatible with the Microsoft C/C++ compiler version 6 and higher.)

The following pages provide a description of each toolkit function using C/C++ syntax to represent argument variables and return types.

⁵ Information on using the standard EPANET toolkit is available through a Windows Help file named TOOLKIT.HLP that is part of the EN2toolkit.zip file available at the web site www.epa.gov/ORD/NRMRL/wswrd/epanet.html.

Table A.1 EPANET-MSX toolkit processing functions.

Function Name	Purpose
MSXopen	Opens the EPANET-MSX toolkit system.
MSXclose	Closes the EPANET-MSX toolkit system.
MSXsolveH	Solves for system hydraulics over the entire simulation period, saving results to an internal scratch file.
MSXusehydfile	Uses a previously saved EPANET hydraulics file as the source of hydraulic information.
MSXsolveQ	Solves for water quality over the entire simulation period and saves the results to an internal scratch file.
MSXinit	Initializes the MSX system before solving for water quality results in a step-wise fashion.
MSXstep	Advances the water quality solution through a single water quality time step when performing a step-wise simulation.
MSXsaveoutfile	Saves water quality results computed for each node, link and reporting time period to a named binary file.
MSXsavemsxfile	Saves the data associated with the current MSX project into a new MSX input file.
MSXreport	Writes water quality simulations results as instructed by the MSX input file to a text file.

Table A.2 EPANET-MSX toolkit data retrieval functions.

Function Name	Purpose
MSXgetindex	Retrieves the internal index number of an MSX object given its ID name.
MSXgetIDlen	Retrieves the number of characters in the ID name of an MSX object given its internal index number.
MSXgetID	Retrieves the ID name of an MSX object given its internal index number.
MSXgetcount	Retrieves the number of objects of a specific type.
MSXgetspecie	Retrieves the attributes of a chemical species given its internal index number.
MSXgetinitqual	Retrieves the initial concentration of a particular chemical species assigned to a specific node or link of the pipe network.
MSXgetqual	Retrieves the concentration of a chemical species at a specific node or link of the network at the current simulation time step.
MSXgetconstant	Retrieves the value of a particular reaction constant.
MSXgetparameter	Retrieves the value of a particular reaction parameter for a given pipe or tank within the pipe network.
MSXgetsource	Retrieves information on any external source of a particular chemical species assigned to a specific node of the pipe network.
MSXgetpatternlen	Retrieves the number of time periods within a source time pattern.
MSXgetpatternvalue	Retrieves the multiplier at a specific time period for a given source time pattern.
MSXgeterror	Returns the text for an error message given its error code.

Table A.3 EPANET-MSX data modification functions.

Function Name	Purpose
MSXsetconstant	Assigns a new value to a specific reaction constant.
MSXsetparameter	Assigns a value to a particular reaction parameter for a given pipe or tank within the pipe network.
MSXsetinitqual	Assigns an initial concentration of a particular chemical species to a specific node or link of the pipe network.
MSXsetsource	Sets the attributes of an external source of a particular chemical species to a specific node of the pipe network.
MSXsetpattern	Assigns a new set of multipliers to a given MSX source time pattern.
MSXsetpatternvalue	Assigns a new value to the multiplier for a specific time period in a given MSX source time pattern.
MSXaddpattern	Adds a new, empty MSX source time pattern to the project.

MSXopen

Declaration:

```
int MSXopen(char * f);
```

Description:

Opens the EPANET-MSX toolkit system.

Arguments:

`f` is a C-style character string containing the name of an EPANET-MSX input file.

Returns:

Returns an error code or 0 for no error.

Notes:

The standard EPANET toolkit function `ENopen` must have been called first to open the EPANET toolkit along with an associated EPANET input file for the network being analyzed as well as to identify the name of a report file to which results are written.

Example:

```
//Open the EPANET toolkit
int err = ENopen("example1.inp", "example1.rpt", "");

//Open the MSX toolkit
if (err == 0) err = MSXopen("example1.msx");

//Add code to perform required analyses here
if (err == 0) ...

//Don't forget to close the toolkits
MSXclose();
ENclose();
exit(err);
```

MSXclose

Declaration:

```
int MSXclose(void);
```

Description:

Closes the EPANET-MSX toolkit system.

Arguments:

None.

Returns:

Returns an error code or 0 for no error.

Notes:

The EPANET toolkit function `ENclose` should be called at some point after calling `MSXclose` to down the EPANET toolkit system.

Example:

```
//Open the EPANET toolkit
int err = ENopen("example1.inp", "example1.rpt", "");

//Open the MSX toolkit
if (err == 0) err = MSXopen("example1.msx");

//Add code to perform required analyses here
if (err == 0) ...

//Don't forget to close both toolkits
MSXclose();
ENclose();
```

MSXsolveH

Declaration:

```
int MSXsolveH(void);
```

Description:

Solves for system hydraulics over the entire simulation period saving results to an internal scratch file.

Arguments:

None.

Returns:

Returns an error code or 0 for no error.

Notes:

Either this function or MSXusehydfile (see below) must be called before any water quality processing is performed.

Example:

```
//Open the EPANET & MSX toolkits
int err = ENopen("example1.inp", "example1.rpt", "");
if (err == 0) MSXopen("example1.msx");

//Solve for hydraulics
if (err == 0) err = MSXsolveH();

//Perform water quality analysis starting here
...
```

MSXusehydfile

Declaration:

```
int MSXusehydfile(char * f);
```

Description:

Uses a previously saved EPANET hydraulics file as the source of hydraulic information.

Arguments:

f is a C-style character string containing the name of a previously saved hydraulics file for the system being analyzed.

Returns:

Returns an error code or 0 for no error.

Notes:

Either this function or MSXsolveH (see above) must be called before any water quality processing is performed.

Example:

```
//Open the EPANET toolkit
int err = ENopen("example1.inp", "example1.rpt", "");
if (err > 0) return err;

//Use EPANET to solve & save hydraulic results
ENSolveH();
ENsavehydfile("example1.hyd");

//Open the MSX toolkit
err = MSXopen("example1.msx");
if (err > 0) return err;

//Utilize the hydraulic solution just saved to file
err = MSXusehydfile("example1.hyd");

//Perform water quality analysis starting here
...
```

MSXsolveQ

Declaration:

```
int MSXsolveQ(void);
```

Description:

Solves for water quality over the entire simulation period and saves the results to an internal scratch file.

Arguments:

None.

Returns:

Returns an error code or 0 for no error.

Notes:

This function does not allow access to computed water quality results as the simulation unfolds. If such information is required use MSXinit in conjunction with step-wise calls to MSXstep (see below).

Example:

```
//Open the EPANET & MSX toolkits
int err = ENopen("example1.inp", "example1.rpt", "");
if (err == 0) err = MSXopen("example1.msx");
if (err > 0) return err;

//Solve for hydraulics & water quality
MSXsolveH();
MSXsolveQ();

//Report results
MSXreport();

//Close the toolkits
MSXclose();
ENclose();
```

MSXinit

Declaration:

```
int MSXinit(int saveFlag);
```

Purpose:

Initializes the MSX system before solving for water quality results in step-wise fashion.

Arguments:

Set `saveFlag` to 1 if water quality results should be saved to a scratch binary file, or to 0 if results are not saved to file.

Returns:

Returns an error code or 0 for no error.

Notes:

This function must be called before a step-wise water quality simulation is performed using `MSXstep`. Do not call this function if performing a complete simulation using `MSXsolveQ`.

Example:

See the example provided for `MSXstep`.

MSXstep

Declaration:

```
int MSXstep(long *t, long *tleft);
```

Description:

Advances the water quality solution through a single water quality time step when performing a step-wise simulation.

Arguments:

Upon returning, `t` will contain the current simulation time at the end of the step (in seconds) while `tleft` will contain the time left in the simulation (also in seconds).

Returns:

Returns an error code or 0 for no error.

Notes:

This function should be placed in a loop that repeats until the value of `tleft` becomes 0. `MSXinit` should be called before beginning the loop.

The water quality time step used by this function is specified in the [OPTIONS] section of the MSX input file.

Example:

```
//Declare time variables
long t = 0, tleft = 0;
int err;

//Open the EPANET & MSX toolkits
...

//Solve for hydraulics
MSXsolveH();

//Run a water quality simulation
MSXinit(0);
do {
    err = MSXstep(&t, &tleft);
    //Use MSXgetqual to retrieve results at time t
} while (tleft > 0 && err == 0);
```

MSXsaveoutfile

Declaration:

```
int MSXsaveoutfile(char * f);
```

Description:

Saves water quality results computed for each node, link and reporting time period to a named binary file.

Arguments:

`f` is a C-style character string containing the name of the permanent output results file.

Returns:

Returns an error code or 0 for no error.

Notes:

For a step-wise simulation using `MSXstep`, this function only applies if `MSXinit` was called with its `saveFlag` parameter set to 1 (see `MSXinit`).

The format of the binary results file is described in Appendix B.

Example:

```
//Open the EPANET & MSX toolkits
...

//Solve for hydraulics & water quality
MSXsolveH();
MSXsolveQ();

//Copy saved results to a permanent file
MSXsaveoutfile("example1.out");

//Close the toolkits
...
```

MSXsavemsxfile

Declaration:

```
int MSXsavemsxfile(char * f);
```

Description:

Saves the data associated with the current MSX project into a new MSX input file.

Arguments:

f is a C-style character string containing the name of the file to which data are saved.

Returns:

Returns an error code or 0 for no error.

Notes:

For a step-wise simulation using *MSXstep*, this function only applies if *MSXinit* was called with its *saveFlag* parameter set to 1 (see *MSXinit*).

The format of the binary results file is described in Appendix B.

Example:

```
//Open the EPANET & MSX toolkits
int err = ENopen("example1.inp", "example1.rpt", "");
if (err == 0) err = MSXopen("example1.msx");
if (err > 0) return err;

//Save the current MSX data to a different MSX file
MSXsavemsxfile("example1a.msx");

//Close the toolkits
...
```

MSXreport

Declaration:

```
int MSXreport(void);
```

Description:

Writes water quality simulations results as instructed by the MSX input file to a text file.

Arguments:

None.

Returns:

Returns an error code or 0 for no error.

Notes:

Results are written to the report file specified in the `ENopen` function, unless a specific water quality report file is named in the `[REPORT]` section of the MSX input file.

Example:

```
//Open the EPANET & MSX toolkits
...

//Solve for hydraulics & water quality
MSXsolveH();
MSXsolveQ();

//Write results to the "example1.rpt" file
MSXreport();

//Close the toolkits
...
```

MSXgetindex

Declaration:

```
int MSXgetindex(int type, char * name, int * index);
```

Description:

Retrieves the internal index number of an MSX object given its name.

Arguments:

`type` is the type of object being sought and must be one of the following pre-defined constants:

- MSX_SPECIE (for a chemical specie),
- MSX_CONSTANT (for a reaction constant),
- MSX_PARAMETER (for a reaction parameter),
- MSX_PATTERN (for a time pattern);

`name` is a C-style character string containing the object's ID name;

`index` is the sequence number (starting from 1) of the object in the order it was listed in the MSX input file.

Returns:

Returns an error code or 0 for no error.

Example:

```
//Declare an index variable
int i;

//Open the EPANET & MSX toolkits
...

//Get the index of the chemical specie named "CL2"
MSXgetindex(MSX_SPECIE, "CL2", &i);
```

MSXgetIDlen

Declaration:

```
int MSXgetIDlen(int type, int index, int * len);
```

Description:

Retrieves the number of characters in the ID name of an MSX object given its internal index number.

Arguments:

`type` is the type of object being sought and must be one of the following pre-defined constants:

- MSX_SPECIE (for a chemical specie),
- MSX_CONSTANT (for a reaction constant),
- MSX_PARAMETER (for a reaction parameter),
- MSX_PATTERN (for a time pattern);

`index` is the sequence number of the object (starting from 1 as listed in the MSX input file);

`len` is returned with the number of characters in the object's ID name.

Returns:

Returns an error code or 0 for no error.

Example:

```
//This code finds the longest specie name within a project

//Declare some variables
int count, i, len, maxlen = 0;

//Open the EPANET & MSX toolkits
...

//Examine each specie
MSXgetcount(MSX_SPECIE, &count);
for (i=1; i<=count; i++) {

    //Update longest specie name
    MSXgetIDlen(MSX_SPECIE, i, &len);
    if (len > maxlen) maxlen = len;
}
```

MSXgetID

Declaration:

```
int MSXgetID(int type, int index, char * id, int len);
```

Description:

Retrieves the ID name of an object given its internal index number.

Arguments:

`type` is the type of object being sought and must be one of the following pre-defined constants:

- MSX_SPECIE (for a chemical specie),
- MSX_CONSTANT (for a reaction constant),
- MSX_PARAMETER (for a reaction parameter),
- MSX_PATTERN (for a time pattern);

`index` is the sequence number of the object (starting from 1 as listed in the MSX input file);

`id` is a C-style character string that is returned with the object's ID name.

`len` is the maximum number of characters that `id` can hold.

Returns:

Returns an error code or 0 for no error.

Notes:

The `MSXgetIDlen` function can determine the number of characters in an object's ID name so that the character array `id` can be properly sized.

Example:

```
//Declare a string to hold a species ID
char id[16];

//Open the EPANET & MSX toolkits
...

//Get the name of the 2nd specie in the MSX input file
MSXgetID(MSX_SPECIE, 2, id, strlen(id));
```

MSXgetcount

Declaration:

```
int MSXgetcount(int type, int * count);
```

Description:

Retrieves the number of objects of a specific type.

Arguments:

`type` is the type of object being sought and must be one of the following pre-defined constants:

- MSX_SPECIE (for a chemical specie),
- MSX_CONSTANT (for a reaction constant),
- MSX_PARAMETER (for a reaction parameter),
- MSX_PATTERN (for a time pattern);

`count` is the number of objects of that type defined in the MSX input file.

Returns:

Returns an error code or 0 for no error.

Example:

```
//Declare a variable for the number of chemical species
int nSpecies;

//Open the EPANET & MSX toolkits
...

//Get the number of species
MSXgetcount(MSX_SPECIE, &nSpecies);
```


MSXgetspecie

Declaration:

```
int MSXgetspecie(int specie, int * type, int * units,
                 double * aTol, double * rTol);
```

Description:

Retrieves the attributes of a chemical species given its internal index number.

Arguments:

`specie` is the sequence number of the specie (starting from 1 as listed in the MSX input file);

`type` is returned with one of the following pre-defined constants:

MSX_BULK (defined as 0) for a bulk water specie,
MSX_WALL (defined as 1) for a pipe wall surface specie;

`units` is returned with one of the following pre-defined constants:

MSX_MG (defined as 0) for milligrams,
MSX_UG (defined as 1) for micrograms,
MSX_MOLE (defined as 2) for moles,
MSX_MMOL (defined as 3) for millimoles;

`aTol` is returned with the absolute concentration tolerance defined for the specie (in concentration units);

`rTol` is returned with the relative concentration tolerance defined for the specie.

Returns:

Returns an error code or 0 for no error.

Example:

```
//Declare some variables
int sIndex, sType, sUnits;
double aTol, rTol;

//Open the EPANET & MSX toolkits
...

//Get attributes of the specie named "Xwall"
MSXgetindex(MSX_SPECIE, "Xwall", &sIndex);
MSXgetspecie(sIndex, &sType, &sUnits, &aTol, &rTol);
```

MSXgetinitqual

Declaration:

```
int MSXgetinitqual(int obj, int index, int specie,  
                  double * value);
```

Description:

Retrieves the initial concentration of a particular chemical species assigned to a specific node or link of the pipe network.

Arguments:

`obj` is type of object being queried and must be either:

MSX_NODE (defined as 0) for a node

MSX_LINK (defined as 1) for a link;

`index` is the internal sequence number (starting from 1) assigned to the node or link;

`specie` is the sequence number of the specie (starting from 1 as listed in the MSX input file);

`value` is returned with the initial concentration of the specie at the node or link of interest.

Returns:

Returns an error code or 0 for no error.

Notes:

The EPANET toolkit functions ENgetnodeindex and ENgetlinkindex can be used to identify the index of a node or link from its ID name;

Concentrations are expressed as mass units per liter for bulk species and as mass per unit area for surface species.

Example:

```
int n, s;  
double c0;  
  
//Open the EPANET & MSX toolkits  
...  
  
//Get initial concentration Of "CL2" in "Tank_A"  
ENgetnodeindex("Tank_A", &n);  
MSXgetindex(MSX_SPECIE, "CL2", &s);  
MSXgetinitqual(MSX_NODE, n, s, &c0);
```

MSXgetqual

Declaration:

```
int MSXgetqual(int obj, int index, int specie,  
               double * value);
```

Purpose:

Retrieves a chemical species concentration at a given node at the current simulation time step.

Arguments:

`obj` is type of object being queried and must be either:

`MSX_NODE` (defined as 0) for a node

`MSX_LINK` (defined as 1) for a link;

`index` is the internal sequence number (starting from 1) assigned to the node or link;

`specie` is the sequence number of the specie (starting from 1 as listed in the MSX input file);

`value` is returned with the computed concentration of the specie at the current time period.

Returns:

Returns an error code or 0 for no error.

Notes:

The EPANET toolkit functions `ENgetnodeindex` and `ENgetlinkindex` can be used to identify the index of a node or link from its ID name;

Concentrations are expressed as mass units per liter for bulk species and as mass per unit area for surface species.

Example:

```
//Declare some variables  
long t, tstep;  
int n, s;  
double c, cMax = 0.0;  
  
//Open the EPANET & MSX toolkits  
...
```

```

//Get the indexes of node "Tank_A" and specie "CL2"
ENgetnodeindex("Tank_A", &n);
MSXgetindex(ENMSX_SPECIE, "CL2", &s);

//Obtain a hydraulic solution
MSXsolveH();

//Run a step-wise water quality analysis
//without saving results to file
MSXinit(0);
do {
    err = MSXstep(&t, &tleft);

    //Retrieve CL2 concentration at Tank_A
    MSXgetqual(MSX_NODE, n, s, &c);

    //Update the max. concentration
    if (c > cMax) cMax = c;
} while (tleft > 0 && err == 0);

//Close the toolkits
...

```

MSXgetconstant

Declaration:

```
int MSXgetconstant(int index, double * value);
```

Description:

Retrieves the value of a particular reaction constant.

Arguments:

`index` is the sequence number of the reaction constant (starting from 1) as it appeared in the MSX input file;

`value` is returned with the value assigned to the constant.

Returns:

Returns an error code or 0 for no error.

Example:

```
//Declare some variables
int i;
double k1;

//Open the EPANET & MSX toolkits
...

//Get the index of the constant named K1
MSXgetindex(MSX_CONSTANT, "K1", &i);

//Get the value of K1
MSXgetconstant(i, &k1);
```

MSXgetparameter

Declaration:

```
int MSXgetparameter(int obj, int index, int param,  
                    double * value);
```

Description:

Retrieves the value of a particular reaction parameter for a given pipe or tank within the pipe network.

Arguments:

`obj` is type of object being queried and must be either:

 MSX_NODE (defined as 0) for a node

 MSX_LINK (defined as 1) for a link;

`index` is the internal sequence number (starting from 1) assigned to the node or link;

`param` is the sequence number of the parameter (starting from 1 as listed in the MSX input file);

`value` is returned with the value assigned to the parameter for the node or link of interest.

Returns:

Returns an error code or 0 for no error.

Notes:

Reaction parameters are only defined for storage tank nodes and pipe links. All other types of nodes and links have parameter values of 0.

Example:

```
//Declare some variables  
int i, j;  
double k2;  
  
//Open the EPANET & MSX toolkits  
...  
  
//Get the value of parameter "K2" for pipe "P1"  
ENgetlinkindex("P1", &i);  
MSXgetindex(MSX_PARAMETER, "K2", &j);  
MSXgetparameter(MSX_LINK, i, j, &k2);
```

MSXgetsource

Declaration:

```
int MSXgetsource(int node, int specie, int * type,
                 double * level, int * pat);
```

Description:

Retrieves information on any external source of a particular chemical species assigned to a specific node of the pipe network.

Arguments:

`node` is the internal sequence number (starting from 1) assigned to the node of interest;

`specie` is the sequence number of the specie of interest (starting from 1 as listed in the MSX input file);

`type` is returned with the type of external source and will be one of the following pre-defined constants:

<code>MSX_NOSOURCE</code>	(defined as -1) for no source,
<code>MSX_CONCEN</code>	(defined as 0) for a concentration source,
<code>MSX_MASS</code>	(defined as 1) for a mass booster source,
<code>MSX_SETPOINT</code>	(defined as 2) for a setpoint source,
<code>MSX_FLOWPACED</code>	(defined as 3) for a flow paced source;

The meaning of these source types can be found in the description of the [SOURCES] section of the MSX input file in section 4 of this manual.

`level` is returned with the baseline concentration (or mass flow rate) of the source;

`pat` is returned with the index of the time pattern used to add variability to the source's baseline level (and will be 0 if no pattern was defined for the source).

Returns:

Returns an error code or 0 for no error.

Example:

```
//Declare some variables
int n, s, t, p;
double c;

//Open the EPANET & MSX toolkits
...

//Get source information for species CL2 at node N1
ENgetnodeindex("N1", &n);
MSXgetindex(MSX_SPECIE, "CL2", &s);
MSXgetsource(n, s, &t, &c, &p);
```


MSXgetpatternlen

Declaration:

```
int MSXgetpatternlen(int pat, int * len);
```

Description:

Retrieves the number of time periods within a source time pattern.

Arguments:

`pat` is the internal sequence number (starting from 1) of the pattern as it appears in the MSX input file;

`len` is returned with the number of time periods (and therefore number of multipliers) that appear in the pattern.

Returns:

Returns an error code or 0 for no error.

Notes:

This function only applies to source time patterns that appear in the MSX input file. There is a comparable EPANET toolkit function, `ENgetpatternlen`, which can be used for the demand patterns defined in the EPANET input file.

Example:

```
//Declare some variables
int i, n;

//Open the EPANET & MSX toolkits
...

//Get the number of multipliers (n) in pattern "P1"
MSXgetindex("P1", &i);
MSXgetpatternlen(i, &n);
```

MSXgetpatternvalue

Declaration:

```
int MSXgetpatternvalue(int pat, int period, double * value);
```

Description:

Retrieves the multiplier at a specific time period for a given source time pattern.

Arguments:

`pat` is the internal sequence number (starting from 1) of the pattern as it appears in the MSX input file;

`period` is the index of the time period (starting from 1) whose multiplier is being sought;

`value` is returned with the value of the pattern's multiplier in the desired period.

Returns:

Returns an error code or 0 for no error.

Notes:

This function only applies to source time patterns that appear in the MSX input file. There is a comparable EPANET toolkit function, `Engetpatternvalue`, which can be used for the demand patterns defined in the EPANET input file.

Example:

```
//Declare some variables
int i, n;

//Open the EPANET & MSX toolkits
...

//Get the number of multipliers (n) in pattern "P1"
MSXgetindex("P1", &i);
MSXgetpatternlen(i, &n);
```

MSXgeterror

Declaration:

```
int MSXgeterror(int code, char * msg, int len);
```

Description:

Returns the text for an error message given its error code.

Arguments:

`code` is the code number of an error condition generated by EPANET-MSX;

`msg` is a C-style character string that is returned containing the text of the error message corresponding to the error code;

`len` is the maximum number of characters that `msg` can contain.

Returns:

Returns an error code or 0 for no error.

Notes:

`msg` should be sized to accept a minimum of 80 characters.

This function only applies to error codes generated by the MSX toolkit. There is a comparable EPANET toolkit function, `ENgeterror`, that applies to EPANET errors.

Example:

```
char msg[81];

//Open the EPANET toolkit & check for errors
int err = Enopen("example1.inp", "example1.rpt", "");
if (err > 0) ENgeterror(err, msg);

//Open the MSX toolkit & check for errors
else {
    err = MSXopen("example1.msx");
    if (err > 0) MSXgeterror(err, msg);
}
if (err > 0) printf("\n%s", msg);
return err;
```

MSXsetconstant

Declaration:

```
int MSXsetconstant(int index, double value);
```

Description:

Assigns a new value to a specific reaction constant.

Arguments:

`index` is the sequence number of the reaction constant (starting from 1) as it appeared in the MSX input file;

`value` is the new value to be assigned to the constant.

Returns:

Returns an error code or 0 for no error.

Example:

```
//Declare an index variable
int i;

//Open the EPANET & MSX toolkits
...

//Get the index of the constant named K1
MSXgetindex(MSX_CONSTANT, "K1", &i);

//Set a new value of K1
MSXsetconstant(i, 0.53);
```

MSXsetparameter

Declaration:

```
int MSXsetparameter(int type, int index, int param,
                    double value);
```

Description:

Assigns a value to a particular reaction parameter for a given pipe or tank within the pipe network.

Arguments:

type is type of object being queried and must be either:

MSX_NODE (defined as 0) for a node

MSX_LINK (defined as 1) for a link;

index is the internal sequence number (starting from 1) assigned to the node or link;

param is the sequence number of the parameter (starting from 1 as listed in the MSX input file);

value is the value to be assigned to the parameter for the node or link of interest.

Returns:

Returns an error code or 0 for no error.

Notes:

Reaction parameters are only defined for storage tank nodes and pipe links. Attempts to set parameter values for other types of nodes and links will be ignored.

Example:

```
//Declare some index variables
int i, j;

//Open the EPANET & MSX toolkits
...

//Get indexes for parameter "K2" for pipe "P1"
ENgetlinkindex("P1", &i);
MSXgetindex(MSX_PARAMETER, "K2", &j);

//Set a new value for the parameter
MSXsetparameter(MSX_LINK, i, j, 0.25);
```

MSXsetinitqual

Declaration:

```
int MSXsetinitqual(int type, int index, int specie,  
                   double value);
```

Description:

Assigns an initial concentration of a particular chemical species to a specific node or link of the pipe network.

Arguments:

`type` is type of object being queried and must be either:

`MSX_NODE` (defined as 0) for a node

`MSX_LINK` (defined as 1) for a link;

`index` is the internal sequence number (starting from 1) assigned to the node or link;

`specie` is the sequence number of the specie (starting from 1 as listed in the MSX input file);

`value` is the initial concentration of the specie to be applied at the node or link of interest.

Returns:

Returns an error code or 0 for no error.

Notes:

The EPANET toolkit functions `ENgetnodeindex` and `ENgetlinkindex` can be used to identify the index of a node or link from its ID name;

Concentrations are expressed as mass units per liter for bulk species and as mass per unit area for surface species.

Example:

```
//Declare some index variables
int n, s;

//Open the EPANET & MSX toolkits
...

//Get the indexes of node "Tank_A" and specie "CL2"
ENgetnodeindex("Tank_A", &n);
MSXgetindex(MSX_SPECIE, "CL2", &s);

//Then set the initial concentration
MSXsetinitqual(MSX_NODE, n, s, 1.25);
```

MSXsetsource

Declaration:

```
int MSXsetsource(int node, int specie, int type,
                 double level, int pat);
```

Description:

Sets the attributes of an external source of a particular chemical species to a specific node of the pipe network.

Arguments:

`node` is the internal sequence number (starting from 1) assigned to the node of interest;

`specie` is the sequence number of the specie of interest (starting from 1 as listed in the MSX input file);

`type` is the type of external source to be utilized and will be one of the following pre-defined constants:

<code>MSX_NOSOURCE</code>	(defined as -1) for no source,
<code>MSX_CONCEN</code>	(defined as 0) for a concentration source,
<code>MSX_MASS</code>	(defined as 1) for a mass booster source,
<code>MSX_SETPPOINT</code>	(defined as 2) for a setpoint source,
<code>MSX_FLOWPACED</code>	(defined as 3) for a flow paced source;

The meaning of these source types can be found in the description of the [SOURCES] section of the MSX input file in section 4 of this manual.

`level` is the baseline concentration (or mass flow rate) of the source;

`pat` is the index of the time pattern used to add variability to the source's baseline level (use 0 if the source has a constant strength).

Returns:

Returns an error code or 0 for no error.

Notes:

The EPANET toolkit function `ENgetnodeindex` can be used to identify the index of a node from its ID name;

Concentrations are expressed as mass units per liter for bulk species and as mass per unit area for surface species.

Example:

```
//Declare some index variables
int n, s;

//Open the EPANET & MSX toolkits
...

//Get indexes for species CL2 and node N1
ENgetnodeindex("N1", &n);
MSXgetindex(MSX_SPECIE, "CL2", &s);

//Assign a constant source strength of 1 mg/L
MSXsetsource(n, s, MSX_SETPOINT, 1.0, 0);
```

MSXsetpattern

Declaration:

```
int MSXsetpattern(int pat, double mult[], int len);
```

Description:

Assigns a new set of multipliers to a given MSX source time pattern.

Arguments:

pat is the internal sequence number (starting from 1) of the pattern as it appears in the MSX input file;

mult[] is an array of multiplier values to replace those previously used by the pattern;

len is the number of entries in the multiplier array mult.

Returns:

Returns an error code or 0 for no error.

Notes:

This function only applies to source time patterns that appear in the MSX input file. There is a comparable EPANET toolkit function, `ENsetpattern`, which can be used for the demand patterns defined in the EPANET input file.

Example:

```
//Declare an array of multipliers
double mult[6] = {1.1, 1.5, 0.8, 0.5, 0.2, 0.0};
int i;

//Open the EPANET & MSX toolkits
...

//Get index for pattern "P1"
MSXgetindex(MSX_PATTERN, "P1", &i);

//Assign multipliers to the pattern
MSXsetpattern(i, mult, 6);
```

MSXsetpatternvalue

Declaration:

```
int MSXsetpatternvalue(int pat, int period, double value);
```

Description:

Assigns a new value to the multiplier for a specific time period in a given MSX source time pattern.

Arguments:

pat is the internal sequence number (starting from 1) of the pattern as it appears in the MSX input file;

period is the time period (starting from 1) in the pattern to be replaced;

value is the new multiplier value to use for that time period.

Returns:

Returns an error code or 0 for no error.

Notes:

This function only applies to source time patterns that appear in the MSX input file. There is a comparable EPANET toolkit function, `ENsetpatternvalue`, which can be used for the demand patterns defined in the EPANET input file.

Example:

```
//Declare some variables
int i, p, n;
double v;

//Open the EPANET & MSX toolkits
...

//Get index & number of multipliers for pattern "P1"
MSXgetindex(MSX_PATTERN, "P1", &p);
MSXgetpatternlen(p, &n);

//Increase each multiplier by factor of 2
for (i = 1; i <= n; i++) {
    MSXgetpatternvalue(p, &v);
    v = 2.0 * v;
    MSXsetpatternvalue(p, i, v);
}
```

MSXaddpattern

Declaration:

```
int MSXaddpattern(char * id);
```

Description:

Adds a new, empty MSX source time pattern to an MSX project

Arguments:

`id` is a C-style character string containing the name of the new pattern.

Returns:

Returns an error code or 0 for no error.

Notes:

The new pattern has no time periods or multipliers assigned to it. The `MSXsetpattern` function can be used to assign an array of multipliers to the pattern.

Example:

```
//Declare some variables
int err, p;
double mult[6] = {0.5, 0.8, 1.2, 1.0, 0.7, 0.3};

//Create a new pattern named "newPat"
err = MSXaddpattern("newPat");

//Assign multipliers to it
if (err == 0) {
    MSXgetindex(MSX_PATTERN, "newPat", &p);
    MSXsetpattern(p, mult, 6);
}
```

APPENDIX B. BINARY OUTPUT FILE FORMAT

The EPANET-MSX system can save the water quality results it computes to a binary output file. This file can be named and saved to disk using the `MSXsaveoutfile` function. The format of the file's contents is described in Table B.1 below.

Table B.1 Format of the EPANET-MSX binary output file.

Quantity	Size and Type
Magic number (516114521)	4-byte integer
Version number (currently 100000)	4-byte integer
Number of network nodes	4-byte integer
Number of network links	4-byte integer
Number of water quality species	4-byte integer
Reporting time step (seconds)	4-byte integer
For each water quality species:	
Number of characters in ID name (N)	4-byte integer
ID name	N character bytes
Specie units	15 character bytes
For each reporting period:	
For each water quality species:	
For each network node:	
Nodal water quality result	4-byte float
For each water quality species:	
For each network link	
Link water quality result	4-byte float
Byte offset where water quality results begin	4-byte integer
Number of reporting periods	4-byte integer
Error code	4-byte integer
Magic number (516114521)	4-byte integer

APPENDIX C. MSX ERROR CODES

- Error 501:** **insufficient memory available.**
There is not enough physical memory in the computer to analyze the pipe network.
- Error 502:** **no EPANET data file supplied.**
A standard EPANET input was not opened with a call to `ENopen` before the MSX system was opened with `MSXopen`.
- Error 503:** **could not open MSX input file.**
The MSX input file does not exist or cannot be opened (i.e., it may be in use by another program).
- Error 504:** **could not open hydraulic results file.**
The hydraulic results file specified in the `MSXusehydfile` function either does not exist or cannot be opened.
- Error 505:** **could not read hydraulic results file.**
The hydraulic results file generated by the `MSXsolveH` function or imported by the `MSXusehydfile` function could not be read correctly. This could happen if, for example, an imported file was not actually a hydraulic results file.
- Error 506:** **could not read MSX input file.**
The contents of the MSX input file were formatted incorrectly or had other errors (such as duplicate ID names for objects of the same type). The specific errors and the offending lines will be listed in the report file named when calling the `ENopen` function.
- Error 507:** **too few pipe reaction expressions.**
The total number of Rate, Equilibrium, and Formula expressions in the `[PIPES]` section of the MSX input file must equal the total number of species defined.
- Error 508:** **too few tank reaction expressions.**
The total number of Rate, Equilibrium, and Formula expressions in the `[TANKS]` section of the MSX input file must equal the total number of bulk species defined.
- Error 509:** **could not open differential equation solver.**
The system's differential equation solver could not be opened, possibly because of insufficient memory available.
- Error 510:** **could not open algebraic solver.**
The system's nonlinear equation solver could not be opened, possibly because of insufficient memory available.

- Error 511: could not open binary results file.**
The binary output file where EPANET-MSX stores its computed results could not be opened or does not exist.
- Error 512: read/write error on binary results file.**
An error occurred when either writing a result to the binary results file or reading a result from the file.
- Error 513: could not integrate reaction rate expressions.**
The differential equation solver employed by EPANET-MSX could not successfully integrate the system's reaction rate equations over the current water quality time step. One could try re-running the analysis using a smaller time step or larger values for ATOL and RTOL (as specified in the [OPTIONS] or [SPECIES] sections of the MSX input file).
- Error 514: could not solve reaction equilibrium expressions.**
The nonlinear equation solver employed by EPANET-MSX could not successfully solve the system's set of equilibrium equations at the current simulation time. Users must insure that the initial conditions set throughout the pipe network are sufficient and consistent so that a solution exists for the governing set of equilibrium equations.
- Error 515: reference made to an unknown type of object.**
The object type code number supplied as an argument in one of the MSX toolkit functions does not equal any of the predefined code numbers.
- Error 516: reference made to an illegal object index.**
The object index number supplied as an argument in one of the MSX toolkit functions is either ≤ 0 or higher than the number of objects of the type of being referenced.
- Error 517: reference made to an undefined object ID.**
The object ID name supplied as an argument in one of the MSX toolkit functions does not belong to any object defined in the MSX input file.
- Error 518: invalid property values were specified.**
An invalid value was supplied as an argument to one of the MSX toolkit functions that modifies a specific property of an object (e.g., an initial or source concentration cannot be a negative value).
- Error 519: an MSX project was not opened.**
A call was made to an MSX toolkit function without having first successfully opened a project with the `MSXopen` function.