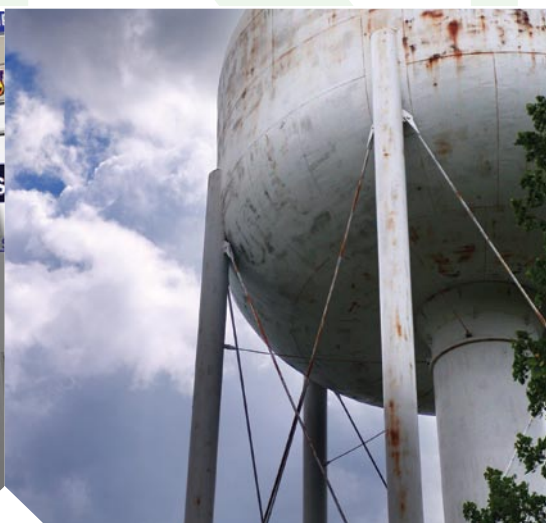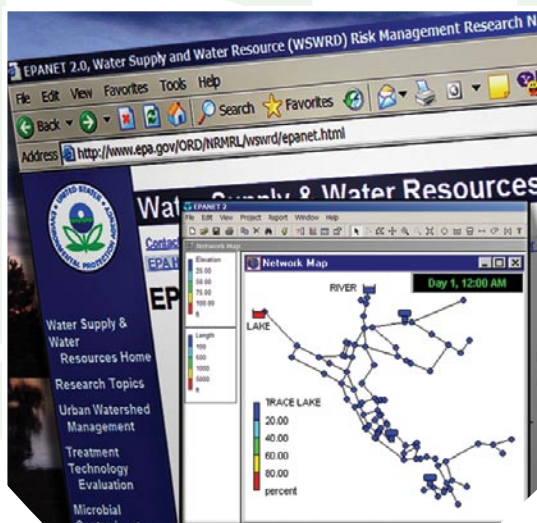# EPANET Multi-Species Extension User's Manual

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

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. Specifications of these terms are included with the EPANET-MSX software distribution.

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.

The National Homeland Security Research Center (NHSRC) was formed in 2002 to conduct research in support of EPA's role in homeland security. NHSRC research efforts focus on five areas: water infrastructure protection, threat and consequence assessment, decontamination and consequence management, response capability enhancement, and homeland security technology testing and evaluation. EPA is the lead federal agency for drinking water and wastewater systems and the NHSRC is working to reduce system vulnerabilities, prevent and prepare for terrorist attacks, minimize public health impacts and infrastructure damage, and enhance recovery efforts.

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.

<div align="right">

Sally C. Gutierrez, Director
National Risk Management Research Laboratory

Jonathan Herrmann, Director
National Homeland Security Research Center

</div>

ACKNOWLEDGEMENTS

CONTENTS

1. INTRODUCTION

EPANET (Rossman, 2000) 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 an extension to the original EPANET that allows it to model any system of multiple, interacting chemical species. 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.
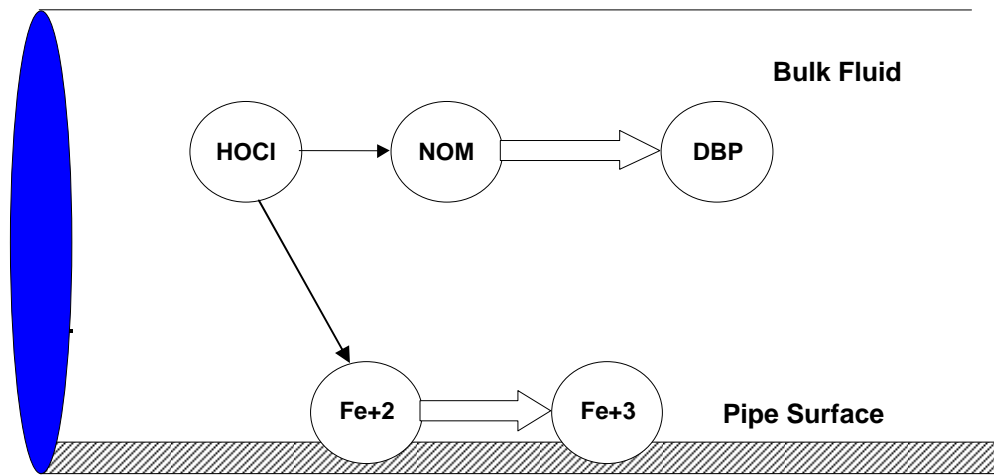
Figure 2-1 Example of reactions in the mobile bulk phase and at the fixed pipe surface phase.

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:
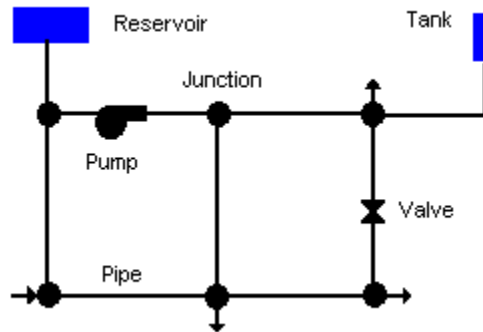
Figure 2-2 Node-link representation of a simple distribution system.

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:

$$\frac{d\mathbf{x_b}}{dt} = \mathbf{f}(\mathbf{x_b}, \mathbf{x_s}, \mathbf{z_b}, \mathbf{z_s}, \mathbf{p}) \tag{1}$$

$$\frac{d\mathbf{x_s}}{dt} = \mathbf{g}(\mathbf{x_b}, \mathbf{x_s}, \mathbf{z_b}, \mathbf{z_s}, \mathbf{p}) \tag{2}$$

$$\mathbf{0} = \mathbf{h}(\mathbf{x_b}, \mathbf{x_s}, \mathbf{z_b}, \mathbf{z_s}, \mathbf{p}) \tag{3}$$

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 in (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 DAEs, consider the oxidation of arsenite (As^{+3}) to arsenate (As^{+5}) by a monochloramine disinfectant residual in the bulk flow and the subsequent adsorption of arsenate onto exposed iron on the pipe wall. (Arsenite adsorption is not significant at the pH's typically found in drinking water.) A more complete explanation and extension of this model is presented in section 5 of this manual. 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 equation:

$$\frac{dAs^{+3}}{dt} = -k_a As^{+3}(NH_2Cl) \tag{4}$$

$$\frac{dAs^{+5}}{dt} = k_a As^{+3}(NH_2Cl) - A_v[k_1(S_{max} - As_s^{+5})As^{+5} - k_2 As_s^{+5}] \tag{5}$$

$$\frac{d(NH_2Cl)}{dt} = -k_b(NH_2Cl) \tag{6}$$

$$As_s^{+5} = \frac{k_s S_{max} As^{+5}}{1 + k_s As^{+5}}, \tag{7}$$

where As^{+3} is the bulk phase concentration of arsenite, As^{+5} is the bulk phase concentration of arsenate, As_s^{+5} is surface phase concentration of arsenate, and NH_2Cl 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, A_v is the pipe surface area per liter pipe volume, k_1 and k_2 are arsenate adsorption and desorption rate coefficients, S_{max} is the maximum pipe surface concentration of arsenate, and $k_s = k_1/k_2$. Thus in terms of the notation used in (1)-(3), $\mathbf{x}_b = \{As^{+3}, As^{+5}, NH_2Cl\}$, $\mathbf{x}_s = \{\varnothing\}$, $\mathbf{z}_b = \{\varnothing\}$, $\mathbf{z}_s = \{As_s^{+5}\}$, $\mathbf{p} = \{k_a, k_b, A_v, k_1, k_2, S_{max}\}$. Example input files for this form of the model are included with the standard EPANET-MSX distribution, while the input file for a more complex version of the model is presented in section 5.

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).
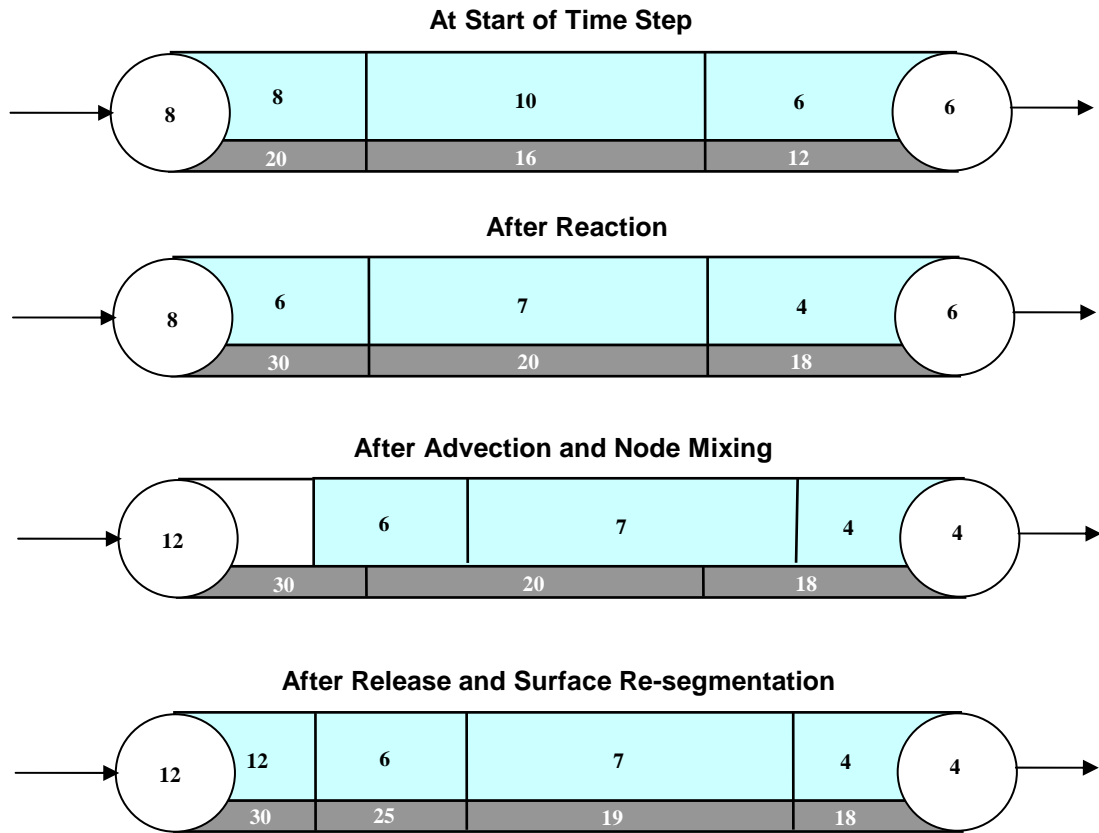
Figure 2-3   Illustration of the 4-step water quality transport method for pipe networks. The upper pipe segments contain flowing water while the lower segments are the pipe wall surface. The numbers in each segment represent hypothetical bulk and surface species concentrations, respectively.

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, 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 **p** can possibly vary by pipe. For the equilibrium reactions it is assumed that the Jacobian matrix of **h** with respect to **z**, $\partial \mathbf{h}/\partial \mathbf{z}$, is unique and nonsingular for all t. In this case, the implicit functions defined by equation (3),

$$\mathbf{z_b} \; = \; \mathbf{z_b}(\mathbf{x_b}, \mathbf{x_s}, \mathbf{p}) \tag{8}$$

$$\mathbf{z_s} \; = \; \mathbf{z_s}(\mathbf{x_b}, \mathbf{x_s}, \mathbf{p}) \tag{9}$$

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 used 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:

$$\frac{dx_b}{dt} = \mathbf{f}(\mathbf{x_b}, \mathbf{x_s}, \mathbf{z_b}(\mathbf{x_b}, \mathbf{x_s}, \mathbf{p}), \mathbf{z_s}(\mathbf{x_b}, \mathbf{x_s}, \mathbf{p}), \mathbf{p})$$

$$= \mathbf{f'}(\mathbf{x_b}, \mathbf{x_s}, \mathbf{p}) \tag{10}$$

$$\frac{dx_s}{dt} = \mathbf{g}(\mathbf{x_b}, \mathbf{x_s}, \mathbf{z_b}(\mathbf{x_b}, \mathbf{x_s}, \mathbf{p}), \mathbf{z_s}(\mathbf{x_b}, \mathbf{x_s}, \mathbf{p}), \mathbf{p})$$

$$= \mathbf{g'}(\mathbf{x_b}, \mathbf{x_s}, \mathbf{p}) \tag{11}$$

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 $\mathbf{x_b}$ and $\mathbf{z_b}$ 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 $\mathbf{x_s}$ and $\mathbf{z_s}$ 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:

$$x_{si}^{new} = \left(\frac{1}{L_i^{new}} \right) \sum_{j=1}^{n} \left(L_i^{new} \cap L_j \right) x_{sj} \qquad \text{for } i = 1, \ldots n^{new} \tag{12}$$

$$z_{si}^{new} = \left(\frac{1}{L_i^{new}} \right) \sum_{j=1}^{n} \left(L_i^{new} \cap L_j \right) z_{sj} \qquad \text{for } i = 1, \ldots n^{new} \tag{13}$$

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 ODEs, equations (1) and (2). These include a forward Euler method (as used in EPANET), a fifth order Runge-Kutta method with automatic time step control (Hairer et al., 1993), and a second order Rosenbrock method with automatic time step control (Verwer et al., 1999). These are listed in order of the numerical work per time step 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 (see, e.g., Golub and Van Loan, 1992).

The algebraic equilibrium equations (3) are solved using a standard implementation of the Newton method (Press et al., 1992). This algorithm requires that the Jacobian of **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 ODEs 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. The top level archive folder contains a *Readme.txt* file that describes the contents of the archive and procedures for reporting bugs.

Table 3-1 Files distributed with EPANET-MSX.

`Readme.txt`	describes the contents of the archive
`\Bin`	
`    epanetmsx.exe`	command line version of EPANET-MSX
`    epanetmsx.dll`	EPANET-MSX function library
`    epanet2.dll`	standard EPANET function library
`\Examples`	
`    example.inp`	example EPANET input file
`    example.msx`	example MSX input file
`\Doc`	
`    epanetmsx.pdf`	EPANET-MSX users manual
`    license.txt`	licensing agreement for using EPANET-MSX
`\Include`	
`    epanetmsx.h`	C/C++ header file for EPANET-MSX toolkit
`    epanetmsx.bas`	Visual Basic declarations of EPANET-MSX functions
`    epanetmsx.pas`	Delphi-Pascal declarations of EPANET-MSX functions
`    epanet2.h`	C/C++ header file for EPANET2 toolkit
`    epanet2.bas`	Visual Basic declarations of EPANET2 functions
`    epanet2.pas`	Delphi-Pascal declarations of EPANET2 functions
`    epanetmsx.lib`	Microsoft C/C++ LIB file for epanetmsx.dll
`    epanet2.lib`	Microsoft C/C++ LIB file for epanet2.dll
`\Src`	EPANET-MSX source files
`\Make`	files for compiling the EPANET-MSX function library and the command line executable on both Windows and Linux platforms.

Most end users will only need to extract the files in the $\backslash bin$, $\backslash examples$, and $\backslash doc$ folders to a directory of their choosing. Developers will also need the files in the $\backslash include$ folder to write custom applications. They should also be aware of the licensing requirements set forth in the license.txt file.

The EPANET-MSX system is supplied as both:

- a stand-alone console application (epanetmsx.exe) that can run standard water quality analyses without any additional programming effort required,

- a function library (epanetmsx.dll) that, when used in conjunction with the original EPANET function library (epanet2.dll), can produce customized programming 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 are provided below.

Regardless of which approach is used, the user must prepare two input files to run a multi-species 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 (Rossman, 2000). 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 Section 4 of this manual.

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 network shown in Figure 3-1 that was adapted from Zhang et al. (2004). Table 3-2 lists the properties associated with the nodes of this network while Table 3-3 does the same for the pipe links.
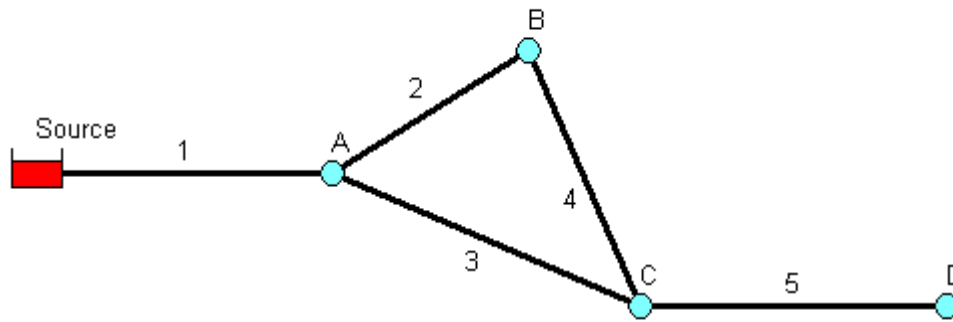
Figure 3-1 Schematic of the example pipe network.

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

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 (6 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 K1   5.0           ;Arsenate adsorption coefficient
  CONSTANT K2   1.0           ;Arsenate desorption coefficient
  CONSTANT Smax 50            ;Arsenate adsorption saturation limit

[TERMS]
  Ks        K1/K2             ;Equil. adsorption coeff.

[PIPES]
  ;Arsenite oxidation
  RATE     AS3    -Ka*AS3*NH2CL
  ;Arsenate production
  RATE     AS5     Ka*AS3*NH2CL - Av*(K1*(Smax-AS5s)*AS5 - K2*AS5s)
  ;Monochloramine decay
  RATE     NH2CL  -Kb*NH2CL
  ;Arsenate adsorption
  EQUIL    AS5s   Ks*Smax*AS5/(1+Ks*AS5) – AS5s
  ;Total bulk arsenic
  FORMULA AStot   AS3 + AS5

[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
```

Figure 3-3 Contents of the *example.msx* input file (Continued on next page)

```
[REPORT]
  NODES    C   D                    ;Report results for nodes C and D
  LINKS    5                        ;Report results for pipe 5
  SPECIES  AStot YES                ;Report results for each species
  SPECIES  AS5   YES
  SPECIES  AS5s  YES
  SPECIES  NH2CL YES
```

Figure 3-3 (Continued)

There are several things of note in this file:

1. The species have been named as follows:
    - AS3 is dissolved arsenite (As^{+3}), expressed in µg/L
    - AS5 is dissolved arsenate (As^{+5}), expressed in µg/L
    - AStot is total dissolved arsenic expressed in µg/L
    - AS5s is adsorbed arsenate, expressed in $µg/m^2$
    - NH2CL is dissolved monochloramine, expressed in mg/L

2. The reaction rate coefficients, Ka and Kb, and the adsorption coefficients, K1, K2, and Smax, 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. Note that the units of these coefficients are as follows: (L/mg-hr) for Ka; (1/hr) for Kb, (L/µg-hr) for K1; (1/hr) for K2; $µg/m^2$ for Smax.

3. The [PIPES] section supplies the three reaction rate expressions and the single equilibrium expression for this system as was presented previously in equations (4) - (7) of Section 2. For example, the rate expression for arsenite oxidation

    $$\frac{dAs^{+3}}{dt} = -k_a As^{+3}\left(NH_2Cl\right)$$

    is expressed in the file as:

            RATE As3 -Ka*As3*NH2CL

    while the equilibrium expression

    $$As_s^{+5} = \frac{k_s S_{max} As^{+5}}{1 + k_s As^{+5}}$$

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

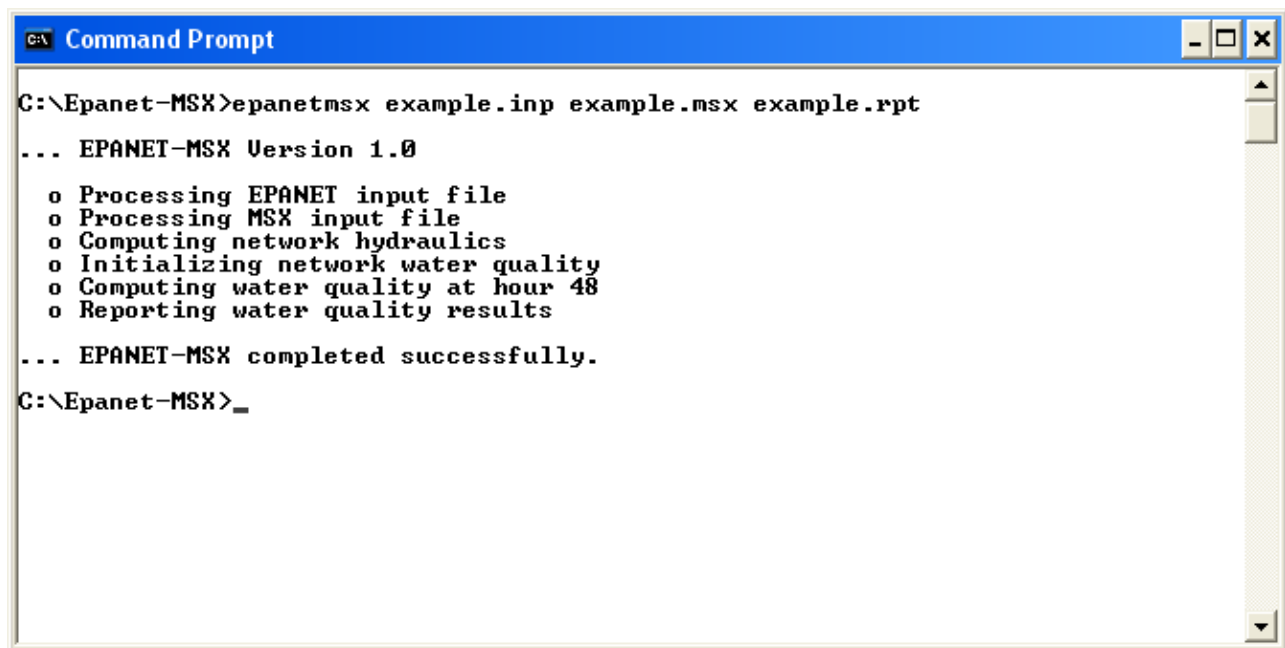   EQUIL As5s Ks*Smax*As5/(1+Ks*As5) - As5s

4. The variable Av that appears in the rate expression for AS5 is a reserved symbol for the pipe wall surface area per unit of pipe volume. It is computed internally by EPANET-MSX and has units of area per liter, where the area units are the same as those specified in the [OPTIONS] section of the file.

5. 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 (Tank water quality reactions can not use wall species, which are associated only with pipes). If only BULK species were present then a redundant [TANKS] section would not be required.

6. 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.

7. 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.

```
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.

```
  Page 1                                          Fri Feb 02 10:00:34 2007

  ****************************************************************
  *                      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           AS5         AStot        NH2CL
  hr:min         UG/L         UG/L         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         0.00         1.10
    10:00        9.17         9.17         1.10
    12:00        9.17         9.17         1.10
    14:00        9.17         9.17         1.10
    16:00        9.17         9.17         1.10
    18:00        9.17         9.17         1.10
    20:00        9.17         9.17         1.10
    22:00        9.17         9.17         1.10
    24:00        9.17         9.17         1.10
    26:00        9.17         9.17         1.10
    28:00        9.17         9.17         1.10
    30:00        9.17         9.17         1.10
    32:00        9.17         9.17         1.10
    34:00        9.17         9.17         1.11
    36:00        9.17         9.17         1.11
    38:00       10.03        10.03         1.11
    40:00       10.03        10.03         1.11
    42:00       10.03        10.03         1.11
    44:00       10.03        10.03         1.11
    46:00       10.03        10.03         1.11
    48:00       10.03        10.03         1.11


  <<< Node D >>>

  Time           AS5         AStot        NH2CL
  hr:min         UG/L         UG/L         MG/L
  -------    ----------   ----------   ----------
     0:00        0.00         0.00         0.00
<Remaining entries not shown>
```

Figure 3-5 Continued.

```
   <<< Link 5 >>>

   Time            AS5         AStot         AS5s        NH2CL
   hr:min         UG/L          UG/L         UG/M2        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          0.00          0.00         0.05
     10:00        0.85          0.85          4.51         0.17
     12:00        1.86          1.86          9.88         0.27
     14:00        2.87          2.87         15.27         0.35
     16:00        3.87          3.87         20.64         0.42
     18:00        4.88          4.88         26.02         0.47
     20:00        5.89          5.89         31.39         0.52
     22:00        6.89          6.89         36.75         0.55
     24:00        7.90          7.90         42.12         0.56
     26:00        8.91          8.91         47.50         0.56
     28:00        9.17          9.17         48.93         0.56
     30:00        9.17          9.17         48.93         0.56
     32:00        9.17          9.17         48.93         0.56
     34:00        9.17          9.17         48.93         0.56
     36:00        9.17          9.17         48.93         0.57
     38:00        9.19          9.19         48.93         0.57
     40:00        9.30          9.30         48.95         0.57
     42:00        9.41          9.41         48.96         0.57
     44:00        9.52          9.52         48.97         0.57
     46:00        9.64          9.64         48.98         0.57
     48:00        9.75          9.75         48.99         0.57

   Analysis ended Sat Jul 14 15:24:38 2007
```

Figure 3-5 Continued.

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 (or a Linux shared object library), such as C, C++, Visual Basic, and Delphi
Pascal. Appendix A describes each function included in the MSX toolkit library. The functions in
the EPANET toolkit library are described in the Help file distributed with that toolkit (available at
www.epa.gov/ORD/NRMRL/wswrd/epanet.html).

As an example of how the library can be used to construct an application, Figure 3-6 displays the C
code behind the command line implementation of the MSX system, *epanetmsx.exe,* which
was just discussed. The listing begins by checking for the correct number of command line
arguments and then attempts to open and read the EPANET file supplied as the first argument. It
uses the EPANET toolkit function *ENopen* for this purpose.

The code then begins a $do-while$ loop that simplifies error detection through the remainder of the program. The $MSXopen$ function is used to open and process the MSX file (the second command line argument), and the $MSXsolveH$ function is used to run a hydraulic analysis of the network whose results are saved to a scratch file. This is followed by a call to $MSXinit$ to initialize the water quality analysis. Note that the argument of 1 to this function tells the MSX system to save its computed water quality results to a scratch file so that they can be used for reporting purposes later on.

The program then begins another do-while loop that will step through each water quality time step. At each such step, the $MSXstep$ function is called to update water quality throughout the network and save these results to the scratch output file. This function also updates the amount of time left in the simulation (stored in the variable named $tleft$). The loop continues until either no more time is left or an error condition is encountered.

If the simulation ends successfully, the $MSXreport$ function is called to write the water quality results to the report file that was named on the command line. If a fourth file name was supplied on the command line, then the $MSXsaveoutfile$ is called to save the results in a binary format to this file. Lastly, the MSX system is closed down by calling $MSXclose$ and the same is done for the EPANET system by calling $ENclose$.

If the source code in Figure 3-6 was saved to a file named $msxepanet.c$ then it could be compiled into an executable named $epanetmsx.exe$ by using the following commands with the Microsoft C/C++ command line compiler:

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

Note that when developing 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. Also, copies of the distributed DLL files epanet2.dll and epanetmsx.dll must be placed in the same directory as the application's executable file or reside in a directory listed the user's PATH environment variable.

Users using other compilers or platforms would need to use the appropriate commands to produce the required object library files and executables.

```
//  EPANETMSX.C  -- Command line implementation of EPANET-MSX

#include <stdlib.h>
#include <stdio.h>
#include "epanet2.h"                    // EPANET toolkit header file
#include "epanetmsx.h"                  // EPANET-MSX toolkit header file

int main(int argc, char *argv[])
/*
**  Purpose:
**    runs a multi-species EPANET analysis
**
**  Input:
**    argc = number of command line arguments
**    argv = array of command line arguments.
**
**  Returns:
**    an error code (or 0 for no error).
**
**  Notes:
**    The command line arguments are:
**     - the name of a regular EPANET input data file
**     - the name of a EPANET-MSX input file
**     - the name of a report file that will contain status
**       messages and output results
**     - optionally, the name of an output file that will
**       contain water quality results in binary format.
*/
{
  int    err, done = 1;
  long   t, tleft;
  long   oldHour, newHour;

// --- Check command line arguments

  if ( argc < 4 ) {
    printf("\n Too few command line arguments.\n");
    return 0;
  }
// --- Open the EPANET file

  printf("\n... EPANET-MSX Version 1.0\n");
  printf("\n  o Processing EPANET input file");
  err = ENopen(argv[1], argv[3], "");
  if (err) {
    printf("\n\n... Cannot read EPANET file; error code = %d\n", err);
    ENclose();
    return 0;
  }
```

Figure 3-6 C source code for the command line version of EPANET-MSX.

```
// ---  Begin an error detection loop

  do {

  // --- Open the MSX input file

    printf("\n  o Processing MSX input file");
    err = MSXopen(argv[2]);
    if (err) {
      printf("\n\n... Cannot read MSX file; error code = %d\n", err);
      break;
    }

  // --- Solve hydraulics

    printf("\n  o Computing network hydraulics");
    err = MSXsolveH();
    if (err) {
      printf(
      "\n\n... Cannot obtain network hydraulics; error code = %d\n", err);
      break;
    }

  // --- Initialize the multi-species analysis

    printf("\n  o Initializing network water quality");
    err = MSXinit(1);
    if (err) {
      printf(
      "\n\n... Cannot initialize EPANET-MSX; error code = %d\n", err);
       break;
    }
    t = 0;
    oldHour = -1;
    newHour = 0;
    printf("\n");

  // --- Repeat for each time step

    do {

    // --- Report current progress
      if ( oldHour != newHour ) {
        printf("\r  o Computing water quality at hour %-4d", newHour);
        oldHour = newHour;
      }

    // --- Compute water quality
      err = MSXstep(&t, &tleft);
      newHour = t / 3600;

    } while (!err && tleft > 0);
```

Figure 3-6 Continued.

```
   // --- Report any runtime error

     if (err) {
       printf("\n\n... EPANET-MSX runtime error; error code = %d\n", err);
       break;
     }
     else
       printf("\r  o Computing water quality at hour %-4d", t/3600);

   // --- Report results

     printf("\n  o Reporting water quality results");
     err = MSXreport();
     if (err) {
       printf(
       "\n\n... MSX report writer error; error code = %d\n", err);
        break;
     }

   // --- Save results to binary file if a file name was provided

     if ( argc >= 5 ) {
       err = MSXsaveoutfile(argv[4]);
       if ( err > 0 ) {
         printf(
         "\n\n... Cannot save MSX results file; error code = %d\n", err);
         break;
       }
     }

// --- End of error detection loop

   } while(!done);

// --- Close both the multi-species & EPANET systems

   MSXclose();
   ENclose();
   if ( !err ) printf("\n\n... EPANET-MSX completed successfully.");
   printf("\n");
   return err;
}
```

Figure 3-6 Continued.

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 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
 COUPLING    FULL/NONE
 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>  ...
 SPECIES     <speciesID>    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.

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	*seconds*
ATOL	*value*
RTOL	*value*

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 rate-dependent 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. Thus the **FULL** coupling option is more accurate, but can require significantly more computation time. 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 species' 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 species name

units species 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) species 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.

- Values for both *Atol* and *Rtol* must be provided to override the default tolerances.

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. The definition and use of TERMS, when those terms are common and appear in multiple rate or equilibrium expressions, may speed computation because the common term expression requires only one evaluation.

Hydraulic variables consist of the following reserved names:
- **D**   pipe diameter (feet or meters)
- **Kc**   pipe roughness coefficient (unitless for Hazen-Williams or Chezy-Manning head loss formulas, millifeet or millimeters for Darcy-Weisbach head loss formula)
- **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**   Surface 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	*specieID*	*expression*
RATE	*specieID*	*expression*
FORMULA	*specieID*	*expression*

Definitions:

specieID   a species identifier

expression   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 requiring one encourages the user 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 species 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 species 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)
```

```
;Bulk C formation plus non-equilibrium sorption between C and Cs
;Using hydraulic variable Av [Area-Units/Liter]
RATE  C   K*C – Av*(K1*(Smax-Cs)*C – K2*Cs)


;Equivalent sorption model, using 1/hydraulic radius = 4/D
;Assumes area units are FT2 and diameter in FT
;CFPL is TERM equal to FT3/Liter, thus (4*CFPL/D) == Av
RATE  C   K*C – (4*CFPL/D)*(K1*(Smax-Cs)*C – K2*Cs)
```

[TANKS]

Purpose:

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

Formats:

EQUIL	*specieID*	*expression*
RATE	*specieID*	*expression*
FORMULA	*specieID*	*expression*

Definitions:

specieID   a species identifier

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

Remarks:

- A **[TANKS]** section is always required when a model contains both bulk and wall species, even when there are no tanks in the pipe network. If the model contains only bulk species, then this section can be omitted if the reaction expressions within tanks are the same as within pipes.

- There should be one expression supplied for each bulk species defined in the model. By definition, wall species do not exist within tanks.

- 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 species 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 species is a simple function of the remaining species.

Examples:

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

Purpose:

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

Formats:

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

Definitions:

sourceType	either **MASS**, **CONCEN**, **FLOWPACED**, or **SETPOINT**
nodeID	the ID label of the network node where the source is located
specieID	a bulk species identifier
strength	the baseline mass inflow rate (mass/minute) for **MASS** sources or concentration (mass/L) for all other source types
patternID	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 species 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 species 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 set point concentration.
- If a time pattern is supplied for the source, it must be one defined in the **[PATTERNS]** section of the MSX file, not a pattern from the associated EPANET input file.

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
```

Purpose:

Specifies the initial concentrations of species throughout the pipe network.

Formats:

GLOBAL	*specieID*	*concen*	
NODE	*nodeID*	*specieID*	*concen*
LINK	*linkID*	*specieID*	*concen*

Definitions:

specieID	a species identifier
nodeID	a network node ID label
linkID	a network link ID label
concen	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.
- All initial concentrations are assumed to be zero unless otherwise specified in this section.
- Models with equilibrium equations will require that reasonable initial conditions be set so that the equations are solvable. For example, if they contain a ratio of species concentrations then a divide by zero condition will occur if all initial concentrations are set to zero.

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  N100  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:

pipeID   the ID label of a pipe link in the network

tankID   the ID label of a tank node in the network

paramID   the name of one of the reaction rate parameters listed in the `[COEFFICIENTS]` section

value   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        node1      node2   ...
LINKS        ALL
LINKS        link1      link2   ...
SPECIES      speciesID  YES/NO  (precision)
FILE         filename
PAGESIZE     lines
```

Definitions:

node1, node2, etc.	a list of nodes whose results are to be reported
link1, link2, etc.	a list of links whose results are to be reported
specieID	the name of a species to be reported on
precision	number of decimal places used to report a species' concentration
filename	the name of a file to which the report will be written
lines	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 **SPECIES** 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 nodal results for species S1 and S2 using
;4 decimal places to the standard EPANET report file
SPECIES  S1  YES  4
SPECIES  S2  YES  4
NODES    ALL
```

5. EXAMPLE REACTION SYSTEMS

This section demonstrates how several different multi-species reaction systems of interest can be modeled 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).
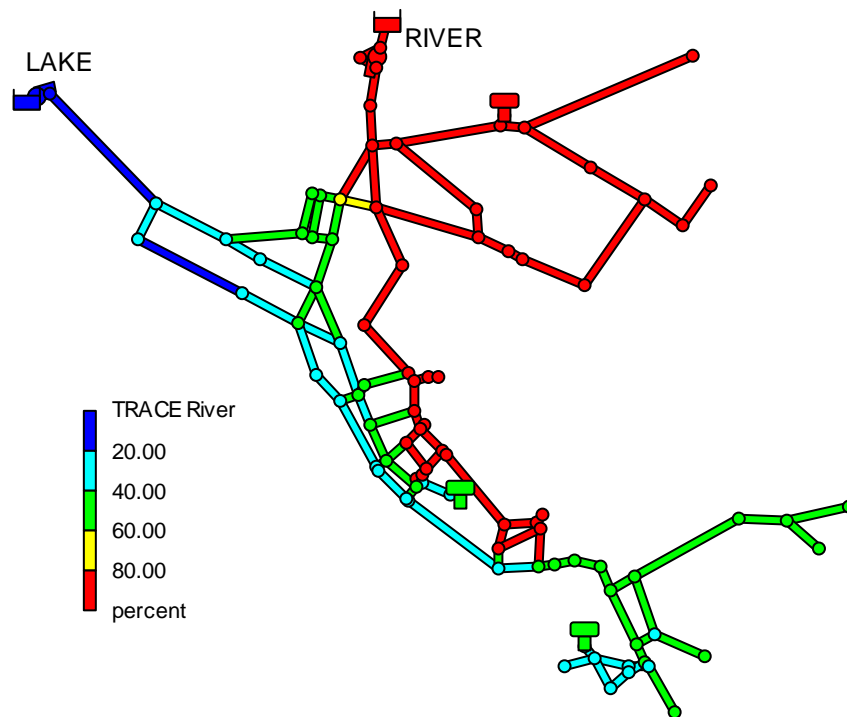
Figure 5-1 Example of a two-source water distribution system showing the average percent of water originating from the River source.

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 days^{-1} while Source 2's water has k = 17.7 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 total organics from the two sources, is to assume that at any time 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 conservative tracer compound at Source 1, denoted as T1, whose concentration is fixed at a constant 1.0 mg/L. Then at any point 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 C. The first-order decay constant k for any pipe in the system would be given by:

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

while the system reaction dynamics would be expressed by:

$$\frac{d\text{T1}}{dt} = 0 \tag{15}$$

$$\frac{dC}{dt} = -\left(1.3\,\text{T1} + 17.7\left(1.0 - \text{T1}\right)\right)C \tag{16}$$

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.

```
[OPTIONS]
  AREA_UNITS  FT2
  RATE_UNITS  HR
  SOLVER      RK5
  TIMESTEP    300

[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 and more complete description 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. We also include 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 the arsenic model. Note that after arsenate is produced by the oxidation of arsenite in the bulk solution it diffuses through a boundary layer to reach a concentration denoted as $As^{+5}|_W$ just adjacent to the pipe wall. It is this concentration that interacts with adsorbed arsenate $As^{+5}|_S$ on the pipe wall. Thus the system contains five 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 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.
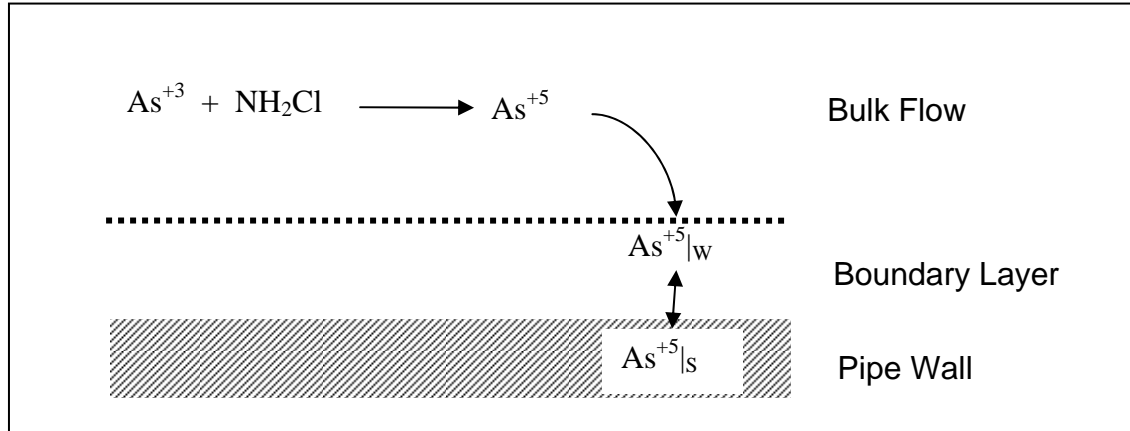
Figure 5-3 Schematic of the mass transfer limited arsenic oxidation/adsorption system.

The mathematical form of this reaction system can be modeled with five differential rate equations in the case of non-equilibrium adsorption/desorption (see, e.g., Gu et al., 1994, for a more complete description of non-equilibrium adsorption/desorption):

$$\frac{dAs^{+3}}{dt} = -k_a As^{+3}(NH_2Cl) \tag{17}$$

$$\frac{d(NH_2Cl)}{dt} = -k_b(NH_2Cl) \tag{18}$$

$$\frac{dAs^{+5}}{dt} = k_a As^{+3}(NH_2Cl) - K_f A_v(As^{+5} - As_w^{+5}) \tag{19}$$

$$\frac{dAs_w^{+5}}{dt} = K_f A_v(As^{+5} - As_w^{+5}) - A_v[k_1(S_{max} - As_s^{+5})As_w^{+5} - k_2 As_s^{+5}] \tag{20}$$

$$\frac{dAs_s^{+5}}{dt} = k_1(S_{max} - As_s^{+5})As_w^{+5} - k_2 As_s^{+5} \tag{21}$$

where As^{+3} is the bulk phase concentration of arsenite, As^{+5} is the bulk phase concentration of arsenate, As_w^{+5} is the bulk phase concentration of arsenate adjacent to the pipe wall, As_s^{+5} is surface phase concentration of arsenate, and NH_2Cl 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), A_v is the pipe surface area per liter of pipe volume, k_1 and k_2 are the arsenate adsorption and desorption rate coefficients, S_{max} is the maximum pipe surface concentration, and K_f is a mass transfer rate coefficient. 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:

$$K_f = \frac{1.6 \times 10^{-4} \, Re^{0.88}}{D}$$ (22)

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

Using the notation defined in eqs. (1)-(3), $\mathbf{x}_b = \{As^{+3}, As^{+5}, As_w^{+5}, NH_2Cl\}$, $\mathbf{x}_s = \{As_s^{+5}\}$, $\mathbf{z}_b = \{\varnothing\}$, $\mathbf{z}_s = \{\varnothing\}$, and $\mathbf{p} = \{k_a, k_b, A_v, k_1, k_2, K_f, S_{max}\}$. The reaction dynamics defined by (17)-(21) conserves total arsenic mass within any pipe segment of length L (and thus bulk volume A×L, and pipe surface area P×L, where A and P are cross sectional area and wetted perimeter, respectively). This can be shown by summing the differential changes in the mass of all arsenic species within a pipe segment, and assuring that they sum to zero: (A×L) (dAs^{+3}/dt) + (A×L) (dAs^{+5}/dt) + (A×L) (dAs_w^{+5}/dt) + (P×L) (dAs_s^{+5}/dt) = 0.

It was mentioned in section 2 that some reactions are reversible and fast enough in comparison with the system's other processes so that a local equilibrium can be assumed, while others are not sufficiently fast and/or irreversible and it is inappropriate to use an equilibrium formulation to represent them. In the case of reversible sorption processes a local equilibrium assumption (LEA) is sometimes assumed between the adsorbed phase and its concentration in the bulk fluid (such an assumption is not always realistic, and no such claim is made here – see, e.g., Koopman et al. (1992)). Under the LEA, the rates of adsorption and desorption must be much faster than the rate of change of the bulk species, and consequently, $dAs_s^{+5}/dt = 0$; the differential equation (21) can then be substituted with the algebraic equation:

$$As_s^{+5} = \frac{k_s S_{max} As_w^{+5}}{1 + k_s As_w^{+5}},$$ (21b)

where $k_s = k_1/k_2$. In this case the DAE system is described by (17)-(20) and (21b), and includes four differential rate equations and one algebraic equation. Thus in terms of the notation used in (1)-(3), $\mathbf{x}_b = \{As^{+3}, As^{+5}, As_w^{+5}, NH_2Cl\}$, $\mathbf{x}_s = \{\varnothing\}$, $\mathbf{z}_b = \{\varnothing\}$, $\mathbf{z}_s = \{As_s^{+5}\}$, $\mathbf{p} = \{k_a, k_b, A_v, k_1, k_2, K_f, S_{max}\}$. The LEA model could be left in this form that explicitly includes the surface adsorbed phase (As_s^{+5}) – or, because the special form of (21b) presents an explicit solution for the adsorbed arsenate, (21b) could be substituted into (20), leaving an equivalent system of only four differential rate expressions in the four bulk-phase variables $\mathbf{x}_b = \{As^{+3}, As^{+5}, As_w^{+5}, NH_2Cl\}$. In this latter case, however, the adsorbed phase arsenate concentration would have to be separately calculated after solution (if desired), using (21b), which could be laborious if it were required at many locations and time steps.

Figure 5-4 shows the MSX input file for this system, with the assumption of local equilibrium (eqs. (17)-(20) and (21b)). The [PIPES] section contains four kinetic rate reactions involving the four bulk species and one equilibrium reaction that includes the lone surface species. The [TANKS] section contains only the bulk species reactions. To complete the model specification, the [QUALITY] section assumes that the network has a single source which is a reservoir node

labeled "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  Smax 50                   ;Arsenate adsorption coeff.
  CONSTANT  K1   5.0                  ;Arsenate adsorption rate coeff.
  CONSTANT  K2   1.0                  ;Arsenate desorption rate coeff.

[TERMS]
  Kf     1.6e-4*Re^0.88/D             ;Mass transfer coefficient (Ft/Hr)
  Ks     K1/K2                        ;Arsenate equil. adsorption coeff.
  LPERCF 28.316                       ;Liters/Cu. Ft. (conversion for Kf)

[PIPES]
  RATE  A3    -Ka*A3*NH2CL                       ;Arsenite oxidation
                                                 ;Arsenate production
  RATE  A5    Ka*A3*NH2CL – LPERCF*Kf*Av*(A5 – A5w)
                                                 ;Arsenate at pipe wall
  RATE  A5w   LPERCF*Kf*Av*(A5 – A5w) – Av*(K1*(Smax-A5s)*A5w – K2*A5s)
  RATE  NH2CL –Kb*NH2CL                          ;Monochloramine oxidation
  EQUIL A5s   Smax*Ks*A5w/(1.0 + Ks*A5w) – A5s ;Arsenate adsorption

[TANKS]
  RATE  A3    -Ka*A3*NH2CL                       ;Arsenite oxidation
  RATE  A5    Ka*A3*NH2CL                        ;Arsenate w/o mass tran.
  RATE  A5w   0                                  ;Not present in a tank
  RATE  NH2CL –Kb*NH2CL                          ;Monochloramine oxidation

[QUALITY]
  ;Initial conditions (= 0 if not specified here)
  NODE  1      A3   10.0
  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.

Note about equation units. The modeler must understand the units of all quantities used in mathematical expressions, and ensure that the resulting expressions are dimensionally correct. In short, a complete unit analysis for each expression and resulting RATE or EQUIL term is an essential quality assurance step for developing MSX models. As an example, consider eq. (20) above, repeated here for convenience:

$$\frac{dAs_w^{+5}}{dt} = K_f A_v \left(As^{+5} - As_w^{+5} \right) - A_v [k_1 (S_{max} - As_s^{+5}) As_w^{+5} - k_2 As_s^{+5}]$$

The units of the left hand side are the units of the bulk species, As_w^{+5}, divided by the units used for time. Since the default time unit is hours, the left hand side has units of [µg/L-hr], and consequently each of the two main terms on the right hand side must have these same units. Since A_v was previously defined as the pipe surface area per liter pipe volume, and the default area unit is [ft^2], this quantity has units of [ft^2/L] (A_v is a pre-defined MSX hydraulic variable – see the description of the [TERMS] input in section 4 of this manual). From this analysis we find that the mass transfer coefficient K_f in the first term on the right hand side must have units of [L/ft^2-hr]. The units of the previous expression for K_f in (22) are, however, the more conventional velocity units of [ft/hr], requiring multiplication by the number of liters per cubic ft: [ft/hr] $\times$ [L/ ft^3] = [L/ft^2-hr]. This unit analysis explains the presence of the term LPERCF in the above MSX input file; without this term the expressions would not be dimensionally consistent and the results would be invalid. The reader should verify that the second term on the right hand side is dimensionally correct – given the units of A_v – provided that rate coefficient k_1 has units [L/µg-hr], and k_2 has units [1/hr] (recall that the surface species has units of [µg/ft^2], since we are using the default area units).

Bacterial Regrowth with Chlorine Inhibition

This next example models bacterial regrowth as affected by chlorine inhibition within a distribution system. The regrowth model is taken from Zhang et al. (2004) and includes the following processes depicted in Figure 5-5:
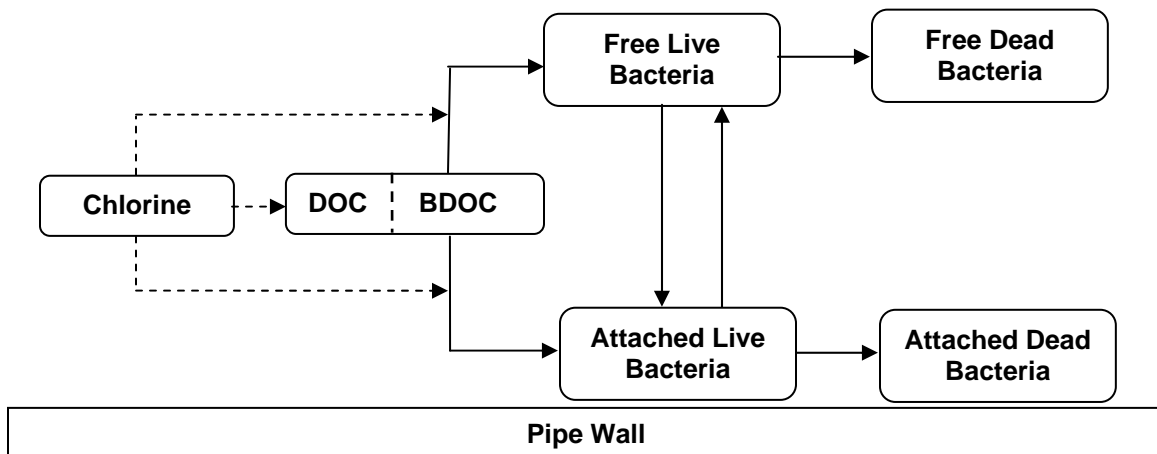

Figure 5-5 Conceptual diagram of bacterial regrowth within a pipeline. Dashed arrows represent reactions with chlorine while solid arrows represent transformation processes.

a. 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:

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

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

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 with the BDOC concentration according to:

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

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:

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

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:

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

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

$$\left.\frac{dX}{dt}\right|_{det\,achment} = k_{det} X U$$

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 I to the bacterial specific growth rate as:

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

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.

```
[OPTIONS]
AREA_UNITS  CM2
RATE_UNITS  HR
SOLVER      RK5
TIMESTEP    300


[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


[COEFFICIENTS]
CONSTANT  Kb       0.05   ;CL2 decay constant (1/hr)
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/(ft/s))
CONSTANT  Kdep     0.08   ;deposition rate constant (1/hr)
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)


[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.

Several notes of explanation require mentioning:

1. There are six species defined for the model: bulk chlorine (CL2), bulk biodegradable dissolved organic carbon (S), bulk bacterial concentration (Xb), bulk bacterial cell count (Nb), 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-workers (Vikesland et al., 2001; Duirk et al., 2005) and is shown in Table 5-1. The principal species are hypochlorous acid (HOCl), hypochlorite ion (OCL^-), ammonia (NH_3), ammonium ion (NH_4^+), monochloramine (NH_2Cl), dichloramine ($NHCl_2$), 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 NH_2Cl 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 5-1. 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 sub-system 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 $CaCO_3$ while the H^+ value of 2.818×10^{-8} moles/L is the same as a pH of 7.75.)

Table 5-1 Monochloramine decay model based on Vikesland et al. (2001) and Duirk et al. (2005).

	Reaction Stoichiometry	Rate Coefficient/ Equilibrium Constant[a]
R.1	$HOCl + NH_3 \rightarrow NH_2Cl + H_2O$	$k_1 = 1.5 \times 10^{10}\ M^{-1}h^{-1}$
R.2	$NH_2Cl + H_2O \rightarrow HOCl + NH_3$	$k_2 = 7.6 \times 10^{-2}\ h^{-1}$
R.3	$HOCl + NH_2Cl \rightarrow NHCl_2 + H_2O$	$k_3 = 1.0 \times 10^{6}\ M^{-1}h^{-1}$
R.4	$NHCl_2 + H_2O \rightarrow HOCl + NH_2Cl$	$k_4 = 2.3 \times 10^{-3}\ h^{-1}$
R.5	$NH_2Cl + NH_2Cl \rightarrow NHCl_2 + NH_3$	$k_5 = 2.5 \times 10^{7}\ [H^+] +$ $4.0 \times 10^{4}\ [H_2CO_3] +$ $800\ [HCO_3^-]\ M^{-2}h^{-1}$
R.6	$NHCl_2 + NH_3 \rightarrow NH_2Cl + NH_2Cl$	$k_6 = 2.2 \times 10^{8}\ M^{-2}h^{-1}$
R.7	$NHCl_2 + H_2O \rightarrow I$	$k_7 = 4.0 \times 10^{5}\ M^{-1}h^{-1}$
R.8	$I + NHCl_2 \rightarrow HOCl + products$	$k_8 = 1.0 \times 10^{8}\ M^{-1}h^{-1}$
R.9	$I + NH_2Cl \rightarrow products$	$k_9 = 3.0 \times 10^{7}\ M^{-1}h^{-1}$
R.10	$NH_2Cl + NHCl_2 \rightarrow products$	$k_{10} = 55.0\ M^{-1}h^{-1}$
R.11	$NH2Cl + S_1 \times TOC \rightarrow products$[b]	$k_{11} = 3.0 \times 10^{4}\ M^{-1}h^{-1}$ $S_1 = 0.02$
R.12	$HOCl + S_2 \times TOC \rightarrow products$[c]	$k_{12} = 6.5 \times 10^{5}\ M^{-1}h^{-1}$ $S_2 = 0.5$
E.1	$HOCl \leftrightarrow H^+ + OCl^-$	$pK_a = 7.5$
E.2	$NH_4^+ \leftrightarrow NH_3 + H^+$	$pK_a = 9.3$
E.3	$H_2CO_3 \leftrightarrow HCO_3^- + H^+$	$pK_a = 6.3$
E.4	$HCO_3^- \leftrightarrow CO_3^{2-} + H^+$	$pK_a = 10.3$

Notes:
   a. All rate coefficients and equilibrium constants are for 25 degrees C.
   b. S_1 is the fast reactive fraction of TOC.
   c. S_2 is the slow reactive fraction of TOC.

```
[OPTIONS]
RATE_UNITS  HR
SOLVER      ROS2
COUPLING    NONE
TIMESTEP    300
RTOL        0.0001
ATOL        1.0e-8


[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]
PARAMETER        k1    1.5e10
PARAMETER        k2    7.6e-2
PARAMETER        k3    1.0e6
PARAMETER        k4    2.3e-3
PARAMETER        k6    2.2e8
PARAMETER        k7    4.0e5
PARAMETER        k8    1.0e8
PARAMETER        k9    3.0e7
PARAMETER        k10   55.0

PARAMETER        kDOC1 3.0E4
PARAMETER        kDOC2 6.5E5
CONSTANT         S1    0.02
CONSTANT         S2    0.50


[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
```

Figure 5-7 MSX input file of the monochloramine decomposition model.

```
[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 Continued.

6. REFERENCES

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

Golub, G. and Ortega, J., *Scientific computing and ordinary differential equations*, Academic press, 1992.

Gu, B; Schmitt, J.; Chen, Z; Liang, L; McCarthy, J (1994). "Adsorption and desorption of natural organic matter on iron oxide: mechanisms and models," *Environ. Sci. Technol.*; 28(1); 38-46.

Hairer, E.; Norsett, S.P.; Warner G. (1993). *Solving Ordinary Differential Equations I. Nonstiff Problems*, 2nd edition, Springer Series in Computational Mathematics, Springer-Verlag.

Koopman, D; Cole, J; Lee, H. (1992). "Assumption of Local Equilibrium in Adsorption Processes", *AIChE J.*, 38(4).

Press, W.H., Flannery, B.P., Teukolsky, S.A., and Vetterling, W.T. (1992). *Numerical Recipes in C,* 2nd Edition, Cambridge University Press, Cambridge.

Rossman, L.A. (1999). "The EPANET Programmer's Toolkit for Analysis of Water Distribution Systems", *Proc. 26th Annual Water Resources Planning and Management Conference,* (E.M. Wilson, editor), American Society of Civil Engineers, Reston, VA.

Rossman, L.A. (2000) *EPANET 2 Users Manual.* EPA/600/R-00/057, U.S. Environmental Protection Agency, Cincinnati, OH.

Verwer, J.G.; Spee, E.J.; Blom, J.G.; Hundsdorfer, W.H. (1999). "A second order Rosenbrock method applied to photochemical dispersion problems", *SIAM J. Sci. Comput.,* 20, 1456-1480.

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

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

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. 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. Tables A-1, A-2, and 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.

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.
MSXgetspecies	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);
```

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();
```

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
...
```

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();
```

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`.

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.

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
...
```

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
...
```

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_SPECIES` (for a chemical species),
   `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 species named "CL2"
MSXgetindex(MSX_SPECIES, "CL2", &i);
```

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_SPECIES` (for a chemical species),
    `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, not counting the *null* termination character.

Returns:

Returns an error code or 0 for no error.

Example:

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

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

//Open the EPANET & MSX toolkits
...

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

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

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_SPECIES (for a chemical species),
    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, not counting the *null* termination character...

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 (to len + 1).

Example:

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

//Open the EPANET & MSX toolkits
...

//Get the name of the 2nd species in the MSX input file
MSXgetID(MSX_SPECIES, 2, id, sizeof(id)-1);
```

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_SPECIES` (for a chemical species),
  `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_SPECIES, &nSpecies);
```

Declaration:

```
int MSXgetspecies(int species, int * type, char * units,
                   double * aTol, double * rTol);
```

Description:

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

Arguments:

`species` is the sequence number of the species (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 species,
  MSX_WALL (defined as 1) for a pipe wall surface species;

`units` is a C-style character string array that is returned with the mass units that were defined for the species in question. It must be sized to hold a maximum of 15 characters plus the terminating null character (for a total of 16).

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

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

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 species named "Xwall"
MSXgetindex(MSX_SPECIES, "Xwall", &sIndex);
MSXgetspecie(sIndex, &sType, &sUnits, &aTol, &rTol);
```

Declaration:

```
int MSXgetinitqual(int obj, int index, int species,
                    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;

`species` is the sequence number of the species (starting from 1);

`value` is returned with the initial concentration of the species 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_SPECIES, "CL2", &s);
MSXgetinitqual(MSX_NODE, n, s, &c0);
```

Declaration:

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

Purpose:

Retrieves a chemical species concentration at a given node or the average concentration along a link 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;

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

`value` is returned with the computed concentration of the species 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 species "CL2"
ENgetnodeindex("Tank_A", &n);
MSXgetindex(MSX_SPECIES, "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);
```

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);
```

Declaration:

```
int MSXgetsource(int node, int species, 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;

`species` is the sequence number of the species 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:

`MSX_NOSOURCE`	(defined as -1) for no source,
`MSX_CONCEN`	(defined as 0) for a concentration source,
`MSX_MASS`	(defined as 1) for a mass booster source,
`MSX_SETPOINT`	(defined as 2) for a setpoint source,
`MSX_FLOWPACED`	(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_SPECIES, "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);
```

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);
```

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;
```

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);
```

Declaration:

```
int MSXsetinitqual(int type, int index, int species,
                   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;

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

`value` is the initial concentration of the species 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 species "CL2"
ENgetnodeindex("Tank_A", &n);
MSXgetindex(MSX_SPECIES, "CL2", &s);

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

Declaration:

```
int MSXsetsource(int node, int species, 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;

`species` is the sequence number of the species 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:

MSX_NOSOURCE	(defined as -1) for no source,
MSX_CONCEN	(defined as 0) for a concentration source,
MSX_MASS	(defined as 1) for a mass booster source,
MSX_SETPOINT	(defined as 2) for a setpoint source,
MSX_FLOWPACED	(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_SPECIES, "CL2", &s);

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

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 int 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);
}
```

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
Species 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 200:   **Cannot read EPANET-MSX file.**
One or more errors were found when processing the MSX input file. See the report output file for details.

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 exisits 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.

Error 520:   **an MSX project is already opened.**
A call was made to open an MSX project without having first closed a currently opened project with the `MSXclose` function.

SC
IEN
CE

EPA

United States
Environmental Protection
Agency

Office of Research and Development
National Homeland Security Research Center
Cincinnati, OH 45268

Official Business
Penalty for Private Use
$300

Recycled/Recyclable
Printed with vegetable-based ink on
paper that contains a minimum of
50% post-consumer fiber content
processed chlorine free