



CSIRO
AUSTRALIA

***ARX:* A SPATIAL EXPERT SYSTEM SHELL FOR MODELLING ENVIRONMENTAL PROBLEMS**

USER'S MANUAL

By Peter Whigham

TECHNICAL MEMORANDUM 93/21

December 1993

Division of Water Resources

**ARX: A SPATIAL EXPERT SYSTEM SHELL
FOR MODELLING ENVIRONMENTAL PROBLEMS**

USER'S MANUAL

By Peter Whigham

Division of Water Resources, Canberra Laboratory

Technical Memorandum 93/21

December 1993

CSIRO

Institute of Natural Resources and Environment

Division of Water Resources

ISBN 0 643 05518 5

Publications enquiries to:

**Divisional Editor
CSIRO Division of Water Resources
GPO Box 1666
Canberra ACT 2601 Australia
ph. (06) 246 5717
fax (06) 246 5800**

ABSTRACT

ARX is a spatial expert system shell which has been developed specifically to represent the spatial information commonly expressed by experts in environmental management and to infer valid conclusions from such (2-D) information.

ARX has been used to develop applications for the Australian Army to predict vehicle movement in the north of Australia and environmental damage from Army training exercises at Puckapunyal Army Base, Victoria. The program exists in several versions. This manual relates to the version connected to ARC/INFO, a commercial geographic information system software package, and describes how to build a knowledge base and the link with ARC/INFO.

The program has been developed as a research tool and, as such, does not comply with Australian software development standards.

TABLE OF CONTENTS

1	Introduction	1
1.1	Companion Documents	1
2	Basic Concepts	2
2.1	An Introduction to Expert Systems	2
2.2	ARX - A Spatial Expert System	4
2.3	Spatial Relationships	5
2.4	Rules	5
2.5	Equations	6
2.6	Attribute Data	6
2.7	The Inference Engine	7
3	The ARX - ARC/INFO Linkage	8
3.1	ARX and the INFO Database	8
3.2	ARX and ARC Mapping Routines	9
4	The Knowledge Base	10
4.1	The Parameter File (.par) (Object Definitions)	10
4.2	The Linkage File (.lnk)	12
4.3	The Rule Base File (.rul)	14
4.4	The Model File (.mod)	18
4.5	User Defined Functions (.fun)	20
4.6	The External Program File (.ext)	20
4.7	The ARX Data-File	21
5	ARX Script Commands	23
APPENDIX A		
	BNF Notation Syntax for Spatial Expressions	24
APPENDIX B		
	Knowledge Base to Predict Spread of Fire	26
B.1	The Parameter File (.par)	26
B.2	The Linkage File (.lnk)	27
B.3	The Model File (.mod)	28
B.4	The Rule File (.rul)	28
APPENDIX C		
	Examples of Script Files	36
C.1	Script to Define a Map Display Window ..	36
C.2	Script to Create an ARX Script Editor	39

1 Introduction

1

This document describes the user's view of *ARX*. Using this document it will be possible to integrate ARC/INFO¹ spatial data with *ARX*, and perform spatial modelling in an expert system environment.

Section 2 introduces features of expert systems and describes the expert system components which have been implemented in *ARX*. This version of *ARX* has been linked to ARC/INFO to provide mapping facilities and these links are described in Section 3. The process of developing a knowledge base and testing it using *ARX* is described in Section 4. A high level script language has been added to *ARX* to aid the rapid development of interfaces to the knowledge base and control facilities of *ARX*. These commands are introduced in Section 5.

ARX has been developed in response to research needs and does not have all the features (or the robustness) of a fully commercial software package. However all the commands and features documented in this report have been tested and have proved reliable.

1.1 Companion Documents

- *ARX* Command Reference Manual

¹ ARC/INFO is a commercial geographic information system (GIS) developed and distributed by ESRI. The version referred to here is V 5.x for SUN workstations using the SUNTOOLS graphics interface running under Unix. The latest version of ARC/INFO, v6.x, uses the OPEN LOOK graphics interface. *ARX* is not presently available for this environment.

2 Basic Concepts

2

2.1 An Introduction to Expert Systems

Expert systems attempt to mimic the deductive or inductive reasoning of a human expert. They belong to a group of information processing programs that grew out of Artificial Intelligence research in the late 1960's. Since the mid-1970s, these systems have been employed in a variety of problem domains, of which perhaps the most successful have been medicine, computing, and mineral prospecting.

FEATURES

Their success has been due largely to a number of distinctive features. First, there are a wide variety of problems where the knowledge is essentially qualitative rather than quantitative, and consequently are better solved with inference rather than calculation. Second, generic expert system development environments, called shell programs, are now widely available. Third, the separation of the 'knowledge base' from the mechanism for applying this knowledge (the 'inference engine'), together with special purpose editors, allows problem-specific knowledge to be incorporated and modified by non-programmers. Fourth, they employ an English-like syntax for representing the qualitative knowledge. Fifth, they are able to explain any answers that they advance - this is probably one of their most powerful features from a user's point of view. In short, they offer a modelling environment which can be easily modified, understood and used by those not literate in a conventional computer language.

Expert systems typically consist of two major parts - the knowledge base and the inference engine. The knowledge base, in turn, consists of:

KNOWLEDGE BASE

- **Parameters** which are representations of the objects being modelled

- **Databases** which store facts about the domain being modelled
- **Rule Base** which is a set of relationships (rules) representing empirical and causal processes operating in the domain

The inference engine contains:

- **Editors** which are mechanisms for creating and modifying the above components of the knowledge base
- **Inference Engine** which is a mechanism for drawing logically valid inferences using the above knowledge
- **Explanation Engine** which is a dialogue mechanism to explain the results of an inference, and to show why certain conditions did or did not occur.

CONSULTATION

The operation of an engine is referred to as a consultation. Most inference engines can operate in either a goal seeking or a fact finding mode.

BACKWARD CHAINING

In goal seeking mode (backward chaining) the inference engine is given a goal parameter (ie. parameter whose value is sought) and it attempts to determine the premise truth of any rule which concludes a value for this parameter. This may lead to a chain of inference, given that in attempting to determine the truth of a premise rule, further rules or database operations may be applied.

FORWARD CHAINING

In fact finding mode (forward chaining), the inference engine asserts the conclusions of all rules whose premises are true. The engine usually repeats (cycles) this operation until a specified number of cycles is exceeded, a particular parameter value has been determined, or no further facts can be asserted.

TRACE

The inference engine keeps a trace of the rules that have been 'fired' and this trace is used by the explanation module to explain how the values

have been determined during the consultation (and possibly to show why certain results were not obtained).

2.2 ARX - A Spatial Expert System

ARX PREDECESSORS *ARX* developed from a series of expert systems to aid managers of natural resource agencies developed within CSIRO Division of Water Resources since 1985. The design of the first program in the series, GEM, was based on the architecture of the EMYCIN² shell. It was apparent at an early stage in the development of these systems that the language provided by EMYCIN needed extension in order to represent the natural and human processes being modelled. One such need was the ability to represent space since many natural resource processes are understood in terms of spatially extensive effects.

Another need was the ability to incorporate both rules and equations in the knowledge base since natural resource processes are often modelled using a mixture of qualitative and quantitative knowledge.

ARX FEATURES The syntax of EMYCIN has been extended considerably in *ARX* to include the representation of spatial processes and indeterminate processes. *ARX* can undertake both backward and forward chaining and can map the results of consultations. Four types of knowledge about a process may be incorporated in *ARX*:

- spatial relationships
- empirical and process knowledge in the form of rules
- mathematical knowledge in the form of equations

² The EMYCIN shell (van Melle, 1979) is a generalisation of the MYCIN medical diagnostic program (Shortliffe, 1976) where a set of symptoms constitutes an instance of the problem and the goal states the range of possible diseases known to the program. EMYCIN was developed to handle similar classification problems in environmental management.

Shortliffe, E.H. (1976). Computer-based medical consultations: MYCIN, New York, Elsevier.

van Melle, W. (1979) A domain-independent production-rule system for consultation programmes, Proc. 6th Int. Conf. Artif. Int., Tokyo, Stanford, Stanford University Press, 923-25.

- facts (attribute data) in the form of database tables

2.3 Spatial Relationships

ARX represents space as a set of spatial domains, each composed of regions (spatial items). At any stage of a consultation, *ARX* is working in one region (called the 'current region') of a particular domain (called the 'current domain').

The user can identify part of a domain by using a <spatial expression> which includes one or more region names and a domain. The regions can be identified explicitly or implicitly, for example:

- For All "P5" "P6" "P7" <Myplace>
- For Any <Myplace> within 3 km <Yourplace>

The first example identifies three explicit regions (called P5, P6 and P7) in the Myplace domain. The latter identifies all (implicit) regions in the Myplace domain that lie within a distance of 3 km of the 'current region' in the Yourplace domain. Note that the current region is the default region in the second example.

2.4 Rules

Each rule consists of a condition and conclusion. Each condition and each conclusion is phrased as a combination of quadruplets of the form

FORMAT

<parameter> <relation> <expression> <spatial expression>

PARAMETERS

where <parameter> is a parameter name, <relation> is one of the relations listed in Section 4.3, <expression> is another parameter name, a value or a mathematical expression and the <spatial expression> identifies regions where the <parameter> <relation> <expression> triplet is to be evaluated.

Parameters can be any of the six types described in Section 4.1. This type helps *ARX* check for consistency and conflict during a consultation.

SOURCE LIST

Associated with each parameter is a source list, which directs the inference engine to the appropriate part of the knowledge base when attempting to determine a value for a parameter. The source list refers to a particular database table, a set of equations, a set of rules, or the user (who, from the point of view of the inference engine, is treated as another source of knowledge).

Some examples of rules are provided in Appendix B.

2.5 Equations

Quadruplets may contain a mathematical expression in the **<expression>** component. Parameters are used as variables in such mathematical expressions and standard mathematical functions (e.g. trig, exp, log) can be included together with user defined functions.

2.6 Attribute Data

Parameters whose values are relatively constant in specific domain/region combinations can be stored in a spatial database. This links the mapping topology with the attribute data. Values inferred during a consultation can be stored in the database for later analyses. The special value 'unknown' is stored for any parameter whose value is not known when the database is developed or is not determined during the consultation.

2.7 The Inference Engine

The inference engine draws upon the parameter descriptions, the rules, the equations and the map/attribute data in order to infer and calculate values for parameters in selected domains and regions. The user can select either a backward or forward chaining strategy to determine a solution to problems. At the commencement of a consultation, the user provides the set of parameters whose values are of interest and the domain and regions where these values are required. The inference engine makes each parameter/domain/ region combination the 'current' parameter/domain/region in turn, and attempts to determine a value for this combination using the sources identified in the parameter source list.

3 The ARX - ARC/INFO Linkage

3

ARC/INFO³ combines a standard tabular database (INFO) with a map representation/manipulation facility (ARC). The standard ARC concept of space is the coverage (or <layer>), which represents a map with features of varying types (feature-classes are points, lines and polygons). Operations over a coverage are generally applied to one feature class of the coverage. Attributes can be assigned to a coverage which may be viewed as a set of values associated with each member of the <layer> <feature-class> collection.

The ARX domain represents a homogeneous class of attributes, and is readily mapped into the corresponding ARC/INFO <layer> <feature-class> pair. Note that one ARC/INFO map coverage may be identified with several ARX domains, covering each feature-class of the coverage.

The integrated ARX-ARC/INFO system draws upon the INFO database, the ARC mapping routines and the spatial searches available in ARC.

3.1 ARX and the INFO Database

Each ARX region is identified with a member of the corresponding ARC/INFO <layer> <feature-class>. This identification is achieved by reference to an internal INFO record number or by selecting an attribute column in the INFO database which contains a unique identifier for each record.

Each ARX parameter (database source) definition includes an alias value which links the parameter with a column in the INFO database. If the

³ This Section assumes the reader has knowledge of ARC/INFO.

source list for the parameter indicates that values for this parameter are stored in the database, then the alias is used to access the current <layer> <feature-class> table and to extract the record associated with the current region in ARX. Type conversions between the ARC/INFO data and ARX internal data are automatically handled, since the parameter type and ARC/INFO data type are both known to ARX.

ARX allows users to create new INFO files to store parameter values inferred during a consultation. The parameter to be stored is given an INFO column name and type, with a given ARX domain used to reference the INFO file for the related values and type information. ARX uses the currently inferred (determined) values for the parameter in each region of the selected domain to create an INFO file. The new file may be used as a separate INFO file for tabular operations or, if the user specified a related field, can be later joined to an existing coverage.

3.2 ARX and ARC Mapping Routines

The INFO files created after a consultation are usually joined to the ARC file corresponding to the particular <feature-class> of the coverage used by ARX. This allows ARC/PLOT, the mapping module of ARC/INFO, to display and manipulate the results of the ARX consultation in the usual way.

4 The Knowledge Base

4

An *ARX* knowledge base may consist of many different files which contain the components that together define the model to be executed. The following file names are used to define a knowledge base:

KNOWLEDGE BASE COMPONENTS

<Directory>

- .par** the parameter definition file which contains the object definitions for a knowledge base.
- .lnk** the linkage file that defines the spatial definitions used by *ARX*. These represent mapping between ARC/INFO coverages and *ARX* internal spatial domains.
- .rul** the rule set for a knowledge base.
- .mod** the model set (mathematical equations and external program calls) for a knowledge base.
- .fun** user defined functions, that may be called in models.
- .ext** the linkage between external programs and their internal *ARX* name.

4.1 The Parameter File (.par) (Object Definitions)

FORMAT

The format for defining parameters in the ".par" file is:

```
<parameter name>
<type>
<source list>
{comment}
```

PARAMETERS

```
<parameter name>
String, e.g. "wetness index"
```

<type>

Integer representing the type (value type) of the parameter. Valid values are:

- 2 REAL
- 3 INTEGER
- 20 ENUMERATED (a list of possible values (strings) follows 20)
- 21 LOGICAL
- 22 STRING
- 23 REFERENCE
- 24 REGIONLIST (not implemented)

This value may be added to one or more of the following **ARX** Flags to give a further definition quality to the parameter:

- 64 RANGE TYPE (parameter is defined with a given range)
- 512 MULTI-VALUED (not implemented)
- 1024 MODIFY-ALLOWED (parameter value may change during inference)
- 2048 REGION-GLOBAL (parameter value transmitted over one spatial domain)
- 4096 DOMAIN-GLOBAL (parameter value transmitted over all spatial domains)

<source list>

String describing the order list of operations to try when the parameter value is being inferred. Key words are:

- "ask" try the user
- "rules" try the rule base
- "dbs" try the 'current' spatial database
- "models" try using a mathematical model
- "file" try looking up an **ARX** data file

{comment} String

The rule and model knowledge bases may be grouped into directories and folders, which may be selectively specified in the source list. This

allows control over the order in which individual or groups of rules and models are applied.

These components are further discussed in Sections 4.3-4.4.

4.2 The Linkage File (.lnk)

The linkage file performs two tasks:

- defines the linkage (name association) between an ARC/INFO coverage and an ARX spatial domain
- defines the linkage (name association) between ARX parameters and data items contained in ARC/INFO coverages

FORMAT

```
<domain> <layer> <class> <col 1> <col 2> <dist> {alias}
<BLANK LINE>
<ARX parameter name> <ARC/INFO Column Name>
<ARC parameter name> <ARC/INFO Column Name>
```

PARAMETERS

<domain>

String - ARX spatial domain identifier that is used in rules and commands to refer to a spatial layer

<layer>

String - full path name to the ARC/INFO coverage being linked, e.g. "/home/usr/arcinfo/test/mycover"

<class>

Integer - 1 (point), 2 (line) or 3 (polygon)

Sets the type of item that the <domain> will use as its spatial elements.

<col 1>

Integer - colour 1 (not currently used, but necessary)

<col 2>

Integer - colour 2 (not currently used, but necessary)

<dist>

Integer - represents the data scale as follows:

0	Scale to follow
1	INCHES
2	CM
3	FEET
4	METRES
5	MILES
6	KILOMETRES
7	CHAINS
8	LINKS

{alias}

String (optional) which identifies the map column name which contains unique labels for each map item

<blank line>

indicates the completion of the spatial domain linkage definitions

<ARX parameter name> <ARC/INFO Column Name>

<ARC parameter name> <ARC/INFO Column Name>

Strings - parameter to column mappings

EXAMPLE

'polys' '/HOME/USR/COVER' 3 ...

'lines' '/HOME/USR/COVER' 2 ...

'points' '/HOME/USR/COVER' 1 ...

'land-use' 'LANDUSE'

'vegtype' 'VT'

An ARC/INFO coverage "/home/usr/cover" may contain points, lines and polygons. These would be referenced by 3 separate ARX spatial domains ("polys", "points", "lines"). Spatial operations may be applied to any of these ARX spatial domains.

The *ARX* parameters "land-use" and "vegtype" are associated with the ARC/INFO column names LANDUSE and VT respectively. Note that the ARC/INFO coverages that contain LANDUSE and VT have not been mentioned, since they are determined by context (i.e. the spatial domain over which *ARX* is currently being applied) when *ARX* is attempting to infer a value for "land-use" or "vegtype".

4.3 The Rule Base File (.rul)

The rule knowledge base is grouped into directories of folders of rules. These are defined using the keywords DIR and FOLDER.

e.g. DIR "traffic"
FOLDER "visibility"

would define a directory called "traffic" which contains the folder "visibility". The rules contained in this folder are defined after this statement in the general format:

```

FORMAT      RULE <rulenum>          /*This is a comment*/
            IF
              <parameter> <relation> <expression> <spatial-expression>
              <parameter> <relation> <expression> <spatial-expression>
            THEN
  
```

Comments may be inserted in the rule base file between /* and */.

```

PARAMETERS <parameter>
            String - a defined parameter, e.g. "vegtype"
  
```

```

<relation>
            String - a relation operator. Legal relation definitions are:
            "<"      less than
            ">"      greater than
            "<="     less than or equal to
  
```

">="	greater than or equal to
"is"	equal
"="	equal
"is not"	is not equal to
"is one of"	is one of (a list of values)
"is between"	is between (lower value - upper value)
"is not one of"	is not one of (a list of values)
"is not between"	is not between (lower value - upper value)
"status_is"	system operation for testing various conditions of a parameter such as whether it has been determined, unknown, etc. See the section below on <expression>.

<expression>

Single value or list of legal values that may be associated with a parameter. Typically, these values are of the type for the parameter, e.g. an integer parameter will have integer values in the <expression>, a string parameter will have strings, etc.

An <expression> may be another <parameter>, in which case the <parameter> of the <expression> will be inferred, and this determined value used in the <expression>. Note that in this case the two <parameter>s must be of the same type.

There are several system values that may be used with the relation "status_is", as follows:

"known"	the <parameter> has been determined (premise only)
"unknown"	the <parameter> is not currently determined (premise only)
"notknown"	the <parameter> cannot be solely determined by premise
"inferred"	directs the inference engine to infer the value of the parameter
"cinferred"	directs the inference engine to delete all knowledge of the parameter, and reinfer its value

"wiped" all knowledge of <parameter> is wiped from the system

<spatial expression>

Represents the spatial locations where the triplet <parameter> <relation> <expression> is to be applied. The BNF notation for this <spatial expression> is given in Appendix A.

The most common form of <spatial expression> involves stating that the 'current location' (i.e. the current focus of the inference engine) is where the rule triplet is to apply. This is designated by the empty string "". Components are:

FOR "All" "Any"
 "Sum" ">Sum" "<Sum" ">=Sum" "<=Sum"
 "Ratio" ">Ratio" "<Ratio" ">=Ratio" "<=Ratio"
 These have a number or parameter after them.

SPAEXP "Within" "Further" (some number)
 "North" "South" "East" "West" "Adjacent"

TARGET <Spatial Domain>

SOURCE <Spatial Domain>

The <Spatial Domain> is the selected set of items to apply the spatial operation. The TARGET represents the resultant set of spatial items to be selected.

PLUS_CURRENT

MINUS_CURRENT

Indicates inclusion/exclusion of current location in spatial selection.

RULE EXAMPLE

DIR 'my'
 FOLDER 'value1'

RULE 1

```

IF
  'id' '=' 25
THEN
  'myvalue' 'is' 7
  'myvalue' 'is' 1
    FOR 'All' TARGET 'grid' SOURCE 'grid' SPAEXP 'Adjacent'
  'myvalue' 'is' 2
    FOR 'All' TARGET 'grid' SOURCE 'grid' SPAEXP 'Within' 1.9
  'myvalue' 'is' 3
    FOR 'All' TARGET 'grid' SOURCE 'grid' SPAEXP 'Within' 2.9
  'myvalue' 'is' 4

```

EXPRESSION

'vegtype' 'is' 1

EXAMPLES

/* vegetation type has the value 1 in the current region */

'vegtype' 'is one of' 2,4,5

/* vegetation type is one of 2, 4 or 5 in the current region */

'vegtype' '<' 3

/* vegetation type is less than 3 in the current region */

'vegtype' 'is not' 2 FOR Any TARGET 'landmap'

/* vegetation type is not 2 for any item on the *ARX* spatial domain 'landmap' */

'soil' 'is' 3 FOR Any TARGET 'soilmap' SOURCE 'vegmap' SPAEXP 'Within' 1.0

/* Soil is 3 for any soilmap item within 1.0 of the current item on the vegmap */

'soil' '>' 1 FOR Any TARGET 'soilmap' SOURCE 'vegmap' SPAEXP 'Within' 1.0 SOURCE
 'vegmap' SPAEXP 'Further' 0.3

/* Soil is greater than 1 for any soilmap item within 1.0 and further than 0.3 from the current item on
 the vegmap */

LINK TO .PAR FILE

The DIR and FOLDER definitions are referenced in source lists in the parameter (.par) file. Some examples are:

- ["rules"]

TRY any rule directories that have been defined

- ["rules", "traffic"]
TRY any rules found in folders under the "traffic" directory
- ["rules", "traffic", "visibility"]
TRY rules found only in the "visibility" folder under the "traffic" directory
- ["rules", "nogo", "conditions"] ["rules", "traffic"]
TRY the nogo rules in the "conditions" folder, and if they fail try the rules in the "traffic" directory.

4.4 The Model File (.mod)

The model file allows the user to execute mathematical functions, external programs and user defined functions. Sets of models can be grouped in directories and rules (similar to the .rul file). A model is essentially a rule without a premise, and does not allow any spatial expression. The lack of spatial expression is not a limitation, since we can control the spatial application of a model by using rules that select some space before the model is called.

FORMAT

```
#<directory> <folder>
<model-name>
<parameter> <relation> <mathematical expression>
<parameter> <relation> <mathematical expression>
<BLANK LINE>
<author>
<comment>
<model-name> ... <comment> repeated as often as necessary
```

PARAMETERS

Standard mathematical operators and functions are supported.

+	Addition
-	Subtraction
*	Multiplication
/	Division

sin (x)	
cos (x)	
tan (x)	
sqr (x)	square root of x
exp (x)	exponential of x
pow (x,y)	x raised to the power of y
sum (x1,x2,x3,..xn)	sum of n arguments
max (x1,x2,x3,..xn)	max. of n arguments
min (x1,x2,x3,..xn)	min. of n arguments
log (x)	log x base e
log10 (x)	log x base 10
neg (x)	the negation of x
rand ()	returns a random integer
rulebase (onoff, dir, fol)	Allows enabling/disabling of rule base folders
avgextent ()	Average boxed extent for the current spatial item
date ()	System call which returns a string for the current date in UNIX format
report (s1,p1,s2,p2,..,sn,pn)	Prints to stdout (standard output) the current determined values of parameters <p1>.. <p>. <s1>..<p>="" are="" args="" as="" each="" for="" headings="" line.<="" one="" parameter="" per="" string="" strings="" td="" the="" used="" value.="" values="" written=""> </p>.>

EXAMPLE

The following example defines two models - the "Summing-model" and "Max-model". These models define how to determine the value for the parameters "bigsum", "smallsum", "biggest" and "funny". The directory and folder have been defined as "bigmodel" and "test" respectively. Note that a parameter that refers to a model in its source list uses the same syntax as shown for rules, ie., the source list may refer to "models", ["models", "directory"] or ["models", "directory", "folder"] as required.

```
# bigmodel test
"Summing-model"
```

```
'bigsum' '=' sum (1,2,3,'max-counter',5,6,7)
'smallsum' '=' sum (1,2,'min-counter')

'Dr Feelgood'
'Note that parameters may be substituted for arguments'

'Max-model'
'biggest' '=' max ('count1', 'count2', min ('smallsum',23))
'funny' '=' rand() * log(2.1)/('biggest' + 'smallsum')

'Davros'
'Note that function calls may be called within an expression or argument'
```

4.5 User Defined Functions (.fun)

This file allows users to define their own functions, using the same syntax as for model definitions.

FORMAT	<function-name> (arg1,arg2,..,argn) <expn>
PARAMETERS	<function-name> String
	<arg1>..<argn> String(s) - Arguments to the function
	<expn> Mathematical expression containing <arg1..argn>, system or user functions and parameter names.

4.6 The External Program File (.ext)

External (compiled) programs may be called from within a mathematical expression as part of a model. The .ext file shows the mapping between

the *ARX* name for this program and the UNIX path name to the program.

FORMAT **<ARX Program Name> <System Path>**

PARAMETERS **<ARX Program Name>**
 String

<System Path>
 String

EXAMPLE **'myprog' '/home/usr/test/theprogram'**

defines the external program "theprogram" to be referred to as "myprog" within the model environment.

Note that arguments to be passed to "myprog" are done in the same way as all other function calls in the mathematical model environment. The following example calls "myprog" with 2 arguments (20 and arg) and assigns the return value to parameter X:

```
'X' '=' myprog(20,arg)
```

4.7 The *ARX* Data-File

The *ARX* data-file allows the user to store data values for a parameter for each item of a spatial domain. The file represents each item address by successive lines, where line one indicates the first item in the ARC/INFO coverage, line two the second, and so on. Each line contains one value - that being the value of the parameter for that spatial item. The source list of a parameter directs the inference engine to look up a specified data file as follows:

FORMAT **{<file>, "some-file-path-name"|"file-param"}**

PARAMETERS

file

String - key word

"some-file-path-name"

String - path to the data file.

"file-param"

String - a parameter set to the path name (often used when a parameter is to obtain different values during the course of an inference).

5 *ARX* Script Commands

5

A high level command language has been added to *ARX* which provides a macro language for writing *ARX* scripts. The set of commands is described in the companion document to this manual, the *ARX* Command Reference Manual.

These commands may be placed in a file that can be executed, thereby creating a user-defined interface to the knowledge base and the control facilities of *ARX*. Some example script files are shown in Appendix C.

APPENDIX A

BNF Notation Syntax for Spatial Expressions

Set out below is the syntax supported by *ARX* for the spatial expression of a rule quadruplet. (Note: If the spatial-expression or the spatial-description component is NULL, the 'current region' is assumed.)

spatial expression	::=	(spatial-selector <target-domain> (<source-domain> spatial description)* NULL
spatial-selector	::=	FOR all FOR any FOR sum = <num> FOR sum < <num> FOR sum > <num> FOR ratio = <fraction> FOR ratio < <fraction> etc.
target-domain	::=	TARGET spatial domain where resultant items are selected
source-domain	::=	SOURCE spatial domain where spatial operations are applied
spatial-description	::=	SPAEXP explicit-regionlist implicit-regionlist NULL
explicit-regionlist	::=	region-identifier explicit-regionlist NULL
implicit-regionlist	::=	distance-expression explicit-regionlist compass-direction explicit-regionlist adjacency-relation explicit-regionlist enclosed-relation explicit-regionlist

distance-expression	::=	"Within" distance "Further" distance
compass-direction	::=	"North" "South" "East" "West" etc.
adjacency-relation	::=	"Adjacent"
enclosed-relation	::=	"Enclosed"
distance	::=	<real number>
region-identifier	::=	<region name> <region number>

The * symbol is the Kleene star operator meaning 'zero or more'.

APPENDIX B

Knowledge Base to Predict Spread of Fire

The example knowledge base represents a system that predicts the spread of fire and classifies grid locations based on their position relative to the oncoming fire and escape access via roads.

Some notes on format:

- comments are designated by a # in the first column of a line.
- if a command is not part of ARX it is assumed that a UNIX system call must be made. The appropriate command is automatically called from within ARX.

B.1 The Parameter File (.par)

```
"fire loc"
3
[ rules, leap, adjacent ]
"Check fire does not spread to areas of low vegetation"

"check spread"
3
[ rules, spread, better ]
"Check fire does not spread to areas of low vegetation"

"vegcode"
3
"db"
"Vegetation code 0 none, 6006 scattered, 6007 medium, 6006 dense"

"vegetation"
20 "none" "scattered" "medium" "dense"
[ rules, vegcode, convert ]
"Convert integer vegcode to word description"

"road-id"
3
"db"
"Road id from the roads coverage"

"v-r"
20 "r-none" "r-scattered" "r-medium" "r-dense" "no-r-none"
```

"no-r-scattered" "no-r-medium" "no-r-dense"
 [rules,fire,"v-r1"] [rules,fire,"v-r2"] [rules,fire,"v-r3"] [rules,fire,"v-r4"] [rules,fire,"v-r1"]
 [rules,fire,"v-2"] [rules,fire,"v-3"] [rules,fire,"v-4"]
 "Combining vegetation and road access"

"fire spread"
 3
 [rules,fire,spread]
 "Fire spread - where the fire may go .."

"fire danger"
 20 "negligible" "low" "moderate" "high" "very high" "extreme"
 [rules,fire,season] [rules,fire,nofire] [rules,fire,danger]
 "Fire danger"

"wind direction"
 2070
 "ask"
 "Wind direction - ie where it is coming from"

"wind speed"
 2051
 "ask"
 "Wind speed in km/hr"

"time-one-km"
 2050
 [models, wind, convert]
 "Time to travel 1 km"

"time till fire"
 2
 [rules,time,spread]
 "Time till fire reaches each cell"

"days since last rain"
 2051
 "ask"
 "Days since last significant rainfall event"

"season"
 2070
 "ask"
 "Season - dry,average,wet"

B.2 The Linkage File (.lnk)

"veg" "/HOME/ARCINFO/LMAS/ALL-VEG" 3 1 2 6 4
 "roads" "/HOME/ARCINFO/LMAS/ALL-ROADS" 2 1 2 6 4
 "fire" "/USR/PETER/SMALLGRID/PETERGRID" 3 1 2 6 4
 "start-fire" "/USR/PETER/SMALLGRID/PETERGRID" 3 1 2 6 4

"road-id" "ALL-ROADS-ID"
 "vegcode" "VEGCODE"

B.3 The Model File (.mod)

```
#wind convert
"conv-wind-speed"
"time-one-km" "=" 60.0 / "wind speed"

"P.Whigham"
"Calculate wind speed per minute"
```

B.4 The Rule File (.rul)

```
/* Conversion of vegetation codes to expression*/
DIR "vegconv"
FOLDER "convert"

RULE 1
IF "vegcode" "is" "0"
THEN "vegetation" "is" "none"

RULE 2
IF "vegcode" "is" 6008
THEN "vegetation" "is" "scattered"

RULE 3
IF "vegcode" "is" 6007
THEN "vegetation" "is" "medium"

RULE 4
IF "vegcode" "is" 6006
THEN "vegetation" "is" "dense"
/***** */
/* First pass check veg and road access to create the value "v-r" representing the combination condition of vegetation and roads */
DIR "fire"
FOLDER "v-r1"

RULE 1
IF "road-id" ">" 0
FOR Any
TARGET roads
SOURCE fire
SPAEXP Within 0.0
"vegetation" "is" "none"
FOR All
TARGET veg
SOURCE fire
SPAEXP Within 0.0
THEN
"v-r" "is" "r-none"
/* Road access - no vegetation */
/* Good escape conditions */

DIR "fire"
FOLDER "v-r2"
```

```

RULE 1
IF "road-id" ">" 0
    FOR Any
    TARGET roads
        SOURCE fire
        SPAEXP Within 0.0
    "vegetation" "is" "dense"
    FOR Any
    TARGET veg
        SOURCE fire
        SPAEXP Within 0.0
THEN
    "v-r" "is" "r-dense"
    /* Road access but dense veg */

```

```

DIR "fire"
FOLDER "v-r3"

```

```

RULE 1
IF "road-id" ">" 0
    FOR Any
    TARGET roads
        SOURCE fire
        SPAEXP Within 0.0
    "vegetation" "is" "medium"
    FOR Any
    TARGET veg
        SOURCE fire
        SPAEXP Within 0.0
THEN
    "v-r" "is" "r-medium"

```

```

DIR "fire"
FOLDER "v-r4"

```

```

RULE 1
IF "road-id" ">" 0
    FOR Any
    TARGET roads
        SOURCE fire
        SPAEXP Within 0.0
    "vegetation" "is" "scattered"
    FOR Any
    TARGET veg
        SOURCE fire
        SPAEXP Within 0.0
THEN
    "v-r" "is" "r-scattered"

```

```

/* Here we have no road access from the grid therefore we base the v-r on vegetation alone */
DIR "fire"
FOLDER "v-1"

```

```

RULE 1
IF "vegetation" "is" "none"
    FOR All
    TARGET veg
        SOURCE fire
        SPAEXP Within 0.0
THEN
    "v-r" "is" "no-r-none"
    /* no road, no veg */

```

```

DIR "fire"
FOLDER "v-2"

RULE 1
IF "vegetation" "is" "dense"
    FOR Any
        TARGET veg
            SOURCE fire
            SPAEXP Within 0.0
THEN
    "v-r" "is" "no-r-dense" /*worst of all!! */
    /* No road,dense veg */

```

```

DIR "fire"
FOLDER "v-3"

RULE 1
IF "vegetation" "is" "medium"
    FOR Any
        TARGET veg
            SOURCE fire
            SPAEXP Within 0.0
THEN
    "v-r" "is" "no-r-medium"

```

```

DIR fire
FOLDER "v-4"

RULE 1
IF "vegetation" "is" "scattered"
    FOR Any
        TARGET veg
            SOURCE fire
            SPAEXP Within 0.0
THEN
    "v-r" "is" "no-r-scattered"

```

```

/*****
DIR "fire"
FOLDER "spread"

RULE 1
IF "wind direction" "is" "north"
THEN
    "fire spread" "is" 1
    "fire spread" "is" 1
        FOR All
            TARGET fire
                SOURCE "start-fire"
                SPAEXP South
    "fire spread" is 0
        FOR All
            TARGET fire

RULE 2
IF "wind direction" "is" "south"
THEN
    "fire spread" "is" 1
    "fire spread" "is" 1 /* Fire goes here */
        FOR All
            TARGET fire
                SOURCE "start-fire"

```

```

                SPAEXP North
"fire spread" "is" 0
  FOR All /* Fire not here */
  TARGET fire

RULE 3
IF "wind direction" "is" "east"
THEN
  "fire spread" "is" 1
  "fire spread" "is" 1 /* Fire goes here */
  FOR All
  TARGET fire
    SOURCE "start-fire"
    SPAEXP West
  "fire spread" "is" 0
  FOR All /* Fire not here */
  TARGET fire

RULE 4
IF "wind direction" "is" "west"
THEN
  "fire spread" "is" 1
  "fire spread" "is" 1 /* Fire goes here */
  FOR All
  TARGET fire
    SOURCE "start-fire"
    SPAEXP East
  "fire spread" "is" 0
  FOR All /* Fire not here */
  TARGET fire

/** FIRE DANGER RULES *****/
/* First do exception rule for the wet season/days since rain */
DIR "fire"
FOLDER "season"

RULE 1
IF "season" "is" "wet"
  "days since last rain" "<=" 7
THEN
  "fire danger" "is" "negligible"
  FOR All
  TARGET fire

RULE 2
IF "season" "is" "average"
  "days since last rain" "<=" 2
THEN
  "fire danger" "is" "negligible"
  FOR All
  TARGET fire

/* First test if no fire reaches us
-> negligible danger */
DIR "fire"
FOLDER "nofire"

RULE 1
IF "fire spread" "is" 0 /* No fire */
THEN
  "fire danger" "is" "negligible"

```

```
/* Otherwise - test using veg and road access values */
DIR "fire"
FOLDER "danger"

RULE 1
IF "time till fire" <= 5.0
THEN
  "fire danger" is "extreme"

RULE 2
IF "time till fire" is between 5.0 10.1
  "v-r" is one of "no-r-medium" "no-r-dense"
THEN
  "fire danger" is "extreme"

RULE 3
IF "time till fire" is between 5.0 10.1
  "v-r" is one of "no-r-none" "no-r-scattered" "r-medium" "r-dense"
THEN
  "fire danger" is "very high"

RULE 4
IF "time till fire" is between 5.0 10.1
  "v-r" is one of "r-none" "r-scattered"
THEN
  "fire danger" is "high"

RULE 5
IF "time till fire" is between 10.0 15.1
  "v-r" is one of "no-r-medium" "no-r-dense"
THEN
  "fire danger" is "very high"

RULE 6
IF "time till fire" is between 10.0 15.1
  "v-r" is one of "no-r-none" "no-r-scattered" "r-medium" "r-dense"
THEN
  "fire danger" is "high"

RULE 7
IF "time till fire" is between 10.0 15.1
  "v-r" is one of "r-none" "r-scattered"
THEN
  "fire danger" is "moderate"

RULE 8
IF "time till fire" is between 15.0 20.1
  "v-r" is one of "no-r-medium" "no-r-dense"
THEN
  "fire danger" is "high"

RULE 9
IF "time till fire" is between 15.0 20.1
  "v-r" is one of "no-r-none" "no-r-scattered" "r-medium" "r-dense"
THEN
  "fire danger" is "moderate"

RULE 10
IF "time till fire" is between 15.0 20.1
  "v-r" is one of "r-none" "r-scattered"
THEN
```

```

"fire danger" "is" "low"

RULE 11
IF "time till fire" "is between" 20.0 25.1
  "v-r" "is one of" "no-r-medium" "no-r-dense"
THEN
  "fire danger" "is" "moderate"

RULE 12
IF "time till fire" "is between" 20.0 25.1
  "v-r" "is not one of" "no-r-medium" "no-r-dense"
THEN
  "fire danger" "is" "negligible"

RULE 13
IF "time till fire" ">=" 25.0
THEN
  "fire danger" "is" "negligible"

/*****
Time till fire spreads to each cell */
DIR "time"
FOLDER "spread"

RULE 1
IF "time-one-km" "is" inferred
  /*always true */
THEN
  "time till fire" "=" 0.0 /*where it starts */
  "time till fire" "=" "time-one-km"
  FOR All
  TARGET fire
  SOURCE "start-fire"
  SPAEXP Within 0.5
  "time till fire" "#=" "time-one-km" * 2.5
  FOR All
  TARGET fire
  SOURCE "start-fire"
  SPAEXP Within 1.5
  SOURCE "start-fire"
  SPAEXP Further 0.5
  "time till fire" "#=" "time-one-km" * 4.0
  FOR All
  TARGET fire
  SOURCE "start-fire"
  SPAEXP Within 2.5
  SOURCE "start-fire"
  SPAEXP Further 1.5
  "time till fire" "#=" "time-one-km" * 5.5
  FOR All
  TARGET fire
  SOURCE "start-fire"
  SPAEXP Within 3.5
  SOURCE "start-fire"
  SPAEXP Further 2.5
  "time till fire" "#=" "time-one-km" * 7.0
  FOR All
  TARGET fire
  SOURCE "start-fire"
  SPAEXP Within 4.5
  SOURCE "start-fire"
  SPAEXP Further 3.5

```

```

*time till fire" "#=" "time-one-km" * 8.5
  FOR All
  TARGET fire
  SOURCE "start-fire"
  SPAEXP Within 5.5
  SOURCE "start-fire"
  SPAEXP Further 4.5
*time till fire" "#=" "time-one-km" * 10.0
  FOR All
  TARGET fire
  SOURCE "start-fire"
  SPAEXP Within 6.5
  SOURCE "start-fire"
  SPAEXP Further 5.5
*time till fire" "#=" "time-one-km" * 11.5
  FOR All
  TARGET fire
  SOURCE "start-fire"
  SPAEXP Within 7.5
  SOURCE "start-fire"
  SPAEXP Further 6.5
*time till fire" "#=" "time-one-km" * 13.0
  FOR All
  TARGET fire
  SOURCE "start-fire"
  SPAEXP Within 8.5
  SOURCE "start-fire"
  SPAEXP Further 7.5
*time till fire" "#=" "time-one-km" * 14.5
  FOR All
  TARGET fire
  SOURCE "start-fire"
  SPAEXP Within 9.5
  SOURCE "start-fire"
  SPAEXP Further 8.5
*time till fire" "#=" "time-one-km" * 16.0
  FOR All
  TARGET fire
  SOURCE "start-fire"
  SPAEXP Further 9.5

```

**** BETTER FIRE SPREAD RULE *****/

DIR "spread"
FOLDER "better"

RULE 1

```

IF "fire spread" "is" 1
  "vegetation" "is one of" "none" "scattered"
  FOR All
  TARGET veg
  SOURCE "fire"
  SPAEXP Within 0.0

```

THEN

```

"fire spread" "is" "wiped"
"fire spread" is 0
"check spread" is 1 /force rule to fire */

```

**** CHECK FIRE DOES NOT LEAP *****/

DIR "leap"
FOLDER "adjacent"

RULE 1

```
IF "fire spread" "is" 1
  "fire loc" "is" 1
  FOR Any
  TARGET fire
  SOURCE fire /* ie. any adjacent cells */
  SPAEXP Within 0.1
THEN
  "fire loc" "is" 1

RULE 2
IF "fire spread" "is" 1
  "fire loc" "is not" 1
  FOR All
  TARGET "fire"
  SOURCE fire /* ie. any adjacent cells */
  SPAEXP Within 0.1
THEN
  "fire spread" "is" "wiped"
  "fire spread" "is" 0
  "fire loc" "is" 0
```

APPENDIX C

Examples of Script Files

C

C.1 Script to Define a Map Display Window

This *ARX* script file implements a simple map display window. The scripts are broken into 5 files.

START	The initial script that builds the windows and links the ARC/INFO maps to the <i>ARX</i> script language
panel.bat	Builds the buttons on the map display panel
map.bat	Displays the named ARC/INFO map to the drawing surface
zoom.bat	Allows the user to zoom into a location of the currently displayed map
quit.bat	Exit the system

ARX commands are highlighted in **bold**.

START

```
#
# Startup file for a simple MAP DISPLAY window.
#
# DATE : 28/5/92
# LOCATION: ex1/START
# AUTHOR: P.A.Whigham
# USAGE: unix%SARX ex1/START
#
# Functionally we require:
#     1. A panel to place buttons, etc. for control
#     2. A map canvas to draw maps/pictures onto
#
# INITIALISATION:
# 1. Shade set for drawing
# 2. The ARX domain files - in this case just the map linkage file ex1/lnk
```

```

shadefile "bw.shd"
"load-all-domain" "ex1"
#
# The canvastoptions command always precedes the canvascreate command.
# It defines how the window will be set up.
# canvastoptions <Scroll (0,1) Panel (0,1) Panel Size 0->100
#                               <Text Window (0,1)> Text Size 0->100
# <No Scroll> <PANEL> <Size = 40% (width)> MAP <Size=60% # Width>
#
canvastoptions 0 1 40 0 0
#
# Now we create the canvas using the canvascreate command
# canvascreate <WIDTH> <HEIGHT> <POS X> <POS Y>
canvascreate 925 570 5 5
#
# And lets give it a name so we can refer to it
canvasname "ARX-MAP"
#
# And place the button controls onto the panel
# The commands for the creation are placed in the file ex1/panel.bat (for convenience)
exefile "ex1/panel.bat"
#
# Finally we need to open the window so it may be used
canvasopen
#

```

panel.bat

```

#
# ex1/panel.bat
# The following controls are placed onto the panel:
# 1. Map Name: (TEXT)          map name to load to the text window
# 2. Quit (BUTTON)           finish execution of map display window (HALT)
# 3. "ARX MAP DISPLAY" (MESSAGE) heading for the panel item
# STAGE 2 *****
# 4. Zoom (BUTTON) zoom in on specified location of map using cross-hairs
#
# Start the WINDOW control command selections
#
WINS
#
# 1. Map Name: (TEXT)
createpi 5 "X"
pioptions 1
"PANEL_ITEM_X" 15 "PANEL_ITEM_Y" 113 "PANEL_LABEL_STRING" "Map:"
pioptions 1
"PANEL_LABEL_BOLD" 1 "PANEL_VALUE_DISPLAY_LENGTH" 30 "PANEL_LABEL_FONT" "F2"
"PANEL_VALUE_STORED_LENGTH" 200
piexe 1 "ex1/map.bat"
#
# 2. Quit (BUTTON)
createpi 2 "Quit "
pioptions 2
"PANEL_ITEM_X" 10 "PANEL_ITEM_Y" 10
piexe 2 "ex1/quit.bat"
#
# 3. "ARX MAP DISPLAY" (MESSAGE)
createpi 1 "X"
pioptions 3
"PANEL_LABEL_STRING" "ARX MAP DISPLAY" "PANEL_ITEM_X" 120 "PANEL_ITEM_Y" 15
pioptions 3
"PANEL_LABEL_FONT" "F5" "PANEL_LABEL_BOLD" 1

```

```

#
# STAGE 2 *****
# 4. Zoom (BUTTON)
createpi 2 " Zoom "
pioptions 4
"PANEL_ITEM_X" 10 "PANEL_ITEM_Y" 200
piexe 4 "ex1/zoom.bat"
#
# Leave the WINDOW control menu
XIT
#
# and we are finished
return

```

map.bat

```

# ex1/map.bat
# OPERATION:
#      i) Clear map surface
#      ii) Load map to surface in linecolour 1
#
defineargs "map"
cavasselectname "ARX-MAP"
#
# Get map name from panel, leave if no name present
varpi 1 "map"
SWITCH map
CASE "" return
#
# Otherwise, clear the map and map extent
cavasclear
clearextent
#
# and draw the map to the canvas
linecolour 1
cavasinser "map"
#
# Done

```

zoom.bat

```

# ex1/zoom.bat
defineargs "map"
cavasselectname "ARX-MAP"
#
# Get map name
varpi 1 "map"
#
# Leave if there is no map name to draw
SWITCH map
CASE "" return
#
# Get new extent from map using crosshairs -
# Then clear the map surface (but do not reset extent)
"interactive-extent"
cavasclear
#
# And redraw the map at the new extent settings
cavasinser "map"
#
# done
#

```

```
quit.bat      # ex1/quit.bat
              # Delete temporary Arc/Info files
              rm *.scr
              rm *.scx
              #
              # Halt the system and shutdown
              haltsarx
```

C.2 Script to Create an ARX Script Editor

The following set of ARX scripts implements a simple ARX script editor. The scripts are broken into seven files.

START	The initial script that builds the windows and links the ARC/INFO maps to the ARX script language
panel.bat	Builds the buttons on the map display panel.
load.bat	Loads a file to the editor window
store.bat	Stores file at named location
list_files.bat	Get directory listing for display
execute.bat	Executes current script
quit.bat	Exit the system and shut down

ARX commands are highlighted in **bold**.

```
START      # Startup file for a simple EDITOR window.
           # DATE : 28/5/92
           # LOCATION: arxed/START
           # AUTHOR: P.A.Whigham
           # USAGE: unix%SARX arxed/START
           #
           # This script will form the basis of an editor for ARX script files (ie. ARX programs)
           # Functionally we require:
           #     1. A panel to place buttons, etc. for control
           #     2. A text window to place scripts for editing
           #
           # The canvsoptions command always precedes the canvascreeate command.
```

```

# It defines how the window will be set up.
#
# canvasoptions <Scroll (0,1) Panel (0,1) Panel Size 0->100
#               <Text Window (0,1)> Text Size 0->100
# <No Scroll> <PANEL> <Size = 100% (width)> <TEXT> <Size=85% Height>
messages 0
canvasoptions 0 1 100 1 85
#
# Now we create the canvas using the canvascreate command
# canvascreate <WIDTH> <HEIGHT> <POS X> <POS Y>
canvascreate 675 970 5 5
#
# And lets give it a name so we can refer to it
canvasname "ARX-EDITOR"
#
# And place the button controls onto the panel
# The commands for the creation are placed in the file arxed/panel.bat (for convenience)
exefile "arxed/panel.bat"
#
# Finally we need to open the window so it may be used
canvasopen
#
# STAGE 2 *****
# and the text window to insert the directory listing
# Make a window which is entirely a text window
# And set it to be the full length of the window
canvasoptions 0 1 100 1 100
canvascreate 500 970 675 5
canvasname "ARX-LISTING"
canvasopen
#

```

panel.bat

```

# arxed/panel.bat
# The following controls are placed onto the panel:
# 1. File Name: (TEXT)           the file name to load to the text window
# 2. Dir Name: (TEXT)           the directory path for the file to load to the text window
# 3. Load File (BUTTON)        Selecting this button will load the file referred to
#                               by "File Name:" into the text window
# 4. Store File (BUTTON)        Selecting this button will store the text window contents
#                               in the file "File Name:".
# 5. Quit (BUTTON) Finish execution of editor (HALT)
# 6. "ARX SCRIPT EDITOR" (MESSAGE) The heading for the panel item
# STAGE 3 *****
# 7. Execute the loaded script file (BUTTON)
#     The current loaded file is executed as an ARX script
# *****
#
# Start the WINDOW control command selections
WINS
#
# 1. File Name: (TEXT)
createpi 5 "X"
pioptions 1
"PANEL_ITEM_X" 10 "PANEL_ITEM_Y" 113 "PANEL_LABEL_STRING" "File Name:"
pioptions 1
"PANEL_LABEL_BOLD" 1 "PANEL_VALUE_DISPLAY_LENGTH" 40 "PANEL_LABEL_FONT" "F2"
"PANEL_VALUE_STORED_LENGTH" 200
#
# STAGE 2 *****
piexe 1 "arxed/load.bat"

```

```

#
# 2. Dir Name: (TEXT)
createpi 5 "X"
pioptions 2
"PANEL_ITEM_X" 10 "PANEL_ITEM_Y" 83 "PANEL_LABEL_STRING" "Directory:"
pioptions 2
"PANEL_LABEL_BOLD" 1 "PANEL_VALUE_DISPLAY_LENGTH" 40 "PANEL_LABEL_FONT" "F2"
"PANEL_VALUE_STORED_LENGTH" 200
#
# STAGE 2 *****
piexe 2 "arxed/list_files.bat"
#
# 3. Load File (BUTTON)
createpi 2 " Load "
pioptions 3
"PANEL_ITEM_X" 10 "PANEL_ITEM_Y" 50
piexe 3 "arxed/load.bat"
#
# 4. Store File (BUTTON)
createpi 2 " Store "
pioptions 4
"PANEL_ITEM_X" 128 "PANEL_ITEM_Y" 50
piexe 4 "arxed/store.bat"
#
# 5. Quit (BUTTON)
createpi 2 " Quit "
pioptions 5
"PANEL_ITEM_X" 510 "PANEL_ITEM_Y" 10
piexe 5 "arxed/quit.bat"
#
# 6. "ARX SCRIPT EDITOR" (MESSAGE)
createpi 1 "X"
pioptions 6
"PANEL_LABEL_STRING" "ARX SCRIPT EDITOR" "PANEL_ITEM_X" 150 "PANEL_ITEM_Y" 15
pioptions 6
"PANEL_LABEL_FONT" "F5" "PANEL_LABEL_BOLD" 1
#
# STAGE 3 *****
# 7. Execute Script (BUTTON)
createpi 2 " Execute "
pioptions 7
"PANEL_ITEM_X" 260 "PANEL_ITEM_Y" 50
piexe 7 "arxed/execute.bat"
#
# Leave the WINDOW control menu
XIT
#
# and we are finished
return

```

load.bat

```

# FILE: arxed/load.bat
# Load new file to text window and display.
#
# OPERATION:
# 1. Get text file and directory names from the panel
# 2. Create the files path by appending the directory name to the file name
#     ie.PATH = "textdir" + "textfile"
# 3. Load file PATH to the text window
# 4. Done
#
#

```

```

# Firstly define some arguments to access the file/dir names
# and to represent the complete file path
defineargs "textdir" "textfile" "PATH"
#
# Select the arx editor window
cavaselectname "ARX-EDITOR"
#
# The first panel item is the file name; the second item is the directory name
varpi 1 "textfile"
varpi 2 "textdir"
#
# Just leave if we have no file or directory name
SWITCH "textfile"
CASE "" return
SWITCH "textdir"
CASE "" return
#
# Otherwise:           Make the complete file path
buildvar "PATH" "textdir" "/" "textfile"
#
# And try to load the file into the text window
textwload "PATH"
#

```

store.bat

```

# arxed/store.bat
# Define arguments for directory and file name
#
defineargs "textdir" "textfile" "PATH"
cavaselectname "ARX-EDITOR"
varpi 1 "textfile"
varpi 2 "textdir"
#
# Do not try and store if we do not have a file or directory name
SWITCH "textfile"
CASE "" return
SWITCH "textdir"
CASE "" return
#
# Otherwise build full file path and try to store
buildvar "PATH" "textdir" "/" "textfile"
textwstore "PATH"
#
# Finished
return

```

list_files.bat

```

# arxed/list_files.bat
# Give text listing of files under Directory:
#
defineargs "textdir"
cavaselectname "ARX-EDITOR"
varpi 2 "textdir"
SWITCH "textdir"
CASE "" return
#
# Use the temporary directory dirtmp to store the directory
# listing before we load it to the LISTING window
#rm "arxed/dirtmp"
echo "===== " > "arxed/dirtmp"

```

```

echo "DIRECTORY LISTING: " "textdir" >> "arxed/dirtmp"
echo "===== " >> "arxed/dirtmp"
ls "textdir" >> "arxed/dirtmp"
echo "" >> "arxed/dirtmp"
echo "===== " >> "arxed/dirtmp"
cavasselectname "ARX-LISTING"
cavasclose
textswload "arxed/dirtmp"
canvasopen
#

```

execute.bat

```

# STAGE 3 *****
# FILE: arxed/execute.bat
# OPERATION:
# Execute the contents of the current text window as an ARX script file.
# First store the text contents in the temporary location arxed/tmpscript
cavasselectname "ARX-EDITOR"
rm "arxed/tmpscript"
textswstore "arxed/tmpscript"
rm "arxed/tmpstartup"
echo "arxed/tmpscript" > "arxed/tmpstartup"
#
# Now execute this file
SARX < "arxed/tmpstartup"
#
# done
return

```

quit.bat

```

# arxed/quit.bat
# Delete temporary files
rm arxed/dirtmp
rm *.scr
rm *.scx
#
# and halt system
haltsarx

```

