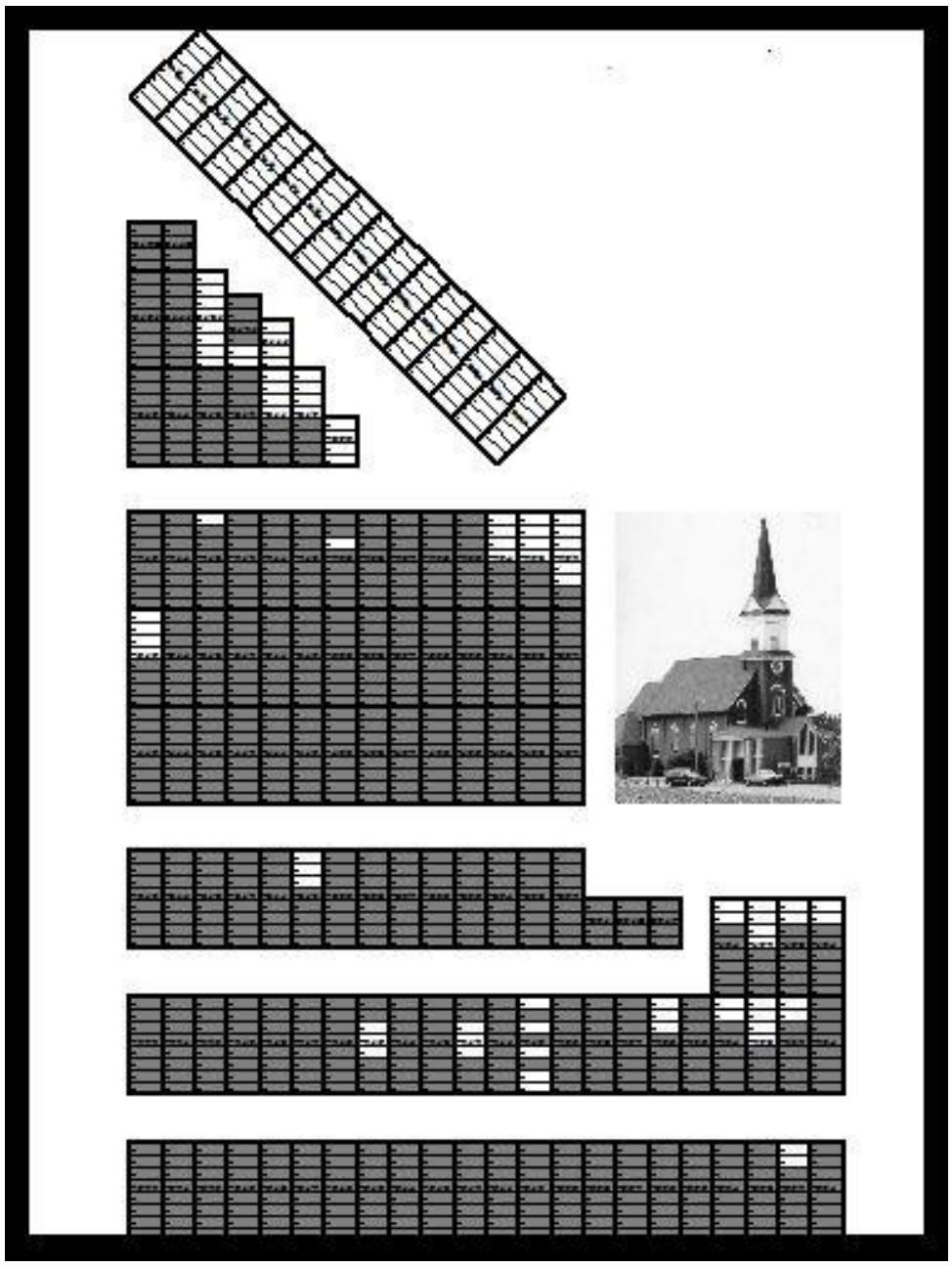
BUUTHILL CEMETERY MAPS

Revision 33



HOW TO USE THIS MANUAL

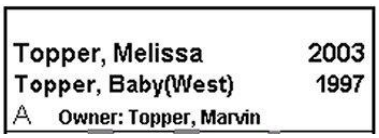
The cemetery layout section will give you a basic understanding of how a cemetery is structured.

We suggest that you ignore the advanced file configuration stuff until you have a firm grasp of the other information.

Be sure that you have a solid understanding of member insert points and configuration before you read the information on groups.

CEMETERY MAP LAYOUT

Graves



If format 1 on the Advanced Options form is specified the year on the right will not be displayed

The lowest level of the structure is the grave. All graves in the cemetery are assumed to be rectangular and may be different sizes. All graves in a given lot must be the same size.

The grave diagram depicted above shows a grave with an id of "A". It has 2 burial records, 1 reserved record and the owner record depicted in it.

Information shown in a grave includes information from the following record types for the grave.

Burial record, which shows the name on the left. If format 0 is used the date of death or burial is displayed on the right. If the date of death field is empty then the burial date will be used. If both of those fields are empty, 4 question marks are entered.

Reserved record, which is, treated much the same as a Burial record except that the death date field will say "Rsvd"

Up to three lines of burial and reserved records may be displayed.

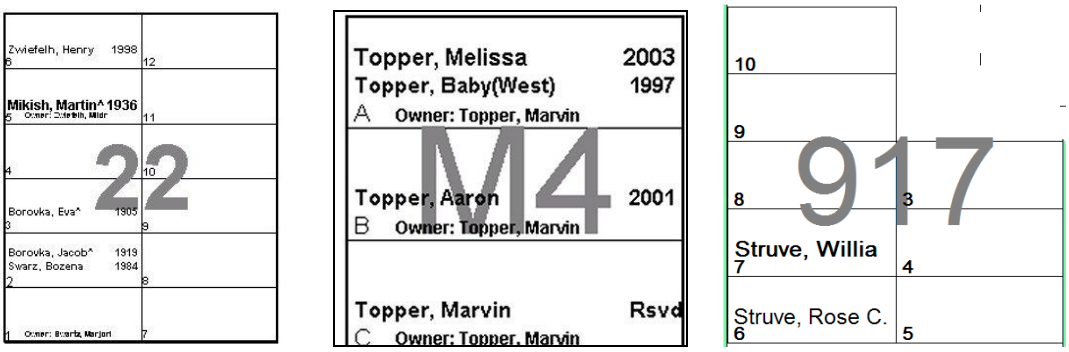
Owner record name information, which is always displayed as the lowermost line with "Own:" in front of it.

Contact record, which is displayed only if there is no owner record for the grave in which case it will say "Cnt:" in front of it instead of "Own:"

The names will be shortened/truncated if too long to fit in the grave space.

It also contains the: **Grave id**, which is displayed in the lower left corner.

Lots

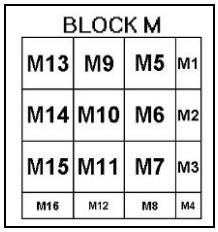


Lots are grouped into rectangular blocks each having a unique id. Different types of lots may be defined. Each type of lot has a specific number of graves arranged in the same way and with grave Id's arranged the same. The lot displayed on the left above shows a lot with an id of 22. This lot has 12 graves lined up in 2 rows of 6 with grave ids of 1 thru 12.

In the middle is a lot with an id of "M4". It has 3 graves with grave id's of A, B and C.

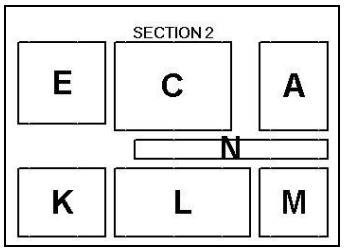
On the right is a 10 grave lot with 2 of the graves nlanked out thus making it an 8 grave lot.

Blocks



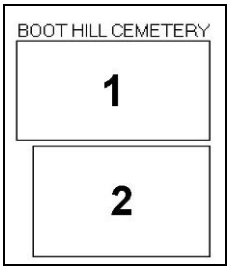
Blocks are rectangular areas each having a unique id within the cemetery. They contain one or more lots. Above is an example of a block containing 16 lots of various types showing their ids?

Sections



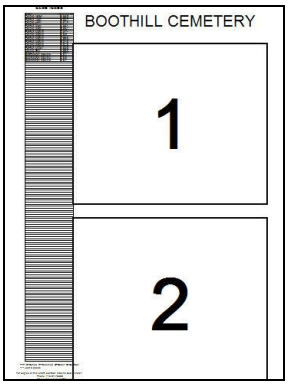
A section is a rectangular area that contains one or more blocks. Each section within the cemetery has a unique id. Above is an example of a section containing 7 blocks with their ids?

Cemetery



A cemetery is a rectangular area that contains one or more sections and the name of the cemetery Above is an example of a cemetery containing 2 sections with their ids.

MAP – Index and Cemetery



MAP CONFIGURATION FILE BASICS.

Definitions

Map coordinate system

The cemetery and all other members of the map are defined by using a grid system. The units of the grid system correspond to inches in your cemetery. Each point in the system is defined by an x,y coordinate. The x,y coordinate of the upper left hand corner is 0,0.

Moving to the right causes the x coordinate to advance and moving downward caused the y coordinate to advance. If you wanted to define a spot 10 units to the right and 20 units down in the grid the coordinate would be 10,20.

Insert points (ip)

Each members insert point is defined as the x,y coordinate where a member will be placed in the grid. The member will be positioned so that its upper left corner is at that coordinate.

Expressions (*exp*)

An *exp*ression is a string of variables, numbers and/or integers connected with +, -,/ or * signs representing addition, subtraction, division and multiplication respectively.

They are not much different than regular algebraic expressions. No spaces are allowed within the expression. No parentheses are allowed either. As in regular algebraic expressions, the * and / functions will be done before the + and – functions.

This document will use the word ***exp*** to specify that an expression can be used for something.

Numbers (*nbr*)

A number is a numeric string with or without a leading plus or minus sign. It may also contain a decimal point. This document will use the word ***nbr*** to specify that a number can be used for something.

Integers (*int*)

An integer is a number without a decimal point. This document will use the word ***int*** to specify that an integer can be used for something.

Members (mbr)

All of the map member have names that are composed of a type name and id. The format is: *type.id* where the *type* defines the configuration type and the *id* defines which occurrence of the type is defined. Examples:

LOT.3 . Defines a lot with id of 3
BLK.A2 . Defines a block with id of A2

The id and type are both case insensitive (For example LOT.H is equal to lot.h). This document will use the word ***mbr*** to specify that member can be used for something.

All *type.id* or *mbr* combinations must be unique. Types are always 3 characters in length. Id's can be of any length but it is generally a good idea to keep them short.

Owners (own)

Many map members can also be owners of other map members. For instance, all BLK members are owners of LOT members and they are also members of SEC members. This relationship will become clearer when we discuss the configuration process.

Variables(vbl)

A variable is an entity that represents a number. It is denoted by its first character being an alpha character. The remainder may be either alpha or numeric. This document will use the word ***vbl*** to specify that a variable can be used for something. Allowed variables are:

T*cmbr*= y coordinate of top of *cmbr*
L*cmbr*= x coordinate of left side of *cmbr*
R*cmbr*= x coordinate of right side of *cmbr*
B*cmbr*= y coordinate of bottom of *cmbr*
W*cmbr*= width of *cmbr*
H*cmbr*= height of *cmbr*
P*1tcmbr*= upper left corner of *cmbr*
P*3cmbr*= upper right corner of *cmbr**
P*5cmbr*= lower left corner of *cmbr**
P7*cmbr*= lower right corner of *cmbr**
P2*cmbr*= midpoint of the top of the *cmbr**
P4*cmbr*= midpoint of the left side of the *cmbr**
P6*cmbr*= midpoint of the right side of *cmbr**
P*8cmbr*= midpoint of the bottom of the *cmbr**

*=The Pncmbr variables are only allowed in I= descriptor expressions.

Cmbr is a compressed version of *mbr*. Its format is *typeid* which is just a *mbr* without the period. Examples:

TLOT5 means the y coordinate of the top of lot 5
WCEM1 means the width of the cemetery
P7BLK1 means the bottom right corner of block 1.

Special abbreviation for lot variable

If the 3 characters following the variable type are not a valid member type, the type is assumed to be "LOT" and the characters following the variable type are assumed to be the id. In other words LLOT5 and L5 would all be equivalent.

This saves a good deal of typing as the largest percent of variables in a map are probably LOT type variables.

This format works as long as you do not start a lot id with any of the 3 character member types. If you had a lot id of BLK5, for instance, the variable BBLK5 would find block 5 instead of the lot with id of BLK5. In this case you should use the full format:

BLOTBLK5 to be sure of finding the correct member.

Descriptors (desc)

Descriptors contain information about the member and are added directly behind the member with a leading semicolon (";"). More than one descriptor may be used by adding more semicolons and descriptors as needed. No spaces may occur anyplace in the member or its descriptors. Each descriptor has the forms *desc=text whe*re *desc* is the descriptor code and *text* is the value for it. The configuration line "**PIK.1;W=120;H=48"** defines picture PIK.1. The first descriptor, "W=120" says that the picture is 20 units wide. The second descriptor, "H=48 says that the picture will be 48 inches high. The descriptor type is case independent. Allowable member descriptors are:

DESCRIPTOR	MEMBER TYPES USED ON	COMMMENTS
GC=*int*	LTY*	Defines the number of grave columns in the lot type
GR=*int*	LTY*	Defines the number of grave rows in the lot type
GW=*int*	LTY*	Defines the width of each grave in the lot type
GH=*int*	LTY*	Defines the height of each grave in the lot type
LTY=*id*	LOT	Defines the lot type id. If not present, the lot type for the lot is assumed to be the first lot type defined in the configuration. The alternate way is to put a / behind lot id followed by the lot type id. Ie LOT.5/AK
L=*exp*	All except LTY	Defines where the left side (x coordinate) of a member is to start
T=*exp*	All except LTY	Defines where the top (y coordinate) of a member is to start
I=*exp*	All except LTY	A combination of the L= and T= descriptors. It allows you to specify both the left and top insert points by using the special P*nmember* variable. See below.
W=*exp*	NAM* TXT*, PIK*	Defines the width of a member
H=*exp*	NAM* TXT*, PIK*	Defines the height of a member
COLS=int **	SEC, BLK, LOT	Defines the max number of columns per row in a group
VG=exp **	SEC, BLK,LOT	Defines the vertical gap between rows in a group
HG=exp **	SEC, BLK, LOT	Defines the horizontal gap between members of a group
A=/gid/gid……..	LTY*	Defines grave ids for the lot type. See Lot type info below.
A=ROT.angle **	BLK	Defines angle to rotate the block through. Discussed later.
A=CTR.x.y **	LOT	Defines where the owning blocks id is placed if it is to be shown. See discussion on block id positioning
A=OTL.int.x.y **	LOT	Defines how the outline of a block is to be drawn if not a rectangular box. See discussion on block boundary outline
A=text.txt.fsz.xp.yp	TXT	Defines the text and how it is to be formatted. Discussed later.
A=T.txt.fsz.xp.yp	TXT	Same as above but multiples allowed
A=intH or A=intV	DIM	Defines the length of a dimension and its direction. Discussed later.
A=F.shadel	BOX,CIR	If present circle or box is filled with the designated grey shade
A=R.begin.end	CIR	Defines beginning and ending degerees for circle
A=B.wid.shd.sty	BOX.CIR	Defines attributes of box and circle boundaries(width,gray shade style)

*=Descriptor required **=Will be explained in Map configuration file advanced section

The member size descriptors H= and W=are not allowed on members that are also owners. On those members, the program computes the sizes based on the rightmost and bottom most members of the that owner.

I= descriptor clarification

When you use the I= descriptor, the program actually converts it to the L= and T= descriptors before evaluation. To further explain, assume that you have Lot 7. The following I= descriptors are converted as shown.

I=descriptor	T=descriptor	L=descriptor	COMMENTS
P17	T7	L7	Lot type is implied
P57	B7	R7	Abbreviated form only used here. page.
P2.7	T7	T7+W7/2	W7/2 gets us halfway
P67	T7+H7/2	R7	H7/2 gets us halfway
P87+20	B7+20	L7+W7/2+20	If no trailing V or H, the number is both.
P37+35V+45H	T7+35	R7+45	Selective based on the trailing V or
420V+200H	450	200	Absolute positioning

Types.

Many member types are also owners of other member types. See the hierarchy of types defined in the cemetery map layout.

The following table lists all allowable types along with allowed owners, how their size is computed and a short description of each. Further discussion on each follows.

Map type	Allowed on owners	Sizes	Usage
MAP*	None	Computed	Defines the map which contains the index and the cemetery map. Also used as a place holder for defining lot types.
VBL	None	N/A	Defines a variable string, See details later
LTY	MAP**	See descriptor table above	Defines a lot type and its associated grave configuration
CEM*	MAP**	Computed	Defines the cemetery level
NAM	CEM**	Required	Defines the name for the cemetery
SEC*	CEM**	Computed	Defines a section within the cemetery
BLK*	SEC**	Computed	Defines a block within a section
LOT	BLK**	Determined by lot type	Defines a lot within a block
TXT	All	Required	Defines text and its placement within the owner
DIM	All	N/A	Defines a dimension and its placement within the owner
PIK	All	Required	Defines a picture and its placement within the owner
LIN	All	Required	Defines a line. See details later.
BOX	All	Required	Defines a box. See details later.
CIR	All	Required	Defines a circle. See details later.

*=Owner type **=At least one the specified map type must be defined for this type.

N/A means that no size is required
Required means that the user must add the W= and H= descriptors.
Computed means that the program computes the sizes based on the rightmost and bottom most members

MAP, CEM and NAM must all have an id of 1 and only one of each may be defined.

MAP

Defines the map which contains the index and the cemetery map. Also used as a place holder for defining lot types.

VBL

Defines Defines a string variable that can be used in the map config later by entering it in brackets. Example

VBL.SIZ 100
PIK.SHED;H=75;W=[SIZ]

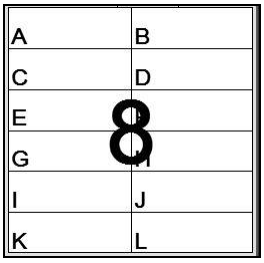
This replaces **=[SIZ]**with 100 in the second line. You may change the variable later in the config by reentering it with a different value.

LTY - Lot Types

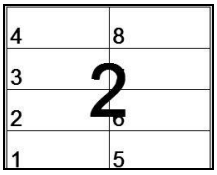
The A= descriptor defines the grave id's for the lot type. Grave id's are listed from the lower left corner of the lot type and proceed up the left column. Upon reaching the top of the column, they start with the next columns lowest grave and proceed up that one and so on. Each id must have a forward slash ("/") on each side of it. If you use numeric ids in a lot type, they must all be numeric. If you use alpha ids in a lot type, they must all be alpha types and only one character per id is allowed. The number of id's must be exactly equal to the number of rows times the number of columns. For example the 2 lot types:

LTY.1;GC=2;GR=4;GW=120;GH=48;A=/1/2/3/4/5/6/7/8/
LTY.AK;GC=2;GR=6;GW=100;GH=40;A=/K/I/G/E/C/A/L/J/H/F/D/A/
LTY.XYZ;GC=2;GR=5;GW=100;GH=40;A=/6/7/8/9/10/5/4/3/*/*/

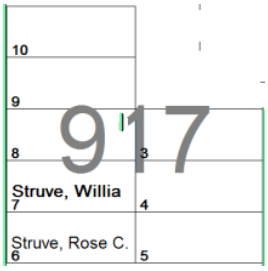
Would look like this:



Type AK



Type 1



Type XYZ

Note that lot 917 has 2 null graves denoted by putting an "*" as the grave id.

See LOT definition below to see how a lot's lot ype is configured

CEM – Cemetery

Defines the cemetery and its members.

NAM - Cemetery name

It is recommended that it be placed in the top part of the cemetery and have a width equal to the width of the cemetery and a height equal to about 6 times the grave height.

Example: **NAM.1;W=WCEM1;H=288**

SEC – Section

Defines a section within the cemetery.

BLK – Block

Defines a block within a section.

LOT – Lot

Defines a lot within a block.

Lot types for lot members are, by default, equal to the first LTY member of the MAP statement.

This can be overridden in two ways.

First, by adding a forward slash("/") behind he id followed, directly by the id of the desired lot type.

Secondly, by using the LTY= descriptor.

Examples:

BLK.1 LOT.Q . Q gets default lot type of 1
LOT.2/AK . 2 gets a type of AK

TXT - Text

Note: TXT is being phased out. Use the BOX statement with the /T option instead.

Format: TXT.id location,size;A=/T.text.fontsize.xposition.yposition/P/V

The /T. is optional. If not there program assumes the text is first
Fonsize* is height of the text desired in cemetery units. (Dflt is .75 times the box height
Xposiction* is horizontal offset from the left side of the defined box. (Dflt is C)
L = left aligned
C = centered
R = right aligned
Number = units horizontally from the left side.
Yposiction* is vertical offset from the top of the defined box. (Dflt is C)
T = topt aligned
C = centered
B = bottom aligned
Number = units verticallyu from the top.
/V* causes the text to be printed vertically upward in the defined box.
/P* causes a pointer to be added behind the text.

Text You may not have spaces anyplace in the definition of the text. For that reason and others the program established the reverse slash as an escape character. If it is entered into the text in the configuration, it and the following character take on a special meaning as defined in the following table.

Slash sequ	Effect	Comments
\s	Replaces \\s with a space on output.	Allows you to enter spaces in the text.
\\	Replaces \\ with a single \ on output.	Allows you to enter back slash character in the text.
\/	Replaces \/ with a single / on output.	Forward slash is used to denote options in the attribute.

		This allows you to enter back slash characters in the text
\u	Replaces \u with the underline Character on output.	The underline character causes problems in the configuration. This allows you to enter them in the text
\c	Replaces \c with a comma on output.	The comma character causes problems in the configuration. This allows you to enter them in the text

Warning: The character following the reverse slash is case dependent. If you enter a reverse slash and the following character is not in this table, those 2 characters will be ignored.

Examples

TXT;P59;H=12;W=80;A=Alleyway . Place text in a 12x80 area below lot 9. It will be centered . with a font size of 9

TXT;P39;H=12;W=80;A=T.East.6.L.5/P . Place text in same area with a pointer. It will be left . aligned 5 units from the top with font size of 6.

DIM - Dimension

A dimension is a line with an arrow on each end and a number in the middle stating how many units exist between the two arrow tips. It can be vertical or horizontal. It starts at the insert point specified and is drawn in the direction specified by the A= descriptor. The descriptor format is of one of the following four formats where *int* is the number of units between arrows. It does not modify the implied insert point.

1. *int*H Draw the line from the insert point to the right *int* units.
2. –*intH* Draw the line from the insert point to the left *int* units.
3. *int*V Draw the line vertically downward *int* units.
4. –*int*V Draw the line vertically upwards *int* units.

If the trailing character is not a V, it is assumed to be horizontal..

Examples:

DIM.SC1;P8SEC1;A=240
. Draws from midpoint of bottom of section 1 downward 240 units
DIM.22;A=-50V
. Draws line from current insert point upwards 50 units

OTL - Outline

This used to be called CTR

By default, when the program draws a block outline, it draws a rectangle that is big enough to wholly contain all of the blocks in it.

You can override this by defining a series of points which tell the program to draw the outline using those points rather than generating a rectangle. The program will draw the described polygon by drawing a line form point 1 to point 2 then from point 2 to point 3 ….. Etc. When the last point is reached, it draws a line from that point to point 1 finishing the polygon. You may define as many points as you wish but just getting close to the desired shape is sufficient.

You specify a point by adding an A=OTL.ptr,x.y where ptr is an integer specifying a point on the polygon you are defining, x is the \horizontal offset from the lots upper left hand corner and y is the vertical offset. If you require more than one point on the same lot, separate the point definitions by a slash. Up to 9 points may be defined, it any points are skipped, the polygon will include only the points up to the first skipped one

Example 5 point polygon:

LOT.1;A=OTL.1.0.0/OTL.2.240.0 . first 2 points along top of lot 1
LOT.3;A=OTL.3.0.240/OTL.4.288.240 . second 2 points along right side of lot 3
LOT.5;A=OTL.5.288.0 . fifth point on lower left corner of lot 5

BOX

Defines Defines a box . Example of a box 100 units wide and 200 units high.

BOX.B1 W=100;H=200

The following attributes may apply:

/B.width.greyscale.drawstyle . Defines the attributes of the boundary line
/F.greyscale,fillstye . If present the box will be filled with the gray scale specified

Width is the line width and defaults to 1.
Grayscale is the level of greyness (defaults to 6-black for boundary and 0-white for fill.
Drawstyle is the style of boundary line. Default is 6(inside solid)

```
0 Solid - 1 Dash  - 2 Dot - 3 Dash-Dot
4 Dash-Dot-Dot -    5 Transparent -
6 (Default) Inside Solid
```

Fillstyle is the style used to fill the fill the box (Default is 0-solid)

```
0 Solid                 1 (Default) Transparent
2 Horizontal Line       3 Vertical Line
4 Upward Diagonal       5 Downward Diagonal
6 Cross                 7 Diagonal Cross
```

/T.text.fontsize.xposition.yposition
/P
/V

These last 3 options are explained in the TXT member discussion. You can have more than onr "/T" Attribute but only the first is affected by the /P and /V parameter

Example:

BOX.addr L=0;T=0;W=400;H=20;A=& . Defines box 20x400
/B.1.6.6& . Draws box 1 line wide around it
/F.0.1 . Not needed as ?B parameter said xparent
/T.Marvin\sTopper.10.0.L . Address in top of nox
/T.Bloomington\c\sMN\s55437.10.10.L . Address in bottom of box

CIR

Defines Defines a circle(or partial circle) . If the lenth and width are different, an ellipse will be drawn.

Example of a ellipse 200 units wide and 100 units high.

CIR.C1 W=200;H=100

/R.begin.end . Defines the begin and end angles of the arc to be drawn
Begin is the starting angle in degrees (Default is 0
End is the ending angle in degrees (Default is 360).
Precede them by a –sign to draw a radius (-0 doesn't work (use –359)
Note: to fill a pie slice arc both radii must b preceded by a – sign.

LIN

Defines Defines a line . It is drawn in an imaginary box specified by the height and width parameters.
Example of a horizontal line 1 unit high .

LIN.L1 W=100;H=1

The following attributes may apply:
/B.width.greyscale.drawstyle . Defines the attributes of the line. Same as for BOX
/U the line will be drawn from the lower left corner to the upper right corner of the box.
Fefault is to draw fron upper left to lower right.

PIK - Picture

Examples:

PIK.CHURCH;W=WSEC1;H=.75*WSEC1
PIK.SHED;H=H7;W=H7*.75

Implied Insert Points

It is certainly possible to configure the whole map without using the implied insert point default logic. You would have to put the positional descriptors on each and every member. Using implied insert points removes this tedium.

The configuration program thinks of the owner type as being comprised of rows and columns of members. When the configuration for a new owner starts the member row is assumed to be 1. The next member row is signified by entering a member of one character equal to a forward slash.

See Appendix A for a walk thru example.

The x insert point is affected in the following ways:

The first member has in insert point of 0,0 relative to its owner

If a member has an L= *exp* descriptor, the x portion is set to the *exp* value prior to configuring the member.
If a member has a T= *exp* descriptor, the y portion is set to the *exp* value prior to configuring the member.

If a member has a I= *exp* descriptor, both the x and y portion are set according to the *exp* value prior to configuring the member.

If a member equal to /HS=*int* is encountered, the x portion is advanced by *int* units.
If a member equal to /VS=*int* is encountered, the y portion is advanced by *int* units.

If a member equal to a single slash is encountered the insert point is set equal to the coordinates of the lower left corner of the first member of the current member row within the owner. The member row number is then advanced.
In all other cases, the x portion is advanced by the width of the configured member.

Groups

We have defined an owner as consisting of single members and positional parameters (/, /HS or /VS) to ease the burden of understanding the configuration process. This section will define a new concept where an owner is defined to be composed of groups. The new configuration format will be "owner group group group ..etc where the group format is "grouptype.ids;descriptors". The below example consist of the owner and 3 member groups..

BLK.W LOT.3-9.12-15;LTY=AK / LOT.K-A;COLS=3

Grouptype is one of the 3 digit configuration types.

Ids can have the following formats

Id1 which defines a one member group.

Id1-id2 defines a range of ids. A value of 1-5 would cause 5 lots to be defined with id's 1 thru 5. A value of A-F would cause 6 lots to be defined with id's A, B, C, D, E and F. A value of D2-D4 would cause 3 lots to be defined with ids of D2, D3 and D4. You may use reverse order such as 9-5 or Z-M or E7-E1 if you wish. If you use the mixed alpha/numeric format in a range such as G1-G6, the letter must be single and the same on both the first and last ids in the range. If you want lot ids like A1 thru A12 and use A1-A12, the lots will have id's A1, A2, A3…..A12. If you configure them as A01-A12, they will have ids A01, A02, A03…..A12. This is desirable when it comes to sorting records by lot into proper order.

1d1.1d2.1d3 defines a list of single ids.

1d1-1d2.id3.id4-id5 defines a mixed list of ranges and singles.

/ defines a group row advance

/HSint defines x coordinate advance

/VSint defines y coordinate advance

Example groups:

LOT.1.A-C.7-9.12.14-16 . Defines lots 1, A, B, C, 7, 8, 9, 12, 14, 15 and 16.
LOT.1.5./.7-9 . Defines lots 1, 5, 7, 8 and 9 with 7, 8 and 9 below 1 and 5

Groups may not have any embedded spaces and must be separated by one or more spaces.

As you will see shortly, the owner member relationships in the previous definition are equivalent to a group definition where all groups have a single id. All of the rules for groups shown below would apply to single member definitions above as though they were a group.

The power of using groups is to allow you to define several members very quickly.

Group insert points

We now think of an owner as being comprised of rows and columns of groups. Groups, in turn are comprised of rows and columns of members.

Within a group, the group is assumed to be an owner and the member insert point applies for each id within the group.

When a configuration for a new group starts the member row within the group is assumed to be 1. The next member row in a group is signified by entering a single member id equal to a forward slash. It is important that you understand this two level concept when we discuss insert points here.

The implied insert point for the first group in any owner is 0,0. The implied insert point for the first member of any group is the same as the insert point for the group.

See Appendix B for a walk through example.

The x insert point is affected in the following ways:

If a group has an L= *exp* descriptor, the x portion is set to the *exp* value prior to configuring the group.
If a group has a T= *exp* descriptor, the y portion is set to the *exp* value prior to configuring the group.
If a group has a I= *exp* descriptor, both the x and y portion are set according to the *exp* value prior to configuring the group.

If a group or member id equal to /HS=*int* is encountered, the x portion is advanced by *int* units.
If a group or member id equal to /VS=*int* is encountered, the y portion is advanced by *int* units.

If a group equal to a single slash is encountered the insert point is set equal to the coordinates of the lower left corner of the first group of the current group row within the owner. The group row number is then advanced. Note that the lower left corner of a group is defined as the lower left corner of the first member of the last row of the group.
If a member id of a single slash is encountered the insert point is set equal to the coordinates of the lower left corner of the first member of the current member row within the group. If there was a VG=*int* descriptor for the group, the y portion is advanced *int* more units. The member row number is then advanced. If there was a COLS=*int* descriptor for the group and *int* non positioning members have been done, a single slash will be emulated and a new row will be started.

In all other cases, the x portion is advanced by the width of the configured member. If there was an HG=*int* descriptor for the group, the x portion is advanced *int* more units.
When a group configuration is complete, it is set equal to the coordinates of the upper right corner of the last member of the first row of the group.

Insert point logic is not used on LTY members.

Group descriptors.

LTY=id

If used, it applies to all ids in the group he without the forward slash behind them specifying their lot type's id. Example:

BLK.1 LOT.Q .R . Q and R get default lot types of 1
LOT.2-4/AK . 2, 3 and 4 get lot types of AK
LOT.7.8/1.9-11;LTY=AK . 7, 9,10 and 11 are type AK and 8 is lot type 1

L=*exp*, T=*exp* and I=*exp*

If any of these are used, they apply to only the first member of the group. All other members are positioned based on implied positioning or positional ids within the group.

VG=*int* or *VG=int1/int2/int3……../intn*

This causes a vertical gap between rows of a group. Technically, it causes this gap to appear between the new member rows first member and the previous rows first member. Unless you are doing some odd positioning, it will apply to the rest of the members in the row because the insert points y coordinate will be the same as that of the first one in the row. The second form will use int1 for gap between 1st and 2nd, int2 for gap between 2nd and 3rd etc.

HG=*int* or *HG=int1/int2/int3……../intn*

This causes a horizontal gap equal to *int* between members within a row in the group. The second form ises the same logic as VG but for column separation.

COLS=*int*

This causes an automatic member row advance within a group if *int* members have been processed and there are more. Basically it emulates a single slash positional id.

Configuration format

Example configuration

```
1       .
 2      .              EXAMPLE CEMETERY MAP CONFIGURATION
3       .
4       MAP.1    CEM.1
5                LTY.1;GC=2;GR=4;GW=120;GH=48;A=/1/2/3/4/5/6/7/8/
6                LTY.AK;GC=2;GR=6;GW=100;GH=40;A=/K/I/G/E/C/A/L/J/H/F/D/A/
7       CEM.1    NAM.1;W=WCEM1;H=288   /  SEC.1
8       SEC.1    BLK.1   BLK.2        . Block 2 will appear on the right of block 1
9       BLK.1    LOT.1   LOT.2    LOT.3   LOT.4
10      BLK.2    LOT.5   LOT.6
11               LOT.7   LOT.8
```

Owner and member format

Owners are always entered starting in column 1.

Members for that owner along with their descriptors are entered between that owner and the next owner. A member is followed by its descriptor(s) if present and all of them are tightly bound with no embedded spaces. Member/descriptor pairs are separated by spaces and or by advancing to the next line.

In the example configuration above for instance:
SEC.1 has 2 members.
BLK.2 has 4 members.

Comments

Comments may be inserted to help explain the configuration and/or make it more readable.

The whole line can be made into a comment by putting a period (“.”) I n the first column. Entering a space-period-space (“ . “) in the line will allow comments to be added on the same line as actual configuration information.

See lines 1,2,3 and 8 in the example

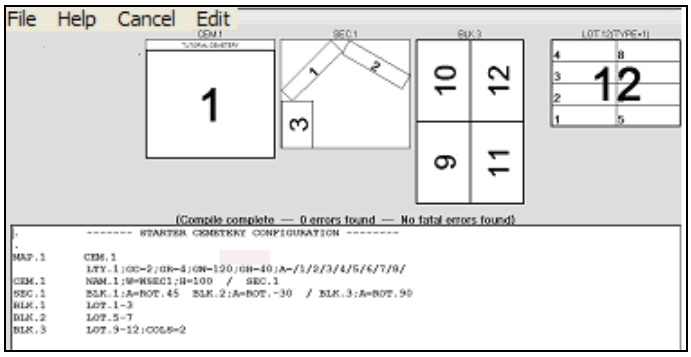
MAP CONFIGURATION FILE - ADVANCED.

Rotating blocks

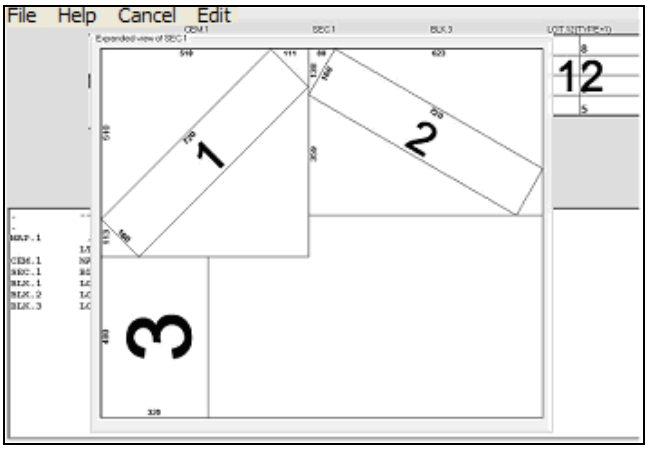
Blocks may be rotated by 1 degree increments up to plus or minus 90 degrees. To rotate a block set the A= descriptor on the block to A=ROT.intdeg. This will cause the block to be rotated intdeg degrees. Configure the block as thought the upper left hand corner is going to be positioned at the desired insert point.

The program will calculate the rectangle that will just hold the defined block and rotate it within that rectangle. The size of the block for display and implied insert point purposes is now the size of the big rectangle.

The following picture shows 3 blocks that are rotated at different angles.



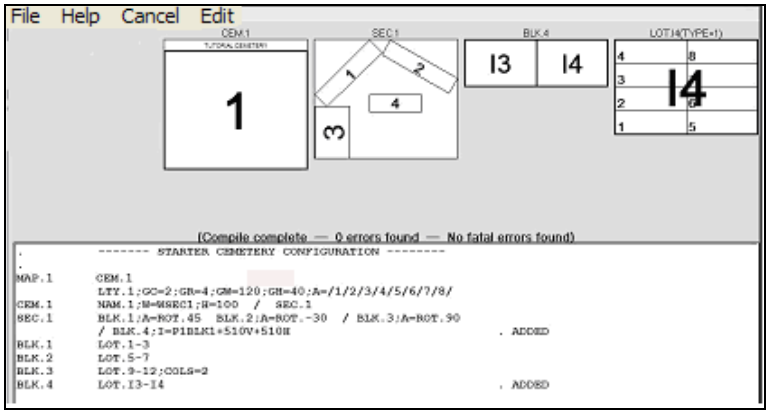
To help to see how it works, we will explode the SEC.1 box to show dimensions.



This exploded view shows the dimensions of the rotated block. It also shows on the left side, the vertical distances from the rotated blocks corner to the left top and left bottom of the calculated square. The same is true of the width measurements at the top. This will come in handy for positioning other blocks as they will be relative to the big square, not the rotated block.

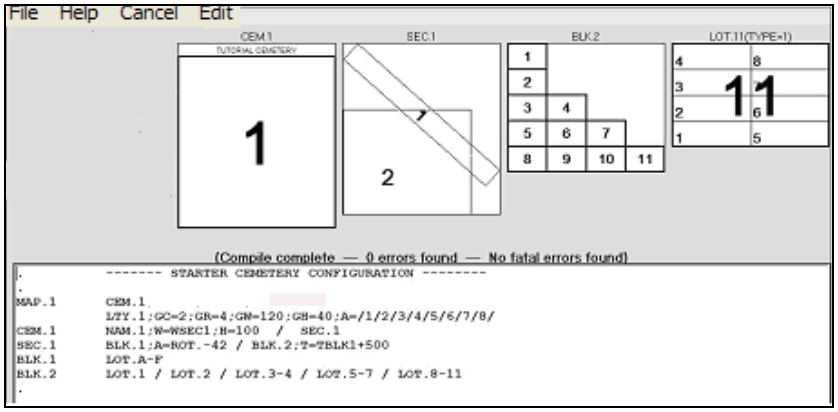
Assume that you wanted to add block 4 with its y coordinate at he same point as the y coordinate of top left of the rotated block number 3 and its x coordinate at the same coordinate as the upper right of rotated block number 3.

The changes required to do that are shown below along with the resulting diagram.



Boundary outline for blocks

Assume that we build 2 blocks. One is rotated to fit along a road on the right. The second one is made to fill in the corner on the left. Following is the configuration and its resultant map. By default, when the program draws a block outline, it draws a rectangle that is big enough to wholly contain all of the blocks in it.

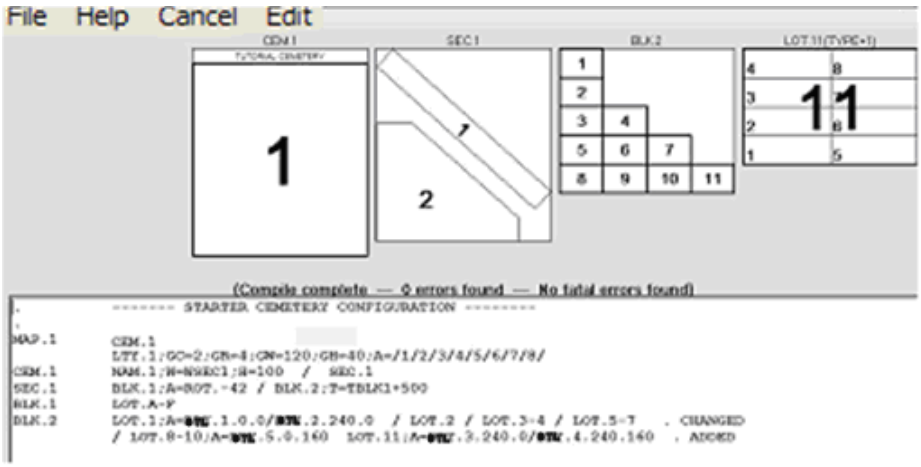


As you can see in the diagram, it looks like block 2 and block 1 overlap. To correct this problem, a series of points can be defined which tell the program to draw the boundary outline using those points rather than generating a rectangle. The program will draw the described polygon by drawing a line form point 1 to point 2 then from point 2 to point 3 ….. Etc. When the last point is reached, it draws a line from that point to point 1 finishing the polygon. You may define as many points as you wish but just getting close to the desired shape is sufficient.

You specify a point by adding an A=OTL.ptr,x.y where ptr is an integer specifying the point you are defining, x is the \horizontal offset from the lots upper left hand corner and y is the vertical offset. If you require more than one point on the same lot, separate the point definitions by a slash. Up to 9 points may be defined, it any points are skipped, the polygon will include only the points up to the first skipped 0ne

The following diagram shows how to describe a 5 point polygon to be drawn setting the points to:

Point 1 upper left corner of lot 1.
Point 2 upper right corner of lot 1
Point 3 upper right corner of lot 10
Point 4 lower right corner of lot 10
Point 5 lower left corner of lot 7.



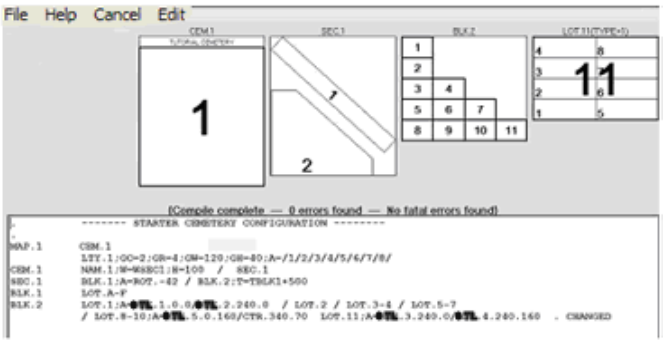
Block ID.

When the block id is to be shown, it is shown as close to the center of the block as possible. If the block is not filled squarely with lots, the block id will tend to be positioned more in the direction where the most lots are configured. Sometimes, that will not seem good enough.

To make the block id print exactly where you want it, add the following attribute to one of the lots in the block. If there is already an OTL point there, insert a slash between them.

A=CTR.*x*.*y*

This tells the program to center the block id on the position specified relative to the upper left corner of the lot. *x* is the integer x offset from that corner and *y* is the integer y offset. The following example shows the id center to be positioned 70 units below the upper right corner of lot 8



MAPEDIT - *THE MAP FILE EDITOR*

Disclaimer

ONE

The map configuration process has many options. If you enter the information correctly, everything works well. If you make errors, however, it is sometimes very difficult to determine what the error is based on the error message. To bring the compiler error diagnostics to a level of sophistication that would correct this problem would require hundreds of hours of programming and testing. This process is being done as time allows but may never be complete.

Given that this is true I recommend that you proceed as follow when configuring:

> ***Make small amounts of changes between each compile. This will help you to zero in on the spot where the error is occurring as it would have to be in the data you entered or changed since the last compile.***
>
> ***On your first cemetery map, let us do it. All we need is the detailed layout. It is free and we can do it a lot faster. We also offer free update help for at least 6 months,***

If you decide to tackle major changes on your own, make sure you understand the process and do the tutorial before starting. It is much cheaper for us to have you send us the current map file with a detailed list of changes required than for us walk you through those changes (and possible errors) on the phone or via email. It is just a matter of you backing up the file, emailing it to us as an attachment, us making the changes, emailing the result back to you as an attachment and you restoring it.

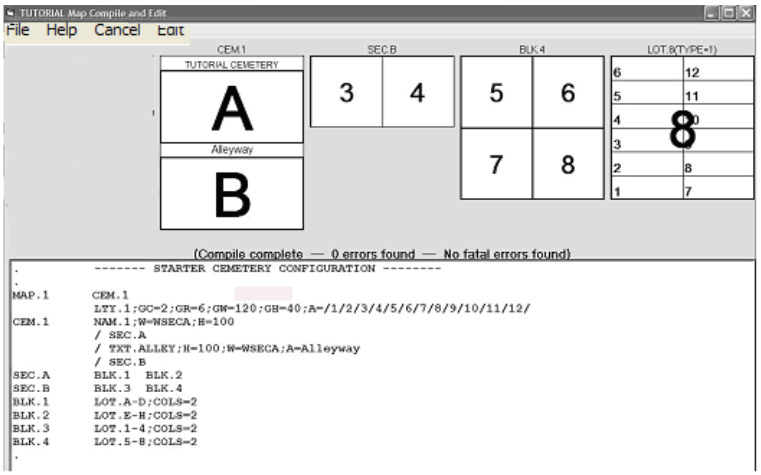
TWO

During the testing of the map edit program, we have experienced problems where the BUUTHILL program errors. If this occurs, none of the changes you made are officially saved. To negate that, you should d a Save and Exit fairly often and then restart the map edit.

The program institutes another recovery point for you. Each time you make a change to the configuration edit box, the contents of the edit box are sent to a special file. If you do a save and exit, that file is deleted. But, if you error or decide to exit the map edit by some other method than Save and Exit the file is left intact. Whenever map edit is entered this file is checked. If present, you are told that it exists and you are given the option to replace the contents of the edit box with that file. You can then continue on and do not have to reenter your data.

The form

You get to this form by going to the file menu on the report form and hitting the Map edit option.



Map

The upper part of the form shows 4 boxes depicting the 4 levels of the cemetery Map.

The first one shows the cemetery with the section layout.
You can select a particular section to be displayed by left clicking on that section in this box.

The second one shows the currently selected section with the block layout for that section,
You can select a particular block to be displayed by left clicking on that block in this box.

The third one shows the currently selected block with the lot layout for that block.
You can select a particular lot to be displayed by left clicking on that lot in this box.

The fourth one is the currently selected lot and shows the grave layout for that lot. It also shows you the id of the lot type.

File text edit box

The lower part of the form shows an edit box which displays the configuration file. This is where you make changes and also where error messages are displayed if any occur during a compile.

If you right click the mouse over this box, you will get the normal edit box which allows you to copy, paste … etc.

Compile error messages

The text line just above the text box tells you how many errors were detected on the last compile and if there were any fatal errors which would cause the upper 4 boxes to be blanked out.

If there are any errors, this message and the text in the text box will be displayed in red.

Menu items

File

The file menu contains 4 sub menus.:

Compile which will take the configuration file with any changes you made and compile it and display the map. If any errors were displayed from a previous compile, they will be ignored. Upon completion, the new configuration file will be displayed with any errors that occurred and the new map will also be displayed.

Save and Exit which will save the current file and its compiled map as the current ones and exit back to the calling form.

Print which prints the configuration displayed in the text box.

New which will fill the file edit box with a starter configuration as though you had entered it yourself? You will have to compile for the map to show this configuration

Pik update which allows you to add pictures to your picture boxes.

Help

This menu calls the help file for the form:

Cancel

This will exit the form. If you made any changes or compiles, you will be warned.

Edit

The file menu contains 3 sub menus.:

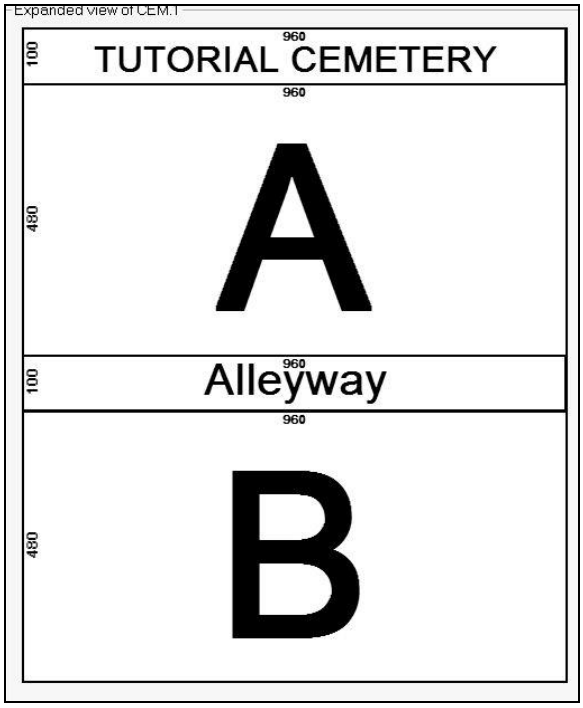
Find Shows a message box in which you enter the text you want to find in the text box.

Find Next Finds the next occurrence of the find text.

Find Error Finds the first error in the text box.

Expanded View of layout boxes

If you right click on any of the map boxes, an expanded view of that box will pop up on the form. I will show the dimensions of the members in inches. There is an advanced option which allows you to display the units in the foot and inches format (ie ***24'8"***). The picture below shows the expanded view of the section box above.



MAPEDIT TUTORIAL

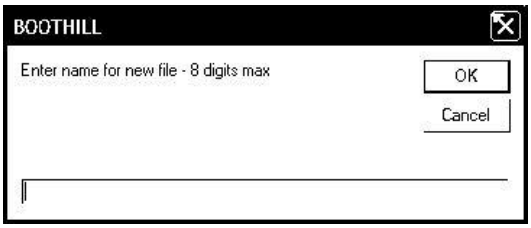
We will assume that you have read the all of the information prior to this in the document and that you are familiar with the BUUTHILL program.

Create basic map file

To start with, we are going to create a new file. The steps to do this are as follows:

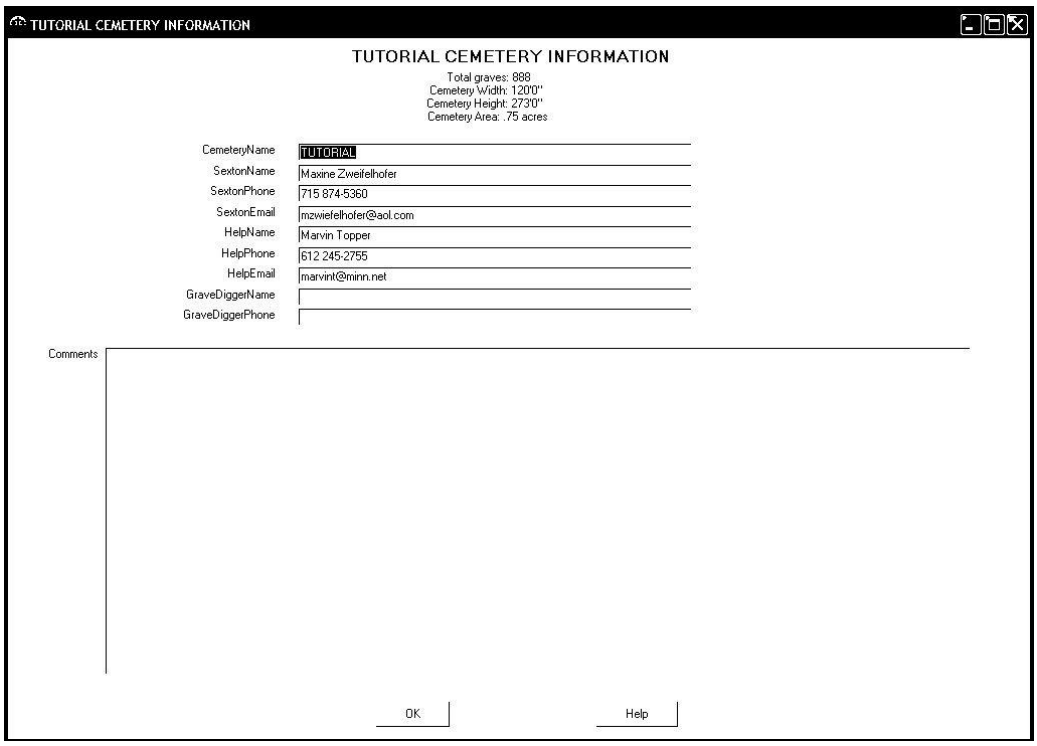
Open to your own or the sample database in the report form.

Go to the File menu and click the new submenu. You should get the following message box.



Enter any name you desire for your configuration file name and hit the OK button.

The following form will appear. Enter the name you wish to call your cemetery into the Cemetery Name field. It does not have to be the same as your file name. Then hit OK.

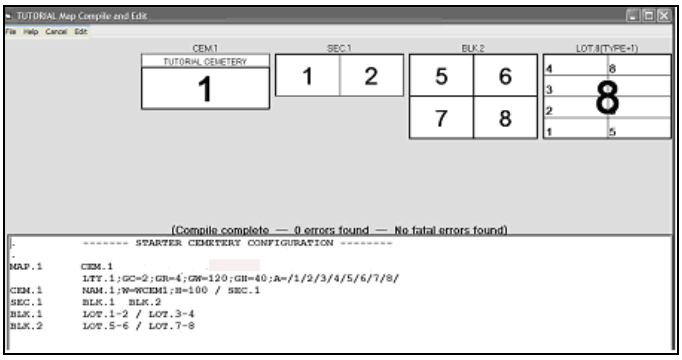


You will probably get the following warning message box. If you do, hit OK.

Now got to the File menu on the report form and hit the mapedit option. You will get a warning reminding you to backup your database. There will be no need to do that at this point. Just answer OK.

If you had previously been making changes to a configuration someplace using map edit and did not do a Save and Exit, you will get a reminder here with an option to load the configuration you were working on. Answer NO to this.

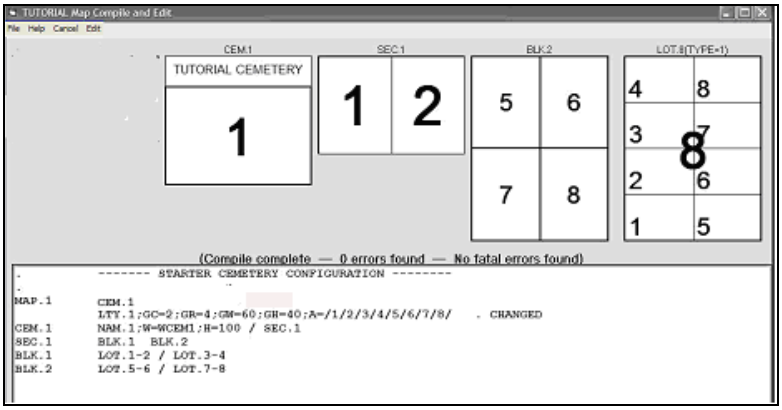
The map edit form will appear. Go to the File menu and hit the New option. A basic configuration will be placed in the edit box. Now go to the File menu and hit the compile menu. The compile will be done and you will see the following information on the Map edit form. This is your starting configuration. We will be modifying and compiling this as we move thru the tutorial.



Throughout the tutorial, the required change lines are marked with a comments called ADDED or CHANGED to help you see where recommended changes were made..

Change the grave size

Change the grave width (GW= descriptor) in the LTY.1 member to 60. Do a compile and note the difference in the map from the one above. See below. Experiment with your own sizes and observe how it changes the map. When done, restore the 120 inch width and 40 inch height.

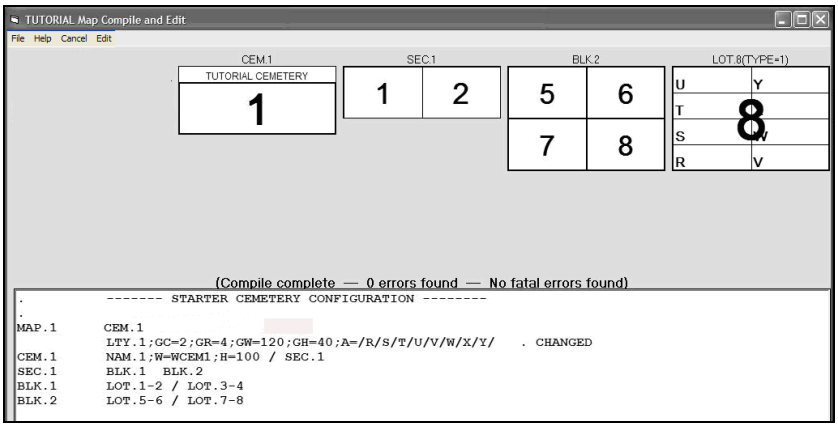


Change the grave letter/number scheme

Change the A= value on the lty.1 member descriptor form /1/2/3/4/5/6/7/8/ to /R/S/T/U/V/W/X/Y/. Do a compile and note the difference in the map from the one above. See below. Experiment with your own grave id's and observe how it changes the map. Compile it.

Try putting less or more entries than graves or leaving some of the slashes out to see what happens. Also try putting a "*" in some of the grave slots to make it a blank area within the lot.

When done, restore the /1/2/3/4/5/6/7/8/ attribute..



Add a new lot type with its grave info

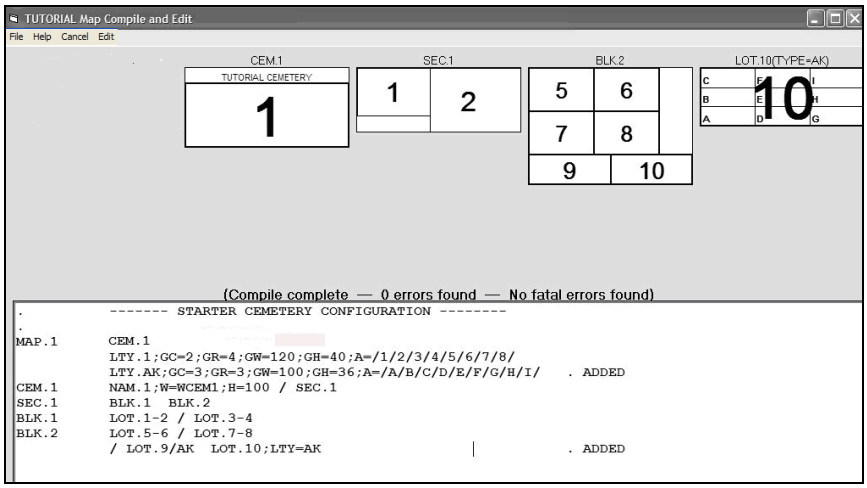
This step is included with the “add a new lot” step which is next.

Add a new lot

Add a new lot type called AK. Set the grave width to 100 and the grave height to 36. Make the lot into a 3 by 3 grave configuration with id’s of A,B,C,D,E,F,G,H,I.

Also add two new lots to block 2 positioning them just below lots 7 and 8. Set the lot type of lot 9 to AK by using the trailing slash and set the lot type of lot 10 to AK by using the LTY= AK descriptor.

Compile it. The recommended changes and resultant compile are shown below:



Note that a leading slash was required to position lots 9 and 10 below 7 and 8. This line could have also been done by using the alternate lines shown below.

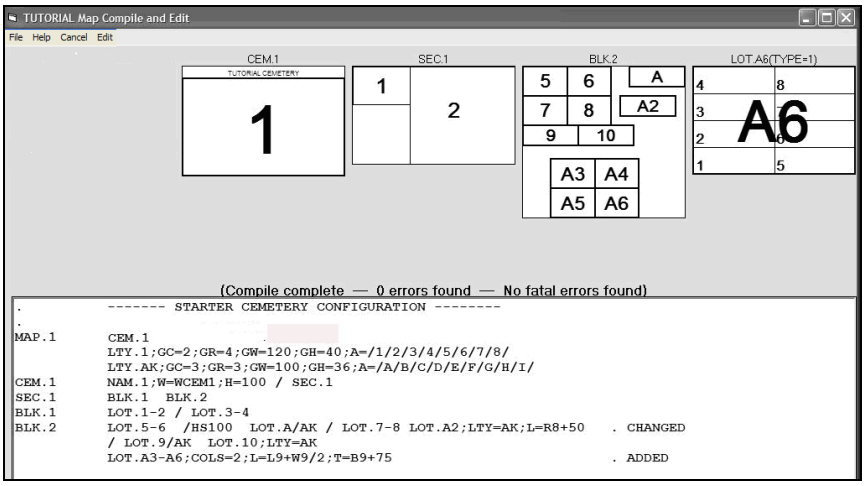
```
LOT.9/AK;L=LLOT7;T=BLOT7     LOT.10;LTY=AK
LOT.9-10/AK;L=L7;T=B7
LOT.9-10AK;I=P57
 /   LOT.9-10AK;I=P57
```

Add a group of lots

Leaving the new information added in the previous section, add 6 more lots to block 2

Add A1 as type AK 100 units to the right of Lot 6.
Add A2 as type AK 50 units to the right of lot 8.
Add the 4 lots A3 thru A6 as a 2 by 2 group with the left boundary halfway between the beginning of lot 9 and the beginning of lot 10. Make the top of the group 75 units below lot 9 and use the default lot type.

Compile it. The recommended change and resultant compile are shown below:



Notes: We used the /HS100 parameter to position A1.
We used the L= parameter to position A2.
We used the L= and T= parameter to position the A3-A6 group.
We used to COLS = parameter to make the group 2 by 2. This could have been done in other ways such as "LOT.A3-A4/A4-A5"

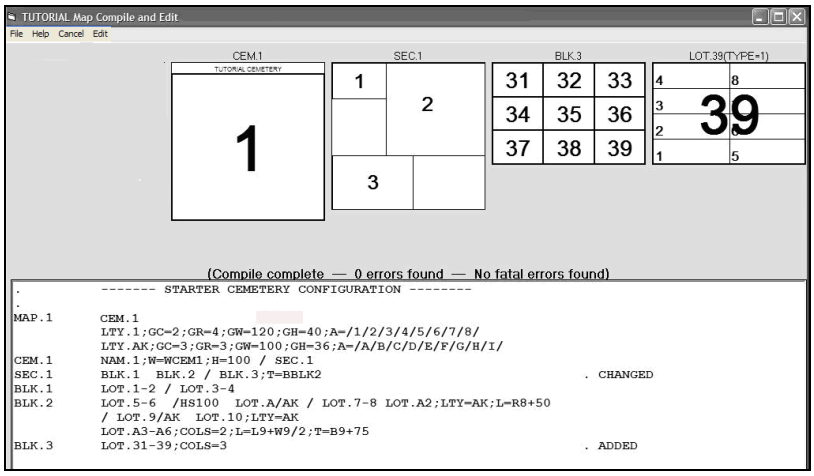
Add a new block

Leaving the new information added in the previous section Do the following:

Add a new block and call it block 3. Position it on the left side of section 1 and just below block 2.

Add a group of 9 lots called 31 thru 39 in a 3 by 3 configuration to that block.

Compile it. The recommended change and resultant compile are shown below:



Notes: We used the slash group to position the insert point of block 3 to the beginning of the row. We then had to override the default top parameter. As the rule states the insert point will default to the bottom left of the first group in the current row. That would be the bottom left of block 1. If we did that, it would cause block 3 to partially overlay block2. Try it without the T= parameter to see how it would look.

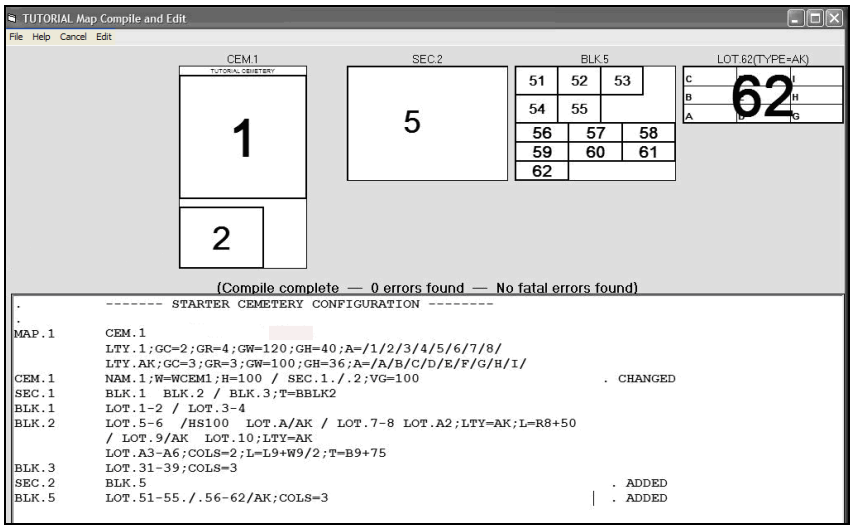
Add a new section

Leaving the new information added in the previous section Do the following:

Add a new section and call it section 2. Position it 100 units under section 1.

Configure the section with one block and call it 5. Configure block 5 with a group of 12 lots called 51 thru 62 with 5 rows with 3 in the first, 2 in the second,3 in the third, 3 in the fourth and 1 in the last. Make the lot type on 56 thru 62 equal to AK.

Compile it. The recommended change and resultant compile are shown below:
:



Notes: We created a section group and used the / to create a new row. We then used the VG=100 descriptor to cause the horizontal gap between the 2 rows to be 100 units.

We used the slash id and COLS=3 parameter to show you one way to do the requested configuration,

There are alternative ways to do both of these. They are left to your imagination..

Delete a lot from a group

This is usually just a matter of changing a range or converting to 2 ranges as required. No example will be given

Add a dimension

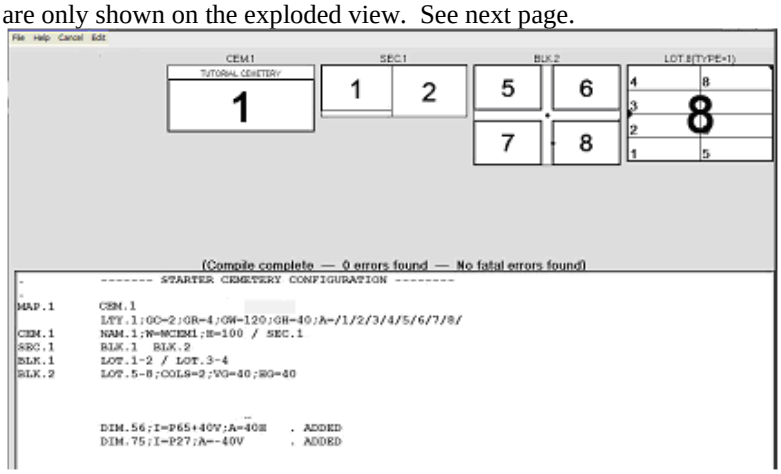
Modify the configuration leaving the markers and other changes from the last exercise.

Add a dimension arrow between lots 5 and 7 depicting a length of 40.

Add another dimension arrow between lots 5 and 6 depicting a length of 40. Put the arrow 40 units below the midpoint of the right side of lot 5.

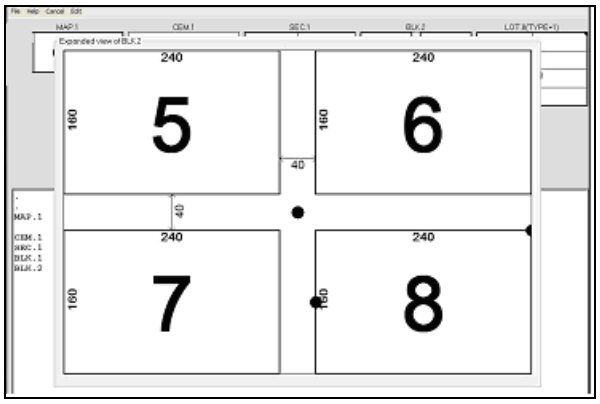
Compile it. The recommended change and resultant compile are shown below:

Note that dimensions are only shown on the exploded view. See next page.



This is the expanded view of block 2 showing the dimension markers. You get this by right clicking on the block picture.

Note: Please ignore the black circles in these pictures. They are present because a previous version of this tutorial was demonstrating how to configure lot markers. Lot markers are now added using the "Markers" menu from the Lot or Report forms.



Add some text

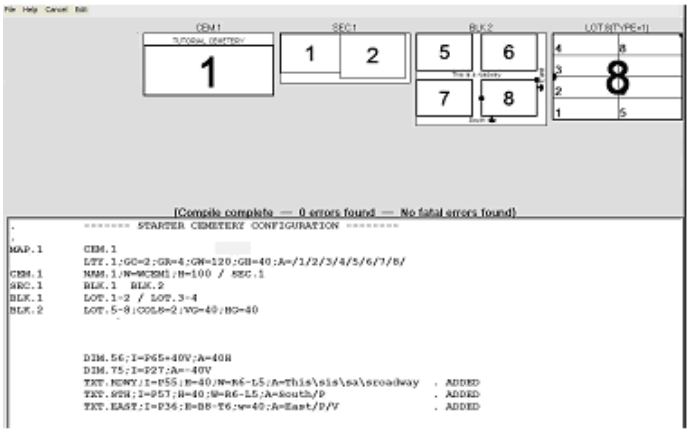
Keep the changes from the previous example.

Add text in the roadway between lots 5 and 7 centered in the block.

Add text on the bottom of the block to indicate that it is south and add a pointer.

Add text to the right side of block 2 to indicate that it is east and add a pointer.

Compile it. The recommended change and resultant compile are shown below:



Note the use of the /P and /V options and their effects.

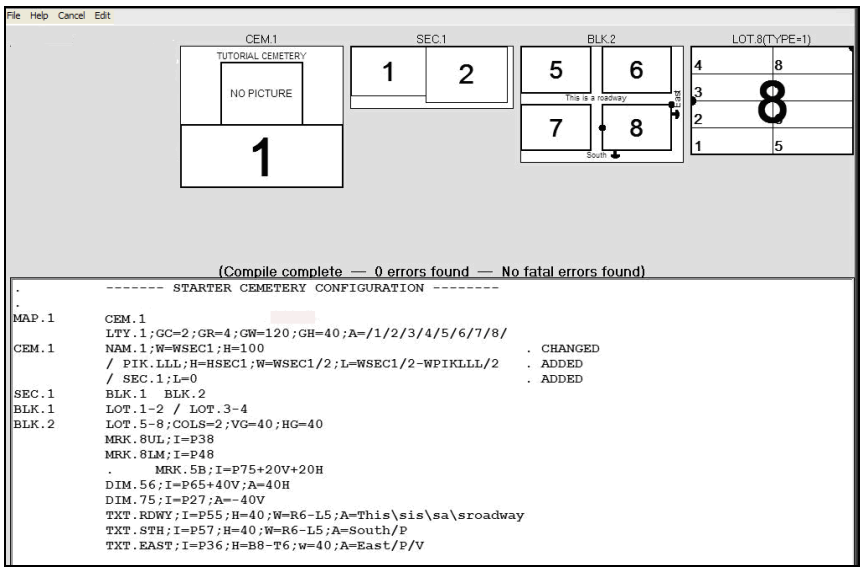
Note that we used the \s characters where we wanted spaces in the text.

Add a picture

Keep the changes from the previous example.

Add a picture with id LLL at the cemetery level. Position it centered on the top of the cemetery. Make its height the same as cemetery ones height and its width half that of cemetery 1.

Compile it. The recommended change and resultant compile are shown below:



Note that the words “NO PICTURE” appear in the picture box. You must now choose a picture to put into that box.
See next page

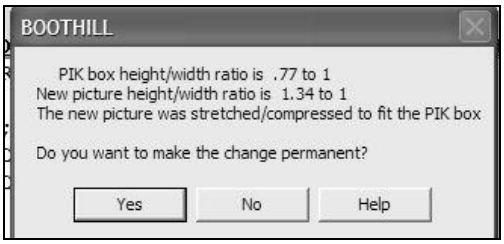
Go to the File menu and select “Pik update”. The following input box will appear.

Enter the id of the PIK you want to select in the text box. If it already appears there, just hit ok.

You will ten be presented with a windows file select menu which will allow you to browse thru your computers directory system to find the “.jpg” picture file you want. Most digital camera pictures are of this type. When you find the one you want, select it.

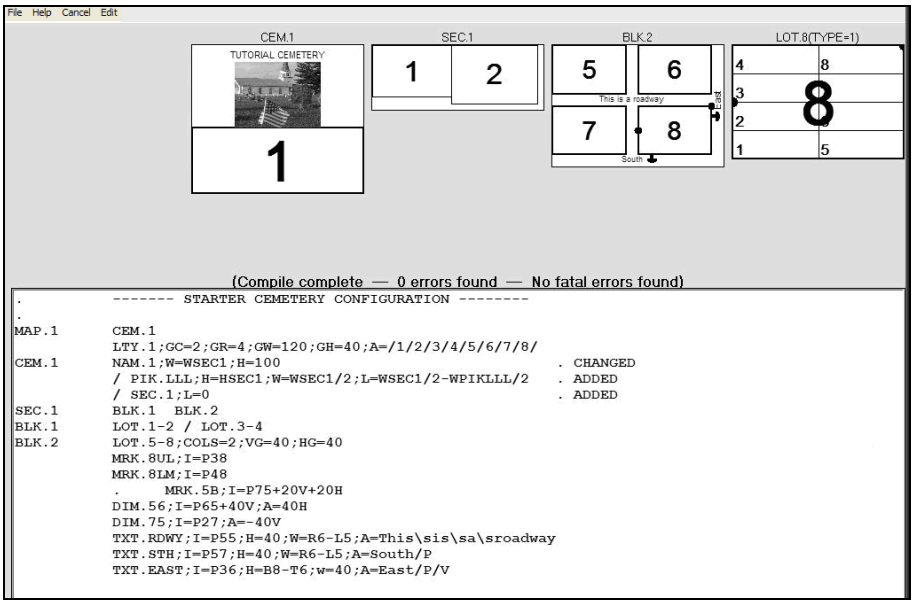
The picture will be resized and put into the picture box defined. If the proportions of width to height are different, the picture will appear skewed. I selected the picture shown on the next page. As you can see, it is visually skewed.

The following message block will be displayed to you.



The resulting picture is shown on the next page with the actual picture shown above it. As you see, looking at the message block above and the picture itself, it was squashed to make it fit in the picture box. If you answer yes, the change will be made permanent.

Actual picture



To make the width of the picture box right, we must divide the height by 1.34 or multiply it by.77 to get the right proportion of height to width. We would change the PIK descriptors as follows:

PIK.LLL;H=HSEC1;W=.77*HPIKLL;L=WSEC1/2-WPIKLLL/2

Experiment with some of your own pictures. Remember that typical photos have a height/width ratio of .75 to 1. Use something like the following suggestions for height and width.

PIK.X;H=HBLK7;W=HPIKX*.75 . Height same as block 7 – height/width ratio = .75
PIK.X;H=.75*WPIKX;W=WBLK7 . Width same as block 7 – height/width ratio = .75

If you have an image processor available like Paint Shop Pro, you can clip the image to the proportions you want before selecting it.

About using color

The program will handle colored images fine but they will cause your cemetery file to be much bigger than if you use grayscale pictures. Loading may also be a little slower and printing will be done using the color as well unless you can set your printer to print in grayscale.

Adding a rotated block

No tutorial instructions will be given. We suggest that you copy the example given in the advanced configuration section for rotating blocks and experimenting with different parameters. You can also experiment with adding an A=ROT.angle on any of the tutorial configurations shown above to see what happens.

Making a block boundary outline

No tutorial instructions will be given. We suggest that you copy the example given in the advanced configuration section for block boundaries and experiment with different parameters.

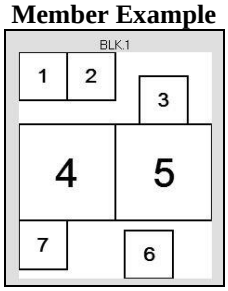
Position a block ID

No tutorial instructions will be given. We suggest that you copy the example given in the advanced configuration section for positioning block ids and experiment with different parameters.

APPENDIX A. IP WALK THROUGH - MEMBERS

In the following configurations, we will assume that the default lot is 50 by 50 and the AK lot is 100 by 100.

Member Example



BLK.1 LOT.1 LOT.2 /HS25 /VS25
LOT.3 / /VS25 LOT.4/AK
LOT.5;LTY=AK LOT.6;T=BLOT5+10;L=LLOT5+10 /
LOT.7

Working member by member:

LOT.1 wil1 be placed at 0,0 by default. The insert point will be advanced to 50,0.
LOT.2 be placed at 50,0 The insert point will be advanced to 100,0.
/HS25 will cause the insert point to advance to 125,0
/VS25 will cause the insert point to be advanced to 100,25
Comments: This is a little abnormal but we add it here to impress how the logic works.
LOT.3 be placed at 100,25 The insert point will be advanced to 150,25.
/ will cause the insert point to go to the lower left corner of first member in the current row. This is equal to the lower left corner of LOT.1. The insert point will now be 0,50
/VS25 will cause the insert point to advance to 0,75
Note: If we do not advance 25 units, lot 5 will end up overlaying part of lot 3.
LOT.4 will be placed at 0,75 and the insert point will be advanced by the width of an AK lot to 100,75
LOT.5 will be placed at 100,75 and the insert point will be advanced by the width of an AK lot to 200,75
LOT.6;T=BLOT.5+10;L=LLOT5+10
This member has a T=*exp*. We calculate where the bottom of lot 5 is and find that it is equal to the top of lot 5 which is 75 plus the height of lot.5 which is 100 and come up with 175 we then add 10 more units as directed in *exp* and come up with 185. So we set the insert point to 200,185.
The member also has an L= *exp*. We calculate where the left side of lot 5 is and see that it is 100. We then add 10 more per *exp* and come up with 110. So we set the insert point to 110,185.

/ will cause the insert point to go to the lower left corner of first member in the current member row. Remember that the current row was started right after the last slash member and the lower left corner of that member turns out to be the lower left corner of LOT.4. The insert point will be set to that which turns out to be 0,175

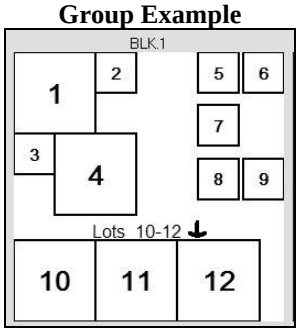
LOT.7 will be placed at 0,175 and the insert point will be advanced by 50 to 50,175

Comments: The positioning of lot 6 and 7 seems contrary to the row and column concept and it is in a pictorial sense. We purposely did this in the example to impress on you that a row is not necessarily lined up as you think. The beginning of member row to the program is the top left of the first member after a single / member

Note hat the portion **BLOT5+10;L=LLOT5+10** could be replaced by: **P5LOT5+10V+10H"**

APPENDIX B. IP WALK THROUGH - GROUPS

Group Example



BLK.1 LOT.1/AK.2./.3.4/AK
/HS75 LOT.5-7./.8-9;COLS=2;HG=5;VG=15
/ /VS50 TXT.1;H=30;W=300;A=Lots/0-12/P
/ LOT.10-12\K

Working group by group:

LOT.1/AK.2./.3.4/AK insert point defaults to 0,0. Working id by id.

1/AK adds lot 1 as type AK at 0,0 and advances to 100,0
2 add lot 2 at 100,0 and advances to 150,0
/ puts the insert point at left bottom corner of lot 1 of 0,100
3 add lot 3 at 0,100 and advances to 50,100
4 adds lot 4 and advances to 150,100
Detects end of group and advances to upper right hand corner of last member of row 1 of the group which is lot 2. IP becomes 150,0

/HS75 advances x portion 75 units to 225,0

LOT.5-7./.8-9;COLS=2;HG=5;VG=15/VS25 Working id by id.

5-7 equates to id's 5, 6 and 7
5 adds lot 5 at 225,0 and advances to 275,0. Detects that group has HG=5 descriptor and advances to 280,0
6 adds lot 6 at 275,0 and advances to 330,0 Detects that group has HG=5 descriptor and advances to 335,0. Detects that number of members has reached the COLS=2 value and emulates a / id. That positions to the lower left corner of the first member of the current row in the group. This is lot 5. Lower left corner is computed to be 225,50. Advances row no to 2. Detects that there is a VG=15 parameter and advances the descriptor to 225,65.
7 adds lot 7 at 225,65 and advances to 275,65.

/ Positions to the lower left corner of the first member of the current row (2) in the group. This is lot 7. Lower left corner is computed to be 225,115 Advances row no to 3. Detects that there is a VG=15 parameter and advances the descriptor to 225,130.

8-9 Adds lots 8 and 9 advancing descriptor by 50+5+50 to 330,130
Detects end of group and advances to upper right hand corner of last member of row 1 of the current group. This is lot 6. Becomes 325,0

/ Causes the insert point to go to the lower left corner of first member of the last row in the first group in the current group row. This turns out to be lot 3. The insert point will be set to that which turns out to be 0,150

/VS50 advances vertically to 0,150.

TXT.1;H=30;W=300;A=Lots\0-12/P adds a text member of dimensions 30 by 300 at 0,150. The text will be centered in the box defined. The /P parameter in the attributes designates that a down pointer should be added behind the text. The insert point advances to 300,150.
The end of group is detected and the inert point is advanced to the upper right corner of the last member of the first row of the group. Because there is only one member in the group, it turns out to be the TXT.1 member. The insert point is advanced to 300,150 (No change)

/ Causes the insert point to go to the lower left corner of first member of the last row in the first group in the current group row. This turns out to be lot TXT.1. The insert point will be set to that which turns out to be 0,180

LOT.10-12/AK Will add lots 10,11 and 12 starting at 0,180. The details are left for you to figure out. The y insert point will be 50 units below LOT.7

APPENDIX C. CEMETERY DESIGN LAYOUT ISSUES.

Scaling

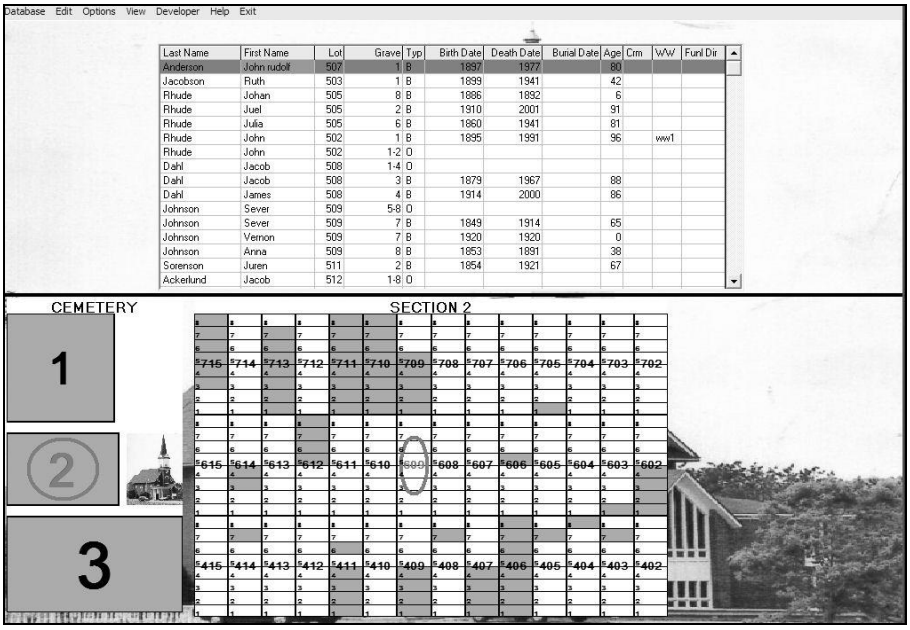
The program assumes the scaling to be on coordinate unit to one cemetery inch. A lot type with 4 foot by 10 foot graves would then be configured with its GW= descriptor set to 120 and it's GH= descriptor set to 48.

Section sizes

If you can reasonably fit the cemetery into one section, that is the way to go. This gives you an "at a glance" picture on the report form screen. If not, you will have to break it into sections. In that case, you will see only one section at any given time.

> ***Note: If you have more than one section there is an option on the report screen that allows you to see the whole cemetery at once. Just right click on the section picture and select the whole cemetery option. You may return to single section view using the same technique. You may also, give the diagram view a bigger portion of the screen thus enlarging it, by putting the mouse over the black bar in the middle of the screen, holding it down and dragging the black bar upwards.***

There will be an additional diagram at the left, depicting which section is on display and allowing you to select others. The picture below depicts that type of cemetery which has been broken into 3 sections.



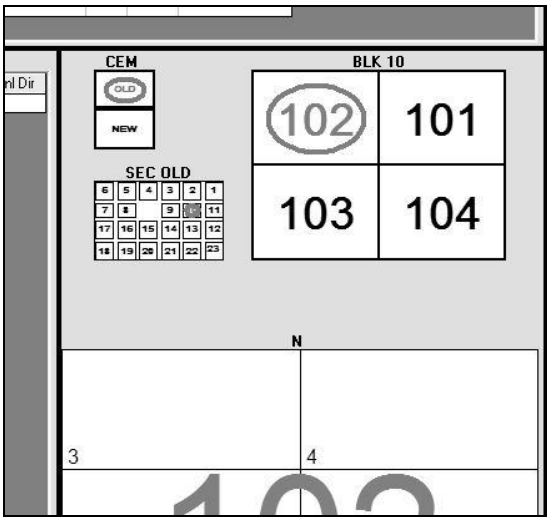
One of the limiting factors you must consider is how easy it will be to read the lot numbers and to be able to make out the graves in the diagram. The biggest reason for splitting this cemetery was the height. If it was made into one, the lot numbers would end up being too small to read. It would also make the diagram appear squashed into the area provided. To get the picture to fit height wise, the width would be very narrow.

A good guideline is to keep the section height to about the height of 30 graves. The lot ids should be readable as well.

A nice height to width ratio is about 3 to 4. If you make the width too wide, the height may be reduced to allow the diagram to fit in the space provided.

Block sizes

In determining how many blocks to fit into a section and how many lots to fit into a block, you must consider the lot locator portion of the lot form. An example is shown here.



The program will print the 3 levels of the cemetery which contain the desired lot in a way that maximizes the fit in to the layout section. As you can see, 6 wide or 6 high is probably a reasonable maximum for any level.

APPENDIX D MAP CONFIG QUICK REFERENCE.

Config Foprmat	
VBLvblid string Owner.id mbrs mbrs mbrs[vblid] mbrs Owner.id members	". " lines are comments . Only data up to the " . " is used. The rest is comments. Any data starting in col 1 is an owner. All mbrs between owners belong to that owner. At least one space required between members. If info starts on a new line, leading spaces are ignored unless a "&" at the line end. Members may not contain any spaces or commas. Any occurrence of [vblid] will be replaced by VBLvblid string

Owner member format	Groups	Mbrspcs (members and spacing)
MAP.1 CEM.1 Lot types CEM.1 group group etc SEC.id group group etc BLK.id group group etc	group=mbrspc.mbrspc.etc& ;control;control;etc Controls VG=int/int/int….. . vert pos HG=int/int/int….. horiz pos COLS=int . cols of mbrs in grps LTY=ltyid . Lot type of id if lots	Typ.id/ltype Typ.id/ltype.id/ltype.id/ltype Type.id-id/ltype / . next row of mbrs /HSint Horz spacing /VSint . vert spacing

Members	Loc siz direction formats	Pos/Size Vriable	txtinfo
CEM.id loc SEC.id loc BLK.id loc;A=ROT.ang LOT.id loc/ltyp;A=& /CTRx.y& /OTL.int.x.y TXT.id loc;siz;A=txtinfo LTY.id GR=rows& ;GC=columns& ;GW=gravewidth& ;GH=igraveheight& ;A=/gid/gid/…etc../ DIM.id loc;A=/direction NAM.id loc;siz	Loc L=topleftx;T=toplefty Topleftx and toplefty are expressions defining the top left cords of the member. Siz W=width;H=height Width and height are expressions defining the size of the member Expression can bec0mbination of +,-,*,/ Signed numbers Pos/size variable Signed integers ending with a V Signed integers ending with an H direction on DIM intV to draw arrow up or down intH to draw arrow fwd or bkwd	Fmt=xmbrid x= T=top of mbrid L=left of mbrid B=bot tomof mbrid R=right of mbrid W=width of mbrid H=height of mbrid Pk where k= crnrs of mbr 1---2—-3 4—-–—6 5—8----7 Lot attributes CTR defines position of owning sections id Default is center) OTL defines a vertex on the outline for the owning section. Int is the vertex no. (If none, the outline will be a box.	Fmt T.txt.fntsz.xp.yp/P/V Txt any text escape chars \s=" " \\="\" \/="/" \u="_" \c="." Fntsz is font size for text in cem units. Default is 75% of member height. Xp and yp are the offsets within the described box where text begins Xp can be L,C,R or intgr Yp can be T,C,B or intgr

Grapic mbrs format	bdyinfo	fillinfo	rangeinfo
LIN.id loc;siz ;A=& /B.bdyinfo& /U BOX.id loc;siz;A=& bdyinfo& fillinfo& txtinfo CIR.id loc;siz;A=& bdyinfo& fillinfo& rangeinfo& txtinfo PIK.id loc;siz	Fmt=/B.width.shade.style width is the boundary/line shade = See fillinfo style ='Line Style for bdy/line 0 Solid 1 Dash 2 Dot 3 Dash-Dot 4 Dash-Dot-Dot 5 xparent - 6 (Dflt) Inside Solid Note: If width is > 1, style settings 1 through 4 produce a solid line Note. /If /U option on LIN line it goes from lwrlft to upper right else it goes from upper left to lower right.	Fmt=/F.shade.style shade= shade of gray from 0-white to 6-black style=Fill styl for box/cir 0 Solid 1 (Dflt) xprnt 2 Hor Ln 3 Vrt Ln 4 Up Diagl 5 Dn Diagl 6 Cross 7 Diag Cross Txtinfo See the TXT attributes	Fmt=/R.begin,end Begin and end are the begin and end in degrees of a circle/ellipse/pie slice Precede them by a –sign to draw a radius (-0 doesn't work (use –359) Note: to fill a pie slice arc both radii must b preceded by a – sign.

Notes

Gid Grave id must be all numeric or all single digit alpha. ("*"means no grave there)
Graves will bepositioned in lower left cell, proceed upwars to top then start at bottom of next column.
BOX and CIR may have multiple "/T" sequences If usin ?P or /V they must precede the text.
NAM Is like TXT with defaults and the cem name is used for the text
BLK.1 LOT.1-28;COLS=7