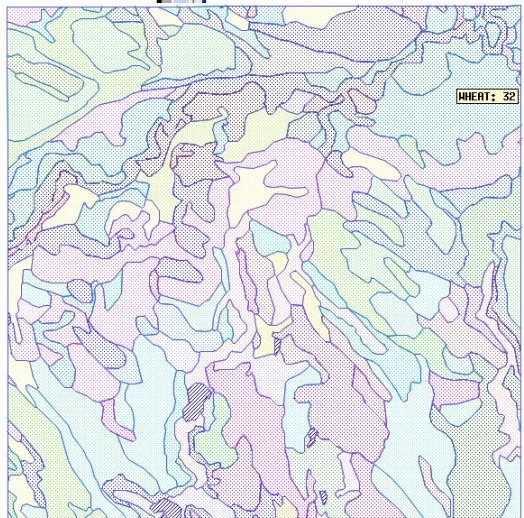
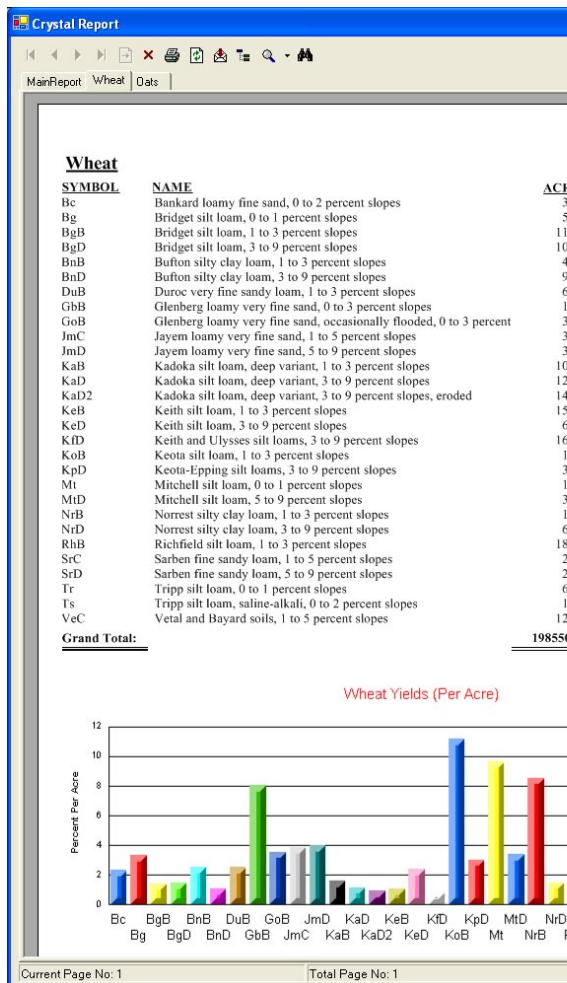
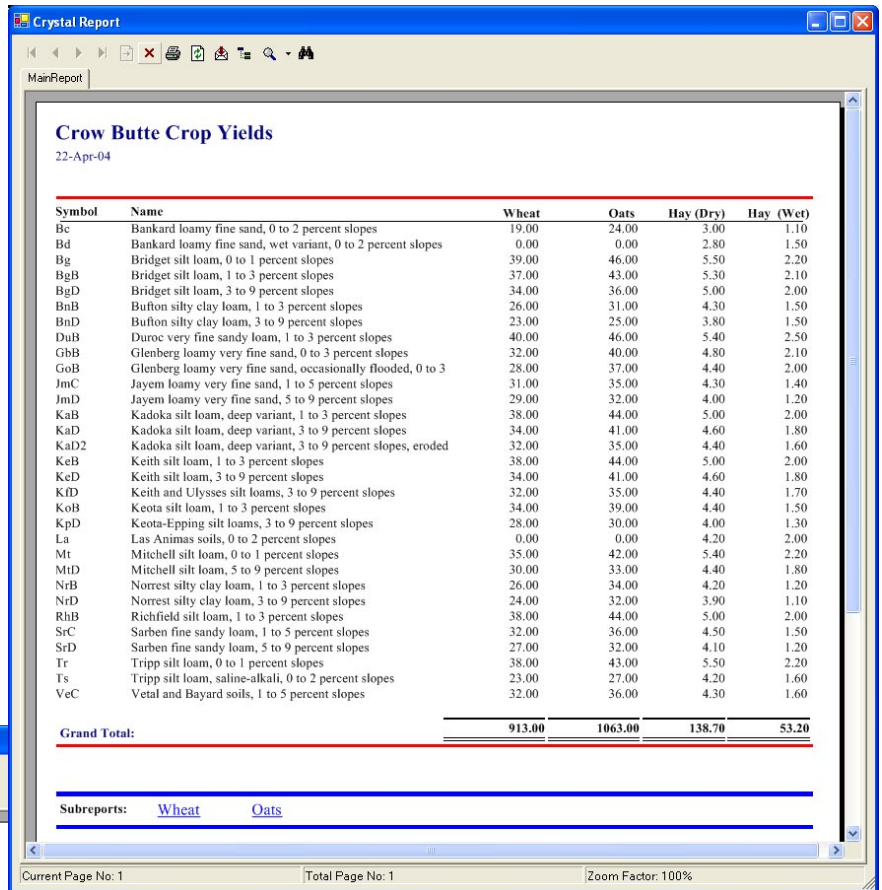


Create Crystal Reports with SML

SML can use Visual Basic to generate a Crystal Report and subreports for a vector database in the TNT products. This script (shown on the back) is data specific but readily adaptable for use with other vector objects. The script imports the form from Visual Basic, creates a class instance for the form, then prompts for the vector object to create the report for. The tables and fields desired for the report are specified in the script. Once the Visual Basic data table is initialized by the script, the specified fields from each record are read into the Visual Basic table, which is used to generate the Crystal Report. The script also makes use of prior settings made in Crystal Reports that specify which fields are available to the Crystal Reports engine and what subreports you want.



The subreports are listed at the bottom of the main report (shown above) until they have been opened and then they are available as tabbed panels (left). The setup of

the graph and which subreports to create are designated in Crystal Reports before running the SML script. If you ran the script in SML/Edit Script mode, you have to close the Crystal Report for the SML Editor functions to be active because the script is executing as long as the Crystal Report window is open. The subreport illustrated at the left makes it immediately clear which soil types are best for growing wheat.

Sample scripts have been prepared to illustrate how you might use the features of TNTmips' SML scripting language. The full script is printed below for your quick perusal. The sample script illustrated can be downloaded from the SML script exchange at www.microimages.com/freestuf/smlscripts.htm.

```
$import cbssoils_report.crystal_form
```

import the Visual
Basic Form

```
class crystal_form cf;
```

create an instance of
the form's class in SML

declare
variables

```
string symbol, name;  
numeric acres, wheat, oats, haydry, haywet;
```

```
vector v;  
GetInputVector(v);
```

get input
vector

```
numeric num_d_records = NumRecords(v.poly.DESRIPTN);  
numeric num_y_records = NumRecords(v.poly.YIELD);  
numeric current;
```

declare more
variables

```
cf.InitializeDataTable();
```

initialize the Datatable
in Visual Basic

add the records
from the vector to
Visual Basic

```
for current = 1 to num_d_records {  
    symbol = TableReadFieldStr(v.poly.DESRIPTN, "SYMBOL", current);  
    name = TableReadFieldStr(v.poly.DESRIPTN, "NAME", current);  
    acres = TableReadFieldNum(v.poly.DESRIPTN, "ACRES", current);  
    cf.AddDescriptnRecord(symbol, name, acres);  
}
```

loops through
records of
DESCRIPTN table

```
for current = 1 to num_y_records {  
    symbol = TableReadFieldStr(v.poly.YIELD, "SYMBOL", current);  
    wheat = TableReadFieldNum(v.poly.YIELD, "WHEAT", current);  
    oats = TableReadFieldNum(v.poly.YIELD, "OATS", current);  
    haydry = TableReadFieldNum(v.poly.YIELD, "HAYDRY", current);  
    haywet = TableReadFieldNum(v.poly.YIELD, "HAYWET", current);  
    cf.AddYieldRecord(symbol, wheat, oats, haydry, haywet);  
}
```

loops through
records of
YIELD table

```
cf.InitializeReport();  
cf.ShowForm();
```

create and show the
report after all the
records are added