



@ddsum

# TAS Premier™ 7i TUTORIAL

*TAS Premier 7i*

### INTRODUCTION

Addsum TAS Premier 7i is designed to be a complete GUI application development tool. It provides application building tools for both new and experienced programmers.

These powerful utilities include:

- Data Dictionary Manager
- Database Maintenance Programs
- Form Editor
- Report Editor
- Source Code Editor
- Runtime Compiler

Addsum TAS Premier 7i includes over 150 commands, each with many options, and over 230 functions that can be used in expressions. Many of the commands can be compared to macro functions found in other high level languages. Add to this the ability to create User Defined Commands (UDCs) and User Defined Functions (UDFs), and you have a very powerful development tool.

One of the reasons Addsum TAS Premier 7i is so flexible and powerful is that it is built around a Data Dictionary. The Data Dictionary is a special Addsum TAS Premier 7i data file that is used to keep track of every specification of every field in every record and file you create. This means that once you define a field, you do not have to define it again. Each time you want to use that field in a program, Addsum TAS Premier 7i automatically looks up the specifications in the Data Dictionary. Storing the specifications in this way greatly increases the ease and speed with which you can create multi-file applications. By using similar fields with the same characteristics in different files, you can retrieve related records, such as finding all invoice records based on a given customer code.

Addsum TAS Premier 7i uses Btrieve (now Pervasive), preferably, to perform all data file I/O, record and file locking in a multiuser system. With Addsum TAS Premier 7i you can typically find any record in a data file in one second or less. With smaller databases, finding a record can be almost instantaneous. Saving or updating records is equally fast. In addition, the speed is consistent regardless of the file size. It takes virtually the same amount of time to save a record in a file with 100,000 records as it does in a file with 1,000 records.

Addsum TAS Premier 7i allows up to 24 different sorts (keys) to be maintained on-line for each datafile, so you never have to re-sort a datafile just to find a record.

## GETTING READY TO RUN THE TUTORIAL

Since this is your first time using Addsum TAS Premier 7i, let's start at the very beginning.

1. On your desktop there should be three new icons, or a new program group that has been created in your Start->Programs list. This tutorial assumes that you have the icons on your desktop. Click on the TP7 Setup icon. The following form will appear:

Set Configuration [c:\adv7\]

Screen Editor | TAS 5.1 | TAS 5.1 Colors

General | User Options | Email Settings | Source Code Editor

Default Path  
C:\Taspro7\

Data Dictionary Path  
C:\Taspro7\

Initial Program Name: ☒ Initial Program is Main Menu  
adwbkmenu.rwn

Startup Company  
B

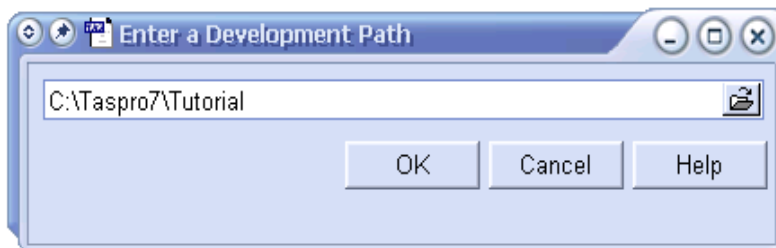
☒ MultiUser (Record/File locking) ☐ Use Btrieve Memos ☐ TP Active for CB  
☐ Use CodeBase ☐ Use File Manager

Set Dev Path Setup Dev Path Icons OK Apply Cancel

The image above reflects a system that is using the "Btrieve" (Zen/Pervasive/PSQL) engine which is not provided with the default installation and that is used for Advanced Accounting. To instead use CodeBase, change the Startup Company to C, uncheck "Initial Program is Main Menu" and blank out the initial program name, and click on the "Use CodeBase" checkbox.

The display above assumes that you have installed Addsum TAS Premier 7i in the C:\TASPro7\ (or where [C:\TASPro7](#) appears substitute [C:\TAS7i](#) subdirectory, which is the default for the Premier version). Your data dictionary path may be different. Make sure Startup Company is set to B, MultiUser is checked, and Use Codebase is unchecked. Note: the above example shows an initial program name and that the initial program is a main menu program – that will not be the case for purposes of the tutorial and those should be left blank.

2. Click on the Set Dev Path button. A form allowing you to enter a new development path will appear.

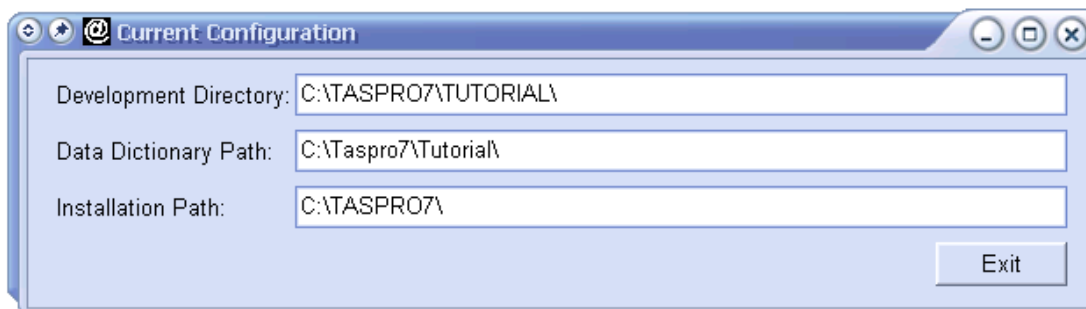


3. Make sure your development path looks like the example above: C:\TASPro7\tutorial. The program will automatically add the trailing slash ("") at the end. Click OK. Then click Yes.
4. This directory was set up for you during installation. You will be back at the main setup screen. Click the OK button again and the program will exit. You have now setup the development directory and the proper data dictionary location.

**NOTE:** You can still access programs in other subdirectories. By defining a development directory it becomes your default directory until you change it using this same process.



5. Click on the TAS 7i icon and the main development screen above should appear. To double check your setup click on the Program menu choice at the top of the screen. Then click on the Current Configuration menu item. The screen below should be displayed. Make sure the values are correct. If they aren't go back to step 1 above and start over again, because you won't be allowed to make any changes to these values here; this is just to display the current information. Click on the Exit button to close the configuration screen and exit the program.



You are now ready to begin the tutorial. You can exit the tutorial at any time and restart. Until you change the Development path using the setup program each time you start Addsum TAS Premier 7i you will start in this subdirectory.

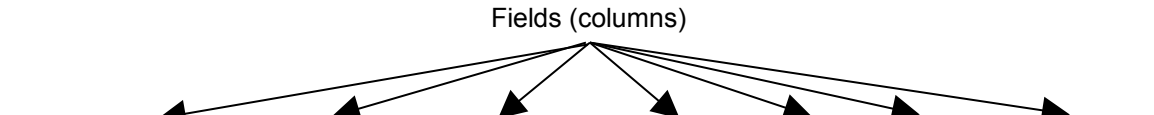
## A BRIEF INTRODUCTION TO DATABASES

Before you begin the tutorial, spend a moment learning about the structure of files in a database and the tools you might use to manage the information contained in those files.

### Database Fields, Records and Files

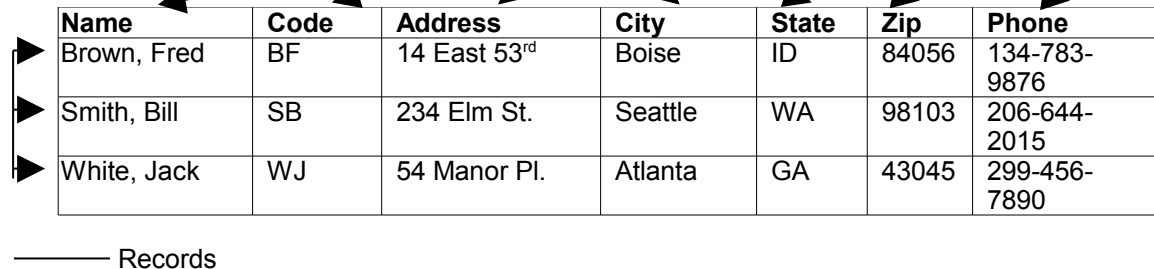
One way to visualize a database is to think of it as a table of rows and columns. In fact, this is the classic manner in which databases are described in computer literature. In the example shown below, each horizontal row of the table is considered a record and contains all the data about that person. Each vertical column is considered a field.

Fields (columns)



Name	Code	Address	City	State	Zip	Phone
Brown, Fred	BF	14 East 53 <sup>rd</sup>	Boise	ID	84056	134-783-9876
Smith, Bill	SB	234 Elm St.	Seattle	WA	98103	206-644-2015
White, Jack	WJ	54 Manor Pl.	Atlanta	GA	43045	299-456-7890

Records



Let's say you wanted to find the address of Bill Smith. First you would go vertically down the Name row until you found the Smith record. Then you would go horizontally across the columns until you found the Address field.

One of the main strengths of a computerized database like Addsum TAS Premier 7i is that it can search thousands of records in the time it would originally take to search just a few.

In Addsum TAS Premier, the term "database" describes a related group of information in an application. "Data file" refers to a specific file in the database. In a database with one file, the data file is the database.

### File Managers, DBMSs and ADEs

There are three general categories of database programs. Each is designed to do different things.

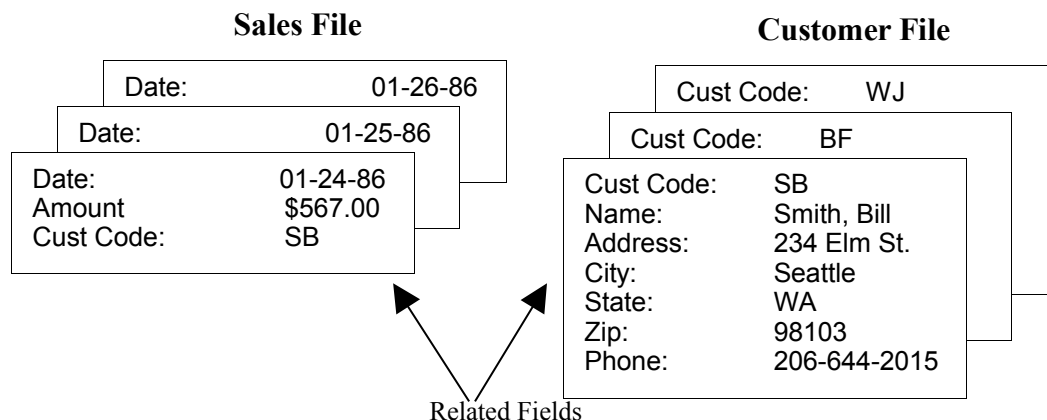
The simplest form of a database program is called the **File Manager**. As the name implies, this type of program is designed to manage a single file at a time. The advantage of a File Manager is that it is fairly easy to use. The disadvantage is it doesn't do very much. File managers allow you to add, delete, and search for records in only one file at a time. Consequently, they aren't typically used for serious business applications.

You could use a file manager to computerize the index card file in the previous example, since that database contained only one file.

But let's say you then wanted to add a second file such as a Sales Journal to your database. At that point a simple File Manager would no longer suffice. You would need a **DBMS** (Data Base Management System) to manage two or more files. A DBMS is more complicated than a File Manager, and so can be more difficult to master. However, a DBMS gives you several advantages over a file manager as you can have multiple files active at the same time.

Even more important, a DBMS allows you to relate files to one another. This is done by selecting certain fields that appear in more than one file. For example, you might have a Customer File which

contained a field called **CUSTCODE** (short for customer code). If a related field (perhaps named **SO.CUST.CODE**) also appeared in our Sales File, it would be possible to relate the two files. (See the figure below.)



Relating files can be very important in creating real world database applications. Let's say you want to send a "valued customer" letter to each person in your Customer File who has made a single purchase over \$500. To do this you will need to search both the files shown above. First you will search the Sales File for records in which the Amount field is over \$500. When you find one, you will need to discover to whom the customer code belongs. You accomplish this by *relating* the record via the **SO.CUST.CODE** field to a record in the Customer file with a matching **CUSTCODE** field.

The next step up from the DBMS is the **ADE** or Application Development Environment. ADE's are systems used to create other programs (i.e. applications). Typically, ADE's include a powerful DBMS, plus utilities, to create custom menus, data input screens and sophisticated reports.

So, Addsum TAS Premier is a very useful program because it can be used as a File Manager, a DBMS or as an ADE. It combines a powerful DBMS with a programming language and an easy-to-use "interface."

Now you're ready to start working through the tutorial. You'll learn how to:

1. Create, link, and maintain data files.
2. Add, change, delete, and search for records.
3. Build and enhance a simple application with a customized data input screen and error-checking.
4. Create reports to summarize and print your data, and develop a menu for using your application.
5. Restructure data files and modify an existing application program.

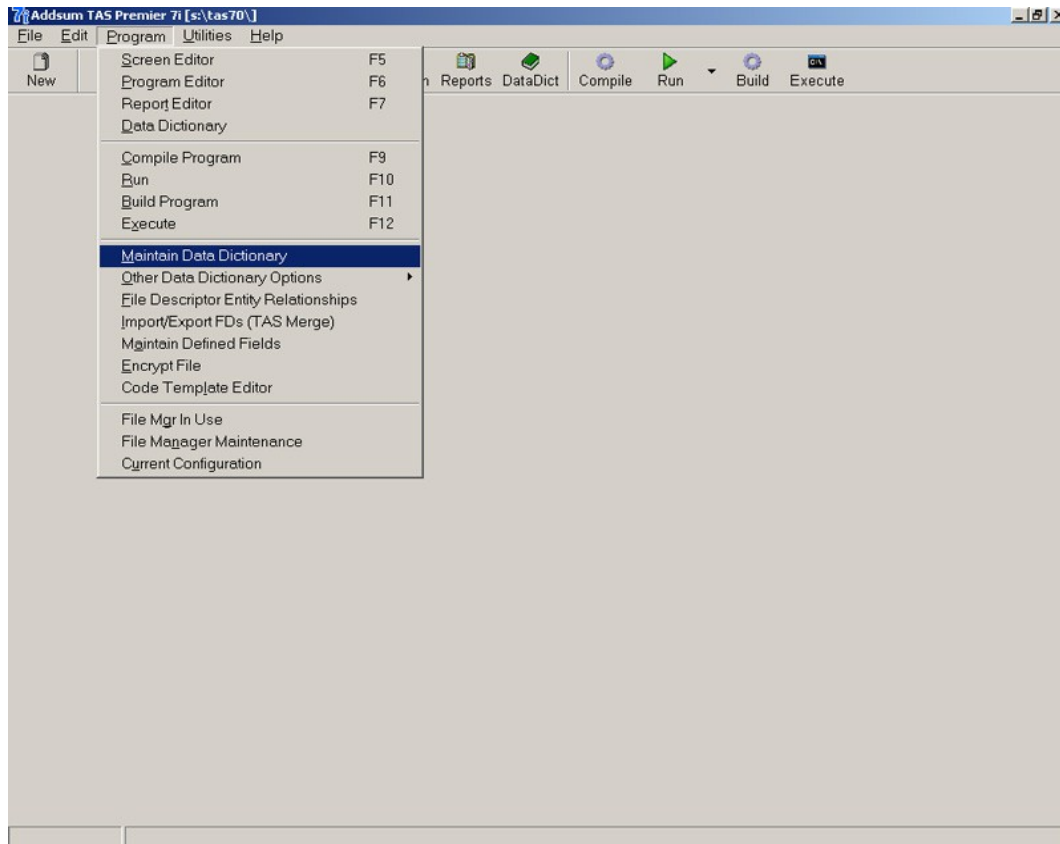
In short, you'll see how easy it is to use TAS Premier.

If you still have Addsum TAS Premier loaded then you are ready to start. If not, please click on the TAS Premier icon or choose the program via Start->Programs->Addsum TAS Premier 7i.

**NOTE:** In the following instructions, whenever you see "enter," as in "enter the name" or "enter CUSTCODE" it simply means type in the relevant data and then press the ENTER key.

## PART 1 - CREATING A DATABASE

In this section you will create a Customer data file using the **Maintain Data Dictionary** option from the Program menu at the top of the screen.



All Addsum TAS Premier files must be 'defined' in the **Data Dictionary** before they can be accessed by any program. This will tell the program how many fields there are, the type and size of each field, and what fields, or group of fields, are to be used as keys (or indices).

### Using the Maintain Dictionary Option

In this example you will create a database to manage the following information about your customers:

- |            |                 |
|------------|-----------------|
| 1. Code    | 6. State        |
| 2. Name    | 7. Zip          |
| 3. Company | 8. Area Code    |
| 4. Address | 9. Phone Number |
| 5. City    |                 |

These nine data fields will form the foundation of your database. After you are finished, you can then add, change, delete or search for records, and use these fields in programs or reports you create in the future.

## Addsum TAS Premier 7i Tutorial

When you choose Maintain Data Dictionary the following screen will appear:

Long Field Name (unique)	Short Field Name	Type	Size	Dec Chrs	Array Elements	Upper Case	Description
						<input type="checkbox"/>	

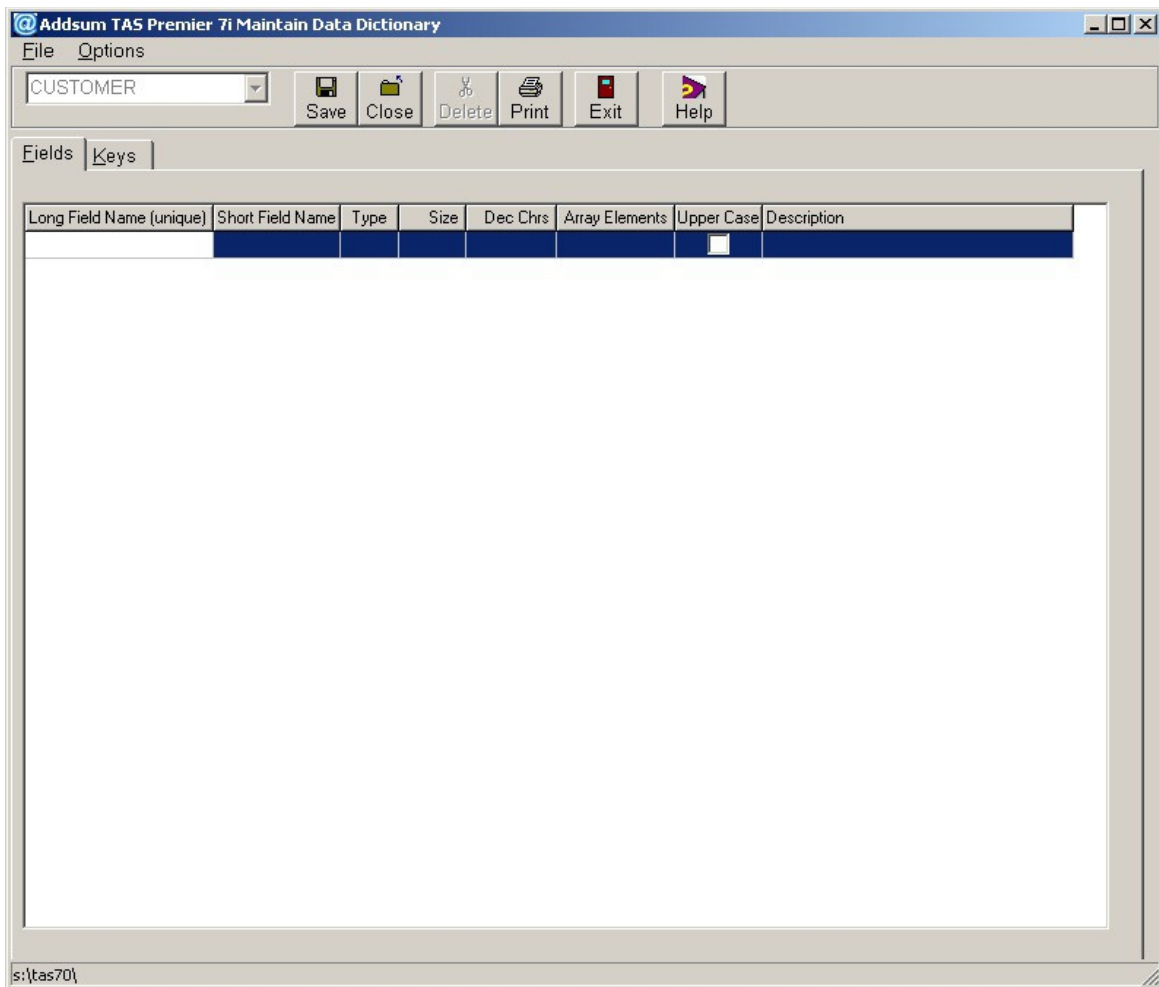
Type **CUSTOMER** in the FD NAME field, then press ENTER.

This is the name of the structure for the file you are creating. Through the use of the Data Dictionary you can have multiple files that use the same structure, also known as the **File Descriptor** (FD). In most cases, however, you will always have a file that has the same name as the FD. This isn't required but it makes it easier when you are trying to remember the FD and/or file name; both will have the same name.

After clicking OK, Press ENTER again, or simply click the space under Long Field Name, and the screen should now look like this:



## Addsum TAS Premier 7i Tutorial



Type **CUSTCODE** in the Long Field Name field. Press ENTER to move to the next column/field.

It is of little consequence what you use for the Short Field Name field, as long as it is unique to the data dictionary. The internal code base structure needs that identifier, but the programmer will never look at it; the programmer will only see the Long Field Name, which should also be unique. So you can type in whatever name you want to use, as long as you don't use it again.

The next entry is the field type. Click on the drop down button for the following choices: **A** - Alphanumeric, **N** - Numeric, **D** - Date, **T** - Time, **L** - Logical, **B** - Byte, **I** - Integer, **R** - Record, **M** - Memo. Since this field may contain any printable character including numbers, letters, and symbols, enter A for alphanumeric.

Next is the field size, which is determined by the largest code you expect to put in the field. In this case it should be **10**, so that the field will allow ten characters.

Next is the number of decimal characters, but because this field is type A (alphanumeric), it is bypassed automatically.

The next entry is the number of array elements for this field. For this one it is 0 because you're only keeping one customer code for the record.

For future reference, an array allows you to keep multiple values of the same type, using the same field name, changing just the element number. This is useful when keeping track of certain types of information, like sales by month. You might have an array called MONTHLY\_SALES with 12 elements. Element 1 (corresponding to the sales for month number 1) would be MONTHLY\_SALES[1], element 2 - MONTHLY\_SALES[2], etc. to element 12, MONTHLY\_SALES[12]. Since you can use any legal Addsum TAS Premier expression as the element specifier (the value within the square brackets), this becomes a very flexible way of accessing field values. For example, using the same field as above the amount of a sales order might be saved in the monthly sales field as follows:

```
MONTHLY_SALES[month(date())] = TOTAL_ORDER_AMT
```

This would save the value of TOTAL\_ORDER\_AMT into the element of MONTHLY\_SALES that corresponds to the month of the current date. (date() returns the current system date and month() returns the month number (1-12) of a date value. In this case Addsum TAS Premier first gets the current system date and then the month number of that date. This is then used for determining the element number in MONTHLY\_SALES. If the date is August 10, 1992 then the element number is 8).

The next entry is Upper Case. Since this is an alphanumeric field you can force the entry to all upper case characters only. Since this is a code that you will be using to search for customers, you will want to make sure that all characters are entered in upper case. The codes abc and ABC are not the same to a computer. You might not check all the possibilities when searching for the customer and the code 'abc' would come after any ZZZ codes, so you might not even see it in a list. To force the field to upper case you can either click on the check box or press the space bar; both will put a check mark in the box.

For Description, you can enter up to 40 characters. This helps you document the fields within the FD.

Once you are finished with the Description, pressing ENTER will move you to the next line in the grid.

If you made an error or if you want to change something, remember that you can simply go back to that line by clicking the appropriate row and column. You can also press the ESC key to exit the editing mode and then the UP and DOWN ARROW keys will allow you to move around the grid. You can make any changes to an FD as desired since you will be allowed to decide whether to keep any changes made at the end before the program updates the data dictionary.

The program automatically moves to the next line. At this point you're ready to enter the next field. Use the field specifications in the following table for the rest of the fields in the Customer data file. (The Array sizes are 0.)

Field Name	Type	Size	Dec Chrs	UpCase
CUSTCODE	A	3	0	Y (Entered)
CUSTNAME	A	25	0	N
CUSTCOMP	A	25	0	N
CUSTADDR	A	25	0	N
CUSTCITY	A	25	0	N
CUSTSTATE	A	2	0	Y
CUSTZIP	A	10	0	Y
CUSTAREA	A	3	0	N
CUSTPHONE	A	8	0	N

It is very easy to manipulate grid lines. If you want to insert a new line, press the ESC key to exit editing mode. Then press the INSERT key. A blank row will appear immediately above the current row. To delete a row again press the ESC key to exit edit mode and then the DELETE key. To return to edit mode all you have to do is press ENTER when the cursor is in the appropriate column.

### Entering Keys for CUSTOMER Database

After you have entered the rest of the fields you are now ready to specify the key fields. Key fields are important because they will help you search and sort through your files. Click on the Keys tab, and this screen will appear:

The screenshot shows the 'Addsum TAS Premier 7i Maintain Data Dictionary' window with the 'Keys' tab selected. The 'Key Name' field is empty. The 'Order' dropdown is set to 'Ascending'. The 'Allow duplicates' checkbox is checked, while 'Modifiable' and 'Ignore Case' are unchecked. The 'Segment Field Name' field is empty. The 'Clear segment field num' field contains the number '1'. The 'More help' button is located at the bottom right. The status bar at the bottom left shows 's:\tas70\'. The title bar at the top reads 'Addsum TAS Premier 7i Maintain Data Dictionary'.

Click on the New button to add a new key. This will make the Order default to Ascending, Modifiable and Duplicates will both be checked, and Ignore Case will be unchecked.

The first step is to enter the key name. For this first one enter **CUSTCODE**. The key name does not necessarily have to be the same as a field name; however, if there is only one field as a part of the key you should use the field name. You can click on the drop down button to get a listing of fields for this FD.

The next entry concerns whether the key will be sorted in Ascending (increasing) or Descending (decreasing) order. Normally this will be in Ascending. Press the ENTER key to accept the default.

The Duplicates box option can restrict whether or not duplicate keys can be saved. Since this is the customer code and you don't want to allow duplicates, press the space bar or click on the check box to set this option to no duplicates allowed.

If the Modifiable box is unchecked, then the user will not be able to make changes to the index field once it has been saved. If you don't care whether or not a key value changes after the initial entry, or you want it to allow changes, the Modifiable option should be checked. In this case we want to allow changes, so leave this box checked. Press the ENTER key to move to the next option.

If you check the Ignore Case box then when the record is saved the program will set the index field(s) into upper case before the index itself is updated. This has no effect on the actual field, just the index. However, by setting the index into all upper case characters the records will sort as though the user entered all upper case characters. This will make searching for the record by this index much easier. The user won't have to remember which characters are upper and which are lower case. We use this feature for the CUSTNAME index.

Any TAS Premier file may have up to 24 different key segments. That may be 24 keys each with only one segment or one key with 24 segments. To search a file using a different method (or field) requires that you have different keys. In this case you will end up with four different keys, each with only one segment. Determining how many keys you should have and how many segments will make up each key is your decision. However, we recommend that you use as few keys as possible and keep the size of each key as small as possible for faster file access.

If you have more than one segment in a key it sorts in the following manner: The first field is used as the primary sort and then the file is sorted further by the following segments. For example, you might have a key with two segments; the first is the customer code and the second is an invoice date. If you were to search within a file by that key you would see all records for a particular customer code in date order. Then when the search reached the end of that customer it would start with the next customer in order again, beginning with the first sale date for that particular customer. If you wanted the file to be accessed in date/customer code order as well, then you would create a second key where the first segment would be the sale date and the next would be the customer code. Then when you accessed the file using that key you would see all sales for that date in customer order, then the next date, etc.

Enter the Segment Field Name **CUSTCODE**. Since there is only one segment in this key, simply click on the Save button. The key will be added to the key list grid (on the left side of the screen). The information you just entered will still show up since that key will be chosen.

Now enter the other three keys using the data in the following table. The entries are as follows (all are Ascending, and the key and segment name are the same):

Key/Segment Name	Duplicates	Modifiable	Ignore Case
CUSTCODE	N	Y	N (already entered)
CUSTNAME	Y	Y	Y
CUSTSTATE	Y	Y	N
CUSTZIP	Y	Y	N

If you want to change a key previously entered, click on the appropriate line in the key list grid. The appropriate information will appear for that key. You can then make any changes you wish, including moving segments around, inserting and deleting segments, and changing the key name. (Be careful about changing the key name after a file is created and part of existing programs since the key name may have been used and will cause problems the next time you compile those programs).

To rearrange the segments of a key name click on the blocks to the left of the segment names, labeled 1, 2, 3 and so on, and while holding the mouse button down, move the segments up or down into the desired order. When you release the mouse button the segment will remain in its new location.

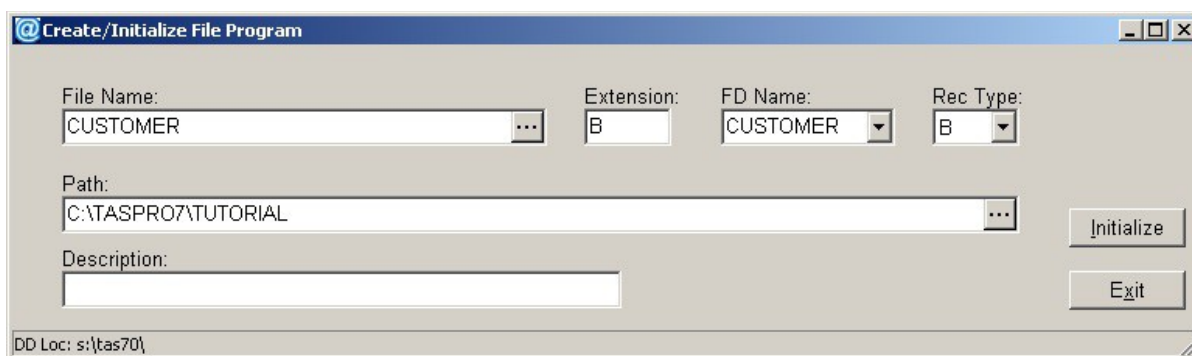
## Addsum TAS Premier 7i Tutorial

If you change or add a key to a FD you must either Initialize the file (if there are no records already entered, or at least none you want to keep), Reindex the file, or Restructure the file. If you have not changed the FD structure you need not Restructure the file; Reindex is the preferable method in this case.

\*Remember that the Save and Cancel buttons to the right will update, but they won't actually save anything. For that, you need to click the Save button at the top. The same goes for the Delete buttons: the one on the left won't permanently delete the entered key(s), but the one at the top will.

### Initializing the New File

Since this is a new FD, the program will automatically ask if you want to create a new file. Press ENTER or click Yes.



**NOTE:** It only automatically asks if you want to create a new file when you have just created a new file descriptor. Otherwise, you must go to Options and then Initialize.

The screen above will appear. The fields are all self-explanatory except for the Rec Type, which simply tells Addsum TAS Premier what type of field this is. Btrieve files are type B, while CodeBase files are type C. The initialization program defaults to type B. Change both the Extension and Rec Type to C for CodeBase files.

**NOTE:** In Btrieve type files the File Name and Extension are put together to get the entire file. So, if the extension was B and the Rec Type was also B the above would be CUSTOMER.B. However, CodeBase files use the extension DBF at all times. The Extension value in this case is more used for which company files are being accessed since you have the capability of keeping separate sets of files for each company. In TAS Premier file names (not including the extension) are limited to 32 characters. The path is limited to 128 characters but can be any legal Windows path.

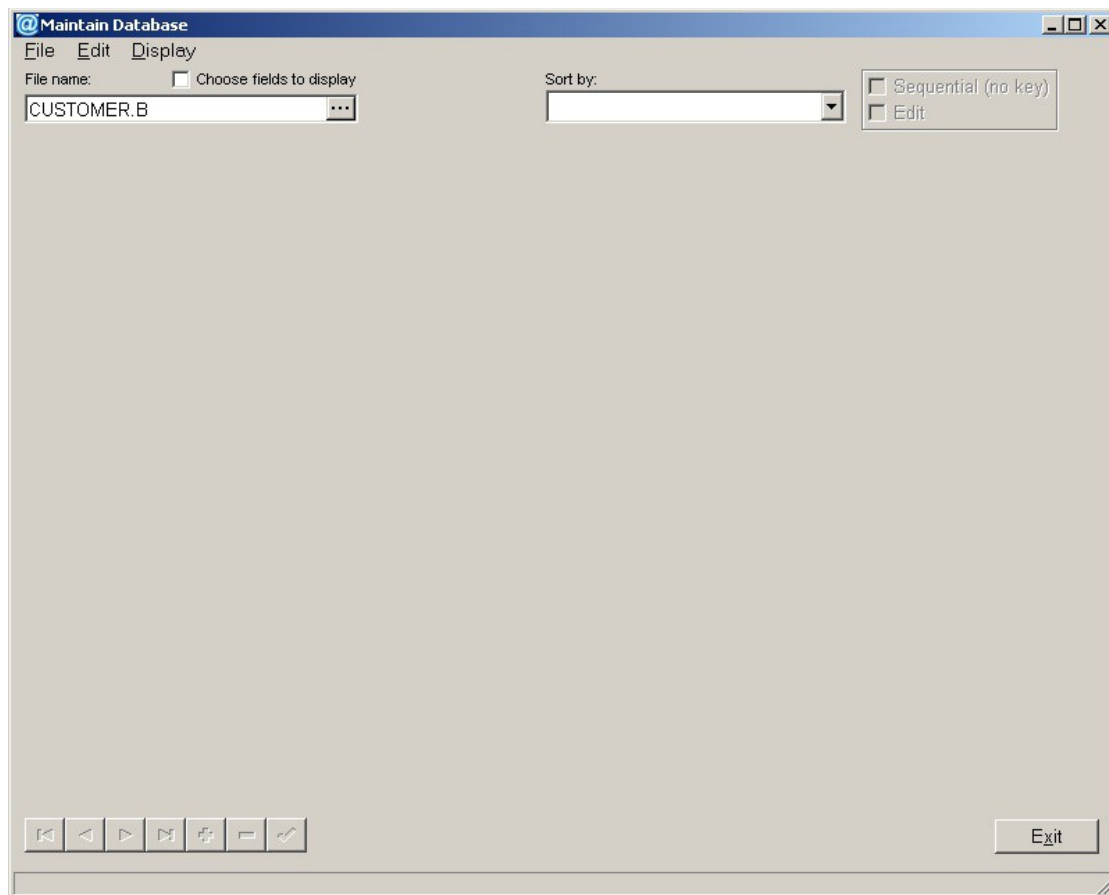
To continue with the process click on the Initialize button. This must be done before any data can be entered. After the initializing process is complete the fields above will be cleared. Press the Exit button to return to the Data Dictionary Maintenance program. Press the Exit button in the Data Dictionary Maintenance program (also referred to as MDD).

This completes Part 1 of the tutorial.

## PART 2 - MAINTAINING A DATABASE

Now that you have created a CUSTOMER database with Addsum TAS Premier, you can start using it right away. You can add, edit, and delete records and generally keep things up to date. This type of activity is known as “maintaining a database.”

In this section you will edit the customer data file using the **Maintain Database** option from the Utilities menu. The following screen will appear:

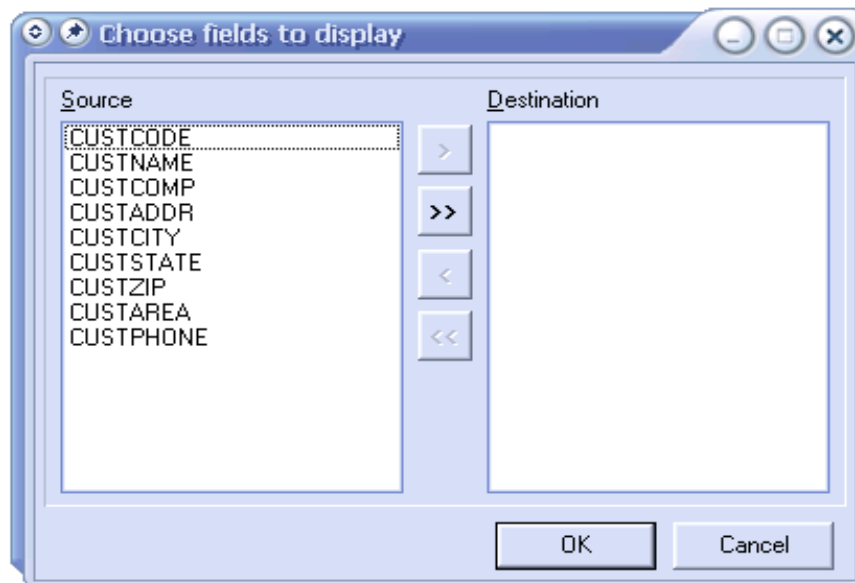


The first entry you will make is the file name. Click on the look up button and all of the files in your data dictionary will be listed. In this case you want to choose CUSTOMER.B. If you using CodeBase files, enter CUSTOMER.C (or CUSTOMER.DBF).

In the future, if you have many files, you can enter one or more characters before you click on the look up button to find the desired file faster. You can also enter the entire file name. Don't forget the extension.

### Choosing Fields to be Displayed

Once you have chosen the file, click on the Open File button. The following screen will appear:



The fields in the Source list are all those available. You can move them over to the Destination list either by moving them all over by clicking on the double right arrow (>>), or clicking on an individual field (or fields by holding down the SHIFT or CTRL key while clicking on the field(s)) and then clicking on the single right arrow (>) to move it (or them) over. You can move them back by doing the same in the Destination list. You can also rearrange the fields in the Destination list by using a standard drag-and-drop process. The order the fields appear in the Destination list will determine how they appear in the editing grid. Each field is a separate column in the grid.

In this case you're going to choose all the fields, so click on the double right arrow and then click on the OK button to pull up the following screen:

The chosen fields make up the columns in the grid, but there are no records in this file so there are no rows in the grid. The keys available for this file are in the "Sort by" drop down box to the right (in this case, just CUSTCODE).

If there were records in this file you could search for them by entering characters in the Fast search field on the left. As you enter each character the program will find the record that matches most closely. If you delete characters the search routine will move backwards.

To the right of the Key List is the Edit check box. If you check this option the grid will go from search mode to edit mode and will allow you to add, change or delete records. Since you want to add new records, go ahead and check the box and you will now be able to do so.

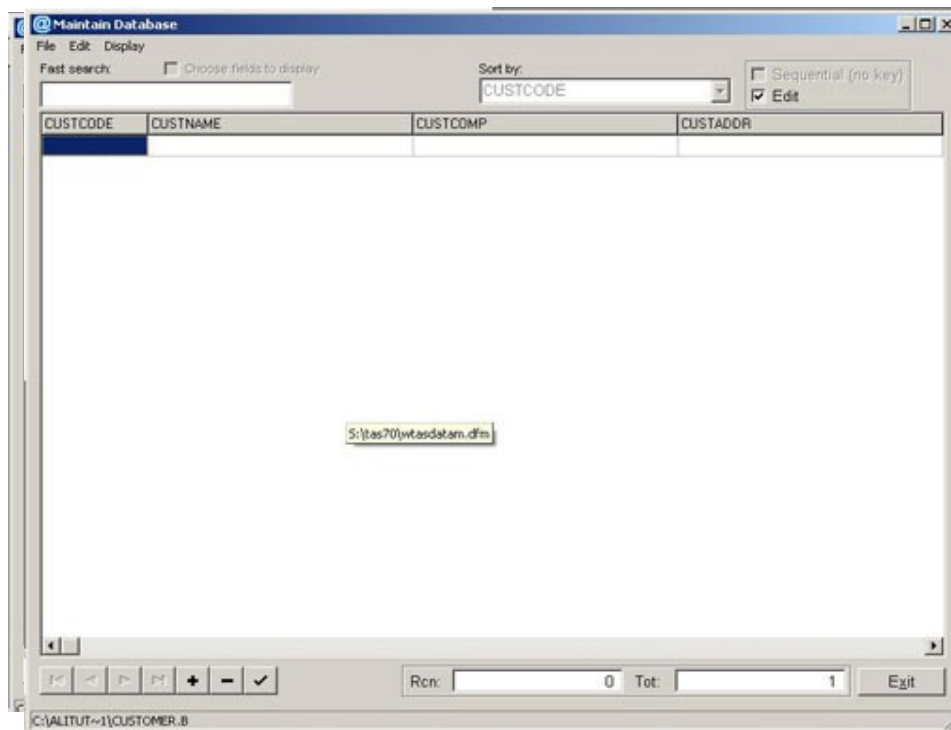
At the bottom of the screen is a standard navigator bar. By clicking on the appropriate navigation buttons on the left side you can move to different rows within the grid, save changes made to a record, delete a record or add a new record (row) to the end of the grid.

**NOTE:** If you modify a record and save it (either by moving to another row or clicking on the Save Record button) you will not be able to move beyond the rows you have already visited. The reason for this is that the records you have already pulled up may now be in a different order. Since you might access records in the new order, the program controls which records can be displayed. To eliminate the restrictions click on the First button (or press the Home key on the keyboard). The records will re-

## Addsum TAS Premier 7i Tutorial

organize as appropriate and you can return to edit mode if you wish.

### Add a Record to a File



After you enter the file name you should see the screen above, with the the name of the open file in the status bar at the bottom.

Make sure the 'edit' box is still checked, then press the ENTER key to start editing the CUSTCODE field. Enter **001**. After you press ENTER the cursor will automatically move to the CUSTNAME field, just like it did when you entered the original fields before. For the customer name enter **Last, First M.**, then **Customer Company** (CUSTCOMP), **12345 Generic Street** (CUSTADDR), **Anytown** (CUSTCITY), **UT** (CUSTSTATE), **55555** (CUSTZIP), **555** (CUSTAREA) and **555-1234** (CUSTPHONE). After you have entered the CUSTPHONE value press the ENTER key. The cursor will move to the next row, automatically saving the one you just entered. Clicking on the Save button (the plus sign) in the Navigation bar also saves any information.

**NOTE:** Until you move to a different row either by adding a new row at the end of the list as you did above, by moving to a row previously entered or by clicking on the Save button, the data you enter will NOT be saved to the file. This applies to both new records or changes to existing records.

Enter three or four more customers using your own data.

### Finding Records in a File

Now that you have created a **CUSTOMER** database and saved some records to it, you can begin experimenting with some of the features of Addsum TAS Premier.

As mentioned earlier in this manual, Addsum TAS Premier can typically find any one record in one second or less. TAS Premier does this by searching for records according to key fields. Recall that when you designed the **CUSTOMER** database, you designated four fields to be keys. They were:



CUSTCODE  
CUSTNAME  
CUSTSTATE  
CUSTZIP

This means you can search for records in the **CUSTOMER** database by customer code, customer name, state or zip code. You could also start at the beginning or the end of the file and sift through the records one by one, but of course this would be more time-consuming than searching with a key. In this program the keys are listed in the "Sort by" drop down box. Since the first key is the default, CUSTCODE is listed as the active key when the file is opened. To search for records in the file you have to be out of Editing mode, so uncheck the box.

If you want to search on a different index all you have to do is click on the "Sort by" drop down box and choose one of the keys in the list. If the first field in the key is of a different type than you are currently searching (e.g. numeric versus alphanumeric), the Fast search field type will change accordingly, the first column will change to the new key field and you will start over at the beginning of the file using the new key.

When you need to sort through the records in the data file to find a particular one, you can use the up and down arrow keys to go through them one at a time. The Home key will move you back to the very first record, and End to the very last. Note that you do not have to be in editing mode to use these keys, and you can always use the navigation buttons at the bottom, as well.

### Change (Edit) a Record

What happens when one of your customers moves? How do you change their address? It's really very easy. Make sure the Edit box is checked. Then you can either simply double-click on the customer field you wish to change, or click it once and press Enter. Clicking on the Save button or moving to a different row will save the data changes.

### Delete a Record from a File

Make sure the Edit box is checked, find the record you wish to delete, highlight any part of it (whether the code, the company name, the zip, etc, it doesn't matter), then either press the Delete key or click on the Delete button on the navigator (the minus sign).

When you are finished click on the Exit button. This will close the WTADATAM program and completes Part 2 of the tutorial.

## PART 3 - CREATING A NEW SALES PROGRAM

In this section you will learn how to use the Screen editor and the Program editor to build an application called **SALES**, which is really a simple sales journal. You can use it to record daily sales activity including Order Number, Customer Code, Sales Amount and other pertinent information. In later parts of the tutorial, we'll learn how to add a menu and report to this application.

TAS Premier allows you to create complete applications, not just simple databases. Let's take a moment to talk about what an application is. In its most basic form an application is a program combined with at least one customized data input screen and one database file.

The real significance of this is the customized data input screen. In TAS Premier the screen editor is a true WYSIWYG (What You See Is What You Get) application; whatever you see in the design environment is what it will look like when you run the program.

There are generally two parts to any TAS Premier application: a screen or form and a source file. The file extension for screens is .DFM and .SRC for source files. You can have an application with multiple screens, one or more report formats and even one that has no screens at all. However, in this case, we're going to have a single form and a single source file. When creating forms, little work need be done in the source file because you can place most of what you want the user to see right into the field objects themselves. Once you have the screen and source file, you can then compile the program. This creates a RWN file. When the program has run, the runtime uses the RWN and the DFM.

**NOTE:** The forms are not part of the compiled program for a very important reason. Once you have compiled the source file you can continue making changes to the forms without having to recompile. As long as you don't add a field to the screen that hasn't been referred to in the program, you never have to recompile. So, you can change colors, text, location, remove fields, etc.

### Design your Database on Paper First

The first and most important step in building a database application is to design it on paper first. Doing this will most likely save you some time and frustration later on.

A good place to start is by listing the types of data that will make up your application. Here's a list of what you will generally need in our **SALES** example:

- |                         |                      |
|-------------------------|----------------------|
| 1 - Order Number        | 7 - Company Name     |
| 2 - Date of Transaction | 8 - Customer Company |
| 3 - Sales Amount        | 9 - Customer Address |
| 4 - G/L Account Number  | 10 - Customer City   |
| 5 - Customer Name       | 11 - Customer State  |
| 6 - Customer Code       | 12 - Customer Zip    |

The next thing to determine is how the data is going to be stored. In this example, you will be entering sales order records based on the customers' purchases. You already have the customer records from the previous sections, so you do not need to keep the customer specific information in the sales file. You do, however, need to keep the Customer Code in the sales file so you can link (relate) the sales records to a specific customer.

This does two things for you: First, it reduces the data entry required to enter a sales order. When the Customer Code is entered, the program will find the corresponding record in the customer file and display the appropriate information. You will be adding the code to do this later. Secondly, it reduces the amount of disk space required to store our application. Each file stores only the data and fields necessary to link to other files in the application.

## Addsum TAS Premier 7i Tutorial

The following table shows how the data you are keeping track of will be stored.

File:	Sales Order	File:	Customer
Fields:	Customer Code Order Number Order Date Sales Amount G/L Account	Fields:	Customer Code Customer Name Customer Company Customer Address Customer City Customer State Customer Zip Customer Area Code Customer Phone

The Customer file already exists, so you will only need to enter the Sales file. The first thing you need to determine is the type and size of each field in the Sales file. As ever, the size of the field is determined by the largest entry you expect to put in that field. For example, you wouldn't make the Sales Amount field only four characters long, since that would limit the sales amount to a maximum of 9.99.

Here are the data types, lengths and key types for the **SALES** file. Use Maintain Data Dictionary to create a new field named SALES. Refer to chapter One for help if needed.

Data	Field Name	Type	Size	Dec	Up
Order Number	SONUMBER	R	6		
Date of Transaction	SODATE	D	8		
Sales Amount	SOSALESAMT	N	9	2	
G/L Account Number	SOGLACCTNUM	A	6		Y
Customer Code	SOCUSTCODE	A	3		Y

Key/Segment Name	Asc/Desc	Duplicates	Modifiable	Ignore Case
SONUMBER	Asc	N	Y	N
SODATE	Asc	Y	Y	N
SOCUSTCODE	Asc	Y	Y	N

**NOTE:** In the field information above we don't include the Short Field Name. Just press the ENTER key when you're in that field and accept the default value.

The keys specified allow searching and reporting based on the Order Number, the Date of Transaction, or the Customer Code. Note that we now allow duplicates of the Customer Code. Since you can have multiple sales for a single customer you need to allow duplicates here; however, we still don't allow the code to change for the same reasons we had when setting the Customer Code key.

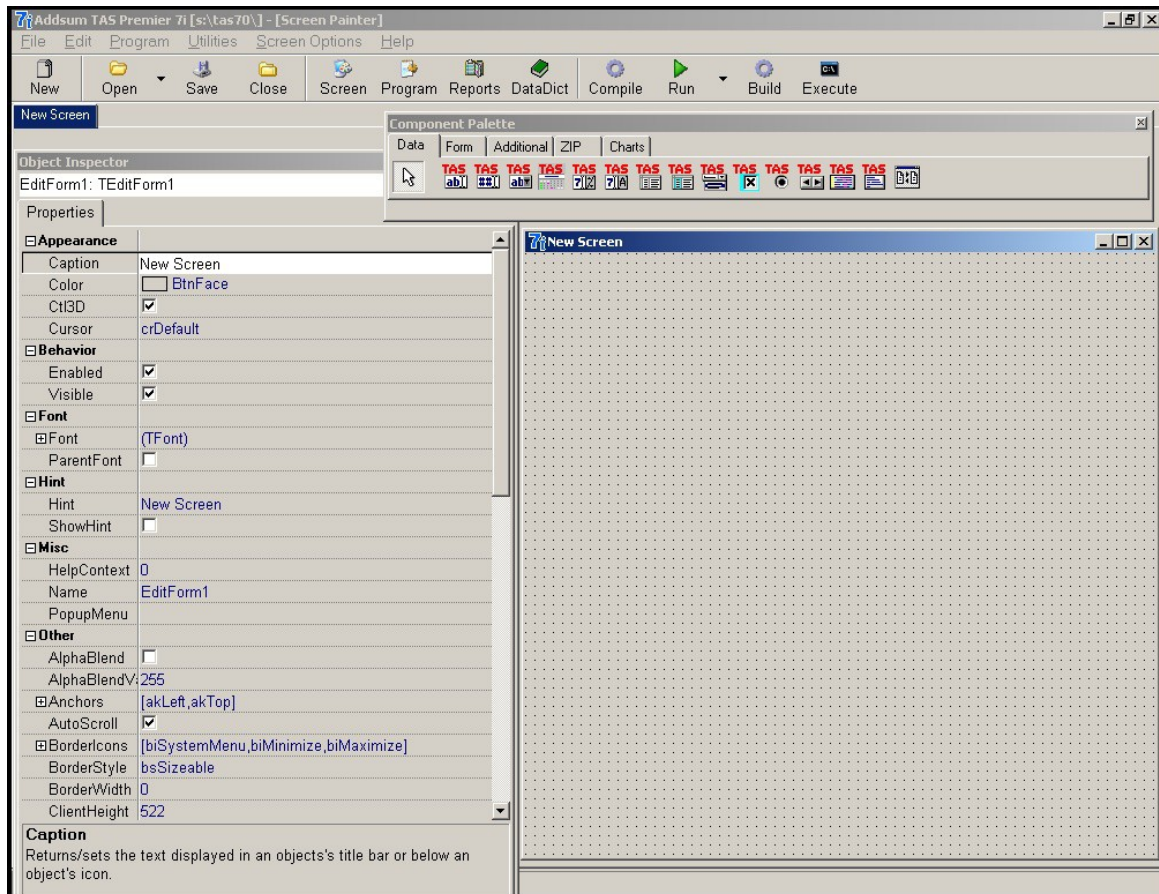
The final thing to design on paper is what you want the computer screen to look like. Often you can think of good ideas by looking at the paper forms around your office. For example, if you were building an invoice tracking system, you might design the computer screen so it looked like your paper invoices. With Addsum TAS Premier you have the ability to make the screen look any way you want.

### Create the Form

Now that you know what you are going to keep track of in your database, you are ready to use the Screen Editor to create a data entry form.

From the Main Menu click on the Screen icon. A blank screen will come up, but if you look at the status bar at the bottom you will see Screen Painter in the first section. You can always refer to this section to make sure you're in the right place.

Now click on the New icon (it's the first one in the menu bar). The screen below will appear. Your screen layout may differ depending on your screen resolution.



This screen is basically a blank canvas. You can put anything you want on it. Above the screen is what is called an object palette. There are three different tabs that make up this palette: Data, Form and Additional (these were called TASWin, Standard and Additional in TAS 6). Each of these is described below:

### Data Page (formerly TASWin)



**TASEnter** - Use this object to enter any type A (alpha - string) field. There are other objects to enter numbers, dates and times, but you can use this object to enter any value in your program.



**TASNumEnter** - Use this object to enter any numeric (N, B, I or R) type field. There are other objects to enter strings, dates and times.



**TASComboEnter** - This is a special version of the standard TASEnter object. Using this you can allow the user to enter a value 'normally', and/or by chaining to a program routine when the user clicks on the button.



**TASDateEdit** - Use this object to enter any type D (date) field. There are other objects to enter numbers, strings and times.



**TASTimeEdit** - Use this object to enter/edit time values. You can also include a check box as part of this object. NOTE: The TASTimeEnter object is recommended for usage instead of this. The TASTimeEdit was retained strictly for compatibility with early versions.



**TASTimeEnter** - This is a newer version of the TASTimeEdit object. It allows for more flexibility when entering time and hours and is the recommended object to use.



**TASDataGrid** - This may be the most important object available. This will display a list of records or array field values on the form in a certain order in columns. You can allow the user to delete rows (records or array elements), insert rows (array elements only), add new rows to the end of the grid, and, through the use of the DATA\_GRID command, add columns, remove columns, shift them around, etc.



**TASDGColTemplate** - This is a duplicate of the TASDGItem (Columns) in a TASDataGrid object. Through the use of this object you can add to or update an existing grid through the use of the DATA\_GRID command. This is a non-visible object.



**TASComboBox** - This object is similar to the TASEnter with a few extra options. You can create a list of values that user can choose from by clicking on the drop down button while typing.



**TASCheckBox** - Use this object to replace a standard Y/N question. If the user checks the box the returned value will be Y; if unchecked, N will be the value. You can also make the attached field a type L (logical). In that case the object will return True (clicked) or False (unclicked).



**TASRadioButton** - The major difference between this object and the TASCheckBox is in usage. Generally, you would put two or more TASRadioButton objects on a TGroupBox. The group box allows only one radio button to be chosen at a time. Using this, you can give the user multiple choices with the knowledge that they can only choose one.



**TASNavigator** - This is a special object that provides the user with a group of buttons that can scroll through the records in a file, delete a record, save a changed record or refresh a record from the file, even after changes have been made. Use the NAVIGATOR command to link this object to a file and key. This is required before this command will have any effect.



**TASStrList** - This object can be used specifically to create a string list. To manipulate the list you can either use the appropriate property editor or the STRINGS() function. This is a non-visible object.



**TASMemo** – This object is used in CodeBase files where you have included a Memo field as part of the FD. It acts as a small word processor and allows you to change and add memos directly to a file.



**DualListDialog** - This is a non-visual object at design time that can display a very useful dialog at runtime. To execute this dialog you will need to use the DUAL\_LIST\_EXEC() function. Refer to this for an image of what the dialog will look like to the user. You will also use the STRINGS() function to determine what options the user has chosen, and perhaps to set the options (strings) the user has to choose from.

## Form Page



**Label** - This object will allow you to put a fixed string on the form. It is the simplest of all objects that you will use and, along with TASEnter, probably one of the most used.



**Image** - Use this object to display an image on the form.



**Button** - This will display the standard Windows button on the form. The user will be able to 'click' on this button (click while the arrow is over the object) and have this execute an Event that will call a routine in your program. This is a common object and will probably be used often on your forms.



**GlyphBtn** - This will display a slightly different Windows button on the form. In this case the button can have a glyph (.bmp graphic) that is part of the button along with the caption. If you don't include the graphic this is identical to the standard Button object. The user will be able to 'click' on this button (click while the arrow is over the object) and have this execute an Event that will call a routine in your program.



**Shape** - This object will allow you to draw a simple shape on the form. It can be used in the same manner as the Bevel – to put a box around an object or group of objects. Other than that, it has no other purpose.



**Bevel** - Use this object to put a bevel around entry objects or other screen objects. Even though this is a visible object there is no interface with the user other than beautifying your form.



**GroupBox** - This object is used in connection with TASRadioButtons. When multiple radio buttons are placed on a single group box, the user will be able to set only one of the options active at a time.



**Panel** - This object is similar to the Bevel in that it can put a border around other objects, but it has other uses as well. The types of styles available are greater than with a bevel and it 'owns' the objects placed on it. This means that if you set the Visible property of the panel to .False., all objects that are on the panel automatically become invisible also. The same applies to the Enabled property. This can be very useful when you know you want to effect a group of objects the same way at the same time.

**NOTE:** The Panel object can also be used as a button since it has a CLICK event just like a button. This is very helpful when you want to use buttons that have a different background color rather than a standard button. You can also place both an image and text (Label) on the panel object.

### Additional Page



**MainMenu** - This object creates the menu that appears at the top of the form. The items that make up the menu are actually MenuItem objects. You create those by using the MenuItemEditor.



**PopUpMenu** - This object creates a menu that appears when the user right clicks on an object (you must specify the Popupmenu object name as the PopupMenu property). The items that make up the menu are actually MenuItem objects. You create those by using the MenuItemEditor.



**Memo** - Use this object to put a miniature word processor on your form. If you want to allow your user to enter messages or notes, etc. you would use this object in connection with the STRINGS() function. This is different from the TASMemo above in that it cannot be connected to a field.



**StatusBar** - This object is generally used to display information to the user at the bottom of the form. You can split up the bar into panels, each of which can be individually modified. The panels that make up the menu are actually StatusPanel objects. You create those by using the Status Bar Panel Editor.



**ToolBar** - This object creates a tool button bar that appears at the top of the form. The items that make up the bar are actually ToolButton objects.



**AlarmClock** - This object will place a clock on your form. It can be either digital or analog in shape and can be used to set an alarm that will call a routine in your program. If you want to execute an event at a regular interval you should use the RtnTimer instead of this object.





**RtnTimer** - This won't appear on your form at runtime; however, it can call a routine in your program at a semi-regular interval. If you want to execute an event only once a day at a specific time you should use the AlarmClock instead of this object. This is a non-visual object.

**NOTE:** In the following documentation, the above objects may be referred to with a "T" before them, like TTASEnter for TASEnter, and TLabel for Label.

### Object Inspector

On the left side of the screen editor is the Object Inspector. Each object has properties. These properties can generally be set either at design time (what you're doing now) or runtime (when the user is running the program). To set them at design time you enter or change the value in the field next to the property name. Each property has been defined in the help file (documentation) for this object. To display that information click on the Help menu option at the top of the screen. Click again on TAS Premier 7i Help. A standard Windows help box will display. By entering the name above, the appropriate help screen will appear.

The Object Inspector will always display the information for the current object. The name of the active object, and its type, will show at the top of the Object Inspector.

### Adding/Moving Objects on the Form

To place an object on the form, click the component icon and then click on the form to place the object. A default version of the object will be displayed. You can then change the size by clicking on one of the sizer (or grabber) blocks that surround the object and, while holding the left mouse button down, move the mouse as appropriate.

Once it is on the form, you can move the object anywhere on it by clicking anywhere inside the object and, while holding the left mouse button down, move the object where you want it. As you're moving the object an outline shape of the object will move with the mouse. When you release the mouse button the object will 'move' to the new location.

Another way to move or resize the object once it is on the form is to change the property values that apply, accessed in the Object Inspector. These are Top, Left, Height and Width. All values are in pixels with the top left of the form position Top=0 and Left=0.

### Multiple forms

You can have more than one form open at a time. Each form that is opened, or created when you click on the New button, will be on their own tab. The name of the forms are shown on tabs. To move between forms all you have to do is click on the appropriate tab.

### Saving/Closing forms

To save an open form click the Save button. If this is a new screen a standard save file dialog will be displayed and you will be able to specify a form name and path. You can also save an existing form to a new name by clicking on the File->Save As menu option. Again, the save file dialog will appear and you can enter a different name, and possibly a new path, for the form. This allows you to easily reuse current forms and create new ones with minor or major differences.

You can also close the form without saving it by clicking on the Close button. If the form has been changed you will be given the chance to save the form before it is closed. The same applies if you exit Addsum TAS Premier completely and there are forms open that have been modified.



## Creating the Sales form

To add the following objects to the form, click on the appropriate icon on the palette and then click on the form. If you click inside the object, and hold down the left mouse button, you will be able to drag the object anywhere on form. While you're moving the object the left and top coordinates will be displayed in a little box immediately under your mouse cursor. It will be in the form of Left, Top. Or, you can enter the appropriate numbers directly in the Left and Top properties in the Object inspector (Object Inspector). The same applies to the Width value. As you adjust the size of the object the Width and Height values will be displayed in the form of Width x Height. As always, the numbers are in pixels.

The first step is to make the form itself the correct size. Depending on your screen resolution, this will probably mean making it smaller. Move the cursor to the lower right corner until it becomes a two-headed diagonal arrow. Hold the left mouse button down and move this double arrow toward the upper left corner. Unfortunately, you won't be able to see the size of the form as it grows smaller; however, you can refer to the ClientHeight and ClientWidth values in the Object inspector to see the results. You can also change those two values directly, rather than using the mouse. Either way, you want to end up with ClientHeight = 235 and ClientWidth = 541.

While you're still on the background form, enter "Sales Order Entry" in the Caption property. This will be displayed at the top of the form.

**NOTE:** Whenever you display an alpha string for entering into a property it will be surrounded by quotes. However, when you actually enter the value do not include the quote marks.

Let's put the first text and field on the form; we'll do the text first. Click on the Standard tab on the palette, then click on the A icon (Label object). Move the mouse cursor anywhere towards the upper left corner of the form and click the left mouse button. This will put a Label object on the form with the default name of Label1. The name will also be put into the Caption property so the name will be displayed as the visible part of the text on the form. Now click somewhere on the Label1 text (inside the outline created by the grabber blocks) and, while holding the left mouse button down, drag the object so that the number under the mouse pointer is 40, 24. You can also enter the appropriate property values. If you'd rather do that, then the values are Left = 40 and Top = 24.

In the Caption property enter the value "Order Number:" Now the Label object on the form should show your text entry. We're ready to move on to the actual field object.

Click on the Data tab on the palette to get back to the TAS field entry objects. Since this is a numeric type field we're going to use the TASNumEnter object. Click on the icon with the '##' inside the entry block. This should be the second icon from the left. If you move the cursor over the icon the object name should appear as a hint. As you did with the Label object, click on the form after you've clicked on the icon.

**NOTE:** The TASNumEnter object automatically creates a button that can be used in your programs to lookup a value, etc. However, in this case, and in the others we use in this tutorial, we have no need for the button to appear. So, in the Object inspector, set the ButtonWidth to 0 and it will disappear.

Set the position of the TasNumEnter object so that Left = 128, Top = 16 and Width = 64. The Height value does not need to change since it should be 21 automatically.

You might want to check that you can activate any object on the form by clicking on it. So, if you click on the Label object the grabber blocks will appear around the text you entered previously and the properties for that object will appear in the Object inspector. Make sure you click back on the TASNumEnter object before you continue.

In this object there are a few extra properties that merit attention. The first we'll look at is Name. Each object you put on your forms will have a name. If you should need to access that object in your program, this is the name the command or function refers to. You should get into the habit of naming your objects logically. So, for

## Addsum TAS Premier 7i Tutorial

example, start the name of this object with “num” since it’s a numeric entry field. Next, you might use something similar to the field that will interface to this object, or what it actually represents; so, in this case, use OrdNum (for Order Number). This will make the Name = numOrdNum. Using a variation of upper and lower case characters makes the name easier to read, but it makes no difference to the program. You could’ve used NUMORDNUM or numordnum; each would refer to the same object. Notice that we didn’t use any underscores or periods. You could use underscores if it makes more sense for you to do so. Periods are not acceptable at all since they are used to separate pieces of the object name internally. So, put “numOrdNum” into the Name property.

In the TASNumEnter object there is a special DisplayFormat property to control how the number value is displayed. Since this field will contain a value that will only be positive and there are no decimal characters, you want to set DisplayFormat = 0. You will see the value in the visible representation of the object change as soon as you press the ENTER key.

Next look for the property called FieldName. This property will tell the object which field to interface to in your program. You don’t have to put a field name into the object, but if you don’t you’ll have a much harder time getting the information in and out and none of the events will work.

### Field Name Lookup

The easiest way to choose a field name is to click on the ellipsis (3 dots in a button) that will appear in the property entry block when you click on FieldName. Do that now and the dictionary lookup screen will appear.

Click on the Data Files drop down box down arrow. A list of FDs will be displayed. Click on SALES. This first object is going to be connected to SONUMBER. It’s already highlighted, so just click on the OK button. The field name will be placed automatically in the FieldName property.

**NOTE:** The next time you load the FieldName search editor the list will start with the FD you last searched. This will make putting a series of fields on your form much easier.

Now you need to put the rest of the objects on the form. The Labels are the easiest so we’ll list those first. All you have to do is place them on the form, make sure they are in the right location, and enter the correct caption (use the list in the box below).

Top	Left	Caption
24	360	Date:
72	31	Customer Code:
120	38	G/L Acct Num:
160	76	Amount:

Next come the entry objects. Each object is given its own paragraph. We start with the object type and then list each of the properties and what you should set as their value. The properties are set in bold and the values are to the right of the equal sign. Any special notes will be given where necessary.

**TASDateEdit:**      **Name**=dateSODate **YearDigits**=dyTwo **Left**=400 **Top**=16 **Width**=88 **FieldName**=SODATE

**TASEnter:**          **Name**=entSOCust **Left**=128 **Top**=64 **Width**=64 **FieldName**=SOCUSTCODE

**TASEnter:**          **Name**=entGLAcct **Left**=128 **Top**=112 **Width**=96 **FieldName**=SOGLACCTNUM

**TASNumEnter:**      **Name**=numAmount **Left**=128 **Top**=152 **Width**=96 **FieldName**=SOSALESAMT

**TASEnter:**          **Name**=entCustName **Left**=264 **Top**=64 **Width**=227 **FieldName**=CUSTNAME

**TASEnter:**          **Name**=entCustComp **Left**=264 **Top**=88 **Width**=227 **FieldName**=CUSTCOMP

**TASEnter:**          **Name**=entCustAddr **Left**=264 **Top**=112 **Width**=227 **FieldName**=CUSTADDR

## Addsum TAS Premier 7i Tutorial

TASEnter:       Name=entCustCity Left=264 Top=136 Width=227 FieldName=CUSTCITY  
TASEnter:       Name=entCustState Left=264 Top=160 Width=24 FieldName=CUSTSTATE  
TASEnter:       Name=entCustZip Left=296 Top=160 Width=91 FieldName=CUSTZIP  
TASEnter:       Name=entAreaCode EditMask="(000);0;" Left=392 Top=160 Width=39 FieldName=CUSTAREA

**NOTE:** This is the first time we used the EditMask property in a TASEnter object. The EditMask above tells the program to put parentheses around the entry and, if the user enters anything, it must be a numeric character. The 0 after the first semi-colon tells the object not to save the parentheses, since these are for entry only. There is a space character after the second semi-colon. It's very important that you include this extra space. It tells the object to use the space character as the default input character. You must have some sort of character in this location; we chose the space character. For more information about the EditMask property refer to the help file. Please be very sure that there are no spaces before the leading "(" in the EditMask field. If there is your data will not display properly. If you use the built-in EditMask editor it may put an extra space at the beginning of the mask. Don't include the quote marks at the beginning and end of the EditMask property value. They are there strictly to show you the extra space at the end.

TASEnter:       Name=entPhone EditMask="(000-0000;1;" Left=440 Top=160 Width=72 FieldName=CUSTPHONE  
TASNavigator:   Name=navSO Left=128 Top=200 Width=232 AfterActionFocus=numOrdNum ClearRecOnSave=Checked  
TButton:        Name=btnExit Left=456 Top=200 Width=75 Caption=E&xit

When you're done the screen should look something like this:

The screenshot shows a window titled "Sales Order Entry" with a grid background. The fields are arranged as follows:

- Order Number:
- Date:
- Customer Code:
- G/L Acct Num:
- Amount:
- entCustName:
- entCustComp:
- entCustAddr:
- entCustCity:
- entCustZip:

At the bottom, there is a row of buttons: a left arrow, a right arrow, a cancel button (X), a save button (checkmark), and an "Exit" button.

### Saving the Sales form

The next step is to save this form to disk. Click on the Save button, the save file dialog will appear. Enter SOENTRY for the File name value and click on the Save button. The form will be saved to disk and the name on the tab will change to SOENTRY, if you look at the bottom of the editor screen, you will see the full path and name for this file.

You're done with the first part of creating this program, the form. The next step is to create the source file for the program that will work with the form.

### Create the Program Source File

Click on the Program button, then click New. A blank file will appear.

On the right side of the screen you will see a tree listing of all the commands and functions for Addsum TAS Premier. As with a standard tree structure, some of the items (commands only) have a plus (+) sign to the left of the entry. If you click on this symbol the options for the command will appear.

If you double click on a command it will be inserted in your code at the current cursor location. If you double click on a command name that has sub-options, the entire command, including all the sub-options, will be inserted into your source file. You can hide or "un-hide" the command tree using the check box in the bottom right hand corner of the screen.

The source editing form is very similar to a standard Windows® editor, except that the lines don't wrap when you reach a certain line length. There are a couple of other options that are unique to a source code editor that you may not be familiar with:

**Goto Line:** Each line in your source file has a line number. The numbering starts from 1 and the lines are numbered sequentially, including all blank lines, etc. The current line and column location are displayed in the left column immediately after the source editing area. To go to a specific line press the CTRL+G keys. A dialog box will come up and you can enter a line number.

**Bookmarks:** You can put up to ten different bookmarks in the source file. These are not saved with the source file, but are active as long as the file is being edited. They are helpful because they allow you to mark a specific line. Then you can return to that line by pressing the appropriate keys or choosing the bookmark from the Bookmark menu option at the top of the screen. When you set a bookmark on a line a small block with the appropriate number (0-9) is placed in the gutter (space to the left of the editing area).

To set a bookmark from the keyboard, press the CTRL+K+number keys where number must be 0-9. To return to the bookmarked line press the CTRL+Q+number key. To turn off a bookmark press the same CTRL+K+number keys you used to set it. Also, you can 'move' a bookmark to another line by doing the same on a different line than it was originally set.

**Undo/Redo:** If you make changes to the source file that you find you really don't want, the easiest way to get rid of them is through the Undo feature. This option (along with Redo) is on the Edit menu. To undo changes from the keyboard enter the CTRL+Z keys. To redo (reverse the undo) enter the SHIFT+CTRL+Z keys.

**Find:** Standard Windows® find option. This is on the Edit menu. From the keyboard use the CTRL+F keys.

**Replace:** Standard Windows® find and replace option. This is on the Edit menu. From the keyboard use the CTRL+R keys.

**Lookup:** An option unique to Addsum TAS Premier is the ability to search for field names using the same lookup routine you first saw in the screen editor. This option is also on the Editor menu or you can enter the CTRL+L keys. The field lookup dialog is displayed. If you choose a field (double click on the field name or press the OK button) it will be inserted into the code at the cursor location.

**Convert Remarks:** Remarks are messages for the future that you, or someone who might follow you, will refer to when trying to figure out what you were doing in a piece of program code. The more remarks you make the easier it will be to maintain or modify the program in the future. In Addsum TAS

Premier there are two 'normal' ways of specifying remarks. If the remark is going to be short, a single line or less, you would use "//" (two forward slashes). Everything after the remark, *on that same line*, will be ignored when the program is compiled.

The second option is when you're going to have a remark that is going to take multiple lines. You can use the // remark at the beginning of each line or you can start the remark with "(\*)" (parenthesis + asterisk), and it will continue over as many lines as you wish until you put the closing characters, which are the "\*)" (asterisk + closing parenthesis).

In TAS 5.1 the semi-colon was the remark character, and that will still work in the compiler. However, the source editor doesn't recognize that as a remark. So, if you choose this option, the editor (on the Edit menu) will change all of the old remarks (;) to new (//).

### Enter the Actual Code Lines

The actual code lines for this program are fairly simple. The code breaks down into five different sections: Form, Open Files, Finding Customers, Saving Records and Exiting the Program.

### Form Section in Source Code

The forms (and report forms for that matter) are not incorporated directly into the final compiled program, for reasons we explained previously. However, you still need to tell the source code what the names of the screen and report forms are. This has two purposes. The first is that during compilation the program will check the forms and will make sure that any field referred to by that form exists in the data dictionary or has been defined in the program. It also adds the field to the list of fields used in the program. The second purpose is that by the location of the form reference, the program is told which screen should be loaded when the program is run.

Add the following line:

```
#Winform SOENTRY
```

This is the line that tells the source code to use SOENTRY.DFM as its main screen. Notice that the name of the form and the source file (the comment above) are the same. This isn't required but you'll probably find it easier to maintain your programs if the screen and report forms use the same or similar names as the source file.

The "#" (pound sign) at the beginning of the line tells the compiler that this is a Compiler Directive. This isn't actual executable code but is a directive or reminder to the program of something that has to be done. In this case, this is the form that needs to be checked and loaded when the program is run (since it's the first form listed). Generally, even if you have multiple forms, you should list them at the top of the source file. Then you will always be able to quickly and easily check which files need to be included with the finished run program.

The "winform" is a particular directive which tells the compiler that the next item on the line is a name of a form. Always put at least one space between winform and the form name.

Next is the form name of "SOENTRY." Notice that there are no quote marks around the form name. This is what we call a "special alpha constant." They are used throughout Addsum TAS Premier in different commands. Most commands allow variables to be used and, in that case, you would've typed the name surrounded by single or double quotes. However, in this case, the compiler wants the actual name, not a variable. Notice also that you don't include the path or extension. The compiler assumes both. You should always have the form file in the same subdirectory as the source file. And, when you run the program, you would have the form file in the same subdirectory as the compiled program.

### Open Files Section in Source Code

Next we're going to open the appropriate files: CUSTOMER and SALES. Each form has four different events that can be used at different times. Two of them are executed when the screen is first displayed by the program. These are the OpenFiles and OnStart events. Events look for line labels in your program that match either what

## Addsum TAS Premier 7i Tutorial

you've entered, in the case of forms, or a combination of the object name and event type in the case of most other objects.

First we will add The OnOpen event and code. These commands will be added between the line label (defined below) and the RET command.

Copy the following lines into your source file, two or three lines below the #winform code.

```
OnOpen:
  define cust_hndl,sales_hndl type i
  openv 'customer' fnum cust_hndl
  openv 'sales' fnum sales_hndl
  navigator 'navSO' fnum sales_hndl key sonumber
  ret
```

For CodeBase files in a mixed system, specify ext 'C' at the end of the openv statements.

The first line is the actual line label. It starts with the first non-whitespace character (whitespace characters are spaces, tabs, etc., anything that doesn't print) and continues until the colon (":"). The colon is not part of the line label but does specify that what comes before is a line label.

The define command creates two temporary fields, cust\_hndl and sales\_hndl, that are active while this program is run. We'll use these two fields to refer to the files elsewhere in the program. Typically these types of fields are defined as type I (integers).

The next two lines are the actual file open commands. These tell the program to access these files and prepare them to be used.

The next line is a special command that links one of the files that was just opened to the Navigator object on the form. After you set the fnum (or file number) and key value (in this case you want to use SONumber), the user will be able to find the appropriate user record by clicking on the navigator button.

The most important line is the last. You must always return from an event or a subroutine with a RET - ALWAYS! The best way to remember this is to imagine that through the form the user controls the program (this is also known as Event Driven Programming). If you don't return (RET) from a subroutine the user will never gain back control. Sometimes you may want this to happen; however, if the user doesn't have control they can never exit the program, will get very frustrated, will start hitting CTRL-ALT-DEL, etc. Then you will get a call. So always try to keep your subroutines as short as possible and always, always return.

To complete this event you need to save the label name in the form. So, click on the Screen button. If you have already closed the form, for whatever reason, it's very easy to get it back. Click on the down arrow right next to the Open button. A list of recently opened form names should appear. Your SOENTRY file should be at or near the top. Just click on that name and the screen will load.

In the properties for the form, find the property OpenFiles. In the entry field for that property, enter OnOpen; this will link it to the same label in code.

**NOTE:** If you make changes to the form don't forget to click on the Save button. Changes made to a form are not saved until you click on the Save button. Nothing will happen to the form, but the appropriate event will be called when the program is run. If the form has been modified the name of the form in the tab will be highlighted.

This is it for the OpenFiles section. Don't forget, if you have questions about the specific commands that we use please refer to the help file. You can search for any command using the command name as provided here.

### Find Customers in Source Code

## Addsum TAS Premier 7i Tutorial

Each time the user enters a customer code (SOCustcode) you will want to find the corresponding record, if it exists, in the CUSTOMER file. This is very easy to accomplish.

Leave a couple of spaces after the last section and enter the following lines:

```
entSOCust.Change:
    findv m fnum cust_hndl key custcode val SOCustcode
    ret
```

The most important part of the code section above is the label at the beginning. This is a very good example of a standard event label. This one will be called each time the user makes a change to the field value. If the standard entry to this field was very long, or if there were lots of fields on the screen, you might do this a different way since each time the user enters a character this routine will be called. However, in this case there are only three characters that can be entered, and it will make a vast difference for the rest of the program. For example, each time the user finds a record using the Navigator, the CUSTOMER record will be updated automatically.

The makeup of this label is very simple. The first part is the object name, in this case “entSOCust.” The second part is the event name, in this case “Change.” Between the two parts is a period (“.”). At the end of the label is the standard colon (“:”) that defines this as a label to the compiler. This is standard event label notation and, except for changing the object name or the event name, this is exactly how you will start event routines in your code.

The next line is the actual find. It tells the file manager to find a record in the file using the file number represented by the variable CUST\_HNDL. This value was set in the OPENV command, and the key CUSTCODE. It needs to find the exact match using the value in SOCUSTCODE. If no record is found, no error will be displayed.

As with the previous subroutine, the RET command sends control back to the form.

If the record is found the CUSTOMER fields on the form will be automatically appear. You do not need to do anything else.

### Exiting the Program

The last section of code we’re going to create is the part that tells the program to quit when the user clicks on the Exit button. You really didn’t need to include this, or the button, since the user could click on the standard Windows® close button in the upper right corner. However, most “standard” programs have such a feature so we’ve included it here.

The code, as the others, is very simple and is listed below:

```
btnExit.click:
    quit
    ret
```

The event label is in standard format. In this case, the event is “click.” This is the only event for the Button object.

The next line tells the program to quit or exit. And, of course, there’s the RET command to return from the event.

### Saving the Source File

The next step is to save the program to disk. As you did with the form, click on the Save button. Enter SOENTRY as the file name.



### Compiling the Source File

Before a program can be run in Addsum TAS Premier it must be compiled. The compiler checks to make sure all commands are correct, that fields not in the data dictionary have been defined, etc. It also puts the programs into a special format that allows the runtime to execute it faster, and makes it so your source code is protected if you don't want to release it to the user.

To compile the program click on the Compile button.

Your program name will already be in Program Name field, so just click Compile. After the process is complete a new tab will appear between Main Screen and Defaults. If you entered the code correctly it should say No Errors. If you didn't, and the compiler found an error, it will say Errors. If it does, click on that tab and you should be able to quickly find what went wrong. Fix it and recompile the program.

Once the compiler has successfully compiled the program, the Run Program button will be set as the default. Again, press the ENTER key or click on the Run Program button. When you do, you should see the same screen you created without the dots on the form itself.

Enter 101 for the Order Number and press ENTER. Click on the button to the right of the Date entry field. A calendar will be displayed with today's date highlighted. Click on the button again and the date will be set to today's date. Press the ENTER key and the cursor will move to the Customer Code field. Enter 002, press the ENTER key, and the customer information for Smith, John should appear. Enter 500 for the G/L Acct Num and press the ENTER key. Enter 150.99 for the Amount and click on the Save button in the Navigator (it looks like a check mark).

After saving, the SALES and CUSTOMER fields will be cleared; that is, they will be cleared if you previously checked the ClearRecOnSave property in the Navigator. If you prefer to leave it un-checked, the information will not clear.

If you click on the First button in the Navigator (far left) you will redisplay the record you just saved. Even the CUSTOMER information will redisplay. This was due to the Change event we included for the Customer Code field above.

Let's say that the user wants to add a new customer. This would be a logical place to do so, but the program will save only the SALES information when they click on the Save button. Let's add the code to your program that will update the CUSTOMER file also.

Click the Exit button to quit the SOENTRY program.

### Saving Changes to the CUSTOMER file

The user will save records by clicking on the check mark button in the Navigator bar. However, this will save only the record in the file that's attached to the object via the NAVIGATOR command. So, it's up to you as the programmer to save the record in the CUSTOMER file via code. Fortunately, the Navigator object has an event that will let you know each time the user clicks on the save button: the SAVE event. The following lines will accomplish this; add them to your source a few lines below the btn.Exit code.

```
navSO.save:
    custcode = socustcode
    save @cust_hndl nocnf
    ret true
```

The first line is the standard event label. In this case the Navigator name is "navSO" and the event is "save" and we've separated the object name from the event with a period.



## Addsum TAS Premier 7i Tutorial

The next line makes sure the customer code value in the CUSTOMER file matches what you entered for the SALES file. If you find an existing record, then both values will already be the same. However, with this line here you will now be able to save new records in the CUSTOMER file.

The next line tells the program to save the record in the file represented by the file number in “cust\_hndl.” It also tells the command to not ask the user to confirm this save (nocnf). If this option were not included the user would get two “Save?” questions: one from the Navigator object itself, and one from this save command.

**NOTE:** You can even eliminate the “Save?” question from the Navigator object by going back to your form and clicking on the Navigator bar. Look at the properties in the Object inspector. One of them is “ConfirmSave” and has been checked. If you un-check that property the user will never see a “Save?” question. Don’t forget to save the form if you make a change. When you make changes like this you *do not* need to recompile the program. However, if you do not save the form, the program will not know the change has been made.

The last line in this code section is a variation on the standard event return command. In this case we are actually returning a value of “true.” This is called a Boolean value. Boolean values have only two possibilities: True or False. In the case of events that expect a Boolean value returned, generally the True value tells the event to continue, the False value tells it to stop, or not continue. If an event expects a return, it is documented in the help file for that object. If you do not include the True or False value the event will act as though you returned True.

In this case always return True, since you are not using this event to check whether or not to save the record; it is being used to save the record in the CUSTOMER file.

It does not matter where you put this routine in your code. You can add it to the end, or put it before the Exit routine. Location is strictly dependent on what works best for you, since each routine is considered almost like a separate program

**NOTE:** If the user answers No to the original save question from the Navigator, you will never get to this event.

Since you’ve already saved the source file once, you can let the compiler do that for you from now on. Just click on the Compiler button. The correct source file name should automatically appear in the program name entry box. Press the ENTER key to compile the program. Before the compiler actually does its work it checks to see if the source file is active (which it is) and if it’s been modified (which it has) and saves the file back to disk before it is actually compiled. So, this saves you the work of saving the program before it’s compiled. Unfortunately, this doesn’t apply to forms. You have to save those yourself.

### Further Modifications to the SOENTRY Program

Let’s make a couple of other changes to the form and the program. If you do not have the screen loaded then click on the Screen button. Then click on the down arrow to the right of the Open button. A list of forms you have edited should appear. At this time it’s probably just the single screen SOENTRY. Click on the appropriate file name and the screen will be loaded into the editor.

The first change to make is to have the date field default to today’s date. Click on the Order Date (dateSODate) object. In the Object inspector find the property DefaultToday. Click on the check box to the right of the property name (if it isn’t checked already). Now, the next time you run the SOENTRY program the date will default to today without any other entry.

Now click on the Order Number (numOrdNum) object. You are going to make two changes here. Scroll down until you get to the ValidExpr property. In this property entry box add the following line:

```
SONumber<>0
```

As with other code, upper or lower case does not matter. What this expression will do is check to see if the SONumber, the field attached to this object, is 0 or something else. If it’s 0 the expression returns False. If it’s anything else, it returns True.

## Addsum TAS Premier 7i Tutorial

In the ValidMsg property click on the ellipsis button. In the text editing dialog enter the following: “You must enter a number here.”

Again, click on the OK button to save the message to the property. This will appear when the VALID\_CHECK() function returns False. We are going to add that function to the source file.

Now save the form to disk by clicking on the Save button. This is very important since the compiler will not be able to see the new ValidExpr property value unless the form is saved. You might also want to close the form since it will be updated by the compiler and you don’t want to save the ‘old’ version over the ‘new’ version that the compiler will create. See the ValidExpr property in the help file for more information.

Now, click on the Program button. As with the form, if the program is not loaded you can load it by clicking on the down arrow next to the Open button and then choose the appropriate source file.

Immediately after the navSO.Save line insert the following line:

```
if .not. valid_check() then ret false
```

This line will execute the VALID\_CHECK() function, and if it returns False (one of the ValidExpr property values returns False) then it will exit the save subroutine and neither of the files will be updated (by returning False from the subroutine it tells the program to *not* save the record in the SALES file, also). If a ValidExpr evaluates to False, the ValidMsg for that object will be displayed and focus will return to that object, forcing the user to enter something other than 0, in this example. The new save subroutine should look like this:

```
navSO.save:
  if .not. valid_check() then ret false
  custcode = socustcode
  save @cust_hndl nocnf
  ret true
```

Now click on the Save button in the Navigator (check mark) while keeping the Order Number value as 0. You will get the message you entered above as the ValidMessage and the cursor will return to the Order Number field.

What’s important about this? The use of the ValidExpr property/VALID\_CHECK() function gives you a quick and easy way to check all important fields at one time without having to remember each and every field. In previous versions of Addsum TAS Premier you could force your user to stay in the field until a proper value was entered. With the change to an event-driven system you want to give your user as much flexibility as possible. You can use the VALID\_CHECK() function anywhere. If there are multiple ValidExpr’s to check, the program will check each one. If multiple fail the user will be returned one at a time to fix them before continuing. This is an important feature of TAS Premier and one that you will probably use quite often.

The last item to notice is that when you load the form, the date field now has today’s date instead of 00/00/00.

We still have one missing feature: What if the user wants to find the order in a group of orders? Let’s put together a quick lookup.

### Add a Lookup Grid to the Sales Program

The first step is to create a new form. If you are not in the screen editor, click on the Screen button now. Then click on the New button to create a new form. Make the size of this form 296 pixels high (ClientHeight) and 552 pixels wide (ClientWidth).

While you are on the form, we’ll make a couple more changes. These are:

**Caption**=Sales Order Lookup **OnStart**=SOLookupStart

Note that a line label is included for the OnStart property for this form. Why not just use START, since it's the default? To keep the program from calling the same subroutine for two different forms, unless you want it to, the program defaults to START only for the first form. All subsequent forms have to specify an OnStart line label, which has been done here.

**NOTE:** While you're placing objects on the form, you may exceed the edges of the form and scroll bars will automatically appear. Just make the form larger and then return to the correct size. The scroll bars will disappear automatically.

Place a TASNumEnter object on the form at the location Left=8 and Top=8. The Width should be 80. This is going to be our FastSearch™ field. The rest of the property entries are:

**Name=numOrdLU DisplayFormat=0 FastSearchType=fsRec**

We do not have to assign a field to this object since it's only going to be used as a FastSearch™ field.

Next drop a TASDataGrid object on the form. The general properties are as follows:

**Name=dgSOLookup Height=216 Left=8 Top=40 Width=536 FastSearchFld=numOrdLU**

Click on the + next to the Navigation property name. Now check the box next to the AdvanceOnEnter subproperty. You can click on the – next to the Navigation property name to close up the subproperty options.

By setting this subproperty, you tell the program to do an automatic select if the user is entering characters in the FastSearch™ field and presses the ENTER key.

Now display the Option subproperties by again clicking on the + next to the property name. Check the box next to the goDrawFocusSelected subproperty.

We now need to add the columns to the grid. To do this, click on the Columns property and then click on the ellipsis button that is displayed. A special column editor will appear. A full explanation of this property editor is in the help file.

When the property editor appears, it is already on column 0 (first column in the grid). The property value entries for this column are:

**Alignment=taRightJustify Name=SONum Header=Order Number Width=85 FieldName=SONUMBER**

As you modify these properties, you will see the appropriate changes to the grid, including header and width. We actually want five columns total in the grid, so you need to add four more. To add a column, click on the + button at the top of the column property editor box. Column 1 should appear in the box and a new set of properties in the Object inspector. For each column listed below, click on the + button to add the column to the grid.

**NOTE:** If the Alignment property is not given, it should be taLeftJustify, which is the default value.

**Column 1: Alignment=taCenter Name=SODate Header=Order Date Width=76 FieldName=SODATE**

**Column 2: Name=SOCust Header=Customer Code Width=95 FieldName=SOCUSTCODE**

**Column 3: Name=SOCustName Header=Customer Name Width=111 FieldName=CUSTNAME**

**Column 4: Alignment=taRightJustify Name=SOAmt Header=Order Amt Width=85  
FieldName=SOALESAMT**

When you are finished creating the columns, click on the close button at the upper right corner of the property editor. The properties in the Object inspector will disappear until you click on one of the objects in the form (or the form itself).

When this form is loaded, you will use the `LOAD_MODAL_FORM()` function. A modal form, in Windows® parlance, is simply one that remains active until the user does something that closes the form. To do that, you have to send a *modal response* to the form. This can be accomplished in two ways: you can use the `SET_OBJECT` command or include one or more buttons on the form that have their `ModalResult` value set. Then, if the user clicks that button, it will send that value back to the form and the form will exit without any further action on your part. So, the last objects you are going to put on the form are these two buttons:

**Button:** `Caption=Select Name=btnSelect ModalResult=mrOK Height=25 Left=376 Top=264 Width=75`

**Button:** `Caption=Exit Name=btnLUExit ModalResult=mrNO Height=25 Left=464 Top=264 Width=75`

The last step is to save the form. As with the `SOENTRY` form, click on the Save button and enter `SOLOOKUP` for the file name.

Next you're going to give the user a way to call this form. You could put a button on the `SOENTRY` form, but this tutorial uses a different option. If the `SOENTRY` screen is already loaded, then click on the tab for that form. If it isn't, then load the form as previously described.

Click on the Order Number (`numOrdNum`) object. Look for the `KeyTraps` property. Click on the property and then click on the ellipsis button. A `Key Traps` property editor will be displayed. As with all other property editors, this is documented in the help file.

You're going to add a key trap to this form. Then, when the user presses the appropriate key (in this case `F2`), the line label you specify will be called in your program, just like a normal event.

Click on the box (also known as a cell) in the `Trap Name` column. A drop down box arrow should appear. You can either enter `F2` directly or click on the drop down arrow and a list of available `Key Trap` Names will display. Press the `ENTER` key and the cursor will move to the `Label Name` column. Enter `SOLOOKUP` here. The property editor should now look like the following:

Click on the `Ok` button and you should see `F2|SOLOOKUP` in the `KeyTraps` property box. Next, click on the `KeyTrapHint` immediately above. Again click on the ellipsis button and, in the text property editor enter the following line:

Press the `F2` key to lookup existing orders.

Click on the `OK` button to save it, then on the `Save` button to save the form. You are finished with the screen changes.

Click on the `Program` button, and, if the `SOENTRY` source file is not loaded, load it now.

You need to put the code in your program that will be executed when the user presses the `F2` key. Add the following code a few lines after your existing code:

```
SOLookup:
  if load_modal_form('SOLookup') <> mrOk
    clr @sales_hndl
    clr @cust_hndl
  endif
  ret
```

The label is the same you entered in the `Key Traps` editor. The `LOAD_MODAL_FORM()` function loads the `SOLookup` form and will not return until the user either selects one of the available order records or clicks on

## Addsum TAS Premier 7i Tutorial

the Exit button. The IF command compares the returned value and, if it's not mrOK (the user clicks on the Ok button or presses the ENTER key), the records of both files are cleared before the user returns to the calling form. As with other event routines, this one also ends with a RETURN command.

The next subroutine you need to apply is “turning on” the lookup data grid. This must be done before any records will be displayed. To accomplish this add the following lines:

```
SOLookupStart:
  wlistf 'dgSOLookup' setup fnum sales_hndl key sonumber
  ret
```

The line label is the same one you specified in the OnStart property for the SOLookup form. The WLISTF command is used in connection with the data grid when you are accessing records from a data file. This is in the OnStart event so that the grid will be active when the lookup form is displayed.

The last subroutine you have to add will be executed if the user presses the ENTER key. All it does is set the Modal\_Result property to mrOK. The GET\_FORM\_NAME() function is included as part of this command line, since we don't know the name of the form like we do for the other objects. Add the following lines:

```
dgSOLookup.Select:
  set_object get_form_name() property 'modal_result' value mrOK
  ret
```

There is data from the SALES and CUSTOMER files in the lookup grid. The grid knows about the SALES file from the WLISTF command, but nothing about the CUSTOMER file. So, you need to tell the program to get the appropriate record in the CUSTOMER file for each record in the SALES file. Look at the routine that is called when the SOCustCode field changes. This will not have any effect here, due to a number of technical reasons; however, you can use the same routine. All you have to do is put an extra line label before the routine. Change the entSOCust.Change routine so that it looks like this:

```
dgSOLookup.Display:
entSOCust.Change:
  findv m fnum cust_hndl key custcode val SOCustcode
  ret
```

This tells the program to call the same routine for both the Display event for the lookup grid (dgSOLookup) and the Change event for the SOCustCode field (entSOCust).

**NOTE:** Line labels are not true executable commands. They are just pointers to a specific location in the program. This means you can put as many line labels as you need at the beginning of a subroutine. Each one will start with the first executable line after the label.

Your modifications are now complete. Save and compile the program by clicking on the Compile button. When the compilation is complete, and assuming you don't have any errors, try running the program.

With the cursor in the Order Number field, press the ALT-F1 key. The KeyTrapHint that you entered should be displayed. Press the ENTER key, or click on the Ok button, to clear the message. Now press the F2 key. The lookup screen should appear.

As you enter an Ord Num value, the selection line will move to the appropriate record. Then if you press the ENTER key the lookup screen will disappear and the chosen record will be displayed in the SOENTRY screen.

This completes part 3 of the tutorial.

## PART 4 - CREATING A REPORT

In this section you will create a report showing sales and customer information. You will be able to specify the sort field at runtime and a range of customers to include on the report.

To create reports with Addsum TAS Premier, you'll probably always start with the **Edit Report Format** program. Then, after you have the format completed, return to the Source Editor, and create the code necessary to setup the data for your report. This is similar to the Screen Editor/Source Editor combination, except that you cannot change Report Format properties at runtime.

To get started click on the Reports button. A completely new window will open up and something similar to the screen below will appear:

As with the Screen Editor, the higher your resolution, the more you will see on the screen.

There is just a single palette in the Report Editor. There are actually only eleven objects that can be placed on the format. However, there are several options that will make your reports look better, along with menu options that will add or subtract bands from the format.

Every report is made up of bands. These will control what will print, where it occurs on the report, etc. When the report format is first displayed there are three bands: Header, Detail and Footer. Many reports will use only these three.

### Report Editor Menu Options

You can easily control the bands that make up your report from the menu. Click on the Report menu item at the top of the screen. Notice that the Header and Footer items are checked. If you un-check either of these (by clicking on the menu item), the band will disappear on the form. Click on it again and it will be back. There is no Detail option – if you don't have a Detail band, you don't have a report! The Header band will print each time you start a new page and the Footer band will print at the bottom of each page.

Two other bands are also part of that menu option: Title and Summary. The Title band prints once at the very beginning of the report. The summary band also prints once, only at the very end of the report.

The Groups option allows you to group records together. This allows you to force a page break when a group changes, keep group records together, and create GroupHeader and GroupFooter bands that print with the Group. If you are going to print checks, invoices, statements, or other similar reports you will end up using the Grouping feature. If you don't use it for those things, you will have items from one check on another!

Portrait and Landscape are mutually exclusive; you choose one or the other. Portrait prints the report in what might be called "standard format" (i.e. tall and narrow). Landscape turns the paper 90 degrees so that the paper will print "sideways" or short and wide. If you need to print a very wide report, try Landscape; otherwise you will probably always use Portrait, which is the default.

Units allows you to choose the measurement system for the report. This defaults to Inches. If you click on the Units menu item, you will see the available options. You can switch to another value at any time and all of the properties where a position value is necessary will be recalculated.

If you click on the View menu item, you will see two options listed. The first, Rulers, should be checked. If you click on this option it will remove the rulers from the report form. You will probably always want the rulers to be displayed.

The other option in View is Grid Options. This will allow you to choose the granularity of the grid that overlays the report form. This is the same as the grid in the Screen Editor, except it doesn't display the dots in the Report Editor. In most cases you will leave these values unchanged. Click on OK or Cancel to close the window.

The Edit menu is fairly standard and should be familiar to you from other Windows® programs.

The File menu is also standard, except for a few options specific to the Report Editor. These are the list of the most recent report formats at the end of the menu. If you wish to reload any of the report formats listed, just click on the file name.

Another option in the File menu is Setup form. This acts the same as Page Setup in other programs. You can make this report for a specific printer by choosing the printer here. However, in most cases the printer should be set to Default, allowing the user to choose at runtime. The Document Name should always be Report. Duplex specifies printing on both sides of the paper. Make sure your printer allows this before you choose this option.

The Paper Size tab allows you to choose a size other than the standard 8.5 x 11 inches. As explained above, you can also choose Portrait or Landscape and see a graphic representation of what it looks like.

The Paper Source tab allows you to choose where the paper will be fed. These are general options and can be safely ignored.

Layout allows you to split the page into columns.

All new report formats start as a single column. However, if you are going to print labels, you'll want to change the Columns value to match the number of columns of labels on the page. You will probably also want to remove the Header and Footer bands since they won't change size. Change the Columns value to 4 and notice what happens to the report format (make sure you're starting with a new form!). You should see the Detail band becoming narrower, while the Header and Footer bands stay the same size. You will also see the ColumnHeader and ColumnFooter bands. These have the same effect as the standard Header and Footer except they will print at the beginning and end of each column. Don't forget to reset the number of columns to 1 before you leave this page.

The last tab is Margins:

In a default report all four margin values will be set to ¼ inch. You can adjust those measurements as needed. However, before you make them smaller, make sure your printer will be able to print closer to the edge of the paper.

Once you have made your adjustments, click on the OK button to save or Cancel, clearing your entries.

### A Simple Report

For your first report you are going to do something fairly simple: list the records in your CUSTOMER file. If the Report Editor is not loaded, do so now by clicking on the Reports button on the main Addsum TAS Premier tool bar. The default Report Editor screen should be displayed. The process for putting objects on the form is very similar to what you did in the Screen Editor with a couple of exceptions: first, you will find the number of properties for each object is less than what you had in the Screen Editor; and second, the report format area is broken up into three blocks or bands, as explained above.

**NOTE:** If you have further questions about the different objects or bands in the Report Editor, please refer to the Addsum TAS Premier Help file. You can access this in the Report Editor by clicking the Help menu option and then the TAS Pro 7 Help item. This is the same Help file that you can access in the Screen/Source Editor.

Click somewhere in the report form above the bar that says “^ Header.” The “^” denotes that the Header band is above this bar., as are the Detail band that comes next and the Footer band that is at the end.

When you click in the Header band, notice that the Object inspector now displays information about the Header. Click in the Detail and Footer and you will see the properties change. Now move the mouse cursor so that it's directly over the bar that says “^ Header.” The mouse cursor should change from the standard arrow to a double-headed arrow that is vertical instead of canted off to the left. While the mouse is positioned over the bar,



press down on the left mouse button. While holding the mouse button down, scroll the mouse up and down. You should see a representation of that bar (just the outline) moving with the mouse. When you release the left mouse button the size of the Header band should change, depending on where you moved the mouse. This is how you resize the band.

**NOTE:** Sometimes you will make a change to the report that will create new bands on the page that have no space that is automatically setup for them. A good example of this is a Group band. When it's created the GroupHeader and the GroupFooter band bar will be tight against the Header and Footer band bar. This means that you will have to resize the band and make it larger if you want to put something in those bands. Always look for the “^” and a name in the bar that specifies what type of band is above the bar. Even when the bars are tight together you can still click on the bar of the band you want to increase and scroll down the screen.

**NOTE:** When you resize a band all the bands below move up or down automatically. As you create large reports, you may move part of the report format off the page. A standard scroll bar will appear on the right side of the editing area that will allow you to move the format up and down. The same applies if the format is too wide to fit the area available.

The first step is to put a title in the Header band. Click on the Label object (the one that looks like an A by itself) and click anywhere in the Header band. It should show a small object that says “Label1”. If you look at the Object inspector, you should see TppLabel in the display box near the top and the properties appropriate to this object should be displayed.

In the Caption property enter “A Simple Report.” When you press the ENTER key or otherwise move off the property, that caption will take the place of the Label1 name. This works just like the TLabel object did in the Screen Editor. Now, click on the object and drag it to the top of the band so that it's tight against the ruler. Once you have it there, try dragging it down one step. You'll see the snap-to-grid process working just like it did in the Screen Editor.

Make the font a little bigger by clicking on the font size drop down box in the palette. Change the font size to 14.

Click on the “Center Horizontally” button in the palette above the format. This is the button that looks like three books between bookends. The hint that appears when you hold the mouse over the button is: “Center Horizontally in Band.” Your caption should now be centered in the band.

Drop a SystemVariable object below the label you added above. Notice that it uses the same font size that was changed above; change the size back to 10. In the VarType property click on the down arrow and choose the vtDateTime property. To finish with this object, click on the Center Horizontally button to center this object in the band.

To the right of the report title, drop another SystemVariable. Set the VarType property to vtPageNoDesc. Set Left to 7.25 and Top to 0.0833 (top should be the same as the report title).

**NOTE:** If you change the size or location of any object, including bands, by adjusting the object directly, the new sizes/locations will not be updated in the Object inspector until you switch to a different object and then switch back. The status bar at the bottom of the report format will give you up-to-date information.

Put the following Label objects on the report form in the Header band:

**TppLabel: Caption=Customer Code Left=2.2 Top=0.6667 Font Size=10**

**TppLabel: Caption=Customer Information Left=3.5 Top=0.6667 Font Size=10**

**TppLabel: Caption=Phone Number Left=6.1 Top=0.6667 Font Size=10**



Now you're going to put fields on the report. Click on the DBText icon on the palette and drop it in the Detail band (directly below the Header band). Set the Top property to 0 and the Left property to 2.2. Now double click in the DataField property. The same dictionary field lookup grid that you are familiar with from the screen and source editors will appear.

If the CUSTOMER file is not displayed as the initial choice, then click on the drop down button and choose the CUSTOMER data file. Then double click on the CUSTCODE field. The value CUSTCODE{3} will be placed in the DataField property. The {3} specifies that the field is three characters in size when displayed.

**NOTE:** You can also enter your own field names. Case doesn't matter (upper or lower). As long as the names you enter here match the values in the record or elsewhere in your program. Whatever you put here will be used when running the report. You can even use arrays, so a legal DataField property value might be test[1] or fld\_lname[cntr], etc.

The reason the program keeps the display size is to help you estimate the width you need when the report runs. To set the default object width, double click on the Width property value. The object will change in size to approximately 0.275. This might be a little narrow since the field will be all upper case characters, so increase the Width to 0.33.

Now add the following fields to the report. The Top property value is provided, since the other fields will be placed under one another. Make sure the Detail band has enough vertical room for the objects. If you need some more space, just click and hold on the Detail band name bar and pull it down a little. It should be at least 1" in height or more. You will shorten it up when you're done.

**TipDBText: DataField=CUSTNAME Left=3.5 Top=0 Width=2.375**

**TipDBText: DataField=CUSTCOMP Left=3.5 Top=0.19 Width=2.375**

**TipDBText: DataField=CUSTADDR Left=3.5 Top=0.39 Width=2.375**

**TipDBText: DataField=CUSTCITY Left=3.5 Top=0.6 Width=2.375**

**TipDBText: DataField=CUSTSTATE Left=3.5 Top=0.8 Width=0.21**

**TipDBText: DataField=CUSTZIP Left=3.8 Top=0.8 Width=0.95**

**TipDBText: DataField=CUSTAREA Left=6.1 Top=0 Width=0.29**

**TipDBText: DataField=CUSTPHONE Left=6.5 Top=0 Width=0.76**

Move the Detail band name bar up so that it's directly under the last text field you put on the form (State & Zip). If you click on the band, you should see the Height value as about 0.9687. Notice that the PrintHeight property has been set to phStatic (the default value). If you add to the Height value, there will be a space between the blocks of data when it prints. If there is no space, then there will be no space between the records when they print.

There is no need for a Footer, so turn it off by clicking on the Report->Footer menu item. The check mark next to the name will be removed and the Footer band will also.

Click on File->Save. The standard save dialog will pop up. Enter SimpleReport for the file name and click on the SAVE button.

Next you need to add the code to your program to link in the report format. Click on the Addsum TAS Premier button in your task bar to activate the screen/source editor. If the screen SOENTRY.DFM is not loaded, load it now.

## Addsum TAS Premier 7i Tutorial

Add a Button to the screen with these properties: **Left=376, Top=200, Width=75, Caption=Print, Name=btnPrint**. The button should be placed between the navigator and the exit button. Save the form.

Load the source file (SOENTRY.SRC) if it isn't already loaded. Add the following line immediately after the #WinForm line:

```
#WinReport SimpleReport
```

This has the same effect as the #WinForm compiler directive. It will load the report form during compilation and will make sure that all of the fields used in the report will be part of this program, even if they aren't used anywhere else.

At the bottom of the current code, add the following:

```
btnPrint.click:
  setup_report_buff rb_num 1 reportname 'simplereport'
  scan @cust_hndl key custcode
  output_report_data rb_num 1
ends
print_report
ret
```

The first line is the familiar standard event label. The second line tells your program to load the report form and setup the first buffer.

**NOTE:** You must always set up buffer 1 first and specify the report name as part of that command.

The third and fifth lines (SCAN and ENDS) scroll through the customer file in customer code order. The line in the middle (output\_report\_data) creates the buffer records that will be used to print the actual report. The fields you have placed on the report form will be accessed to create the buffer records. The PRINT\_REPORT command will do the actual printing. The last line, of course, is the standard RET from all event routines.

You can easily change the records that appear in the report by modifying the SCAN command. Restrict the records printed or change the order in which the records appear by changing the KEY value.

Compile the program by clicking on the Compile button or by pressing the F9 key. After the program compiles (assuming it compiled properly), click on Run Program. Click on the Print button (if the button doesn't show up on the form make sure you've saved it).

To print the report to a printer, click on the printer icon (far left). The next three icons and the percentage entry field control how large the preview is on your screen and how much of the page you see. The second entry field that is surrounded by navigator type buttons controls which page is displayed. You can enter the page number directly or use the first page, previous page, next page or last page buttons. The Close button exits the print preview.

## PART 5 – ADDING A MENU TO AN ADDSUM TAS Premier PROGRAM

It's very easy to add the standard Windows type menu to a form. Reload the SOENTRY.DFM file in the screen editor if it's not already loaded.

From the Standard palette tab drop a MainMenu object on the form. This is a non-visual object and will simply place the icon on the form. Click on the ellipses next to the Items property in the object inspector. Click on the button that looks like a plus sign (hint is Add Item). A MenuItem will be placed in the tree and will appear at the top of the form.

**NOTE:** When you start adding menu items, the screen may exceed the size you have allowed and horizontal/vertical scroll bars will appear. Stretch the screen out from the lower right corner and the scroll bars will disappear.

Click on the item (in the tree box) and you can change the Caption property directly. Enter "&File" as the new caption. The "&" (ampersand) before the "F" in File tells Windows to allow the user to enter ALT-F to choose this menu item. Press the ENTER key and File should replace MenuItem at the top of the form. This is a standard first file menu in any Windows program.

To add an additional top level menu item, click on the Add Item button again. To add a sub-level item, one that won't appear until the user clicks on the top level item, click on button to the right of the Add Item (it looks like an arrow over the + symbol). Once again, click on the tree item that says MenuItem and change the caption to "&Print" and the Name property value to mniPrint (menu item Print). Now exit from the property editor (click on the X in the upper right corner). Save the form by clicking on the Save button.

**NOTE:** If you click on the File menu item, at the top of the form, the Print option should appear just like a standard drop down menu.

Reload the source file (SOENTRY.SRC) if it isn't already loaded. Add the event label "mniPrint.Click:" directly after or before the existing event label "btnPrint.Click:". This will allow the user to either choose the item from the menu or simply click the Print button.

**NOTE:** You cannot have two objects on the same screen with the same name. This means that you must have multiple event labels if they both/all call the same routine.

Compile the program and run it. When you click on File and then choose the Print option, the report should print just like it does when you click on the Print button.

The only difference between this simple example and more complex menus is the number of MenuItems and the event labels you add to your program.

This completes the tutorial. See the directory COMPLETED under TUTORIAL for the completed tutorial. To look at other sample programs, see the SAMPLES directory as well as the SRC files in the TAS 7 installation directory.