Technical Appendix: Using Company Accounts data from Datastream[1]

Nick Bloom[2], Alexander Klemm[2], Rain Newton-Smith[3], and Gertjan Vlieghe[3]

June 2004

Abstract

This note provides detailed information on the construction of UK firm level panel data sets using Datastream accounting data. It describes how to perform automated downloading of large datasets and how to transfer data into Stata format. Then it suggests basic procedures for cleaning the data. Finally, it explains how to calculate a measure of the capital stock from company accounts. The appendix contains all programme files described.

Keywords: Datastream, Stata, Capital stock, Panel Data

Acknowledgements: This work has been financed by the Bank of England and the programme of research at the ESRC centre for the Micro-Economic Analysis of Fiscal Policy at the IFS. Financial support of the ESRC is gratefully acknowledged.

Correspondence: a.klemm@ifs.org.uk; 7 Ridgmount Street, London WC1E 7AE, UK.

[1] This paper forms a technical appendix to Bond et al (2004), 'The roles of expected profitability, Tobin's Q and cash flow in econometric models of company investment', Bank of England Working Paper no. 222.

[2] Institute for Fiscal Studies.

[3] Bank of England.

Working with data from Datastream (DS)[4]

This paper aims to provide a brief manual for the use of Datastream (DS) data on company accounts. It describes how to perform automated downloading of large datasets, how to transfer data into Stata format and suggests a basic cleaning procedure. Finally, it explains how to calculate a measure of the capital stock from company accounts.

1. Getting data from DS

Data can be downloaded from DS either manually or using a macro programme to download several companies at the same time automatically. Section 1.1 describes the different ways in which data can be downloaded manually and Section 1.2 describes several macro programmes.

1.1 Manual downloading of data

DS data is normally accessed with the help of commands. The three most common commands used to extract data are the following: 900A, which downloads time invariant data, 900B, which downloads time variant data and 900c, which downloads small panels of data. For an exhaustive list of commands please refer to the current Datastream manual. Note, however, that which commands can be run depends upon the contract with Datastream.

A manual download is started by typing in one of these commands. Every command will bring up a different screen. The following describes the three commands mentioned in more detail:

900A This command downloads company time invariant data, such as a firm's name. First, you need to type a company number – e.g. "900479" for GlaxoSmithKline. Then you are asked for an item – type "name", for example. When prompted to enter a date, just leave it blank, as the information is time-invariant.[5]

900B This command downloads company data that is time variant, such as the share price. At the "code or expression" prompt, a combination of company name and data type needs to be typed in. For example typing "900479(P)" gives GlaxoSmithKline's share price, and "900479(MV)" gives this firm's market value. The default if you just type in "900479" is the share price. Next enter the start date – e.g. 1/1/90, and then the end date, e.g. 1/1/00. Then enter the frequency, anything from D (daily) to Y (yearly). The amount of data that can be displayed is limited. If your choice is rejected, try asking for a shorter time span. For example, the maximum time span for daily data is about ten years.

900C This command downloads a panel of company accounts data, but it is limited to about 10 companies over ten years. This is the most complex of the downloading options. First, a format code needs to be specified. A format code is a short programme created by the user that contains a list of variables that the user wishes to download. Programme 190X (defined below) can be used to create and edit such format codes. Programme 190Y can be used to

[4] Please note that in order to access these data via the macros provided in this paper, you must have a current subscription agreement for Thomson Financial's DataStream and IBES services. Your access to and use of the data is subject to the terms of that subscription agreement. To the fullest extent permitted by law, Thomson Financial Limited on behalf of itself and its affiliated companies disclaims all liability in relation to the use of the macros and instructions contained in this paper.

[5] Strictly speaking, the information may not be time invariant. A company's name for example may change. In that case the command will provide the most up to date information, i.e. the company's current name.

review pre-existing format codes. Once a format code has been created it can be used until it is overwritten. To access the data, type the format code required ("F###"). Second, type the Datastream code (Dscode) for the company you want e.g. "900479", then type the desired time span, e.g. 1/1/90 and 1/1/00 for start and end date. Third, press return in response to annualise.

190X This programme is used for creating and editing a list of variables you wish to download. The result is a format code. If you wish to edit an existing format code, type A (for amend) and the format code number, "F001" for example. First of all, you should ensure that you are not unintentionally over-writing a pre-existing format code (see Programme 190Y below). The set of variables in F001 will then be listed on the screen and you can add or delete variables. If you press return and then enter in a company number such as "900479" in the first line and press return; you will obtain an example output for those data items. Press yes when asked to amend the format code in order to confirm your selection.

190Y This programme is used to review existing format codes. This can be useful if you or any of your colleagues have previously created format codes since it gives a list of existing format code numbers. This can be used to ensure that you are not unintentionally overwriting them.

The output from the three programmes (900A, 900B or 900C) can be saved. To save manually you need to click on file then either "capture to plain text" or "convert data channel". In order to facilitate the transferral to a statistical package such as Stata, the best way to save the data is as a text or CSV file ('comma-separated').

1.2 Automated Downloading of Mass Data sets

A more flexible and robust way of downloading data is to use a macro programme.

We provide three variants of "mac" files – or DS macro programmes – to automate the process. We have named them to correspond to the particular downloading procedure they aim to automate. The general guide to these files is the "DSWindows Macros" user guide.

900A.mac This downloads time invariant data – e.g. company names

900B.mac This downloads time variant company level data – e.g. company daily share prices

900C.mac This downloads time variant company data for more than 1 company

The 900C.mac programme is the key programme for downloading large panel data sets, and we therefore describe its workings in detail.

The 900C.mac file begins by configuring some settings, e.g. about the format in which the data are to be saved. "Configuredc (separator: "[TAB]")" for instance means that the data are to be separated by tab stops (as opposed to say commas or blanks).

Then a file which contains a list of DScodes (each company has a unique Datastream code of six digits which identifies it) is opened. This file should contain a list of Dscodes for the companies you wish to download accounts for. An example of such a list is provided in the appendix. If you already have a list of DS codes, all you need to do is format it as in the

example and change the path in the 900c.mac file. If you do not have a list of DS codes, you can obtain one from DS.[6]

When the programme is run for the first time, the path after "opendata" should refer to the complete list of companies. In our example this list is called "list.lst". If the programme crashes (which may happen, e.g. due to connection failures), then it would be a waste of time to repeat the programme for those firms that have already been downloaded. Therefore, the list of DS codes should be manually edited by deleting all of the DS codes for the firms for which data has already been downloaded and saved. The new, shorter list can be saved under a different name, temp.lst2, for example. To read from the new list, the OPENDATA line using the complete list needs to be commented out (using a semicolon at the start of the line) and instead the OPENDATA line using the temporary list should be used.

The macro uses a loop to download the same set of variables (Format code F079 in our example) for each firm. Format code F079 was created using programme 190X described in section 1.1. It contains a list of company account items that we are interested in, eg sales, inventories, cash balances, investment etc. The macro will download these variables at two points in every year from 1968 to the most recently available for each firm. The data are then saved in CSV format for each company. We request data at two points in any given year because some companies change the date of their accounting years. In this case, the information in May may be different from the information in October. Downloading twice ensures all data are captured.

Data for each company is saved separately because this is safer in case of an interruption (either intentional or unintentional). The data are also downloaded year by year because experience has shown this to be more stable than downloading time series several years at a time.

Once the data for a firm have been downloaded, they will be saved as "*.csv" (where * is the DS code for that company). This process is repeated for each firm. Hence, if there are 4000 firms there will be 4000 "*.csv" files. We recommend saving these in a separate folder (e.g. called "csvfiles"). This has two advantages: it makes it easier to find files later, and the computer opens up the folder more quickly.

The 900a.mac and 900b.mac work in a similar way, but are less complicated and are therefore not explained in detail here.

2. Transfer of data to Stata format

2.1 Overview:

Once the data are downloaded and saved in CSV files, they need to be transferred into Stata format. This is not just a matter of transferring data from one format into the other. The more demanding part of this step is to organise the data into variables and observations. This is done using a Stata do file.

For each type of downloaded data (i.e. the result of each *.mac file), there is an associated *.do file, that transfers the data to Stata format. For example, the programme 900b.mac downloads daily market values and saves them in CSV format. The 900b.do file then reads the CSV files, rearranges some rows and columns, and saves the data in Stata format.

[6] This can be done with the 900a command. In DS type 900a and then a list. FBRIT for example will give you all existing UK quoted companies. These lists will not include companies that no longer exist owing to things such as bankruptcies or mergers). If you want a large list, you may wish to consider downloading several lists and the merging them together.

2.2 An example: 900c.do

Each of the Stata *.do files work slightly differently, since the Datastream output is always in a different format. As with the *.mac files, we describe one in detail, the 900c.do file. The other do files are less complex.

The 900c.do file contains the following two programme, Readdata and Appendall:

Readdata will read the data in CSV format, reshape the data and save it in Stata format. For each firm a separate file, called "dscode.dta" is saved.

Appendall takes up all the firm data sets and merges them into a single set called 900c.dta.

The remainder of the file deals with purely aesthetic issues, such as labelling and ordering variables. So the Stata do file 900c.do creates a Stata data file for each company named after the company's dscode and one large Stata data file, 900c.dta, with all the company accounts together. Another approach would be to simply have one programme that immediately saves all data into a single dataset. But our method has two advantages. First, it is easier to detect mistakes, as you can compare a *.csv file directly with its corresponding Stata *.dta file. Second, it is easier to add companies. In that case readdata, a comparatively slow programme, only needs to be rerun for the new companies. Appendall, which is a comparatively fast programme, can then be used to merge all companies (old and new) into a single data set

Both programmes take as their argument the list of DS codes (**list.lst**) that was already used above in the 900c.mac programme. This list does not need to be formatted or transferred, as this is done automatically in the programmes. You can also use **Readdata** with the shorter temp.lst lists, e.g. after a mistake occurred. **Appendall** should always be run with the complete list in order to make sure that no company is missing or double in the final dataset.

It is not really necessary to understand the details in order to run the programme, but it may be helpful if you need to alter the programme. For instance, if a problem appears or if something new needs to be added.

2.2.1 Readdata

The programme starts by reading in the list of DS codes. This is done using the "insheet" command. Because the file contains some information that is only relevant to Datastream, some observations need to be dropped (the first two and the last). After renaming the variable and sorting, this list is then saved in Stata format. Since it is of no particular interest, this can be saved in a temporary folder.

The next step is to define the loop. Then the *.csv files are read in one by one. This is done with the "insheet" command. The advantage of this command over alternatives such as "infix" is that it recognises tabs or commas, and hence can flexibly deal with differences in spacing and different numbers of variables and/or observations.
In order to find out whether there are any data in a particular *.csv file, (some csv files will be empty if a company no longer exists on DS) the second observation of the data is listed. We choose to list the second observation because sometimes Stata will read an empty dataset as a variable, v1, which contains one missing observation. If data are available, then there will be either three or four variables. Normally there will be three variables. The first mostly contains the DS numbers of the account items, the second contains the names of the account items, and

the third contains the data. There is only a forth variable if a change in the accounting year end date occurs. This case is dealt with in section 2.2.1a. When there are only three variables, the process is straightforward: Some minor operations are perfomed, such as the construction of a variable called "dscode" that contains the DS company identifier. Then the data are reshaped so that they become yearly, with the DS account items as variables. After this the data are saved in Stata format as "*.dta". Then the programme returns to the beginning of the loop and opens the next "*.csv" file.

2.2.1a Dealing with a change in accounting years

Sometimes firms change their accounting years. In this case, there will be two accounting year-ends in the same calendar year. For most firms, the *.csv file contains three variables. But if there is a change in the accounting year, that company's *.csv file will contain a fourth variable. The fourth variable contains data that are from the same year as the data in the third variable, but at a later date. For example, if a firm changes its accounting year-end from March 1st to July 1st, then the third variable will contain data from March and the fourth will contain data from July. This happens for about 4.5% of the firms. Later, during the cleaning procedures, we will need to decide how do deal with these companies. Section 3 below describes how in practice, we drop the information contained in the "short year" e.g. March to July and then rename the company with the new July year-end as a new company. But at this stage, we simply want to ensure we keep all the data we have downloaded. Readdata.do, therefore, deals with such cases in the following way. First it establishes whether or not there is a fourth variable (or in other words whether there is an accounting year change). Then a dummy variable called "ind" is created which is equa1 to 1 if there is a fourth variable, or equal to 0 otherwise. If there is none, then all of the following steps are skipped.

If there is a fourth variable, then the data are temporarily saved. The third variable is deleted and data from the fourth variable are used instead. These data are then reshaped and saved following the same procedures as before. The reshaped data from the fourth variable are then saved in temporary folder. The programme then returns to the previously saved results. Now the fourth variable is deleted and the programme continues, as it would have without a fourth variable. Finally, the data from the fourth variable that were saved in a temporary folder are appended to data. These are saved in Stata format as before. The only difference to the simpler case is that now there can be two observations per year.

When all is done a message "finished successfully" is displayed and the folder containing Stata files will be fairly full (around 3500 data files).

2.2.2 Appendall :

This is a comparatively straightforward programme that simply appends all individual datasets into one large panel dataset. Note that this programme again reads in the list of DS company codes (**list.lst**). This is done to ensure that you can run it separately from the first programme. Otherwise we could have used the list saved as temp1.dta.

3. Merging data from different datastream commands

Often data from the three different types of Datastream downloads need to be merged into a single dataset. A researcher may, for example, wish to combine company accounts data (from 900c) with share price data (from 900b) and time-invariant data from (900a). Time invariant data can easily be merged in Stata with the usual Stata merge command (For example: merge dscode using filepath/filename). Merging daily 900b data with yearly 900c data, however, is more challenging. In order to do that, the daily share price data need to be annualised first. This can be done in several ways: either by taking the share price on the day of the accounting year-end, or by taking an average. The average could be, for example, over the three months

preceding the accounting year-end, or over the whole accounting period, or any other suitable period. The merging needs to be done on a company-by-company basis, because merging daily data for all 4000 companies would create a data set that would be too big for most computers to handle.

An example of a programme in which share price data are merged with yearly accounts data is **merge.do**. A copy is provided in the appendix. The following is a detailed description of how **merge.do** works. First, it opens the list of DS codes and then works on a company-by-company basis, a process familiar from the **900c.do** file. Then the dates are determined on which the averages are to be calculated. To save time, averages are only calculated on accounting period end dates rather than for every day in the year. Once a temporary list of these dates has been saved (as **tempdate.dta** in the example), the file containing daily market values is opened. Then a number of averages are calculated: over the month, over the quarter, and over the year preceding the accounting year-end. Furthermore the standard deviation of the share price over the year is calculated. The share price on the day is also kept, even though this information is also contained in the yearly account data. Keeping it will enable us to double-check the merging procedure. It may be that there is no market value for the exact year-end because the year-end is a bank holiday. In that case, the previous data point is used.

Then this information on the firm's share price is temporarily saved (**tempMV.dta**). Next, the dataset containing yearly company accounts data for this firm is opened and merged with its market value data (saved in **tempMV.dta**). Once this procedure has been completed for all firms, another programme within **merge.do** puts all the separate companies together into one dataset, just as the **Appendall** programme did in the 900c.do file. Finally some aesthetic issues are dealt with, such as labelling of DS variables.

4. Cleaning the data

General Issues

We now have a complete set of Datastream company accounts panel data (**900c.dta**), which needs to be cleaned. If, however, you plan to merge DS data with data from other sources, it is best to merge the data first, and do the cleaning once the data set is complete.

Cleaning refers to a systematic examination of the data. The aim is to replace or drop observations that are believed to be wrong or are misleading for any other reason. Such a procedure is necessary, because unfortunately the data can contain mistakes and inconsistencies. These range from simple typographical errors to more complicated issues such as breaks in company time series due to mergers. If the data are not cleaned, then outliers can have a strong influence on any subsequent regression results.

Much of the cleaning is inevitably subjective. If a company's sales rise tenfold, for example, then this would probably, but not necessarily, reflect a typographical error. It is inevitable that some observations will be dropped that should have been kept, while other mistakes may remain undetected. If a cleaned data set is used, it is advisable to check the sensitivity of any results to the cleaning procedure.

It should also be noted that the suggested cleaning is in no way exhaustive. Depending on the subsequent use of the data, more cleaning may well be desirable. If a certain variable is going to be studied, for example, then this variable should be cleaned first.

The Clean.do file

The first thing to do is to determine the length of the accounting year. This is assumed to be the difference between two accounting dates. Whenever this difference is missing (which will happen for the first year of data for each company), it is assumed to be 365 days. Any observation with an accounting year that is more than 30 days longer or shorter than 365 days is dropped. This will affect many of the cases in which accounting years changed. It also deals with companies that do not report profits regularly. Note: This must be the first step in the cleaning procedure because once variables are dropped, the difference between two dates for any given variable will no longer give the length of an accounting year.

The next step is to drop any company years for which we do not have core data. This is done not only because we need these items, but also because a firm that lacks these core items will probably have unreliable data elsewhere. The core data we require to exist in every observation are sales and cash flow.

The remaining operations are more subjective. The next thing is to get rid of large outliers, because they are likely to be cause by unusual firms (e.g. the oil extraction industry) or typographical errors. For convenience this is done with the help of a programme, which we named trimdrop. This programme drops any observation that has increased (decreased) by more than a specified percentage. The percentage is specified with the global command. Typing global g=200, for example, sets the maximum percentage by which a variable can increase from one year to the next to 200%. The corresponding drop in percentage terms (-67% in this case) is automatically calculated. If one of the two observations is missing (in the first year accounts for each company for example), then the programme does not drop the observation. The format for this programme is simply "trimdrop" followed by the variables to be trimmed. The percentage can be specified at a different level for each variable if required, but this is not done in our example, so as not to introduce further subjectivity.

One issue that arises in the cleaning file is that each time an observation is dropped, the difference between two accounting periods may no longer be one year. To deal with this problem, a programme called "series" is defined at the top of the clean.do file. This programme detects such cases, and deals with them by constructing new DS codes. Whenever a year is missing, the years before and after the missing year are assigned a new DS code, which will be ten times the original DS code plus a number from 1 to 9. It is important to run this programme before any lagged variables are constructed or referred to.

Finally, we drop any companies of which we have less than four years of uninterrupted data.

After cleaning we address another issue, the assignment of a year variable. One possibility is to define the year variable as the year of the account date. We have chosen instead to count any account date that lies in or before May as the previous year. Whatever cut-off date is chosen, there may be problems with firms that occasionally report just before and occasionally just after this date. For example, a company label its account year as 31st of December one year and in the next labelled it as the 1st of January. The result is that a year variable defined as the calendar year will seem to have jumped by two years between these two observations, even though in reality only one year (and one day) has passed. To get around this, a few lines of code assign the previous year to such cases. Checking for firms that have accounting year-end in the month preceding and following the cut-off date allows for this. In this case, all observations in the month following the cut-off date are assigned to the previous year.

This completes the data work. Figure 1 summarises the steps described so far. The last section discusses the calculation of a capital stock measure.

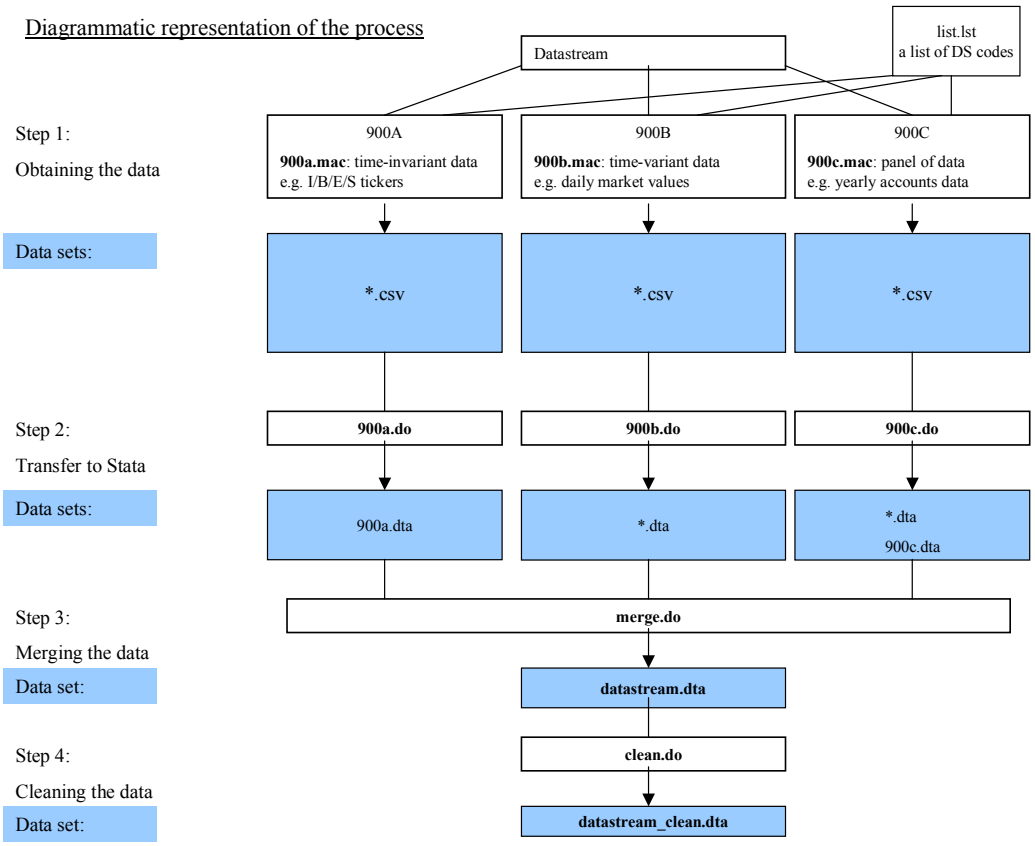

Figure 1

5. Capital Stock

The standard way to construct the capital stock valued at replacement cost for company accounts is to use the perpetual inventory method, similar to Blundell et al (1992) and Bond, Harhoff and Van Reenen (1999). The basic procedure of this method is to estimate a starting value of the capital stock, K_0, and then augment it each year with new investment, deduct depreciation and adjust for inflation. The data used for the inputs into the model such as the starting value of the capital stock, the investment series, the depreciation rate(s) and the deflators can vary depending upon what the capital stock will be used for and the availability of data. What follows is a basic procedure for calculating capital stock series from UK company accounts which can be modified to suit your purpose. The numbers in brackets refer to the relevant Datastream code.

We define investment as follows:

$$I = ds1026 + ds479$$

where ds1026 is net payments for fixed assets (where net means less sales of fixed assets) and ds479 is the net payments for fixed assets of subsidiaries. If ds1026 is not available, we define investment as:

$$I = ds431 - ds423 + ds479$$

where ds431 is purchases of fixed assets by the parent company, ds423 is sales of fixed assets by the parent company and ds479 is the net purchase of fixed assets of subsidiaries. We calculate investment in two ways since ds1026 replaces ds435 (total new fixed assets) after an

accounting change in 1990. This is why we calculate investment in two different ways. Users should note that if they are calculating an investment series prior to 1980, an alternative measure should be used for the period until 1990:

$$I = ds435 - ds423$$

where ds435 is total payments for fixed assets. Note that

$$ds435 = ds431 + ds432$$

where ds432 is purchase of fixed assets by subsidiaries. This is because prior to 1980, ds431 is not normally reported but ds435 is. After 1990, ds435 is replaced by ds1026.

Since companies report investment in nominal terms we then deflate investment using the quarterly business investment deflator implied by the UK National Accounts to create an investment series in constant (1995) prices.[7]

.

Next, we need an estimate of the initial capital stock (K_0) for each firm. As a general rule, we use the book value of fixed capital (ds339) in the first year of data for each firm (1968 where available). This can be modified to allow for inflation in the previous years, by increasing the first available book value by three years of inflation.

We can now estimate the evolution of the real capital stock as

$$K_t = (1-\delta)K_{t-1} + I_t$$

For the baseline estimate of the capital stock we use a depreciation rate (δ) of 8% for all capital goods. This is in line with Bond et al (1999).

We drop observations if the estimated capital stock is negative, or if our estimate out of line with book value by more than a factor of four.

For the ratios, which we will construct later, it will be necessary to define the nominal capital stock, which is simply $P_t^K K_t$. For the same reason, we also reflate the investment series to obtain a nominal investment series.

Quarterly price indices of various types are merged with the original dataset as follows: First, the quarterly price data (which is available from National Statistics), along with the appropriate date series, are transformed into a STATA dataset using STATA TRANSFER. Then a variable is created in the main dataset for the quarter that the company year-end belongs to. The quarterly price data is then merged using the quarter as the matching item.

[7] Source: ONS codes: NPEK/NPEL.

Appendix:

This appendix contains the files discussed or mentioned in the text. In order to run any of the programme, paths to the data need to adjusted and in some cases directories have to be created. Paths and programmes mentioned in the text are marked bold.

This example of a list of datastream codes shows the format required for the *.mac files to work. Note that DS codes can occasionally be alphanumeric.

list.lst

```
LIST:
DATA
"981397"
"981405"
"981406"
"981407"
"981418"
"981466"
"981482"
"981483"
"981486"
"981493"
"982089"
"983512"
"988567"
"988942"
"991000"
“13016J”
ENDDATA
```

The following programmes downloads a variable called “IBTKR”. This is the company identifier used by I/B/E/S and is useful if one wishes to merge the data with data from I/B/E/S. Of course, “ibtkr” can be replaced by another variable of interest.

900a. mac

```
ALLOWDUPLICATETIMESERIES (FALSE)
CONFIGUREDC (CODES900C: 1)
CONFIGUREDC (SEPARATOR: "[TAB]")
CONFIGUREDC (ENDOFLINE: "[CR][LF]")
CONFIGUREDC (QUOTECHAR: " ")
CONFIGUREDC (QUOTETEXT: 1)
CONFIGUREDC (QUOTENUMBERS: 0)
CONFIGUREDC (NOTAVAILABLE: "#N/A")

OPENDATA "m:\industry\datastream\900a\lists\list.lst":list

Loop:
IF &ENDOFDATA = FALSE THEN
INPUT dscd
SEND("[CLEAR]")
STARTDC (CSVFILE,"c:\csv_files\900a\d"+ dscd +".csv")
SEND ("900a," +dscd+ ",IBTKR,01/01/1990")
;SEND ("900A")
;SEND (" +dscd+ ")
;SEND ("IBTKR")
;SEND ("1/1/90")
SEND ("[CLEAR]")
ENDDC
GOTO Loop
ENDIF
END
```

This file downloads DS daily data. In the example it is market value data from 1960 onwards, but again this can be changed to any other variable of interest. Note that this file needs to be updated every year to download the newest available year.

900b.mac

```
send ("[CLEAR]")
CONFIGUREDC (QUOTECHAR: " ")
CONFIGUREDC (colHEADINGS900A: 0)

OpenData "c:\list.lst":LIST
Loop:
IF &ENDOFDATA = FALSE THEN
INPUT dscd
STARTDC (CSVFILE,"c:\csv_files\900b\MV\d"+ dscd +".csv")
Send(  "900B " + dscd + "(MV),1/1/60,31/12/60,D" )
send ("[CLEAR]")
Send(  "900B " + dscd + "(MV),1/1/61,31/12/61,D" )
send ("[CLEAR]")
Send(  "900B " + dscd + "(MV),1/1/62,31/12/62,D" )
send ("[CLEAR]")
Send(  "900B " + dscd + "(MV),1/1/63,31/12/63,D" )
send ("[CLEAR]")
Send(  "900B " + dscd + "(MV),1/1/64,31/12/64,D" )
send ("[CLEAR]")
Send(  "900B " + dscd + "(MV),1/1/65,31/12/65,D" )
send ("[CLEAR]")
Send(  "900B " + dscd + "(MV),1/1/66,31/12/66,D" )
send ("[CLEAR]")
Send(  "900B " + dscd + "(MV),1/1/67,31/12/67,D" )
send ("[CLEAR]")
Send(  "900B " + dscd + "(MV),1/1/68,31/12/68,D" )
send ("[CLEAR]")
Send(  "900B " + dscd + "(MV),1/1/69,31/12/69,D" )
send ("[CLEAR]")
Send(  "900B " + dscd + "(MV),1/1/70,31/12/70,D" )
send ("[CLEAR]")
Send(  "900B " + dscd + "(MV),1/1/71,31/12/71,D" )
send ("[CLEAR]")
Send(  "900B " + dscd + "(MV),1/1/72,31/12/72,D" )
send ("[CLEAR]")
Send(  "900B " + dscd + "(MV),1/1/73,31/12/73,D" )
send ("[CLEAR]")
Send(  "900B " + dscd + "(MV),1/1/74,31/12/74,D" )
send ("[CLEAR]")
Send(  "900B " + dscd + "(MV),1/1/75,31/12/75,D" )
send ("[CLEAR]")
Send(  "900B " + dscd + "(MV),1/1/76,31/12/76,D" )
send ("[CLEAR]")
Send(  "900B " + dscd + "(MV),1/1/77,31/12/77,D" )
send ("[CLEAR]")
Send(  "900B " + dscd + "(MV),1/1/78,31/12/78,D" )
send ("[CLEAR]")
Send(  "900B " + dscd + "(MV),1/1/79,31/12/79,D" )
send ("[CLEAR]")
Send(  "900B " + dscd + "(MV),1/1/80,31/12/80,D" )
send ("[CLEAR]")
Send(  "900B " + dscd + "(MV),1/1/81,31/12/81,D" )
send ("[CLEAR]")
Send(  "900B " + dscd + "(MV),1/1/82,31/12/82,D" )
send ("[CLEAR]")
Send(  "900B " + dscd + "(MV),1/1/83,31/12/83,D" )
send ("[CLEAR]")
Send(  "900B " + dscd + "(MV),1/1/84,31/12/84,D" )
send ("[CLEAR]")
Send(  "900B " + dscd + "(MV),1/1/85,31/12/85,D" )
```

```
send ("[CLEAR]")
Send(  "900B " + dscd + "(MV),1/1/86,31/12/86,D" )
send ("[CLEAR]")
Send(  "900B " + dscd + "(MV),1/1/87,31/12/87,D" )
send ("[CLEAR]")
Send(  "900B " + dscd + "(MV),1/1/88,31/12/88,D" )
send ("[CLEAR]")
Send(  "900B " + dscd + "(MV),1/1/89,31/12/89,D" )
send ("[CLEAR]")
Send(  "900B " + dscd + "(MV),1/1/90,31/12/90,D" )
send ("[CLEAR]")
Send(  "900B " + dscd + "(MV),1/1/91,31/12/91,D" )
send ("[CLEAR]")
Send(  "900B " + dscd + "(MV),1/1/92,31/12/92,D" )
send ("[CLEAR]")
Send(  "900B " + dscd + "(MV),1/1/93,31/12/93,D" )
send ("[CLEAR]")
Send(  "900B " + dscd + "(MV),1/1/94,31/12/94,D" )
send ("[CLEAR]")
Send(  "900B " + dscd + "(MV),1/1/95,31/12/95,D" )
send ("[CLEAR]")
Send(  "900B " + dscd + "(MV),1/1/96,31/12/96,D" )
send ("[CLEAR]")
Send(  "900B " + dscd + "(MV),1/1/97,31/12/97,D" )
send ("[CLEAR]")
Send(  "900B " + dscd + "(MV),1/1/98,31/12/98,D" )
send ("[CLEAR]")
Send(  "900B " + dscd + "(MV),1/1/99,31/12/99,D" )
send ("[CLEAR]")
Send(  "900B " + dscd + "(MV),1/1/00,31/12/00,D" )
send ("[CLEAR]")
Send(  "900B " + dscd + "(MV),1/1/01,31/12/01,D" )
send ("[CLEAR]")
ENDDC
GOTO LOOP
ENDIF
END
```

The following downloads a variables specified in format code **F079**.

900c.mac

```
;These commands controls the data channel (see macro handbook).
ALLOWDUPLICATETIMESERIES (FALSE)
CONFIGUREDC (CODES900C: 1)
CONFIGUREDC (SEPARATOR: "[TAB]")
CONFIGUREDC (ENDOFLINE: "[CR][LF]")
CONFIGUREDC (QUOTECHAR: " ")
CONFIGUREDC (QUOTETEXT: 1)
CONFIGUREDC (QUOTENUMBERS: 0)
CONFIGUREDC (NOTAVAILABLE: "#N/A")
ENDDC

;This command just clears to make sure nothing else is in running from earlier
SEND("[CLEAR]")

;this specifies the location of the list of datastream codes:
OPENDATA "c:\list.lst":list

;Once the programme has extracted data for some firms the rest are extracted using
;the shorter "temp.lst". This is changed after each crash.
;OPENDATA "c:\ temp.lst":list

;This starts the dataextraction loop.
Loop:
IF &ENDOFDATA = FALSE THEN
;This inputs line after line from our "*.lst" file. Each line is another DS code.
INPUT dscd
;This line defines what the output data file will be called - according to the DS code.
;and of course where it will be saved. This leads to many files, so choose somewhere
;with lots of capacity.
STARTDC (CSVFILE,"c:\csv_files\900c\d"+ dscd +".csv")
;These commands ask for data in format F079 for each year for our dscode
SEND ("900c,F079," +dscd+ ",5/5/68,10/10/68")
SEND ("900c,F079," +dscd+ ",5/5/69,10/10/69")
SEND ("900c,F079," +dscd+ ",5/5/70,10/10/70")
SEND ("900c,F079," +dscd+ ",5/5/71,10/10/71")
SEND ("900c,F079," +dscd+ ",5/5/72,10/10/72")
SEND ("900c,F079," +dscd+ ",5/5/73,10/10/73")
SEND ("900c,F079," +dscd+ ",5/5/74,10/10/74")
SEND ("900c,F079," +dscd+ ",5/5/75,10/10/75")
SEND ("900c,F079," +dscd+ ",5/5/76,10/10/76")
SEND ("900c,F079," +dscd+ ",5/5/77,10/10/77")
SEND ("900c,F079," +dscd+ ",5/5/78,10/10/78")
SEND ("900c,F079," +dscd+ ",5/5/79,10/10/79")
SEND ("900c,F079," +dscd+ ",5/5/80,10/10/80")
SEND ("900c,F079," +dscd+ ",5/5/81,10/10/81")
SEND ("900c,F079," +dscd+ ",5/5/82,10/10/82")
SEND ("900c,F079," +dscd+ ",5/5/83,10/10/83")
SEND ("900c,F079," +dscd+ ",5/5/84,10/10/84")
SEND ("900c,F079," +dscd+ ",5/5/85,10/10/85")
SEND ("900c,F079," +dscd+ ",5/5/86,10/10/86")
SEND ("900c,F079," +dscd+ ",5/5/87,10/10/87")
SEND ("900c,F079," +dscd+ ",5/5/88,10/10/88")
SEND ("900c,F079," +dscd+ ",5/5/89,10/10/89")
SEND ("900c,F079," +dscd+ ",5/5/90,10/10/90")
SEND ("900c,F079," +dscd+ ",5/5/91,10/10/91")
SEND ("900c,F079," +dscd+ ",5/5/92,10/10/92")
SEND ("900c,F079," +dscd+ ",5/5/93,10/10/93")
SEND ("900c,F079," +dscd+ ",5/5/94,10/10/94")
SEND ("900c,F079," +dscd+ ",5/5/95,10/10/95")
SEND ("900c,F079," +dscd+ ",5/5/96,10/10/96")
SEND ("900c,F079," +dscd+ ",5/5/97,10/10/97")
```

```
SEND ("900c,F079," +dscd+ ",5/5/98,10/10/98")
SEND ("900c,F079," +dscd+ ",5/5/99,10/10/99")
SEND ("900c,F079," +dscd+ ",5/5/00,10/10/00")
SEND ("900c,F079," +dscd+ ",5/5/01,10/10/01")
;this will need to be updated each year to get extra data - i.e. add in 02 in 2002

;This just clears at the end of the loop to stop crashing
SEND("[CLEAR]")
;This closes each dscodes data file
ENDDC
GOTO Loop
ENDIF
```

This converts the time invariant data form **900a.mac** to Stat format, The resulting data are saved in a single data set called "**900a.dta**".

900a.do

```
cap log close
set more off
clear
set memory 60000
set matsize 800
cd c:\

insheet using list.lst,clear
drop in 1/2
drop in l
ren v1 code
keep if code~=""
global num=[_N]
global c = 1
sa temp\temp, replace

while $c<=$num {
  u temp\temp,clear
  gsort - code
  global compcode = code[$c]
  cap insheet using csv_files\900a\"d$compcode.csv",clear names
  if _rc ~= 0 { di "data set empty: d$compcode" }
  if _rc == 0 {
  di "d$compcode"
  cap g str6 dscode = string(date)
  cap ren date dscode
  cap drop v3
  append using temp\temp
  sa temp\temp, replace }
  global c = $c+1 }

drop code*

ren v2 ticker
replace ticker = "" if ticker =="#N/A"
replace ticker = substr(ticker,3,6)
keep dscode ticker
drop if dscode==""
sa 900a.dta,replace
```

This file transfers the data downloaded by **900b.mac** into Stat format. Unlike in the case of **900a**, here the files are saved company-by-company rather than in a single dataset. This is to avoid dealing with a very large dataset (as it is daily data), which could cause problems on computers with limited RAM.

900b.do

```
cap log close
set more off
clear
set memory 60000
set matsize 800
cd c:\

***Read in Csv files and save in Stata format
insheet using list.lst,clear
drop in 1/2
drop in l
ren v1 code
global num=[_N]
global c = 1
so code
sa temp\tempMV,replace

while $c<=$num {
  u temp\tempMV, clear
  global compcode = code[$c]
  clear
  cap insheet using csv_files\900b\d$compcode.csv
  cap l in 2
  if _rc == 198 { di "d$compcode - data set empty" }
  if _rc==0 {
    di "d$compcode"
    drop if v1=="Name" | v1=="Currency"
    g str10 dscode = v2 if v1=="Code"
    replace dscode = substr(dscode,1,6)
    replace dscode = dscode[_n-1] if dscode[_n-1]~=""
    drop if v1=="Code"
    g date = date(v1,"dmy",2020)
    format date %d
    g MV = real(v2)
    drop if MV==.
    drop v1 v2
    so dscode date
    sa stata_files\900b\d$compcode,replace }
  global c = $c+1 }
```

This file transfers the data downloaded by **900c.mac** to Stata format. Data are saved both company-by-company and in single data set.

900c.do

```
set more off
clear
set memory 80000
cap log close
cd c:\
*********************************************************************************************
*This is the programme that reads in csv data, transforms it to Stata format
*and saves the data in individual datasets

cap prog drop readdata
prog def readdata
  *first the list of dscodes is needed (this is the argument when using the readdata programme)
  insheet using ${_1}
  drop in 1/2
  drop in l
  ren v1 code
  so code
  sa  temp\temp1,replace

  *This is the loop that transforms the files one by one and saves them
  global num=[_N]
  global c = 1
  while $c<=$num {
    u temp\temp1,clear
    global compcode = code[$c]
    clear
    cap insheet using csv_files\900c\d$compcode.csv,tab

    *this deals with empty data sets
    cap l in 2
    if _rc == 198 {di "d$compcode data set empty" }

    if _rc == 0 {
      di "d$compcode data available"

      *this creates a variable containing the DS code
      g str6 dscode =v1 in 1
      replace dscode = dscode[_n-1] if dscode==""

      *this creates a variable containing the company name
      g str80 name = v2[1]
      drop v2

      *this creates the date
      g str8 tempdate = v3 if v1=="TYPE"
      g date = date(tempdate, "dmy", 2020)
      format date %d
      replace date = date[_n-1] if date==.

      *This deals with cases where there is a forth variable (i.e. change in accounting year)
      g tempind = .
      cap g tempind2 = real(v4)
      cap replace tempind=tempind2
      egen ind = sum(tempind)
      replace ind = 1 if ind~=0
      if ind~=0 in 1 {
        sa temp\temp,replace
        keep if v4~=""
        drop tempdate date v3
        g str8 tempdate = v4 if v1=="TYPE"
        g date = date(tempdate, "dmy", 2020)
        format date %d
```

```
          replace date = date[_n-1] if date==.
          drop if v1=="TYPE"
          g ds = real(v4)
          drop if dscode == v1
          drop v4 temp*
          reshape wide ds, i(dscode date) j(v1) string
          sa \temp\d$compcode,replace
          u temp\temp,clear }

        *this is to tidy up
        drop if v1=="TYPE"
        g ds = real(v3)
        drop if dscode == v1
        drop v3 temp*
        cap drop v4
        reshape wide ds, i(dscode date) j(v1) string
        sa stata_files\900c\d$compcode,replace

        *this adds the data from any change in accounting years
        if ind~=0 in 1 {
          u temp\d$compcode,clear
          append using stata_files\900c\d$compcode
          sa  stata_files\900c\d$compcode, replace }
    }
    global c = $c+1 }
  di "Finished successfully"
end
*******************************************************************************

*******************************************************************************
*This puts all company data sets together
cap prog drop appendall
prog def appendall
  insheet using ${_1},clear
  drop in 1/2
  drop in l
  ren v1 code
  gsort - code
  global num=[_N]
  global c=1

  while $c<=$num {
    gsort - code
    global compcode = code[$c]
    cap append using stata_files\900c\d$compcode
    if _rc==0 {di "$compcode"}
    if _rc~=0 {di "$compcode - not available"}
    global c = $c+1 }

  drop if code~=""
  drop code
  sa 900c, replace
  di "Finished successfully"
end
*******************************************************************************

readdata  c:\list.lst
appendall c:\list.lst

label var ds104    "TOTAL SALES"
label var ds160    "CORPORATION TAX"
label var ds214    "EMPLOYEE REMUNERATN (DOMESTIC)"
label var ds315    "MINORITY INTERESTS"
label var ds336    "PLANT & MACHINERY-DEPN"
label var ds364    "TOTAL STOCK AND W.I.P."
label var ds423    "SALES OF FIXED ASSETS"
label var ds1026   "NET PAYMNT FOR FIXED ASSETS"
```

```
label var ds117   "TOTAL EMPLOYMENT COSTS"
label var ds164   "IRRECOVERABLE A.C.T."
label var ds215   "TOTAL EMPLOYEE REMUNERATN"
label var ds321   "TOTAL LOAN CAPITAL"
label var ds337   "OTH FIXED ASSETS-DEPN"
label var ds365   "FINISHED GOODS(O/W)"
label var ds429   "EQUITY & PREFERENCE ISSUES"
label var ds1035  "PAYMENTS: SUBS ETC."
label var ds119   "RESEARCH AND DEVT."
label var ds166   "TOTAL DOMESTIC TAX"
label var ds216   "NO. DOMESTIC EMPL. (UNITS)"
label var ds324   "LEASED ASSETS-GROSS(O/W)"
label var ds338   "TOT FIXED ASSETS-DEPN"
label var ds375   "TOTAL CASH & EQUIVALENT"
label var ds431   "FIXED ASSETS PURCHASED"
label var ds1036  "RECEIPTS: SUBS ETC."
label var ds135   "PROFIT BEFORE PROVS (ADJ)"
label var ds169   "TOTAL OVERSEAS TAX"
label var ds219   "TOTAL NO. OF EMPL. (UNITS)"
label var ds327   "TOTAL LAND & BLDGS-GROSS"
label var ds339   "TOT FIXED ASSETS-NET"
label var ds376   "TOTAL CURRENT ASSETS"
label var ds432   "FIXED ASSETS OF NEW SUBS."
label var ds1037  "NET PAYMENTS: SUBS ETC."
label var ds136   "DEPRECIATION"
label var ds172   "TOTAL TAX CHARGE-ADJ"
label var ds275   "BANK BORROWING > 1YR(O/W)"
label var ds328   "PLANT & MACHINERY-GROSS"
label var ds342   "RESEARCH & DEVMT."
label var ds387   "BANK BORROWING < 1YR(O/W)"
label var ds435   "TOTAL NEW FIXED ASSETS"
label var ds1038  "NET PYMNT: LOANS & ADVANCES"
label var ds137   "OPERATING PROFIT-ADJ"
label var ds175   "AFTER TAX PROFIT-ADJ"
label var ds305   "EQUITY CAP. AND RESERVES"
label var ds329   "OTH FIXED ASSETS -GROSS"
label var ds344   "TOTAL INTANGIBLES"
label var ds389   "TOTAL CURRENT LIABLITIES"
label var ds436   "RESEARCH & DEV.EXP."
label var ds1045  "CASH INFLOW FROM FINANCING"
label var ds143   "INTEREST INCOME"
label var ds181   "PREFERENCE DIVIDEND FOR PERIOD"
label var ds307   "TOT. SHARE CAPITAL & RESERVES"
label var ds330   "TOT FIXED ASSETS -GROSS"
label var ds356   "TOTAL INVESTMENTS (INC.ASS.)"
label var ds390   "NET CURRENT ASSETS"
label var ds479   "FIXED ASSETS (SUBS)"
label var dsMV    "MV"
label var ds153   "TOTAL INTEREST CHARGES"
label var ds182   "EARNED FOR ORDINARY-FULL TAX"
label var ds309   "BORROWINGS REPAYABLE < 1 YEAR"
label var ds331   "LEASED ASSETS-DEPN (O/W)"
label var ds359   "OTHER ASSETS"
label var ds406   "TOTAL EQUITY ISSUED"
label var ds666   "CONSTRUCTN IN PROGRESS"
label var ds155   "PRE-TAX PROFIT EXC ASSOCS-ADJ "
label var ds183   "NET E.P.S. FULL TAX"
label var ds312   "TOTAL DEFERRED & FUTURE TAX"
label var ds332   "LEASED ASSETS-NET (O/W)"
label var ds360   "STOCKS"
label var ds412   "EQUITY ISSUED FOR CASH"
label var ds1024  "PAYMENTS: FIXED ASSETS"
label var ds156   "ASSOC. PRE-TAX PROFITS"
label var ds187   "ORDINARY DIVIDENDS"
label var ds313   "TOTAL LT PROVN EXC DEF TAX "
label var ds335   "TOTAL LAND & BLDG-DEPN"
label var ds361   "WORK IN PROGRESS"
```

```
label var ds418    "CHANGE IN LOAN CAPITAL"
label var ds1025   "RECEIPTS: FIXED ASSETS"
label var ds113    "WAGES AND SALARIES"
label var ds993    "OPERATING PROFIT"
label var ds981    "ADJ TO OPERATING PROFIT"
order dscode name date dsMV ds104 ds113-ds993 ds1024-ds1038
so dscode date
sa 900c, replace
```

merge.do

```
clear
set more off
set memory 100000
set matsize 800
cap log close
cd c:\

insheet using list.lst,clear
drop in 1/2
drop in l
ren v1 dscode
so dscode

***so as to save time: only go through the process if we have accounts data:
merge dscode using 900c.dta
tab _merge
keep if _merge==3
keep if dscode~=""
keep dscode
so dscode
keep if dscode~=dscode[_n-1]
global num=[_N]
global c = 1
sa tempx,replace

while $c<=$num {
  u tempx, clear
  global compcode = dscode[$c]
  clear

  *determine the relevant dates
  u 900c.dta
  keep if dscode == "$compcode"
  so date
  keep if date~=date[_n-1] & date~=.
  keep date
  g dateno=_n
  so date
  sa tempdate, replace

  *Market value data
  clear
  cap u stata_files\900b\d$compcode
  cap l in 2
  if _rc == 198 { di "d$compcode - no RI (900b) data available" }
  if _rc == 0 {
  so date
  merge date using tempdate
  drop _merge
  g aMV1m = .
  g aMV3m = .
  g aMV1y = .
  g sdMV  = .
  egen tempmax = max(dateno)
  local m=tempmax[1]
  local f = 1
  while `f'<=`m' {
  g temp1 = date if dateno==`f'
  egen temp2 = min(temp1)
  gen tempdummyyear   = 1 if date <= temp2 & date > temp2 - 365
  gen tempdummy3month = 1 if date <= temp2 & date > temp2 - 91.5
  gen tempdummymonth  = 1 if date <= temp2 & date > temp2 - 30.5
  egen tempaMV1y = mean(MV), by(tempdummyyear)
  egen tempaMV3m = mean(MV), by(tempdummy3month)
  egen tempaMV1m = mean(MV), by(tempdummymonth)
```

```
   egen tempsdMV  = sd(MV)  , by(tempdummyyear)
   replace aMV1m = tempaMV1m if dateno==`f'
   replace aMV3m = tempaMV3m if dateno==`f'
   replace aMV1y = tempaMV1y if dateno==`f'
   replace sdMV  = tempsdMV  if dateno==`f'
   drop temp*
   local f = `f'+1 }
   so date
   replace MV = MV[_n-1] if MV==. & date-date[_n-1]<=7
   drop dateno dscode
   so date
   sa tempMV, replace }

   cap u 900c
   if _rc==0 {
   keep if dscode == "$compcode"
   so date
   cap merge date using tempMV
   cap drop _merge
   drop if dscode==""
   so dscode date
   sa firms\d$compcode, replace
   cap erase tempMV.dta }

   global c = $c+1  }

insheet using list.lst,clear
drop in 1/2
drop in l
ren v1 code
gsort - code
global num=[_N]
global c=1

while $c<=$num {
   gsort - code
   global compcode = code[$c]
   cap append using firms\d$compcode.dta
   if _rc==0 {di "$compcode"}
   if _rc~=0 {di "$compcode - not available"}
   global c = $c+1 }

drop if code~=""
drop code

* now just make it a bit nicer:
label var ds104     "TOTAL SALES"
label var ds160     "CORPORATION TAX"
label var ds214     "EMPLOYEE REMUNERATN (DOMESTIC)"
label var ds315     "MINORITY INTERESTS"
label var ds336     "PLANT & MACHINERY-DEPN"
label var ds364     "TOTAL STOCK AND W.I.P."
*etc.

so dscode date
sa c:\datastream, replace
```

clean.do

```
cap log close
set more off
clear
set memory 60000
set matsize 800


**********Programmes that are used in this file:
* This programme generates a different series number for each new run of consecutive data
* The dscode is replaced by dscode * 10 plus series number
* Note this programme needs to be run before any command that refers to a previous date (i.e. [_n-1])
cap prog drop series
prog def series
   cap gen str6 odscode=dscode
   cap label var odscode "Original DScode"
   bys odscode (date): g tempdays = date - date[_n-1]
   ge ok = 0
   replace ok = 1 if (tempdays>395|tempdays<335)
   ge series = 1
   replace series = 2 if ok==1
   bys odscode (date): replace series = 2  if series[_n-1]==2
   local x = 3
   while `x' <= 15  {
      by odscode (date): replace series = `x' if series[_n-1]==`x'-1 & ok==1
      by odscode (date): replace series = `x' if series[_n-1]==`x'
      local x = `x'+1 }
   g tempdscode      = real(odscode)*10 +series if series<=9
   replace tempdscode= real(odscode)*100+series if series>=10
   replace dscode = string(tempdscode)
   drop series tempdays ok tempdscode
end

cd c:\
log using clean.log, replace

*specify dataset to be cleaned:
u datastream,clear
*u 900c, clear

**********determine length of accounting year
*this needs to be done before any observation is dropped, otherwise don't get correct length of
accounting years
bys dscode (date): g days = date - date[_n-1]
replace days = 365 if days==.
label var days "Days in accounting year"
drop if days>395|days<335

**********dropping now if lacking core data
*Datastream:
drop if ds104==. | ds104==0
gen cash=(ds182+ds136)
drop if cash==.

**********Trimming:
*the following deals with unbelievable changes (up by g% or down by -1/(1+g) %)
*use only with core variables (because each time used you lose observations)


cap prog drop trimdrop

prog def trimdrop
   qui series
```

```
  while "${_1}"~="" {
    bys dscode (date):gen temp_${_1}=(${_1}-${_1}[_n-1])/${_1}[_n-1]
    su temp_${_1}, de
    drop if (temp_${_1}>=$g/100 | temp_${_1}<=-($g/100)/(1+($g/100))) & temp_${_1}~=. & ${_1}~=0
    drop temp*
    macro shift }
end

global g=200
trimdrop ds104 ds219 ds339

********make sure we have at least four observations per firm (or virtual firm)
qui series
cap drop noj
egen noj=count(date), by(dscode)
drop if noj<4

**********dealing with the year variable
cap g year = year(date)
*if _rc==0 {replace year = year-1 if month(date)<=5}

*sometimes we have two observations in a year, because accounts change from the 31/5 to the 1/6.
generat temp = 0.03 if month(date)==5
replace temp = 10   if month(date)==6
egen tempprob = sum(temp), by(dscode)
replace year = year -1 if month(date)==6 & tempprob>10 & tempprob-int(tempprob)~=0
*test whether this worked (if nothing listed, successful):
so dscode year
l dscode year date if (year==year[_n-1] & dscode==dscode[_n-1]) | (year==year[_n+1] &
dscode==dscode[_n+1])

/* this for year from 1 january to 31 december
*sometimes we have two observations in a year, because accounts change from the 31/5 to the 1/6.
generat temp = 0.03 if month(date)==12
replace temp = 10   if month(date)==1
egen tempprob = sum(temp), by(dscode)
replace year = year -1 if month(date)==1 & tempprob>10 & tempprob-int(tempprob)~=0
*test whether this worked (if nothing listed, successful):
so dscode year
l dscode year date if (year==year[_n-1] & dscode==dscode[_n-1]) | (year==year[_n+1] &
dscode==dscode[_n+1])      */

drop temp*
sa datastream_clean, replace

log close
set more on
```

References

More information on Datastream can be found at www.Datastream.com or by ringing their helpdesk on +44 207 778 0801.

Blundell, R., Bond, S., Devereux, M., Schiantarelli, F. (1992) "Investment and Tobin's Q – Evidence from company panel data" Journal of Econometrics 51

Bond, S, Harhoff, D, and Van Reenen, J, (1999) "Investment, R&D and financial constraints in Britain and Germany," IFS Working Papers W99/05, Institute for Fiscal Studies.

Stata Corporation (2001), Intercooled Stat 7.0 for Windows, Stata Corporation, College Station, TX, USA
___ (2001) Intercooled Stata 7.0 Reference Manual, Stata Corporation, College Station, TX, USA