Table of Contents

List of Tables

3. Sources and Drivers

3.1 All about Sources and Drivers

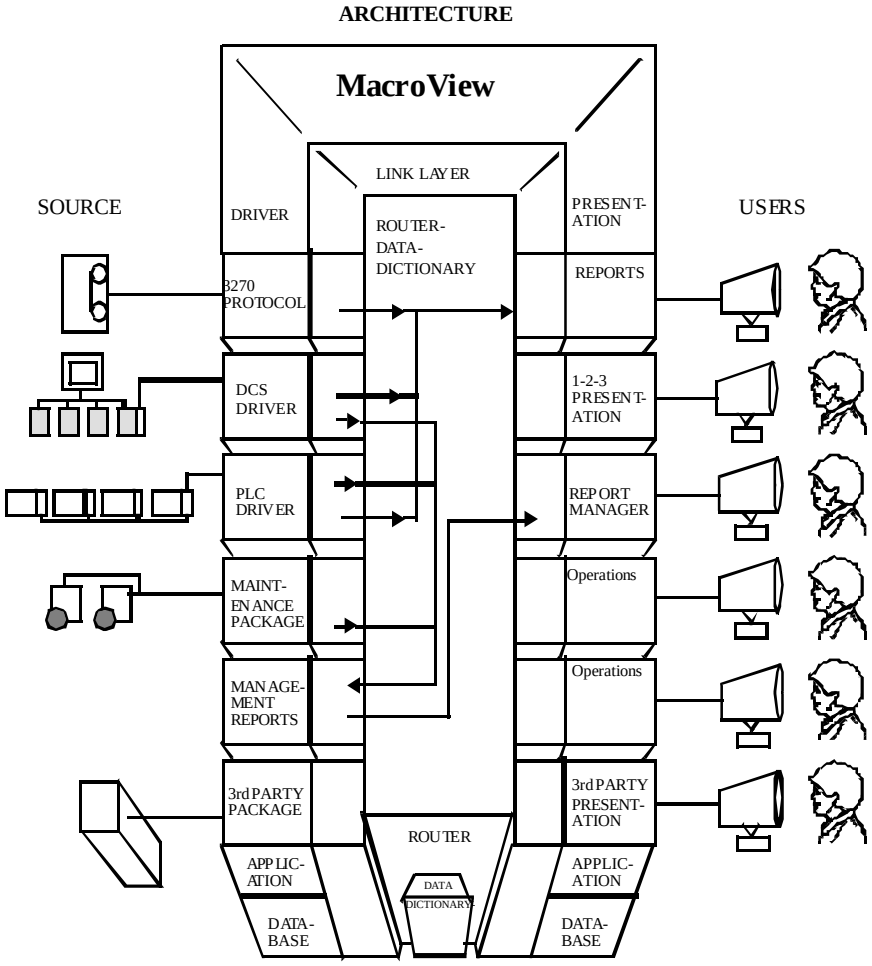
One of the primary functions of the ***MacroView*** product is the integration of data from multiple sources for management and operations. This chapter describes how this is achieved by putting together the various driver building blocks in a clearly defined procedure. In this context:

A Source is where the information comes from. It can be a DCS or a PLC or it could be a recipe or laboratory database.

A Driver is the actual software module that performs the communication with the device or the database. ***MacroView*** allows multiple drivers to be added to the system.

The integration of data occurs in the data dictionary layer where requests from the users are routed to the individual drivers and the data from the drivers is routed back to the users.

The diagram below shows how new sources of data can be added to the system in an organised structure and how the data is routed through the data dictionary to the users



This chapter describes how you add data sources to your system by adding and configuring the communication drivers.

What Drivers you receive with *MacroView*

In the base ***MacroView*** system, you receive the following drivers and sources that have been installed by your distributor:

i. A single Type 1 (usually serial) external driver that communicates with your primary equipment- whether it is a DCS or PLC or a data acquisition unit.

ii. A dBase source - which enables you to address dBase files as entities and,

iii. A local source - for the storage of variables, for simulations, etc.

iv. A text source - for using text files as source data.

Most users have more than a single source of external data. A new driver is added for each external source of data.

Generally speaking, with each driver, you receive:

- The driver software itself.
- A set of standard or example group and detail faceplates.
- Documentation which describes among other things:
  - How you load the driver.
  - How you start the driver.
  - How you configure the driver - in particular, how you specify the entity addresses.

The Sources and Drivers Appendix lists the drivers available "off the shelf" at the time of printing.

If the driver you require is not in the list, you can ask your distributor to write the driver or, if you are proficient in C, you can use the driver development kit to write your own driver. (See the Programs Chapter.)

Installing the Driver

Simply do the following:

1. Place the driver floppy disk in the drive.
2. Type `mvinstall`

At this point, the system will read the files from the floppy disk and will also update the types database (if standard types exist), the group faceplates and the detail faceplates.

Starting the Driver

To get the driver running, you will have to:

i. Configure the various parameters: Before starting the new driver, you need to configure various parameters such as the register block start and end addresses. This is discussed in the detail parts of the next section - "Configuring the Source".

ii. Start the driver: Drivers are normally started up automatically when the machine is switched on. You can arrange this by adding the driver start up commands to the ~/system/start script file. The command, which needs to be added to this file, is defined in the driver documentation.

Typically the commands consist of two lines. The first to change directories to the configuration directory and the second line which actually starts the driver.

For example:

```
cd $MACRODIR
../bin/a-b A-B /dev/tty1a Station 3 > /dev/null
```

3.2 Configuring the Source

To configure the source (and the driver), use the engineering configurator and ***select Data Sources:Sources***.

In summary, each source requires the following information to be specified:

1. **Source Name** - this is the source name is later referred to in the configuration of the entities.
2. **Description** - used for your documentation only.
3. **Source Type** -this is used to decide how the driver data is converted to attribute entity form. The different types of sources and drivers are discussed in more detail later in this Chapter.
4. **Other Information** - such as a key to access the shared memory.

The following pages explain in detail how you fill in the blanks in the Source configuration.

Source Configuration (Name, Description)

Source Name

What you type

Enter the name of the source. By convention, the name is capitalized.

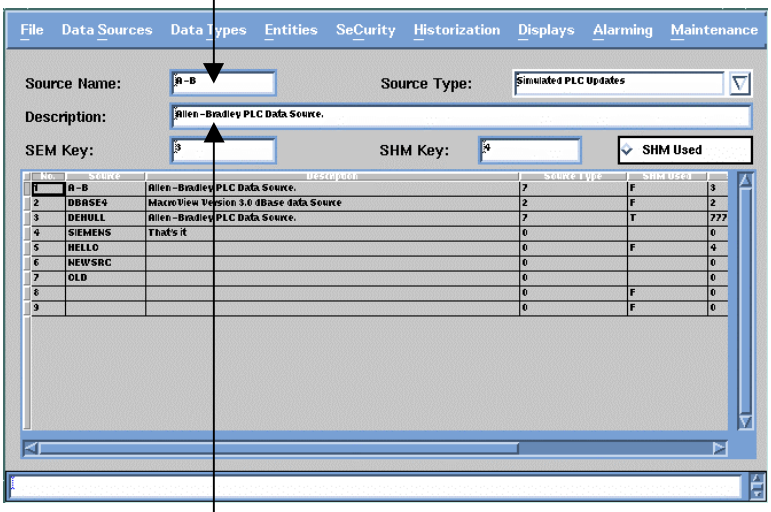
Example

UXL, A-B, DBASE, SQUARE - D, CENTUM etc.

How it Works

The source name is used in the start up command in the `~/system/start` directory. Your driver documentation will tell you how to write this start up command.

Group and Detail faceplates are stored in the directories: `groups/<source_name>` and `detail/<source_name>` where source_name is the name you enter here.



Description

What you type

Enter a description of the source. This description is not used by the system, it is for your reference only.

How to get there

1 ***Data Sources: Sources:Detail*** This brings up the sources detail screen.

2 Click on the source to be modified: The source detail will appear in the window.

3 Alternatively, you may add a blank record using ***Data Sources:Sources :Add Blank*** and edit the new record.

Source Configuration (Type, Shared Memory, Semaphore)

Source Type

What you type

Select from the list of source or driver types i.e.
Off-line Data Source
Simulated updates
DBase compatible databases
Text file updates
Demand database updates
Sorted image updates
PLC Image updates
Simulated PLC updates

How it Works

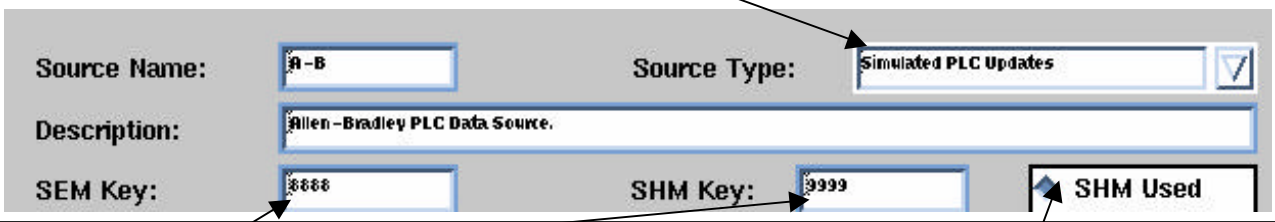
This source type tells ***MacroView*** how to convert the driver data into the attributes and entities structure that the user expects to see.
Each source type is discussed in more detail in the next section of this chapter.
The driver documentation that comes with the system will tell you what needs to be entered here

How to get there

1 ***Data Sources: Sources:Detail***
This brings up the sources detail screen.

2 Click on the source to be modified: The source detail will appear in the window.

3 Alternatively, you may add a blank record using ***Data Sources:Sources: Add Blank*** and edit the new record.



SHM key
SEM key

What you type

Enter a description of the source. This description is not used by the system, it is for your reference only.

How it Works

The shared memory key is used by the various user programs to identify the shared memory for this particular source. It is like a bank PIN number for accessing shared memory. The semaphore key is used by the same programs to avoid a write clash.

You only need to enter these values if your system uses shared memory sources (Consult the documentation provided with the driver). The number needs to be unique. If you choose one already used, the system will inform you.

SHM Used

What you type

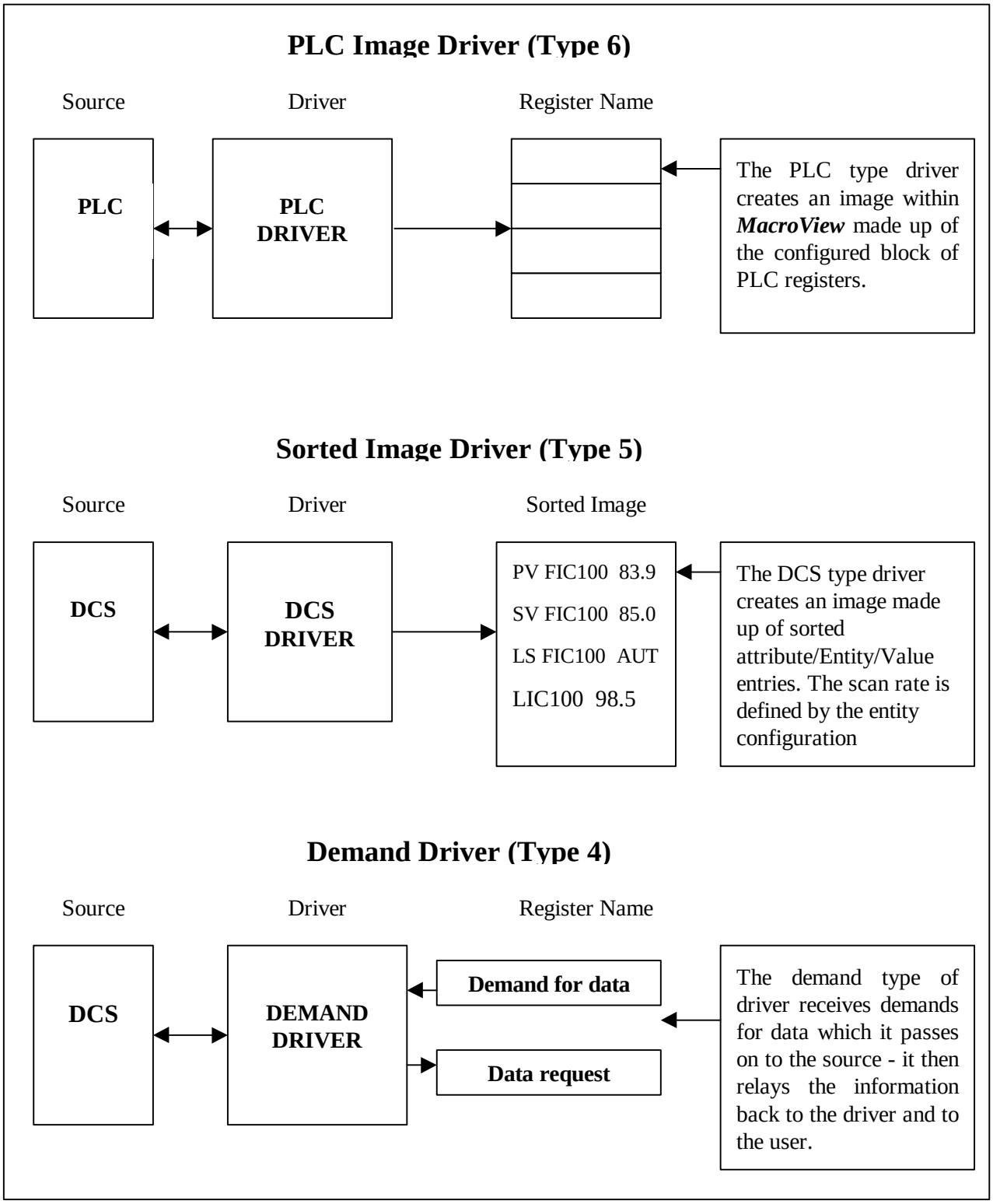
Most drivers use shared memory that enables multiple users to access or share the data that is collected from the source.

How it Works

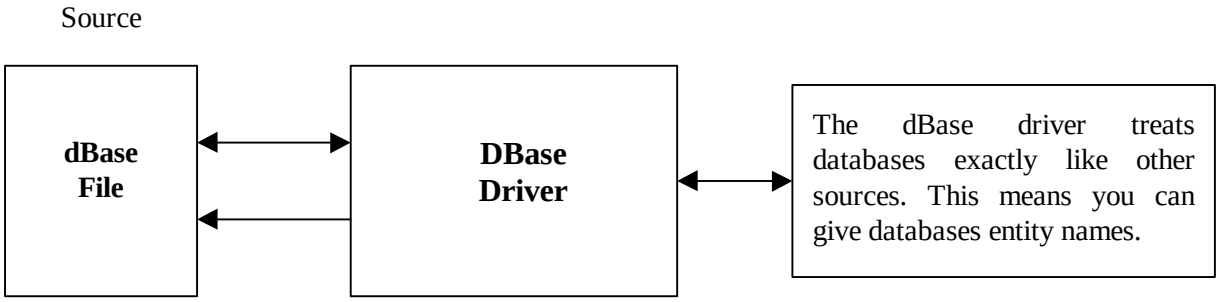
Select whether you want Shared memory or not. The documentation you receive with the driver will tell you what to select here.

3.3 Type of Drivers

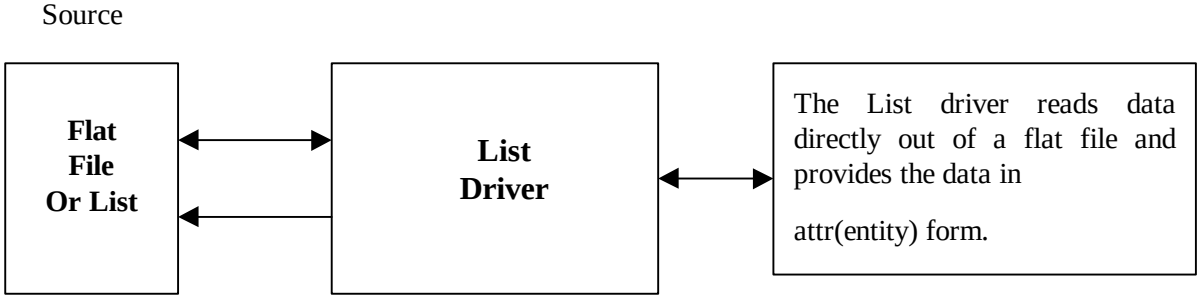
There are various types of drivers. The type of driver you choose depends largely on the nature of the source - whether it is a PLC, a DCS or some data held internal within the ***MacroView*** system. The diagram below shows examples of some typical drivers. The various types of drivers are discussed in more detail in this part of the chapter



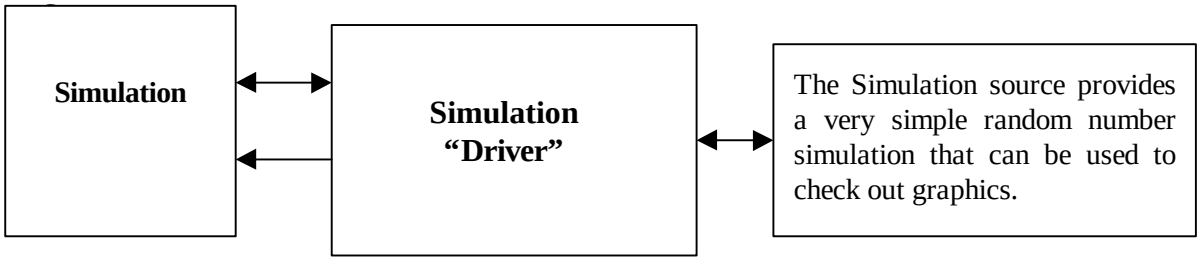
dBase Source (Type 2)



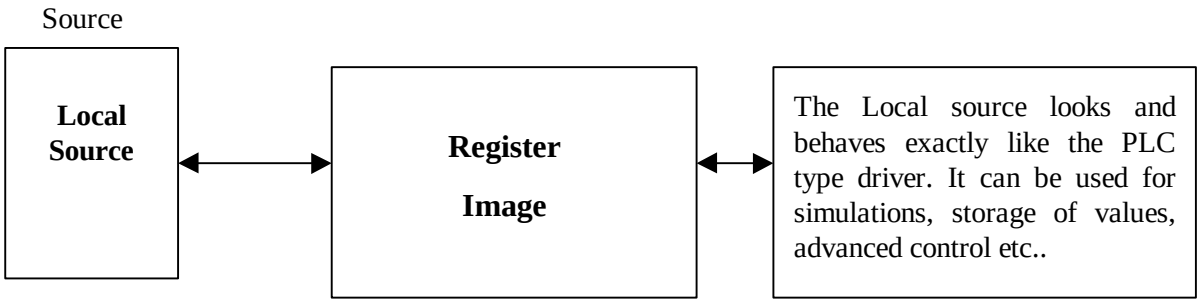
List Source (Type 3)

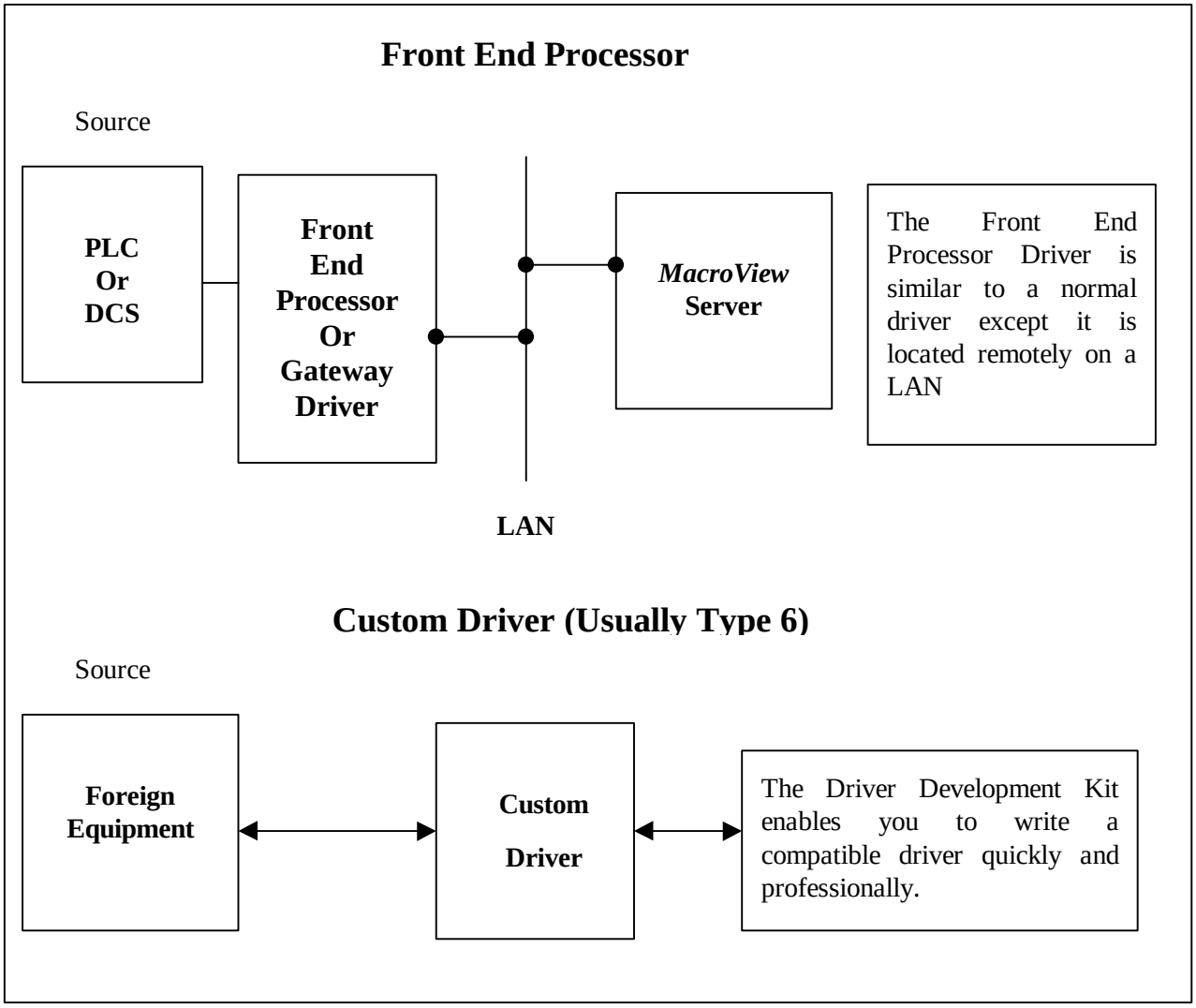


Simulation Source (Type 1)



Local Source (Type 6)



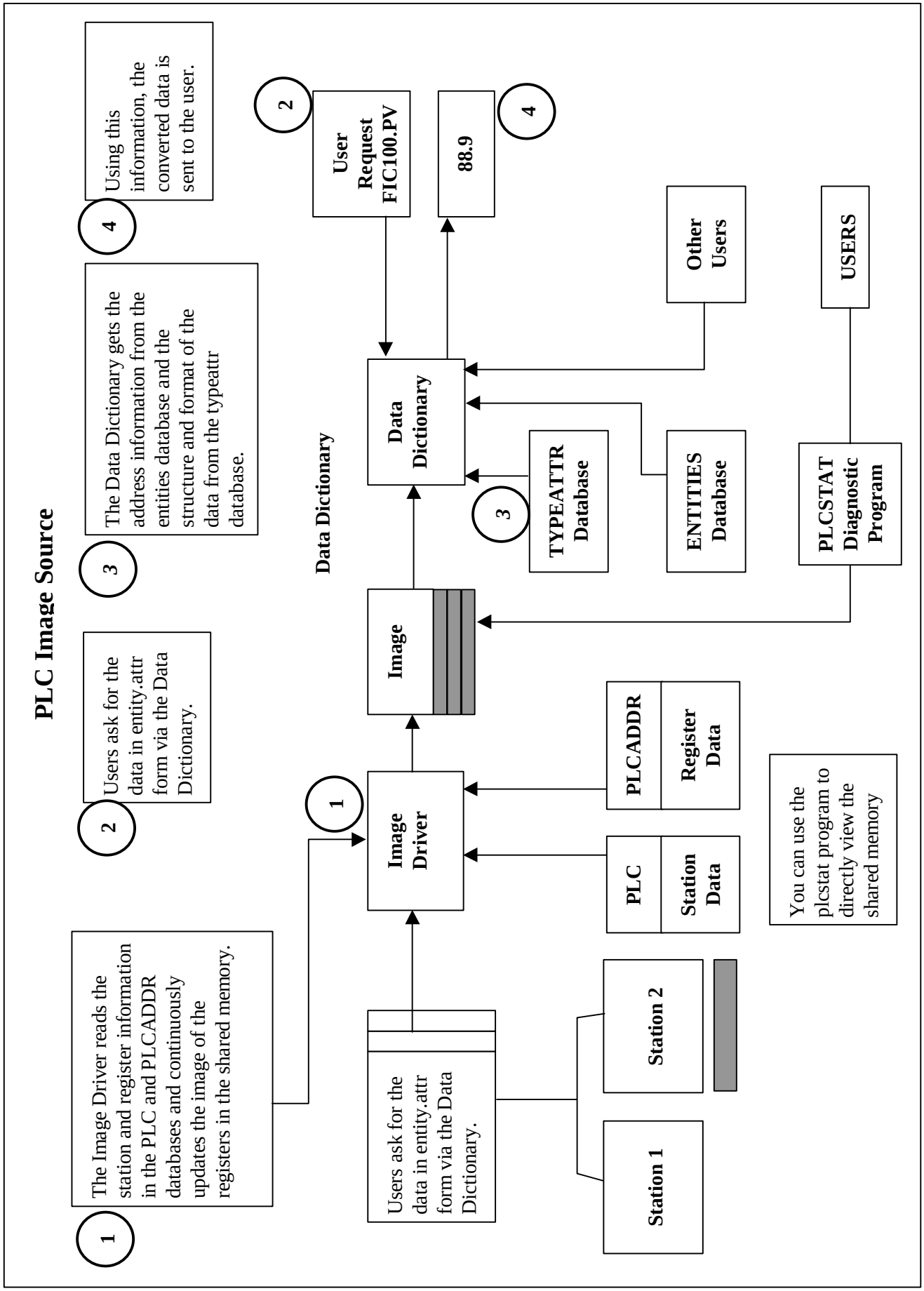


3.4 PLC Image Source (Type 6)

The PLC Image Source is responsible for creating and continuously updating an image, in ***MacroView***, of the register blocks defined in the configuration. The image is accessed by multiple users and, using the entity information such as entity type and address, is converted into the desired standard attr(entity) form.

It is important to note that the PLC Image Source, despite its name, is used for many systems that are not PLC's. The main requirement is that the data is stored in blocks of multiple registers.

The diagram on the next page shows how the Image Source works in more detail.


PLC Image Source
1
The Image Driver reads the station and register information in the PLC and PLCADDR databases and continuously updates the image of the registers in the shared memory.
2
Users ask for the data in entity.attr form via the Data Dictionary.
3
The Data Dictionary gets the address information from the entities database and the structure and format of the data from the typeattr database.
4
Using this information, the converted data is sent to the user.
Data Dictionary
Image Driver
Image
Data Dictionary
User Request FIC100.PV
88.9
TYPEATTR Database
ENTITIES Database
Other Users
USERS
PLCSTAT Diagnostic Program
PLCADDR
Register Data
PLC
Station Data
Station 1
Station 2
Users ask for the data in entity.attr form via the Data Dictionary.
You can use the plcstat program to directly view the shared memory

Configuration

From a configuration standpoint, the following information must be completed:

- **Source Information:**

  This names the driver and makes the location of the shared memory public by giving it a shared memory key.

- **Station Information:**

  This associates a name to the route part of the register address (Most systems have as the first part of the address some kind of routing information).

- **Register Information**:

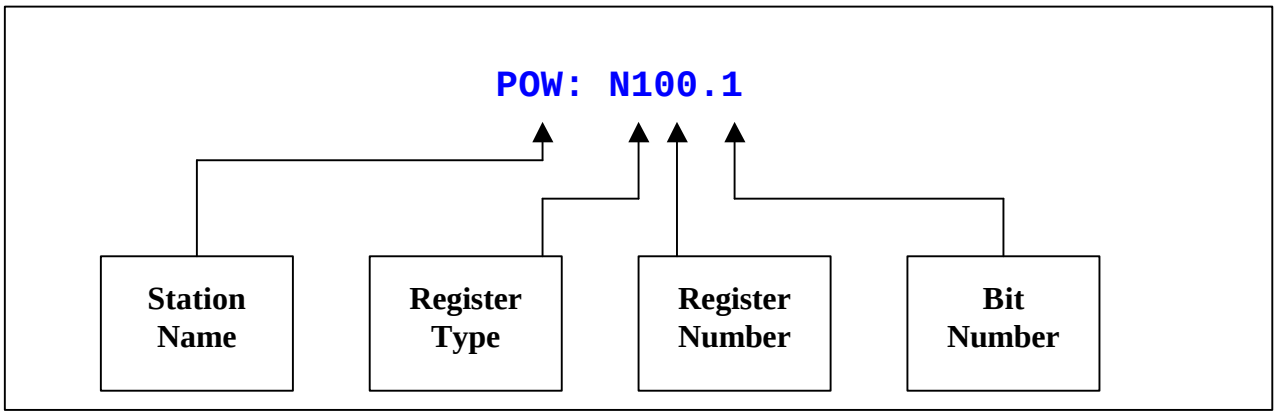
  Here, for each register block, you need to configure the start address, the size of the block and the priority of the block. The priority determines how often the image of that block is updated.

- **Type Information:**

  If you are not using standard types, you will need to define your own. Essentially, this is a process of grouping and defining several attributes into a type. This is explained in detail in the Types Chapter.

- **Entity Information:**

  Each entity must have the source name and the register address stored in its address field. A typical address field is shown below:



Loading the PLC Image Driver

This section is a step by step description of the configuration of the PLC Image Source and Driver.

Load the driver from the floppy using the `mvinstall` instruction discussed earlier in this chapter. Now you need to configure the source.

Configuring the Source

Start up the engineering configurator, add a blank source, and follow the instructions below:

Item	Configuration task
Source Name	Select the source name as the name you used in the driver start up command.
Source Type	Select the source type as PLC Image Updates. This sets up the shared memory location and makes it available to all users
Description	You may use any description here as it is for your benefit only.
SEM Key, SHM Key	Enter a number that you haven't used yet. For more information on shared memory configuration, see the section on Source Configuration earlier in this chapter
SHM Used	Select this item. (I.e. the button should be highlighted.)

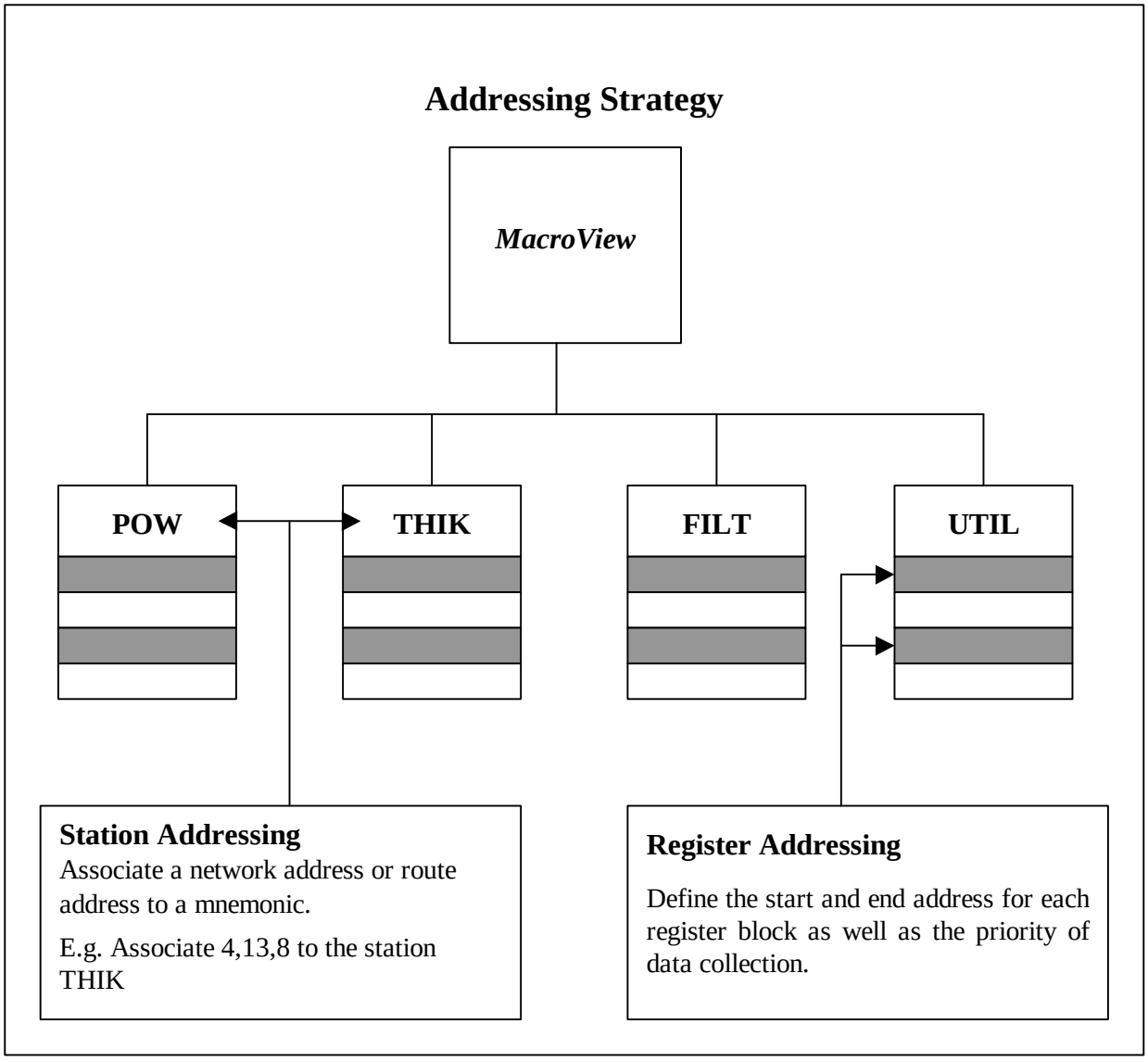
Note: For more information on each of the items above, please see the detailed descriptions in the section "Configuring the Source" earlier in this chapter.

3.5 Addressing

The fundamental mechanism of data collection for the image-based driver is to read in various register blocks. The definition of these register blocks consists of two parts:

(i) Associating a PLC station's network address to a mnemonic that will be used in the entity address (PLC station definition) and

(ii) Defining the start and end address of the registers within the PLC stations (Register definition).



Configuration Station (Source, Station, Route)

Source

What you type

Select the Source from the pull-down combo

How it Works

This tells ***MacroView*** what driver to use.

Example

SQUARE-D, A-B, GE.

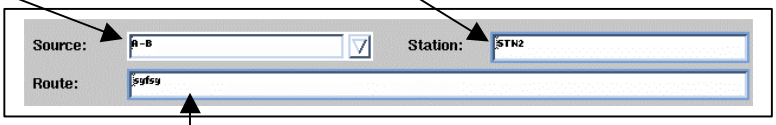
Station

What you type

Enter the name of the station.

How it Works

This name is used as a mnemonic by the driver. If the entity address is MON: 0400 then the driver substitutes the route address for MON to make up the complete address.



How to get there

1 ***Data Sources: PLC Station:Detail***
This brings up the Station detail screen.

2 Click on the Station to be modified: The source detail will appear in the window.

3 Alternatively, you may add a blank record using ***Data Sources:PLC Station:Add Blank*** and edit the new record.

Route

What you type

Enter the address of the station on the network. This is specific to the driver used. You should consult the driver document to determine the format of the route for the specific station.

How it Works

Although it is normal to associate a simple station address to each PLC, remember the PLC station address is just a mnemonic for the route address. It is therefore, quite acceptable to have more than one station name per physical PLC - for example, you may wish to use mnemonics for ACM, TRD and SCAN for the Alarm area, Trend area and Scanned area all within a single PLC.

Configuring the Register Blocks

Station, Start Address, End Address.

What you type

Enter the Station name, Start address and End address of the block.
E.g. STN3, 0,400 for a register block called STN3 starting at register 0 and ending at register 400.

How it Works

The driver cyclically refreshes the internal image of these registers at a rate dependant on their priority.
To increase the update speed, try to organize the data into fewer but larger register blocks. You can of course have several blocks per station.

Hint

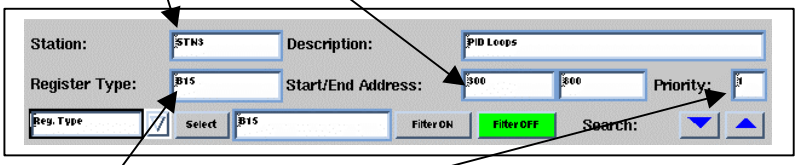
Note: Some drivers may impose a maximum register block size.

How to get there

1 ***Data Sources: PLC Register:Detail*** This brings up the Station detail screen.

2 Click on the Station to be modified: The source detail will appear in the window.

3 Alternatively, you may add a blank record using ***Data Sources:PLC Register:Add Blank*** and edit the new record.



Type

What you type

Enter the register type:
A for analog.
N1 for file type N1.

Example

PLC manufacturers store different types of data in different formats. For example, counters and timers are stored in a different way to digital inputs.

How it Works

The driver needs to know how they are stored. Again consult the appropriate driver reference sheet for specific examples.

Priority

What you type

Enter the priority of the refresh rate
(0 to 99).
0 = don't update unless demanded.
1 = low priority.
99 = high priority, quick update.
(i.e. 99 times more often than a block with a priority 1)

How it Works

MacroView uses an intelligent scanning technique to satisfy the updating speed for all users. If more users are looking at a point, it will be updated faster. Likewise if the priority is high, the point will be updated more often.

Hint

If a value is only required for display updates, set the priority to zero to minimize the communications loading

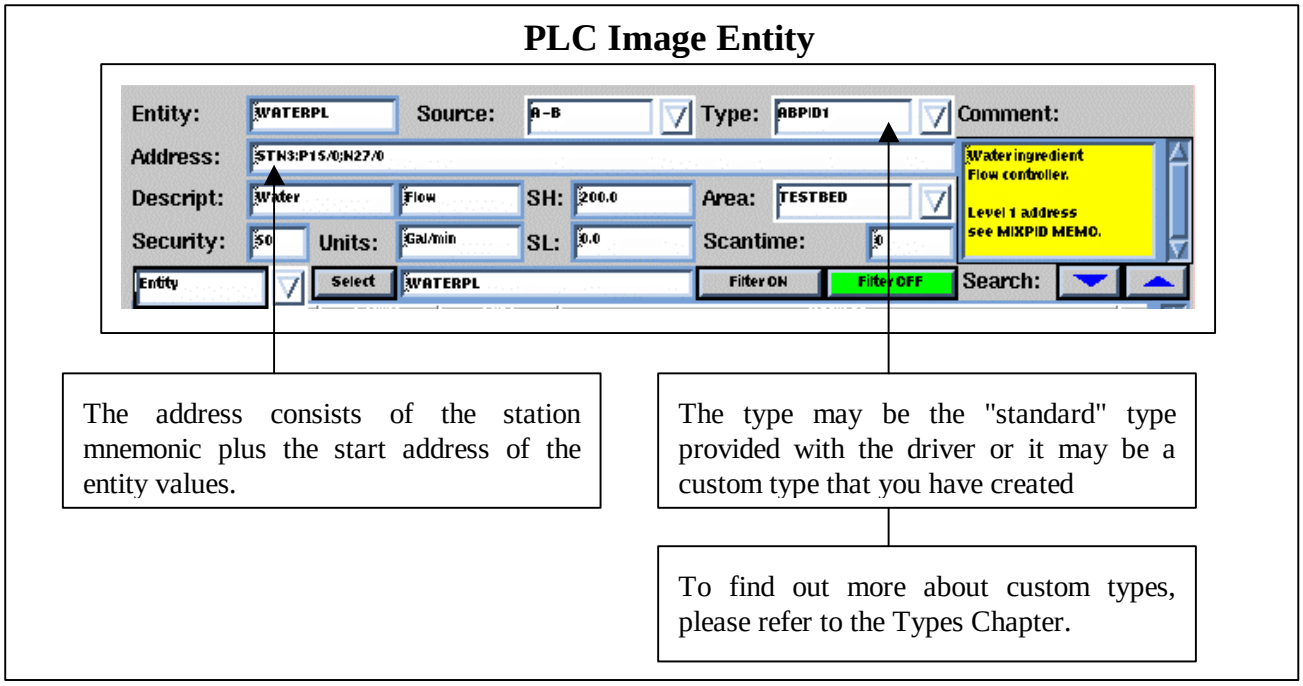
Starting the PLC Image Based Driver

Once you have set up the station names and the addressing, you can start the driver using the technique discussed earlier in this chapter.

Remember to get the specific driver command from the driver documentation and please note that, if you change the register block definition, you need to restart the driver. This is because the driver only reads this configuration when it starts up.

Adding the Entities

The Entities Chapter of this document described how you configure entities in detail. For a PLC image source, a typical entity could look like this:



Checking out the PLC Image Source

To see that the driver is functioning and to get statistics on the rate of communication and any errors, you can use the `plcstat` program.

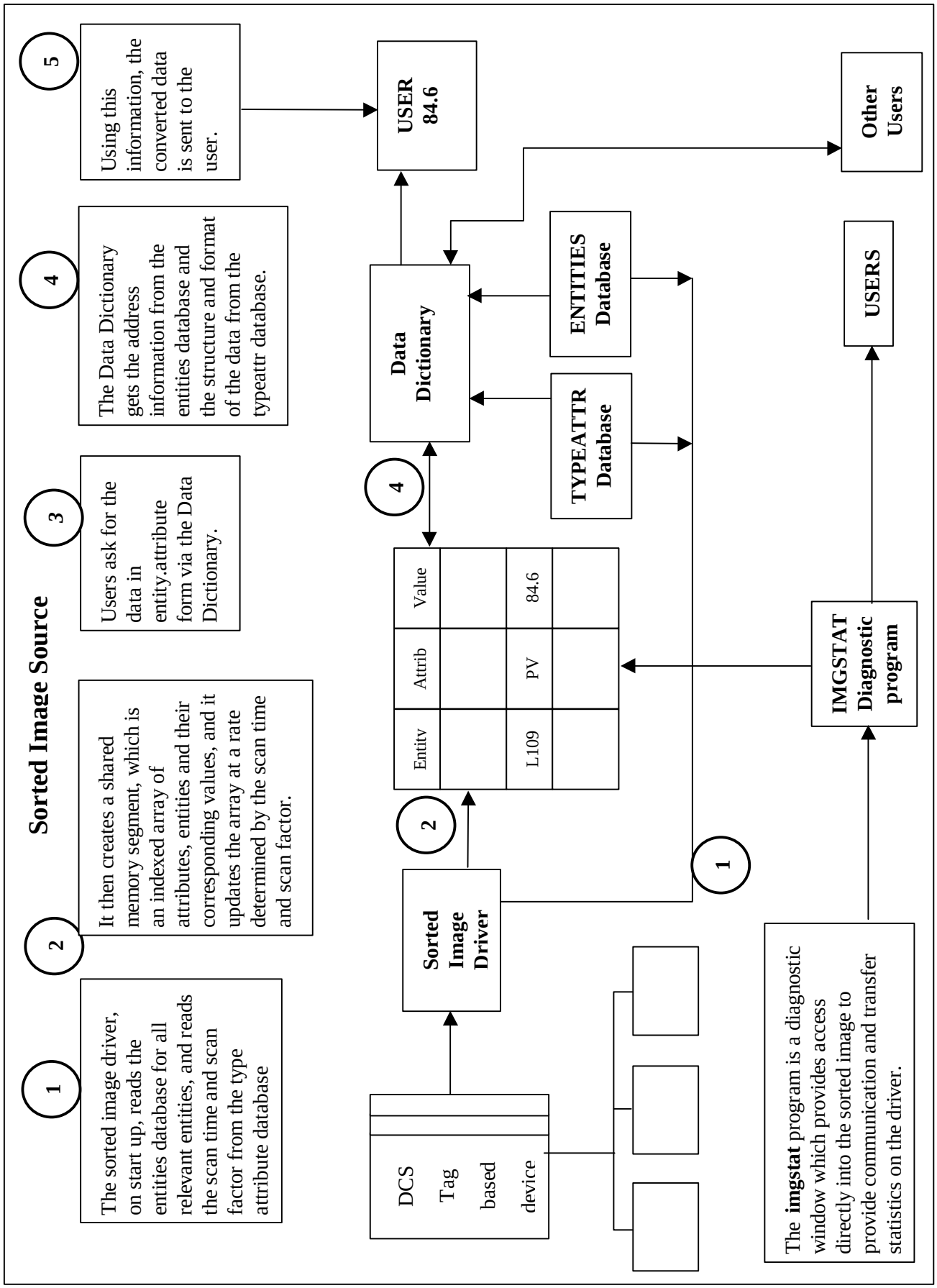
i. Just type in `plcstat` at either the UNIX prompt or DOS prompt, depending on your system type, and you can interactively move through the image produced in shared memory. For more information on `plcstat`, please ask your distributor for the document `plcstat(C)`

ii. Once you are satisfied the driver is operating and updating the image, start the operations program, and call the entity you have created to the scratchpad. You should see the entity name appear in the scratch pad with the correct PV.

iii. Now call up the Detail display for the Entity and check the other attribute values.

3.6 Sorted Image Source (Type 5)

- The sorted image driver is responsible for creating and continuously updating an image of the data held in the entity-based device.
- In contrast to the PLC image driver, the sorted image driver asks the device for data by attribute and entity name as opposed to reading in blocks of registers. This form of communication is typical for DCS and other systems.
- Also in contrast to the PLC image driver the sorted image driver stores the data (as its name implies) in a sorted image or table of entities, attributes and the updated values. The image is indexed allowing a fast access retrieval of the data.
- The rate at which each attribute is updated is defined in the configuration of the scan time in the entities configuration and the scan factor in the types configuration. The diagram shows how the sorted image driver functions.


Sorted Image Source
1
The sorted image driver, on start up, reads the entities database for all relevant entities, and reads the scan time and scan factor from the type attribute database
2
It then creates a shared memory segment, which is an indexed array of attributes, entities and their corresponding values, and it updates the array at a rate determined by the scan time and scan factor.
3
Users ask for the data in entity.attribute form via the Data Dictionary.
4
The Data Dictionary gets the address information from the entities database and the structure and format of the data from the typeattr database.
5
Using this information, the converted data is sent to the user.
USER 84.6
Other Users
Data Dictionary
ENTITIES Database
TYPEATTR Database
USERS
Entity
Attrib
Value
L109
PV
84.6
Sorted Image Driver
IMGSTAT Diagnostic program
DCS Tag based device
The imgstat program is a diagnostic window which provides access directly into the sorted image to provide communication and transfer statistics on the driver.

Configuring the Sorted Image Driver

To load, configure and start the driver, just follow the instructions in the first part of this chapter.

A careful strategy in choosing the scan time and the scan factor is advised.

The actual rate at which the attribute is updated is known as the update period. The update period is defined as:

- Update period = scan time x scan factor (in seconds)

Every entity has a defined scan time in seconds, and every attribute for a given type has a scan factor.

It is wise to choose these values carefully for a large system because, obviously, every communication channel has a finite bandwidth.

Note: If the driver has idle periods, it will use up these idle periods by updating certain attributes at a faster rate dependant on the users demands.

These extra updates are referred to as demand updates. This way, the driver makes the best possible use of the communication channel bandwidth.

Checking Out the Sorted Image Driver

To see how the driver is functioning and to get statistics and communication details, you can use the diagnostic program **imgstat.**

This provides a direct window into the sorted image shared memory. Just type `imgstat` at the UNIX prompt and you will be able to interactively examine the various parts of the sorted image.

For more information on `imgstat`, please ask your distributor for the document `imgstat(C)`

Once you are satisfied that the driver is functioning correctly, you can check out the entities using the usual procedure (i.e. call up the entity in the scratchpad, and call up the detail display).

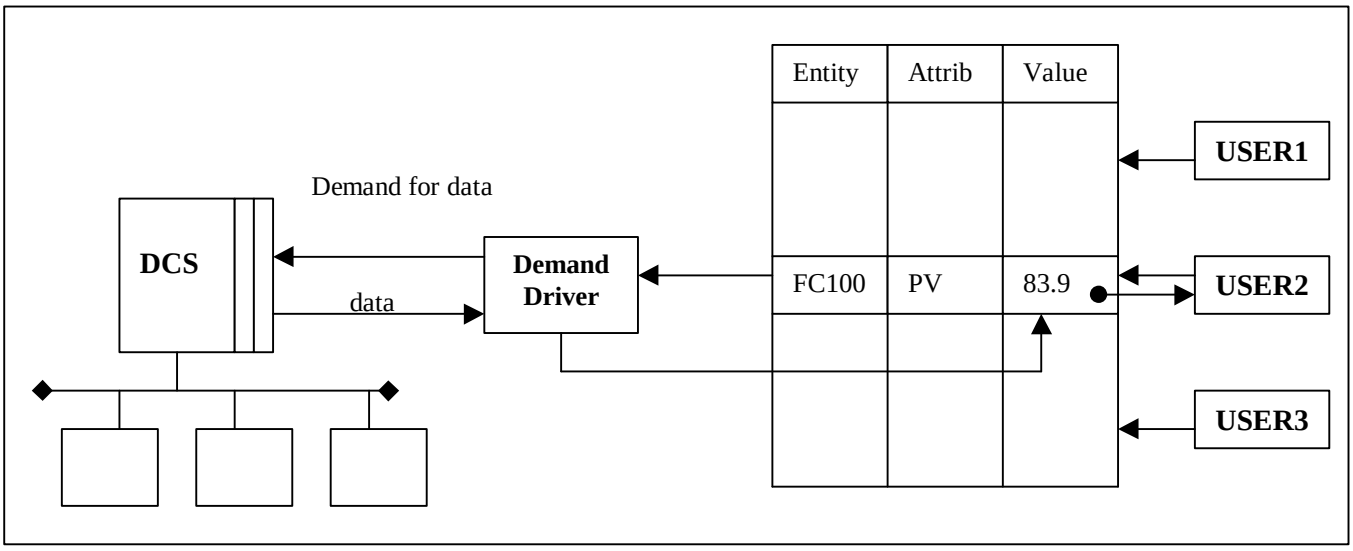
3.7 Demand Based Source (Type 4)

With the demand-based source, each program is assigned a partition in a database where the program writes its demands in terms of entity and attribute for that particular moment.

For example, the user may require all the attributes for an entity FIC100 if the detail display is being shown.

The demand driver reads the same database for the list of attributes and entities from all the users and in turn asks the source equipment for the same values.

Once the source equipment (usually a DCS) responds, the values are written in the value column where the user retrieves them and puts them on the screen.



Note: that each program has a separate partition in the database and therefore the amount of communication traffic is **dependant** on how many programs there are and on the amount of data requested by each program.

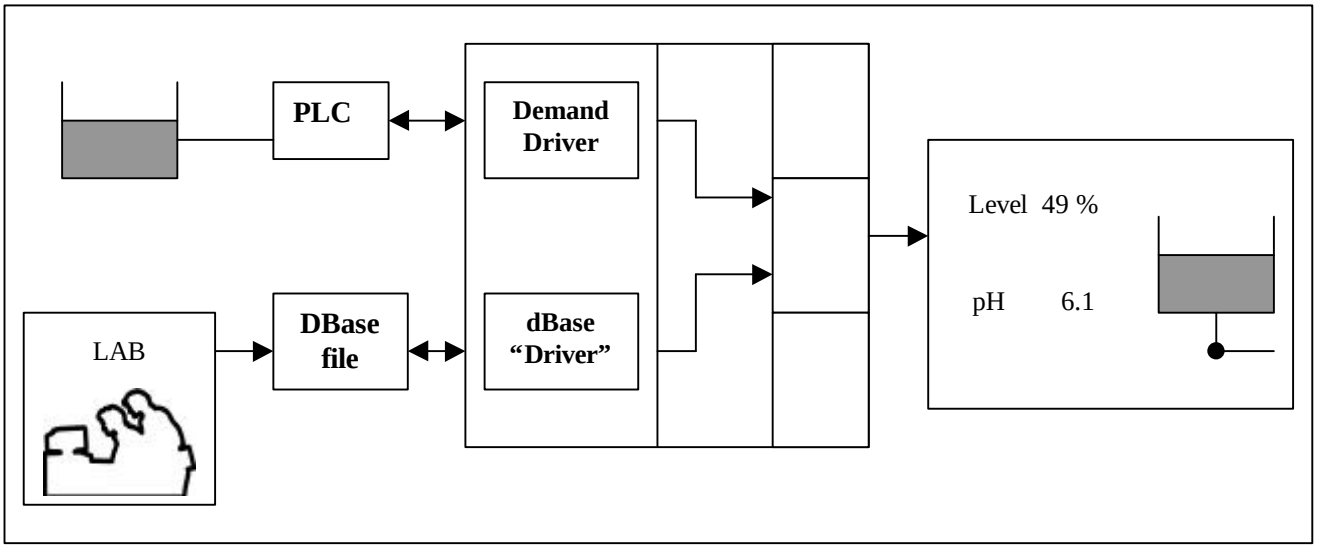
- This is in contrast to the PLC image based source and the sorted image based source where the communication rate is not directly related to the number of users and their demands.
- For this reason, it is recommended that this type of driver is used only for small to medium sized systems.
- Additionally, please note that the process of reading data involves a number of sequential processes.
- That is the user writes the demanded data, the driver reads the demanded data and asks the equipment for the data, the equipment answers and sends the data to the driver which writes it in the database, the user reads the database.
- This means that the screen updates can be relatively slow in contrast to image based drivers where the user reads the data directly from the database.

For more information on this type of database, please see the documents **supervis**(F), **setvalue**(F) and **updates**(F)

3.8 dBase Source (Type 2)

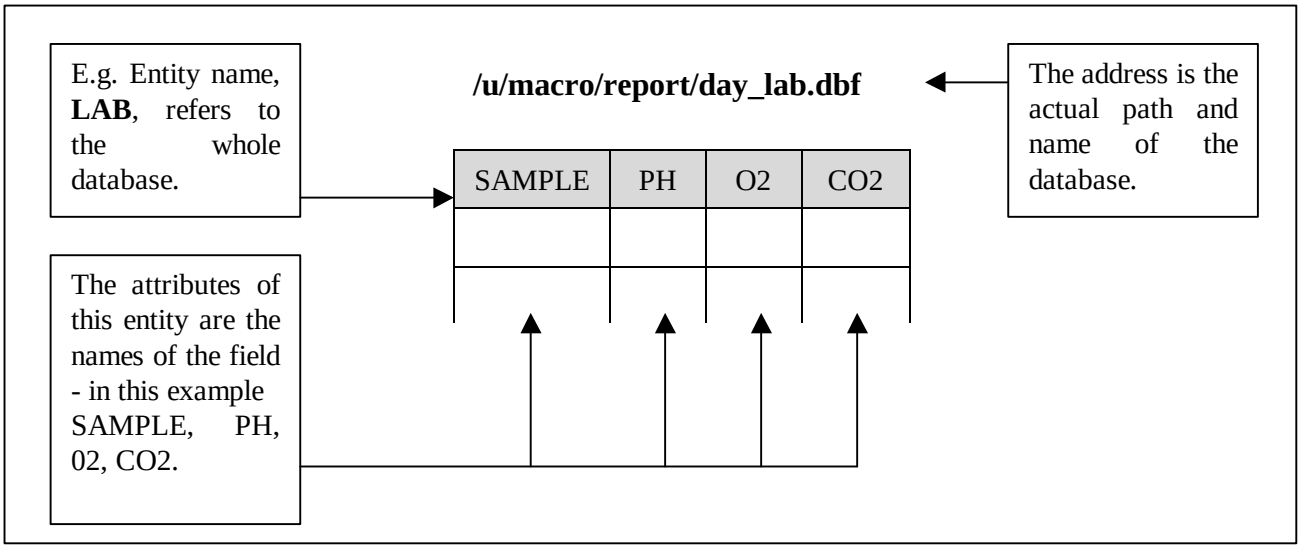
Using the dBase source, you may look at a dBase file in exactly the same way as you would look at an entity say from a PLC. For example, you can call up the entity name of the dBase file in the scratchpad and ask for the detail display of the dBase file by pressing the D for detail key.

The dBase source makes dBase compatible files "look like" entities. This means that you can freely mix data from a real world source with data from a dBase file in a graphic or on a trend. For example:



In essence, with a dBase source:

- The **entity name** refers to the whole database.
- The **attribute names** are the names of the fields.
- The **entity address** is the actual path and name of the database itself i.e.



When you first access the database entity, the values of the attributes are taken from the fields of the last (or in the case of a historical file, the latest) records.

You can also access any of the records by changing the record number or by referring to record numbers **relative** to the current record. (This is discussed in the section "Using the dBase Source" later in this chapter.) But first, let us discuss the configuration of the dBase source.

Installing the dBase Source

There is no need to install the dBase source, it comes standard with your ***MacroView*** system. Likewise, it is not necessary to start the "driver"

Creating the dBase File

- You create the dBase file using dBase itself either in **ASSIST** mode or at the dot prompt with `CREATE <dBase-name>` where `<dBase-name>` is the name of the database. Remember that the names you assign to the fields will be the names of the attributes of this entity. In fact, the dBase source will read these names from the header of the dBase file.
- If you intend to use the dBase file for historical collection, i.e. storing data on a time basis, then the first field must be called TIME and (if required) the second field must be called **DATE**. This will enable you to use the data on a trend page.
- Once you have created the database file, you are ready to configure the entity.

IMPORTANT NOTE

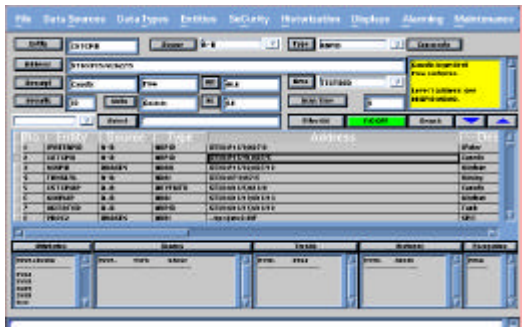
The following description on the configuration of dBase Entities will make mention of a special attribute for dBase Entities, called RECNO, and the value of this attribute can be set by a program, for example, to change the current record pointer position in the data base. While this reference to, and use of, the RECNO attribute is still supported in the current version, this is mainly for backward compatibility reasons. The preferred method in dealing with the manipulation of data and setting current record pointers is to use the tools provided in the metascript language discussed in chapter 10 of this Engineering manual.

You will also note that the examples refer to the DBASE Source. This again is relevant for pre Version 3 ***MacroView***. The preferred source for Version 3 Database Entities is the DBASE4 Source.

Configuring dBase Entities

Entity Name

Refers to the complete database



Source

This should be the name you configured in the source configuration earlier in this chapter (typically **DBASE4**).

Address

This is the path and name of the dBase file including the **.dbf** extension. Note that you can use absolute addressing e.g. `/u/macro/data/lab1.dbf` or relative addressing

e.g. `../data/lab1.dbf.`

Relative addressing is preferable because the address is relative to the configuration directory.

Security

The security refers to the whole database. Therefore, setting the security of individual fields is not possible.

Scales

You must set the scales to something other than zero. Typically, you set the scales to correspond to the most significant field.

Type

Use any name for the type. This is the name you will use for the group and detail faceplates. ***MacroView*** will determine the structure of the dBase file by reading the dBase header. In this way, you may refer to the attributes of the entity by their field name, e.g. LAB1.PH refers to this field.

RECNO

The special attribute for the dBase source is **RECNO**, and this is number to which the record pointer is currently set.

It is a local variable, which means any changes of the **RECNO** attribute only affect the program in which the change was performed.

In this example, if the record pointer was on the last record, RECNO would = 5.

SAMPLE	PH	O2	CO2

The first field may be referred to as the PV or the field name. When you first call it up in the scratchpad, the "PV" will be displayed.

Note: You do not create a type for dBase entities since the structure is held in the dBase header

If you intend to use the dBase file for trending, you must call the first field **TIME** and, if necessary, the second field **DATE**.

3.9 Using dBase Entities

Once you have created the dBase entity, you may access the entity by referring to the attribute (field) and entity exactly the same way as you would refer to a normal "real world" entity.

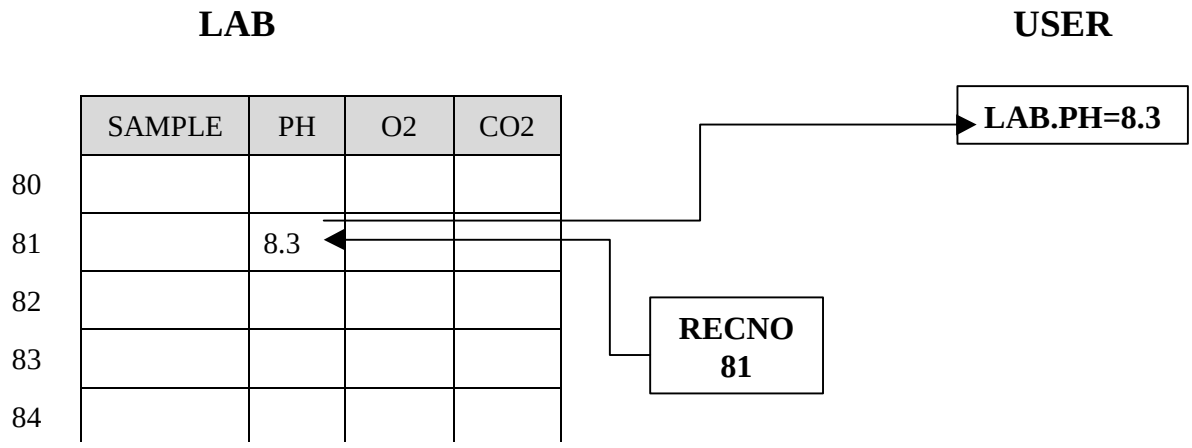
Example	Call the dBase entity to the scratchpad and then call up the pop up attribute list. You will see that all the field names in the data base are listed as attributes and these attributes can be called to the scratchpad to display their values in the same way any other entity and attribute value is displayed. Other applications such as Lotus, Scripts and meta scripts may also access the data in the usual way.

RECNO (The special Record Number Attribute)

Unlike "real world" entities, database files are two dimensional arrays and you generally need to be able to access the attributes of the database in any of the records in the database.

To do this, you can set the special dBase attribute called RECNO.

RECNO is the current record number of the database.



The special dBase attribute RECNO.

1. Can be set using a program.
2. Each user has a private copy of the attribute RECNO - this means one user can look at one record, another user at a different record.
3. When the database is first accessed, the RECNO is set to the last record or in the case of historical files, to the latest record.

Note: You do not create a field called RECNO. The field is automatically there for every dBase entity.
Note: The exception to this is when we use another entity as the record number pointer. See the next section.

The special dBase attribute RECCOUNT

This is similar to RECNO but always holds the number of records in the database

Assessing the Data using Relative Addressing

When you access the data using the normal method you are actually looking at the data in the record pointed at by the attribute RECNO (which you can change, if required). You may also look at records relative to the current record if you use the square brackets as follows:

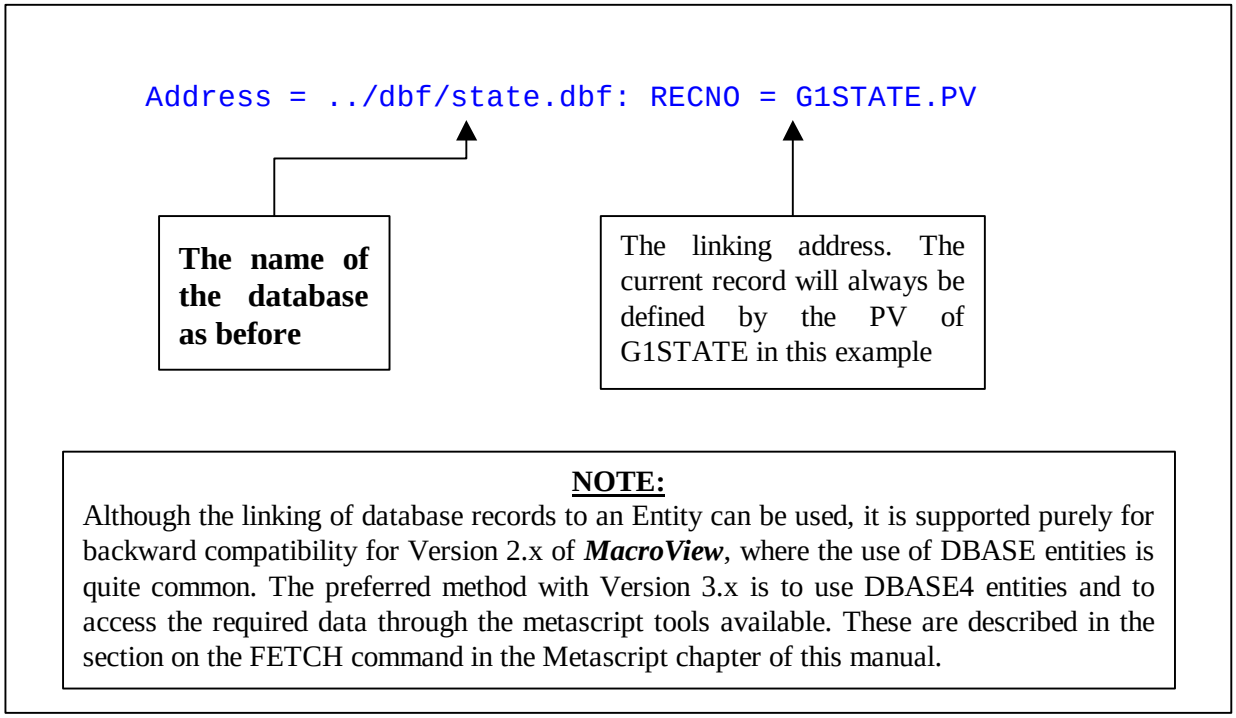
Table 1: Relative Addressing

Entity.Attr definition	Attr(entity) definition	Refers to:
LAB.PH	PH(LAB)	The record pointed at by RECNO.
LAB.PH[1]	PH[1](LAB)	One record **back** from the current record.
LAB.PH[2]	PH[2](LAB)	Two records **back** from the current record.
LAB.PH[-1]	PH[-1](LAB)	One record **forward** from the current record.

This facility is useful for tabular data in graphics, in spreadsheets etc. **Note** If you have used a TAG to index the database, the offset will be relative to the current record in the ordered database.

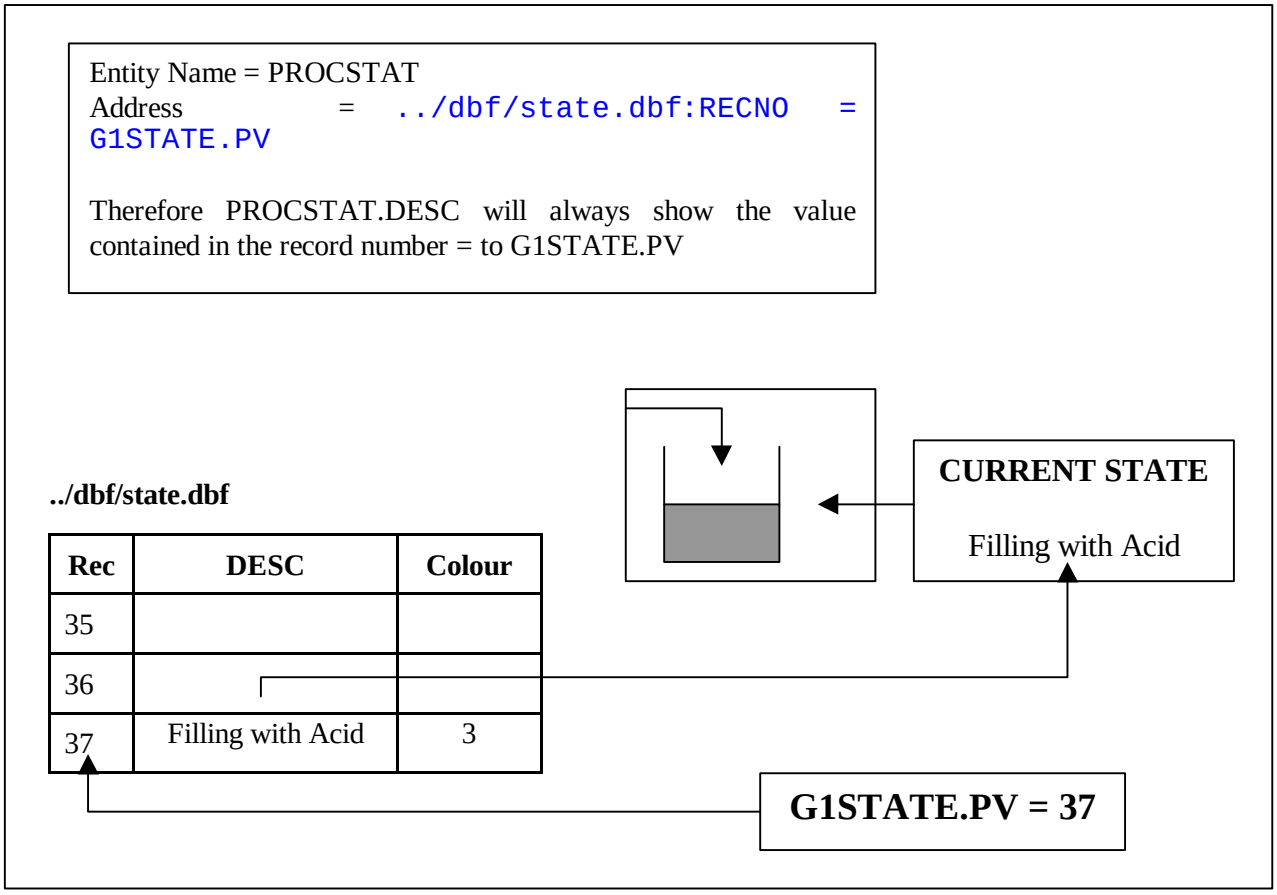
Linking the Record Number to an Entity

You may also link the RECNO attribute directly to another entity. In other words, the record number is set to the value of the other entity. You do this by setting the address in the entity configuration as follows:



NOTE:
Although the linking of database records to an Entity can be used, it is supported purely for backward compatibility for Version 2.x of ***MacroView***, where the use of DBASE entities is quite common. The preferred method with Version 3.x is to use DBASE4 entities and to access the required data through the metascript tools available. These are described in the section on the FETCH command in the Metascript chapter of this manual.

In the following example, the "state" entity PV reflects the state of the process. This is used to point to a description that must be displayed on the screen for each new state.



As the state entity changes, so different descriptions are displayed on the screen.

Other dBase Functionality

Since all ***MacroView*** configuration files are in a dBase form, there is a close coupling to the dBase functionality. Please refer to the Database functionality section of the Metascript Chapter 10 of this Engineering manual for more details on such things as historical access and real-time access from a dBase program.

Using Indexes with dBase4 Entities

For large databases, it is more efficient (and fast) to set up an index or **mdx** file so that the record numbers are ordered according to the way you wish them presented. You can set up to 47 indexes in a single **mdx** (multiple index file) for a given database file. To set up an index, you should:

1. Create the index file (**mdx** file) in dBase using the **INDEX ON** command at the dot prompt. In the command, give the index a tag name. E.g.,

   INDEX ON PCODE TAG PCODE

2. Configure the entity as a **dBASE4** source as described above.

3. In the address field, set the Address as

   `<FileName> TAG=<TagName>`

e.g. `batch2.dbf TAG=TimeOfBatch`

You need one entity for every index. I.e. several entity names (with different indexes) be referencing a single database.

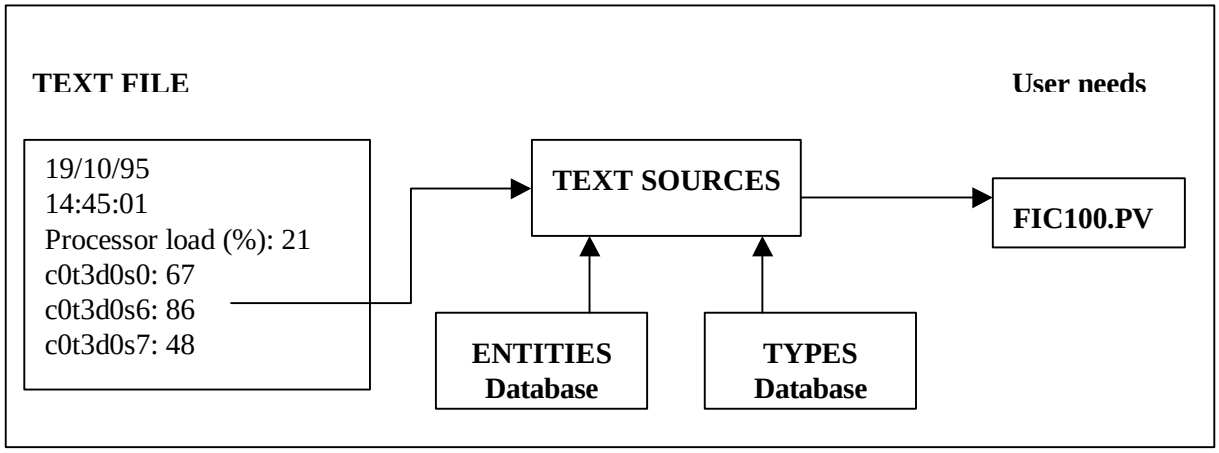
4. Once you have created the entity name, you can check that the index works by creating a Browse widget and confirming that the order is as required

3.10 Text Source (Type 3)

One of the most useful sources is the TEXT source.

This source reads data from an ASCII file (flat file) and provides the data in the standard ***MacroView*** entity.attr format. Once it is in the entity.attr format, the data that originated in the text file can be used in exactly the same way as any other "real world" entity. For example, it can be used in graphics, group or detail displays, in high level programs etc.

This source is particularly useful for reading in data that has been sent to the ***MacroView*** system from some another computer system - for example, an in-stream-analyser.



As the diagram shows, when the user asks for a value in entity.attr form, the text source reads the entities and types databases to find the name of the text file and the position of the attribute value. The position is defined using a combination of search strings and line and character offsets from the search strings. You configure the search string and offset information in two levels.

1. The entities level - which locates the block of data relating to the entity and
2. The attribute level - where the search continues to find the actual location of the attribute itself.

Using this mechanism, you can set up a simple and flexible file structure that can be expanded without any program changes and which has the required transparency expected from ***MacroView***.

Installation and Start up

There is no need to install the Text source or to start it up. This source comes standard with the ***MacroView*** product.

Text File Structure

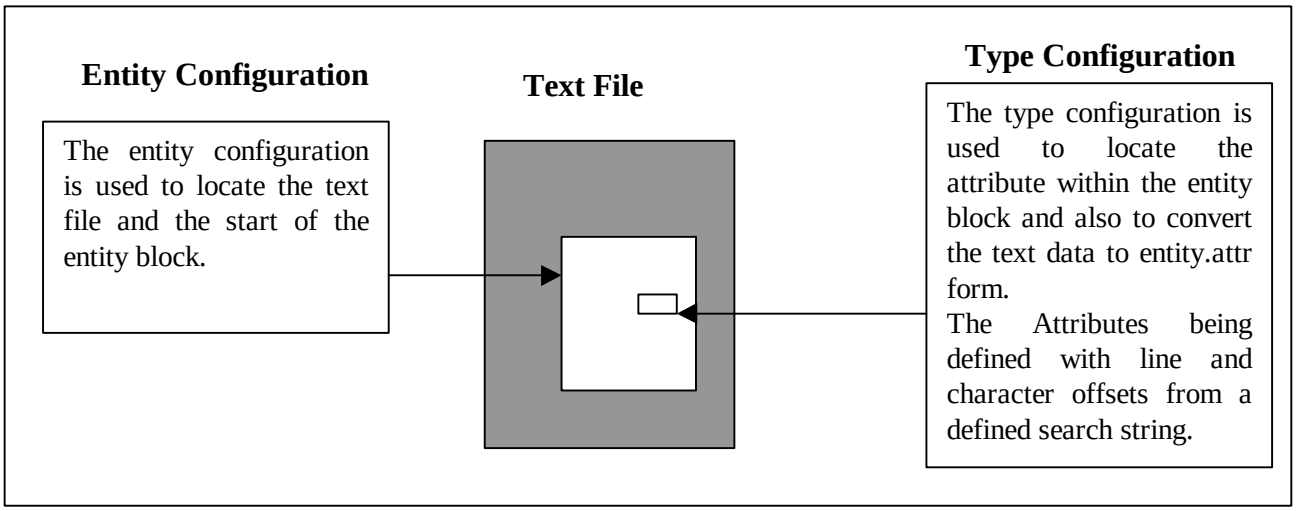
Decide on a text file structure that can be produced by the foreign device easily. It is preferable to use search strings as opposed to line and character offsets.

When using the search strings, the files are easier to read for debugging purposes and are simple to expand. The choice of file structure is worth careful consideration.

Entity and Type Configuration

The entity and type configuration is used by the text source to locate and **convert** the text file into entity.attr form.

The diagram below shows how the Entity and Type configuration locate the data.



The configuration process therefore involves:

1. The setting up of a type (using the Maintenance, Types and Attributes menus in the configuration software) and

2. The configuration of the entities themselves.

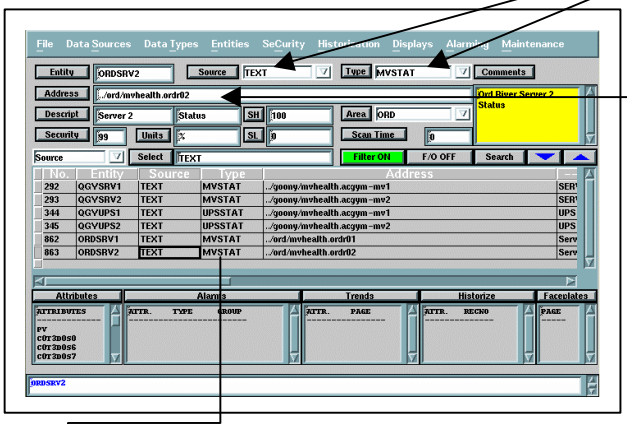
Of course, once you have set up the type structure, you can use the same type for a multitude of entities, provided they have the same form. Likewise, the group and detail faceplates need only be designed once for each type.

The procedure is best shown in the following example, which reads the space left on a server's disk drive into an attribute:

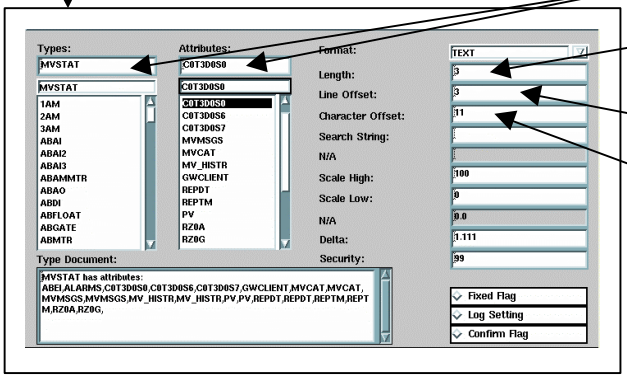
Text Source Configuration Example

ORDSRV2.C0T3D0S0 =67

ENTITY CONFIGURATION



TYPE CONFIGURATION



The entity configuration of ORDSRV2 indicates the **source is TEXT**,

The type (attribute information) is MVSTAT.

The **location of the text file** is in the directory "ord", filename "mvhealth.ordr02".

The **Type information** holds the key to locating and formatting the attribute value.

The TEXT "driver" reads the attribute C0S3D0S0 of the type MVSTAT.

The attribute value is 3 characters in **Length**.

The attribute is situated 3 lines from the top **(Line Offset = 3)**, and 11 characters from the start of the search string **(Character Offset = 11)**.

There is no **Search String** in this example. The **Search String** may be used to find a certain line in the file.

The **SH and SL** may be used as clamps when writing to this source.

Entity data block

TEXT FILE NAME

../ord/mvhealth.ord02

19/10/95
14:45:01
Processor load (%): 21
c0t3d0s0: 67
c0t3d0s6: 86
c0t3d0s7: 48

3 lines below the top of the file (or search string).

11 characters offset to the actual value (Attribute Character Offset)

Length of field is 3 characters (percentage).

Variation to the Example

1. Using line and character offsets

   The general form of the address in the entity configuration is filename "search-string" (x,y) where x is the number of lines (i.e. CR and LF or CR or LF's) past the search-string and y is the number of characters past the search-string. If you do not use the search-string, the number of lines and characters will be from the top of the file.

2. Using the command form of the address

   Instead of using an existing file, you may generate one at the time of retrieval by setting the address to one of two forms:

   ! command "Search-string" (x,y)

   OR

   ! "command arg1 arg2" "search-string" (x,y)

In each case, command is the name of a program, which has as its standard output a text file. The "search string" and (x,y) fields have the same meaning. In the second case, arguments may be used by the calling program. An example of this type of command is to display the computer usage in a graphic display using the vmstat(C) command.

WARNING

- By its nature, the Text source is not intended for high speed updates. If this is required, it is better to write a custom driver with the driver development kit.
- Also please note that the text source must open the text file and there is a limit to the number of open files a system can handle. You should ensure that the number of users with open files does not approach the maximum number available - or, if it does, increase this number.
- Further information on the Text source can be found in the sources(F) manual page obtainable from your distributor. Also, in defining your types, you may need to consult the Types Chapter in this configuration manual.

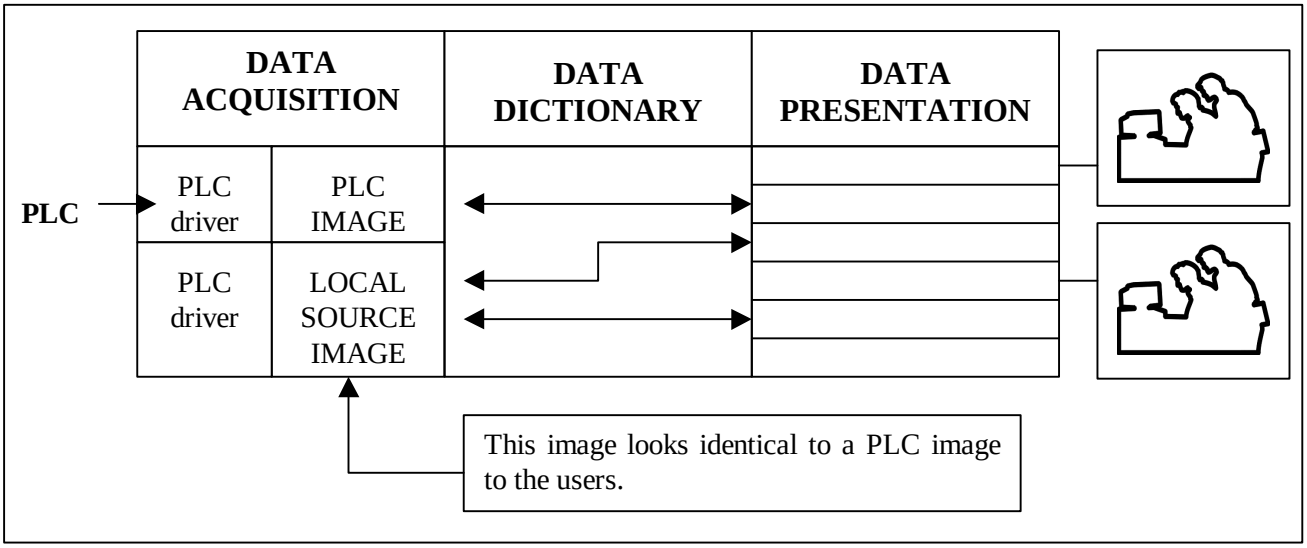
TXT Attribute

The Text view files all have a standard attribute called TXT. This is equivalent to the PV of the view.

3.11 Local Source (Using type 6)

All About the Local Source

Very often, in process control systems, there is a need for entities that have no input or output. For example, a simulation entity or a reference entity. There is no need for this kind of entity to be loaded into a PLC or DCS system. The virtual source is a source whose registers reside within the ***MacroView*** system itself and behave in exactly the same manner as a real world PLC system.



In summary then, the LOCAL source:

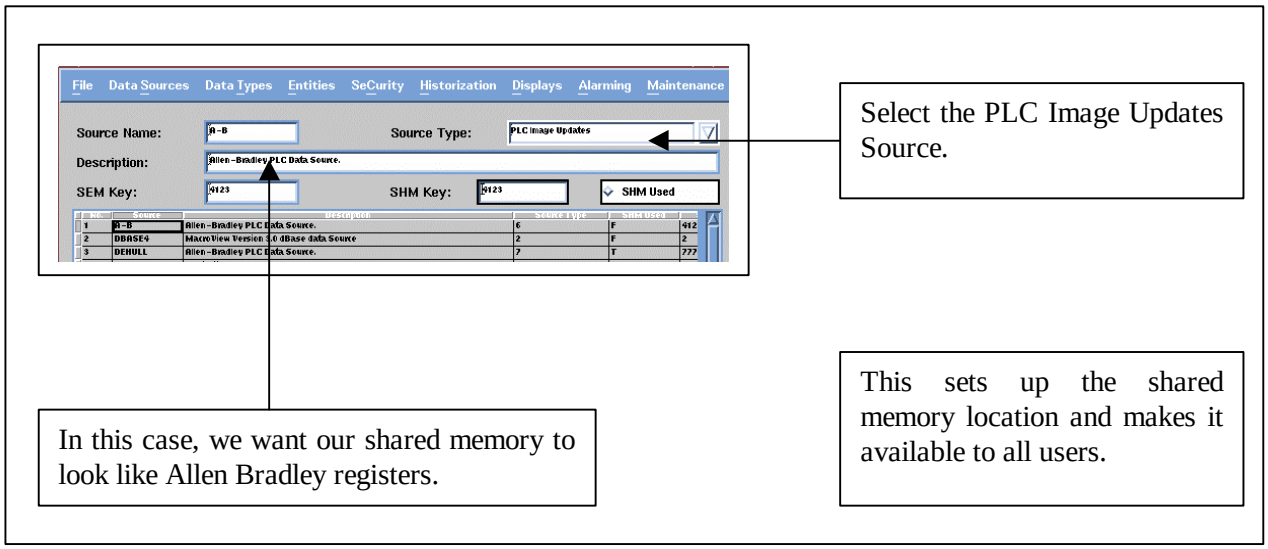
- Acts like a PLC source but has no input or output to the real world.
- Is set up in a similar way to a real world source - i.e. SOURCE, REGISTER, ADDRESSING etc. This means you can use your existing types and templates in exactly the same way as an actual PLC i.e.

    1. Because there is no input or output, the source entities rely on programs such as dBASE, Shell Script or Lotus programs to set their values.

    2. You set up various parameters (such as how often the image is saved to disk) in the start up script of the source.

Configuring the Local Source

- Loading the source - You do not need to load the local source because it comes with your ***MacroView*** system.
- Adding the source to the system - Start up the Engineering Configurator, then choose Maintenance, Sources and then Add. Now follow the directions below:



- Starting the Local Driver

Just like any other driver, you need to include the virtual source driver in the start up script. The location and name of this script is typically as follows:

For UNIX systems - `$MACRODIR/../system/start.modicon`

For Windows NT systems - `%MACRODIR%\System\MVstart.NT`

Each of these files contains information on starting the ***MacroView*** programs required for normal operation, e.g. Alarms and History managers etc, and any programs you require for ***MacroView***. The instructions, found in the start file, for starting a Local Source driver would be as follows:

For a UNIX system

The driver must be started from the configuration directory.

```
cd $MACRODIR
../bin/localsrc    A-B    T1    C2 F 3 > /dev/null    &
```

The execution program

The source name

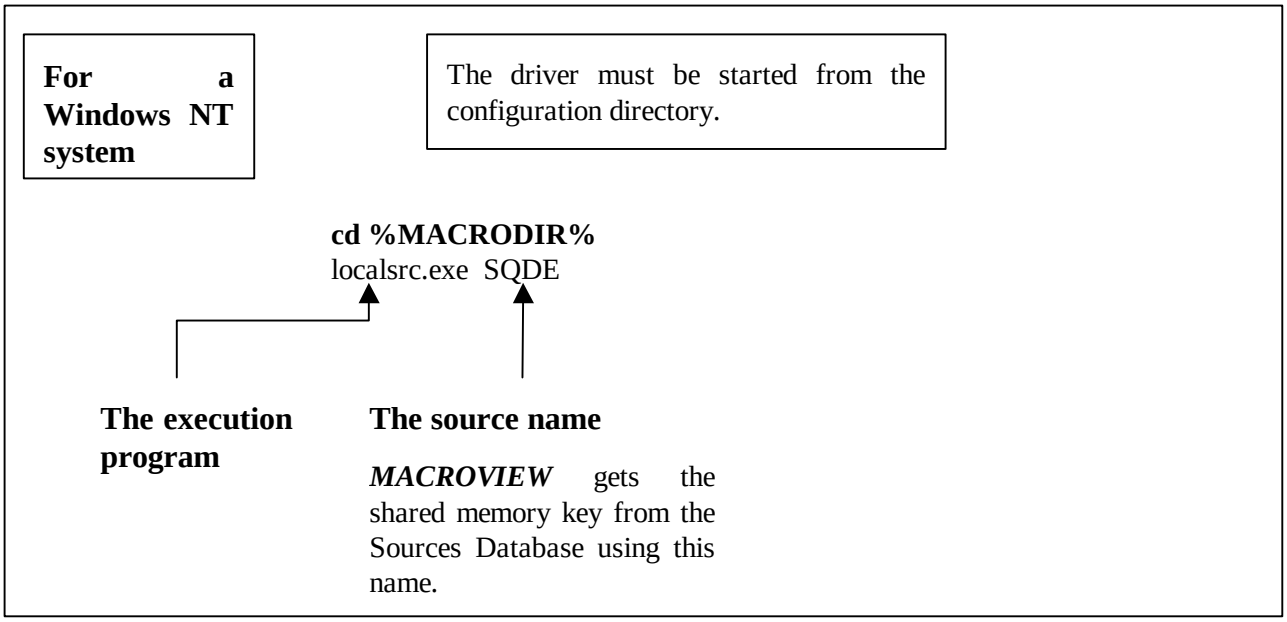
MACROVIEW gets the shared memory key from the Sources Database using this name.

Various parameters

That specify the format of the internal registers. See the following section for more details.

Any diagnostics messages go here.

In this case, they are discarded.



NOTE :

The specific details on Driver installation and configuration are covered in the Driver installation document, which is delivered with the driver software.

The optional arguments for the Local Source program can be displayed by simply typing localsrc at the command line prompt. For your reference, the format for UNIX and Windows NT platforms is:

For a UNIX System

`localsrc driver_name -nap <ms> -store <secs> <reg_type reg_size>` where:

Argument	Description
`driver_name`	The same name you specified in the sources configuration.
`-nap <ms>`	The program must nap in between write requests otherwise it will use up too much processor time. Set this to at least 1000.
`-store <secs>`	The program periodically writes a file to the hard disk at this interval so that at boot up, there is an up-to-date image of the entities.
`<reg_type reg_size>`	The default width of registers is 16 bits. If you want to override this, you use these arguments. The register types in this example are: C Counters T Timers F Floating points The register sizes are: 1 16 bits 2 32 bits 3 48 bits etc.

Argument	Description
	In the example above, as we are simulating an A-B PLC, we have used these arguments to reflect the A-B memory.

For Windows NT System

Typical Usage: `localsrc <src_name> [-v] [-nap <ms>] [-store <secs>] [[<reg_type reg_size>],..]`

Argument	Description
`<src_name>`	Specifies the name of the data source that the program is representing.
`v`	Prints this banner.
`-nap <ms>`	Specifies the time in milliseconds between the processing of write requests.
`-store <secs>`	Specifies how often the shared memory image is to be written to the `<src_name>.sts` file for crash recovery reasons. 0 indicates no saves.
`reg_type <reg_size>`	Optional list of register type code and size pairs. Default register size is 1 (ie 16 bits).

Configuring the Local Source Retrieval

You set up the local source in exactly the same way as a normal PLC Image driver. I.e. you need to:

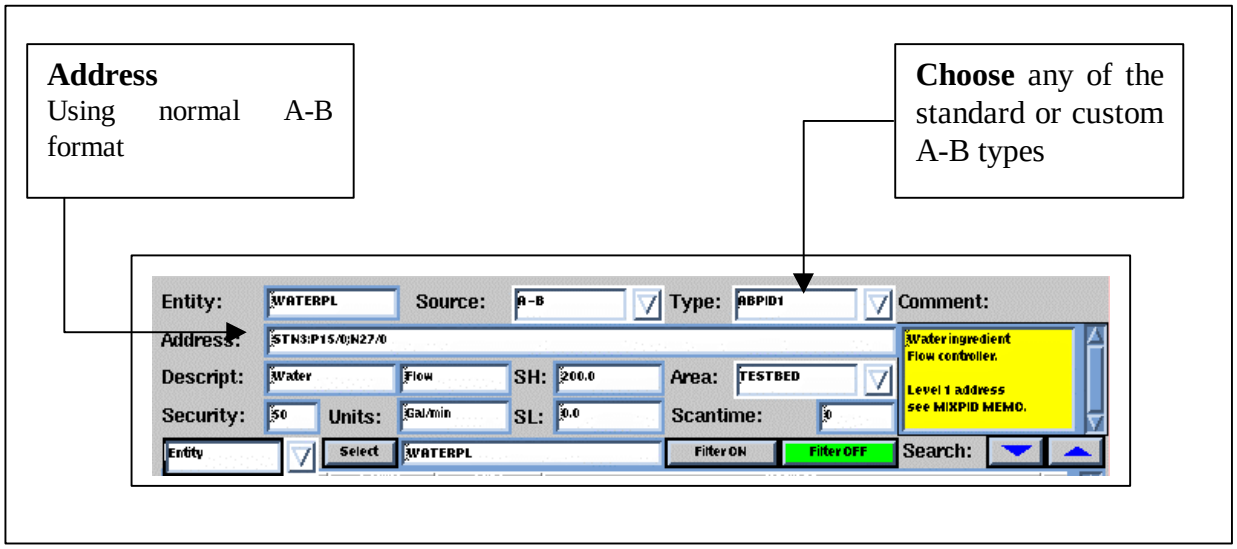
1. Set up the Station Name, and Source Name (as defined in the start up script.) Use Data ***Sources:PLC Station:Detail*** to do this. (Note that you do not need to add a route address.)

2. Now set up the register blocks once again exactly as you would set up a normal PLC image source. I.e., use ***Data Sources:PLC Register:Detail*** You would:
   - Use the mnemonic defined in (i) above.
   - Set the register types as for a normal PLC (e.g. for analog, use A).
   - Set up the start and end addresses to simulate the PLC as before.

Adding the Entities

A typical entity would look like this: (Note: we have used A-B as an example)

Configuring a Local Source Entity



Checking Out the Local Source

1. You can type `plcstat` from the command line prompt and select the desired local source register block. You should see the registers displayed as if it were a normal PLC.
2. Now start up the operations program and call the entity WATERPL (in this example) to the scratchpad. You should see the PV.
3. Now call up the detail display and you should see the detail of the entity in A-B format.
4. You can now set the attribute values using the usual method, or write a dBase, Lotus or Shell Script program to set the values. These changing value are indicated in the PLCSTAT program display.

3.12 Simulated Source (Type 1)

The simulated source is a very simple source that is designed to be used primarily to test graphics. The heart of the simulator is a random number generator. The simulator essentially responds to the individual requests from the user by providing randomly varying numbers. It does however, treat some special attributes differently. For example, the Manipulated Variable, MV (or output), ranges from 0 to 100% (as we would expect). The way in which these special attributes are treated, is shown in the table below. All other attributes are simply varied in a random fashion between SCALE HI and SCALE LOW if they are analogue and if they are digital, they are switched between the High and Low state in a random way

Table 2: Simulator Special variables

Attribute	Description	Simulation Method	Value Changes	Value Range
PH	Process High Limit	Analog	No	75% to 100% of Scale
PL	Process Low Limit	Analog	No	0% to 25% of Scale
HH	High High Limit	Analog	No	75% to 100% of Scale
LL	Low Low Limit	Analog	No	0% to 25% of Scale
OPHI	Operating High Point	Analog	No	90% to 100% of Scale
OPLO	Operating Low Point	Analog	No	0% to 10% of Scale
DL	Differential Limit	Analog	No	0% to 25% of Scale
MH	Manipulated Variable High Limit	Analog	No	75% to 100%
ML	Manipulated Variable Low Limit	Analog	No	0% to 25%
P	Proportional Band Constant	Analog	No	0 to 100
I	Integral Band Constant	Analog	No	0 to 100
D	Derivative Band Constant	Analog	No	0 to 100
LS	Loop Status	Special	Yes	"AUT" or "MAN"
ALRM	Alarm Status	Special	Yes	"NR" or "HI"
ALM	Alarm Status	Special	Yes	"NR" or "LO" or "HI"
MV	Manipulated Variable	Analog	Yes	0% to 100%

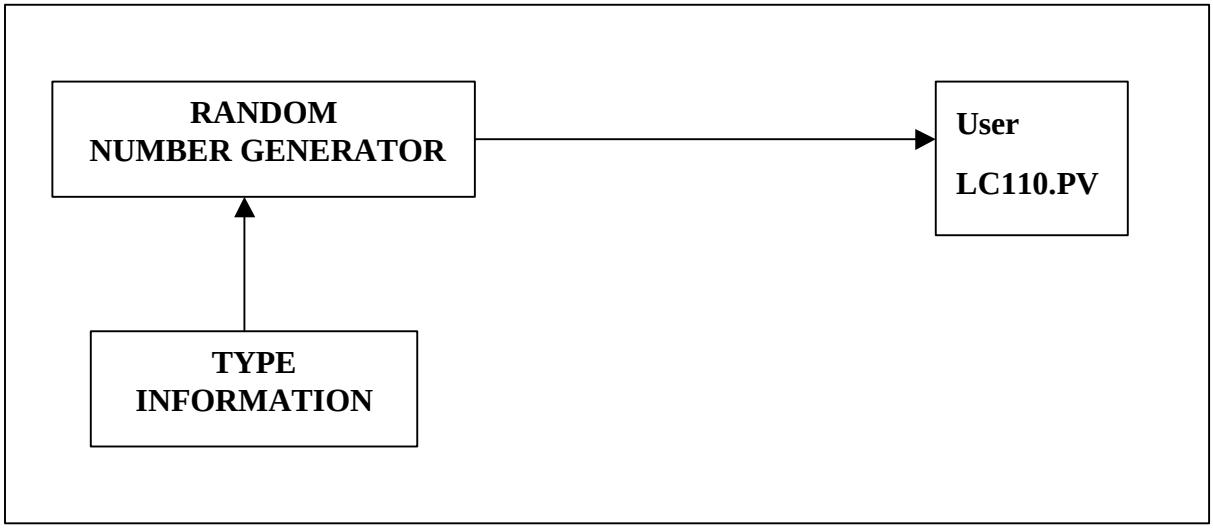
Note: Because this is a very simple simulator, there is no link between one update on the screen and a second update that is displaying exactly the same attribute. Clearly, this is not realistic but it must be remembered that the primary function of the simulation source is to test graphics and not to portray realistic plant situations. Realistic process simulations are best done with the meta script facility discussed in the meta script chapter.

Configuring the Simulated Source

It is extremely easy to set up an entity to act in a simulation mode. Just set the source to Simulated Dated Source (Type 1) and the operations program will show the attributes varying in a random fashion.

3.13 Simulated PLC Image Source (Type 7)

The Simulated PLC Image Source (Type 7) is different from the Simulated Data Source in that it uses the Type information structure to set up the random number generation.



- The simulation has access to the conversion data in the type attribute database. It therefore has an indication of such information as the range of the attribute. It can provide simulations of scaled analogs, integers, floating point values, digital status bits, etc.

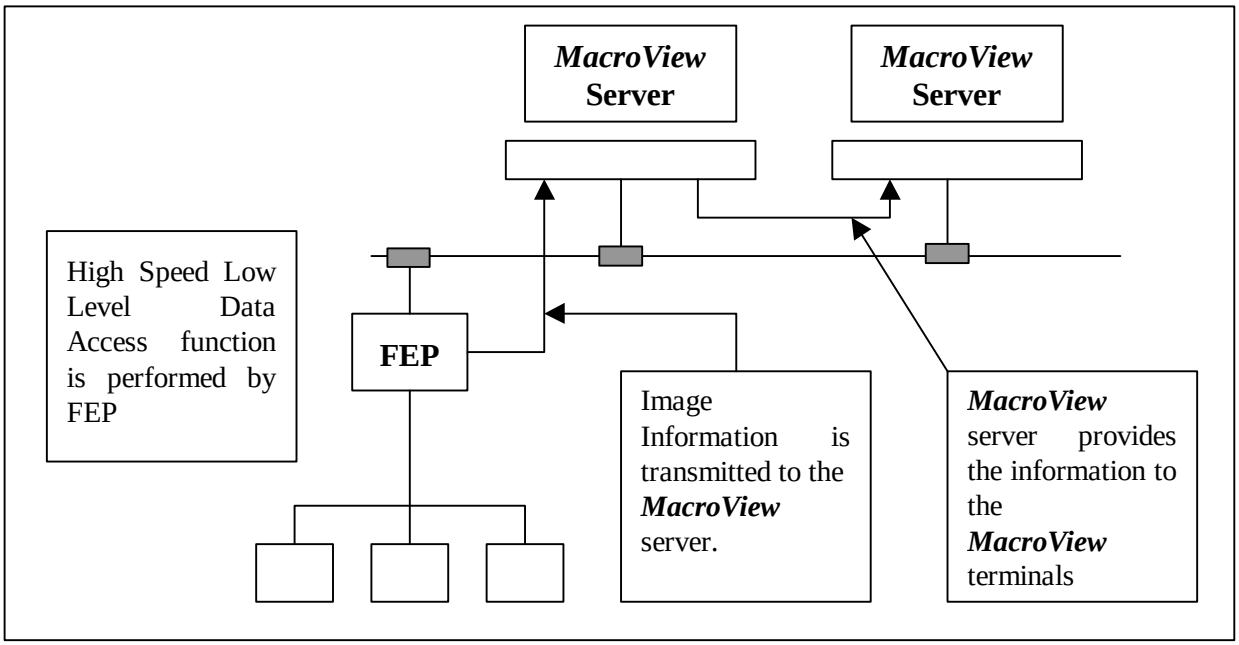
Configuring the Simulated PLC Source

Configuring an entity with a simulated PLC source is simple.

- Just select the Simulated PLC type and the entity attributes will start moving in a random fashion.
- You must, of course, have valid types defined.

3.14 Sources in the Front End Processor (Usually Type 6)

The Gateway or Front End Processor (FEP) is a device located remotely on a LAN that performs a high-speed data gathering function with an external device. An image of the external device data is held within the FEP and this data is continuously sent up to the ***MacroView*** server. Please see the diagram below:



The Front End Processor is:

- Usually a DOS based rack-mounted industrial computer with no screen or keyboard.
- Typically running a PLC image based driver.
- Typically linked to the server via Ethernet.

The advantages of the FEP approach as opposed to running the driver in the server are as follows:

Off-loads Server

The FEP off-loads the highly interruptive and processor intensive low level program from the server, thus giving the server more time to perform higher level tasks.

Specialist Cards

The FEP, being PC based, supports a range of specialist cards such as the modbus plus card. This means that a high data throughput can be achieved and the server hardware need not be dependent on what cards fit into the server.

Remote Connection

The FEP may be mounted remotely (i.e. close to the external devices) thus eliminating the need for long high-speed communication lines.

Specialist Cards

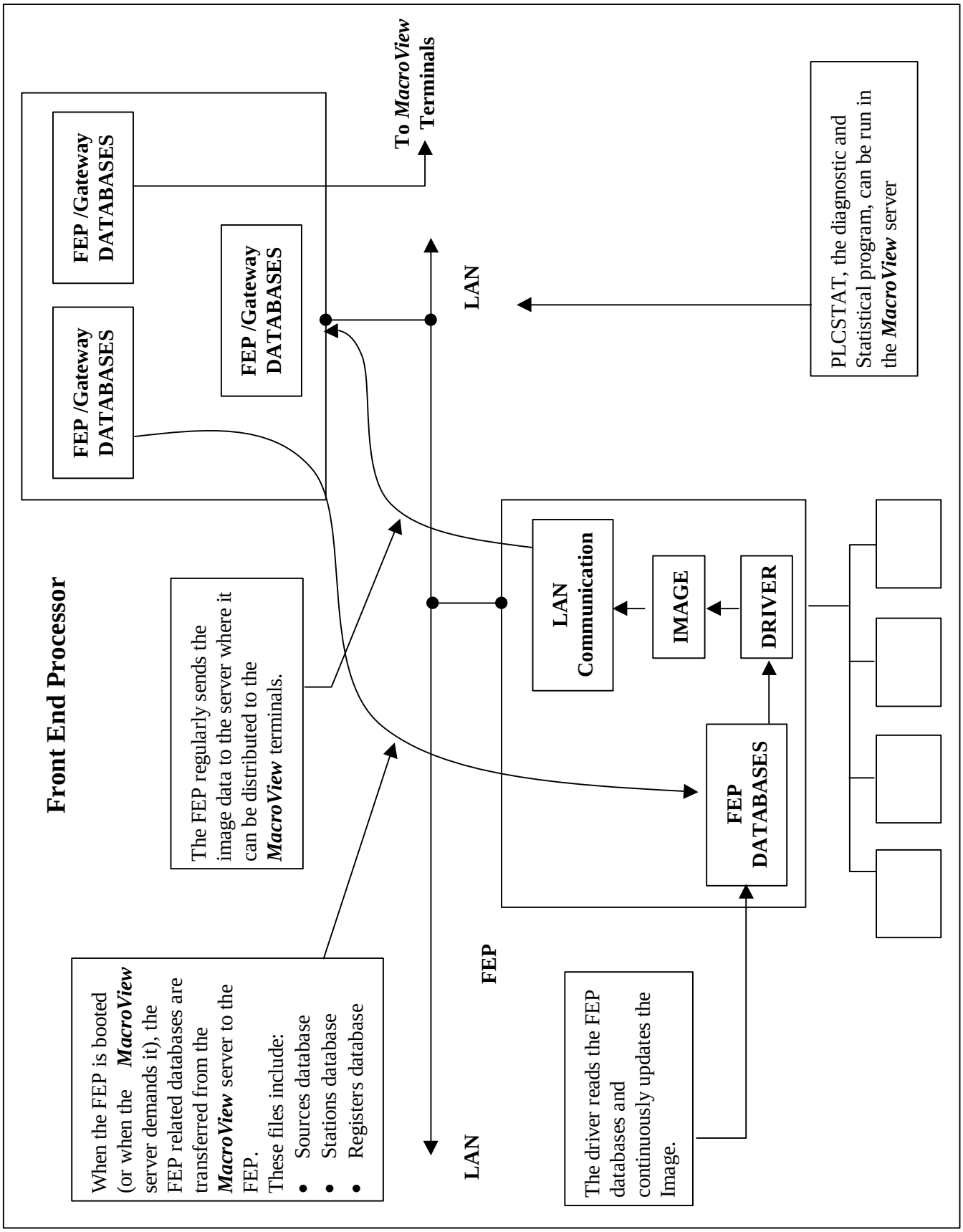
The FEP may serve multiple ***MacroView*** servers on the same LAN. The FEP operates in a true client/server role.

Efficiency

The FEP, because it is serving multiple servers and multiple ***MacroView*** terminals, reduces the loading of the PLC data bus.

FEP Functions

The diagram on the following page shows the mechanisms of data transfer of the Front End Processor or Gateway in more detail.


Front End Processor
FEP /Gateway DATABASES
FEP /Gateway DATABASES
FEP /Gateway DATABASES
To MacroView Terminals
LAN
PLCSTAT, the diagnostic and Statistical program, can be run in the MacroView server
The FEP regularly sends the image data to the server where it can be distributed to the MacroView terminals.
LAN Communication
IMAGE
DRIVER
FEP DATABASES
When the FEP is booted (or when the MacroView server demands it), the FEP related databases are transferred from the MacroView server to the FEP.
These files include:
Sources database
Stations database
Registers database
FEP
LAN
The driver reads the FEP databases and continuously updates the Image.

Installing and Starting up the FEP

- During installation of the FEP, a screen and keyboard are connected. The software is loaded and, in the start up batch file, the location of the configuration files is defined. (I.e. the ***MacroView*** server LAN address and FEP directories on the server.)
- On the server, the relevant types, group and detail faceplates are loaded. The installation software also adds the Source entry to the Sources database.
- Starting up the FEP is simply a matter of re-booting the machine (You would only do this once you have defined the station and register addresses).

Note: For more details of the installation procedure, please consult the specific FEP driver document.

Configuring the FEP

Since the FEP normally operates with a PLC Image based driver, the configuration of the FEP is very similar to the normal PLC type configuration.

The configuration is carried out in the server and the files are transferred to the FEP on re-boot.

The configuration consists of:

1. Modifying the Source default entry if desired.
2. Configuring the Stations on the FEP network.
3. Configuring the Register Blocks that the FEP will be reading.

As mentioned, the configuration is very similar to the PLC Image based driver, however there are minor differences. The following section discusses the configuration in more detail.

Configuring the FEP Stations (Sources Station, Route)

Source

What you type

Select the Source from the pull-down combo

How it Works

This tells ***MacroView*** what driver to use.

Example

SQUARE-D, A-B, MODICON.

Station

What you type

Enter the name of the station.

How it Works

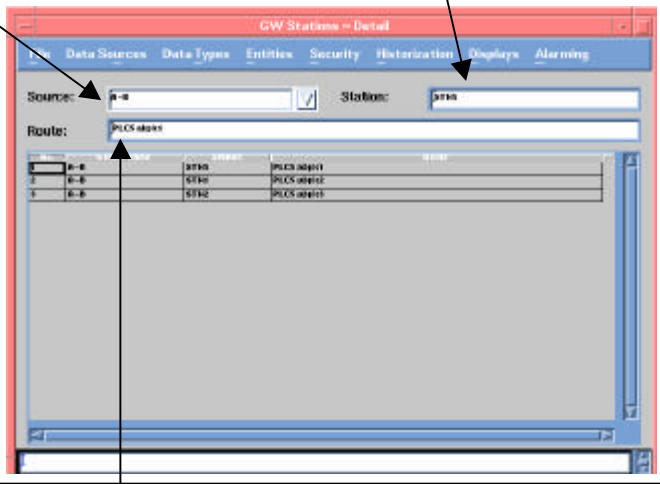
This name is used as a mnemonic by the driver. If the entity address is MON: 0400 then the driver substitutes the route address for MON to make up the complete address.

How to get there

1 ***Data Sources: GW Station:Detail*** This brings up the Station detail screen.

2 Click on the Station to be modified: The source detail will appear in the window.

3 Alternatively, you may add a blank record using ***Data Sources:GW Station:Add Blank*** and edit the new record.



Route

What you type

Enter the address of the station on the network. This is specific to the driver used. You should consult the driver document to determine the format of the route for the specific station.

Configuring the FEP Register Blocks

Here you define the location and type of the register blocks i.e. Station, Start and End address and Type

Station, Start Address, End Address.

What you type

Enter the Station name, Start address and End address of the block.
E.g. STN3, 0,400 for a register block called STN3 starting at register 0 and ending at register 400.

How it Works

The driver cyclically refreshes the internal image of these registers at a rate dependant on the Normal and Demand Scan time.

Hint

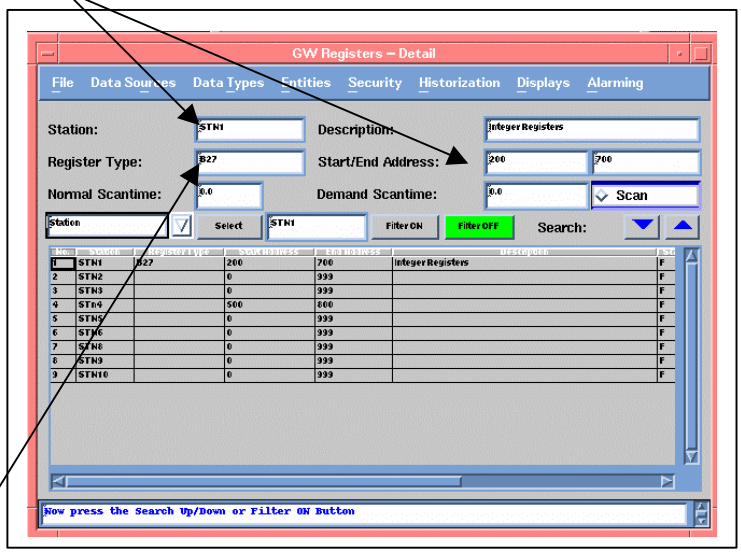
To increase the update speed, try to organize the data into fewer but larger register blocks You can of course have several blocks per station.

How to get there

1 ***Data Sources: GW Register:Detail*** This brings up the Station detail screen.

2 Click on the Station to be modified: The source detail will appear in the window.

3 Alternatively, you may add a blank record using ***Data Sources:GW Register:Add Blank*** and edit the new record.



Type

What you type

Enter the register type:

Example

A for analog.
N1 for file type N1.

How it Works

PLC manufacturers store different types of data in different formats. For example, counters and timers are stored in a different way than digital inputs. The driver needs to know how they are stored. Again consult the appropriate driver reference sheet for specific examples.

Configuring the FEP Register Scanning

Normal Time

What you type

Enter the Scantime in seconds for normal update rate.

Example

4.0 - Update image of these registers every 4 seconds.

How it Works

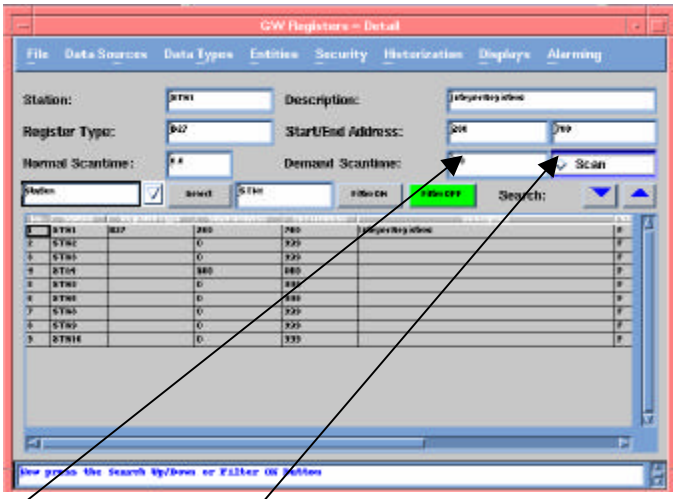
The driver refreshes the internal image of these registers at a rate dependant on the Normal Scantime as long as registers are not demanded.

How to get there

1 ***Data Sources: GW Station:Detail*** This brings up the Station detail screen.

2 Click on the Station to be modified: The source detail will appear in the window.

3 Alternatively, you may add a blank record using ***Data Sources:GW Station:Add Blank*** and edit the new record.



Demand Scan time

What you type

Enter the Scan time in seconds for registers being demanded.

How it Works

2.0 - Update internal image every two seconds.

Example

The update rate usually needs to be more frequent for registers being demanded in operations.

Scan

What you type

Select the check box ON or OFF to indicate that you wish the driver to scan the block of registers. A PLC program can decide when to inform the FEP of register block values. Check the FEP documentation for the particular driver of interest.