# MICROCOMPUTER WORKSTATIONS.
# THE PATH TO FULL INTEGRATION.

Richard Linnett
Cognos Inc. Ottawa Ontario Canada.

## Summary.

With the entry into the market by major computer manufacturers, microcomputers came out of the hobbyists den and became a legitimate tool in the office environment. A large number of microcomputers have been installed to handle small applications within an office - word processing and spreadsheet type applications. To a large extent these microcomputers are completely self contained, any data required from existing minicomputer systems being promoted into the system manually.

Many microcomputers are now being used not only for standalone wordprocessing, but also as replacements for terminals connected to central minicomputers or mainframes.

Exchange of data between minicomputer and microcomputer is becoming a major requirement - whether it is the extract of data from the corporate database for inclusion in a wordprocessor report, or the transfer of files between the two environments.

We are now seeing tools on the market that address the data transfer requirements. Within the next few years we should expect to see application development tools on the market that go far beyond file transfer to full integration.

This paper looks at the state of microcomputer integration, and paints a broad scenario of how we can expect the situation to change over the next few years as we move towards true integration.

## What is MicroComputer Integration?

Much attention is being paid to the integration of software packages on the microcomputer, and we now have a number of integrated wordprocessor, spreadsheet and communications products available. These products, despite the criticisms, have made a step in the direction of integration on the microcomputer. However, the integration stops at the microcomputer, there is no real integration into the minicomputer world.

Integration of microcomputers into an office minicomputer environment entails far more than an easy to use front end, a full distributed environment is needed where:

The minicomputer and Microcomputer work together in processing an application. A micro can typically support far better end user interaction than a minicomputer. A minicomputer can support larger databases and can control sharing of data across multiple users far better than a microcomputer. To be fully integrated, these capabilities must be used to the best ability.

Load sharing between minicomputer and microcomputer has to be intelligent - load sharing should be self balancing depending on the load of the systems at the time of processing.

There will always be a number of dumb terminals in use, or times when the central minicomputer system is unavailable. An integrated mini / micro environment must also be able to adapt to this.

Data exchange between microcomputer and minicomputer will not be on a "batch" basis as with file transfer, but will be on a transaction basis.

Data in the environment has to be available using terminology that the user understands. A data dictionary will have to be powerful enough to shield the user from the intricacies of Image database structure, allowing dynamic user views of data to be developed independantly of where or how data is stored.

Much of this could be achieved now with extensive coding at both mini and micro to develop an integrated application, but the magnitude of the work required precludes it. Future developments in application development software will make this intelligent use of all resources available to all users.

Where Are we coming from?

Microcomputers and minicomputers cannot really be considered as being integrated now. The uses of both environments is to all intents, completely divergent.

We have a very traditional environment in the minicomputer world:

There is still has much faith in COBOL, structured programming techniques and complex databases.

The minicomputer DP department suffers from the same problems as a large mainframe department - large maintenance loads and long backlogs of work requests are making the department appear very unresponsive to user demands.

These backlogs are causing the DP department to move from COBOL to products that can aid productivity. But even with the increased turnover in development, the backlog is not being reduced because more and more requests are being added to the backlog.

Many DP departments are experimenting with, or actively encouraging users to develop their own ad hoc requests using the same development tools as the DP are now using. However, with data in the corporate database normalised in a way that makes for efficient processing, it is not readily meaningful to the user.

The microcomputer environment is very much at odds with the structured minicomputer world.

Small spreadsheets or Basic programs are developed as needed and thrown away just as quickly.

A knowledgeable spreadsheet user can build very complex models that reflect an entire operation, and that model can be adjusted quickly to incorporate any operational changes that occur.

As long as these two worlds stay apart, the different methodologies can co-exist, the distain of the professional DP professional for the ignorant microcomputer user balanced by the distain of the microcomputer user for the DP department that never gets anything done.

Where are we now?

With few exceptions the level of integration between microcomputer and minicomputer is now limited to the downloading of data from the minicomputer for processing in a microcomputer application. Typically this involves:

A mini program - COBOL or possibly a report writer extracts data from the corporate database and build a file on the minicomputer.

Transformation of data into the optimal format for use by the microcomputer is not always attempted. Flat ASCII or DIF files are the common transfer medium.

Actual data transfer is managed using a terminal emulation package. The microcomputer, which may be capable of 750,000 instructions per second does no more than wait on a terminal port and transfer the occasional character to a disk file.

The spreadsheet user then has the data available for use.

There are many problems with this situation:

A program needs to be written on the minicomputer to extract the data. That requires knowledge of the host system and frequently scheduling of work by DP staff to write the programs.

Communications is a black box to most people. To configure a terminal emulation package to match the protocols used by the host system requires knowledge that most users do not have and do not care to have.

Transformation of the data to ASCII or DIF formats looses format information. When loaded into the destination spreadsheet, the user frequently sees misaligned columns and truncated values.

Very few sites look at the uploading of data back to the minicomputer as a viable option. Considerable problems exist with data integrity if significant portions of a database are offloaded from the host to distributed microcomputers.

Although these integrity problems preclude file transfer as a viable basis for distributed processing, there will be instances where the nature of the data allows for file transfer. As an example, a corporate budget file could be safely distributed across the individual cost centres assuming that a cost centre could only update that cost centres budget.

By limiting the process to data download only, many of the problems of data integrity can be avoided. For that reason, the first products that we can expect to see appear on the path to shared processing will appear in the guise of more efficient extract programs than available now.

As previously, mentioned, a microcomputer acting in terminal mode wastes a large amount of processing power. The more powerful microcomputers that have become available during the last year, and microcomputers based on announced but not yet available processors will have even more spare processing power during what is basically terminal mode.

By moving to multitasking (if not multiuser) operating systems, the option will be there to move a file transfer program into the background:

> This will allow the user to continue with micro based work during the time transfer takes.
>
> By hiding the file transfer inside such an environment, probably with an attractive windows interface, the microcomputer will become more effectively used than now. This is however very much a dead end.
>
> Nothing is done using this approach to offload the host, and the problems of currency control are not addressed.

The alternate, and preferable approach to using the spare microcomputer power is to offload the host system as much as possible.

> Definition of the extract request will be offloaded completely from the host system. By formatting and editing the request on the microcomputer, considerable savings in host resources can be made while improving the user interaction.
>
> There is no need for the host processor to use valuable cycles to reformat the data into a microcomputer format. That work can adequately be managed during the time that the microcomputer is waiting for input.

Together, the two approaches provide a major advance in what is available now.

Concurrent with the improved data extract packages we can look towards intelligent terminal emulation for microcomputers that do more than simple terminal emulation.

Many users are now using microcomputer terminal emulation products to provide an easy to use front end to the HP3000. The command language being used to control access to the host, providing autologon facilities and navigation through the available applications by a series of menus.

We should be looking for combined minicomputer / microcomputer programs that take this menu driven approach a step further. As microcomputer users begin to require more access to the minicomputer, some integration of minicomputer and microcomputer operating system commands is needed. This could take the form of:

a menu driven option to MPE commands, much along the lines of PAM on the HP150.

menu driven equivalents to utilities such as FCOPY.

Beyond such an enhanced terminal emulation feature, we should be looking for intelligence being distributed to the microcomputer during application processing. With such a front end, it will be possible to make significant reductions in host processing and improve the user interface dramatically. It is possible to offload the following:

Screen formatting - boxes, windows etc require a considerable amount of communications traffic in addition to the host processor time. Major improvements in turnaround could be managed by this process.

Data entry editing. If a field is known to be numeric and within a certain range, it would be reasonable to make such checks on the microcomputer before transmitting the data to the host. By keeping the processing local until database access is required, the microcomputer can be used to its best, providing a fast effective front end to the host database machine.

Even with all data remaining on the host system, a significant housekeeping process is implied, as meaningful savings require screen formats and editing rules to be downloaded to the microcomputer. Changes in the central system would need to be propogated to all microcomputers when they use the system.

It is likely that a fourth generation product will be enhanced to provide this level of integration in a fairly automated manner.

These product initiatives would seem to be fairly simple at a first glance, they can be implemented using existing technology. They do however, provide a major improvement in the utilisation of microcomputers in the minicompter environment and set the basis for further product developments.

Further advances require developments in the following areas:

A Data dictionary will need to be available to the environment that goes beyond the simple file and element descriptions that are in use now:

> The data dictionary is going to need to be far more active than now. All data access will need to be routed through the dictionary if dynamic data formats, as required by the microcomputer user are to be supported.
>
> The data dictionary will need to support distributed data, and the dictionary itself will need to be distributed and replicated across several systems.
>
> If central data storage is to remain in complex data base formats such as IMAGE, the dictionary is going to need to hold information necessary to navigate across the database.
>
> User views of data, based on multiple file access and selection criteria will need to be available.
>
> Database security and integrity are major requirements of the DP department. Strong controls will need to be available in the dictionary.

The same application development language will need to become available on both the minicomputer and the microcomputer.

> This may not necessarily entail mirror images of the language being available on both systems. There are many features required of a language by a DP guru that are not at all relevant to the microcomputer user. Within limits, the fine tuning commands that the guru needs only lead to confusion and rejection of the language by a no DP user.
>
> Despite the question about how exactly a microcomputer language duplicates the minicomputer language, there is a strong need for the ability to develop and run the same application on both mini and micro. Many small applications would benefit from being transfered to a microcomputer, with the ensuing release of minicomputer resources.

Application development is frequently a resource hog, and the availability of a microcomputer development environment is a good way to reduce the mini load.

A question does remain about the power of microcomputers being sufficient to fully support a development language that was based on the size and capabilites of a virtual memory minicomputer.

Communications is an area that requires some change before the true integrated environment can develop:

Standards in data communications are needed if the integrated environment is to be available freely. This is happening now with the OSI layered protocol being supported in more and more products. The IBM LU 6.2 peer to peer protocol will also drive more standardisation into the communications world.

Packet switch networks need to become more widespread. The cost reductions for long distance connections and data integrity available through an error checking link are necessary.

Error checking and handling need to become more a part of the communications environment, allowing some reliance on data transmission. X25 networks and the newer error checking modems are beginning to provide this integrity.

The current mess of inter microcomputer communications needs to settle before much progress can be made. Few software developers can make significant progress in development of network software until some standardisation happens, and until the software developers begin to adapt to the networks real multiuser versions of microcomputer software will not be available.

## The Fully Integrated Workstation.

Given the required enhancements in the data dictionary, the "universal language", strong communications and microcomputer power we can look for applications software that will allow the power of the microcomputer to be integrated with the power of the minicomputer. We will then have a truly integrated microcomputer workstation:

Applications will be developed using a common language. It may be that the microcomputer user has a series of screen painters and application generators while the DP expert has a concise syntax that allows for very exact specifications, but the base syntax will be the same.

Once developed, the application will be able to run on either microcomputer or minicomputer, depending on the data volumes and the scope of the application.

If an application is implemented on the microcomputer, it will be possible to port the application to the minicomputer as data volumes grow.

If an application is run on the minicomputer, the microcomputer terminal will offload as much of the terminal management as reasonable. The same application will be run side by side on a terminal and a micro, yet the microcomputer will provide much faster response times, and an easier interface than the pure terminal.

Truly distributed processing will be possible, with data available locally on the microcomputer, from a departmental minicomputer, or even through the local minicomputer to the central corporate mainframe database.

With standardisation of communications links, and program to program communications across a link, it will become possible for programs from different vendors to be integrated to a level only dreamed of today.

Microcomputer software such as a spreadsheet or a word processing package will be able to access data from the central host system easily and without need for host system knowledge. Just as it is possible to move data from the word processor part of an integrated microcomputer package into the spreadsheet component, it will be possible to extract from the host database into the word processor and spreadsheet.

The two components, microcomputer and minicomputer will be connected through a very strong data communications link, and that link will provide an intelligent highway between the two.

Biography.

Richard Linnett has a wide background in the technical aspects of Data Processing, gained over eighteen years of practical experience with a number of organisations.

Richard Linnett has been with Cognos Inc for four years. During this time he has been responsible for the implementation of a number of projects to integrate different hardware and software environments for clients with mainframe, minicomputer and microcomputer applications.

He is now a Project Manager in the HP Marketing and Development Division of Cognos.