Software Licensing Basics

Issues arising in technology transfer are reviewed; paper based on workshop information exchange

BY CHESTER E. MARTINE JR.*
and EDWARD G. TUTLE**

COMPUTER SYSTEMS

A simple computer system is a combination of hardware and software that is capable of receiving an input, processing the input in accordance with specified instructions, and producing an output in a form desired by the user.

The hardware includes machine elements that physically receive and store the input, that actually perform calculations or other manipulations to process the input and that produce, store and deliver the output to the user.

The hardware of a computer system is useless without computer programs (the "software"). Software provides the specific instructions that control the step-by-step operations of each element of the hardware. Software is generally divided into two categories; first, system software (or "operating system") controls the operation of the hardware in a generic sense, without regard for the specific function for which the user is using the system; second, application software is selected by the user to work with the system software and enable the system to perform desired functions.

The hardware of the simple computer system includes a central processing unit (CPU) comprising a control unit and an arithmetic unit. The control unit interprets the system and application program instructions and directs the sequence of operations as determined by those instructions. The arithmetic unit performs arithmetic functions, such as "add" and "subtract," and performs logical operations, such as "true/false," "and" etc., under the direction of the control unit. The arithmetic unit has its own high-speed memory (called "registers") in which the results of the arithmetic and logical operations are stored during processing. The CPU also has an internal (or "main") memory, also relatively high-speed, that stores the system and application programs and the results of computations performed by the arithmetic unit.

Because the size of main memory is limited, peripheral storage devices are connected to the CPU for storing data that the CPU needs or produces and for storing application programs that are not in current use. The peripheral storage operates slower than main memory, but costs less and can store much more data than main memory. The system software selects and controls the storage of data in the peripheral storage devices.

"Firmware" can be part of the computer's operating system providing special characteristics or "personality" to a generalized computer system. These are usually fixed, or factory alterable, memory devices causing specific operations or outputs to occur for specific users. Firmware may be necessary to enable specific users to run certain applications software within a generalized computer. An example is a specific readout upon interrogation commands resulting from the introduction of an object program. The program can be encrypted to test if the computer is the one for which the program is licensed.

The hardware also includes imput/output devices, which can simply be a standard terminal that has a keyboard (input) and a video screen (output). These devices are used to enter data and requests for output into the system.

Please see the "Computer Subject Glossary" for terms used here and in other computer texts.

OPERATING SYSTEMS

Operating systems are divided into groups according to whether they permit more than one operator to use the computer system, whether such use can be simultaneous and whether many functions can be performed by the system. For instance, in the simplest case two operators can take turns. This approach is called "scheduled access." The operators themselves do the scheduling. At the next level of complexity, the operating system schedules access to the computer. In a "batch" operating system, many users submit their jobs to the computer at once and the computer then executes these jobs as quickly as possible, one after the other. By doing the scheduling, the computer minimizes the time between jobs.

However, CPUs are so fast that when dedicated to one job they spend very little time actually computing. One solution to this efficiency problem is "multistream batch" that allows multiple jobs (batches) to run apparently simultaneously by having one job use the CPU, another the peripheral storage, etc. at the same time.

In "time-sharing" operating systems, the CPU devotes a given amount of time to each operator's program in sequence. There are two kinds of time-sharing. The oldest and least efficient is called "swapping." When job A's turn has expired, it is transferred out of main memory onto peripheral storage. Job B is then read from peripheral storage into main memory, executed for a time, and then

**Attorney at Law, Boulder, CO.*
***Manager, Technology Evaluation and Transfer, Martin Marietta Orlando Aerospace, Orlando, FL; paper based on workshop at LES U.S.A./Canada Annual Meeting, October 1984.*

returned to peripheral storage, so that the computer can turn its attention to job C. When job A's turn comes around again, it is read back into main memory and execution is resumed. The disadvantage of swapping is that a significant portion of the computer's time is spent in switching from job to job (taking programs and data to and from storage) — a process during which no actual execution of the users' instructions is carried out.

A modern method of time-sharing is one in which all the jobs to be time-shared remain resident in internal memory until they are completed — whether or not they are actively being executed. With the decreasing cost of memory, this has become the more common method.

"Event-Driven"

An operating system that can respond to events as they happen is "event-driven" (called a "real-time" operating system). This allows external devices to input data and process the data when the event occurs. Even time itself can be used to drive the system; in this case an electronic signal occurring at a given rate (called a "real-time" clock) is the "event" to which the system responds. Real-time operating systems come in simple and complex varieties. An event-driven system dedicated to a single task ("job") simply waits for events to occur and then does what it is supposed to do (such as perform a measurement or a computation). A multitasking or multiprogramming system is one that attempts to cope with several tasks at the same time by interspersing their component computational parts and by scheduling access to system resources. In this case, requests for computing must be coordinated because they are driven by external happenings in the real world.

Sooner or later conflicts will occur between jobs contending for the same resource at the same time. At this point, the only solution is to allocate resources according to some pre-defined priority, favoring one task while sacrificing others. The favored job is served immediately, but lower-priority tasks will not get processed until the other job has been finished. It is up to the system designer to ensure that conflicts are resolved quickly enough and that even if there are delays, the system in general "keeps up."

Besides permitting sharing of resources, a useful operating system will provide other services essential to the operation of a shared computer system. The operating system can protect individual users and their data from one another and from intruders by requiring use of passwords. An operating system can also provide a way to permit "backup" of data — the copying of all data and programs contained on the system's disk peripheral storage onto other disks or less accident-prone media such as magnetic tape. Transaction logging, a process that keeps a copy (usually on magnetic tape) of every user interaction with the computer, may also be provided.

APPLICATION SOFTWARE

The process of application software design involves carefully analyzing the functions that the computer system must perform in order to achieve the result that the user desires. One method of functional analysis is to prepare a data flow diagram by which the overall result is broken down into small functional systems (called "modules") and the data flow between the modules is defined. Each module is then designed to relate as necessary with the other modules and to perform its function.

Following detailed design of each module, programming occurs by writing a set of instructions in a certain sequence that will produce the desired result when the sequence is executed by the computer. This writing, which is called "coding," is performed using a programming language. The first type of language that can be used is called "machine language" and is a "low-level" language that is easily understood only by skilled programmers.

Symbolic languages were developed to simplify programming by being more easily understood by the user. As written in the symbolic language, the so-called "source code" does not operate with the operating system, but must be "translated" into so-called "object code" that runs on the hardware.

Symbolic Language

A "high-level" symbolic language is one that is most easily understood by a programmer, e.g. FORTRAN and BASIC. A computer program called a "compiler" is used to translate source code written in the high-level language into the object code form that is understood by the computer. The compiler is actually a computer system that is controlled by special application programs that cause the compilation to occur.

When source code is written, separate English language "comments" are also written to explain the operation of the source code. This makes it easier to correct (or "fix" or "debug") the code and to modify it if "enhancements" (new functions) are desired.

Functional analysis, design and programming are essentially intellectual tasks. Thus they are subject to the errors and limitations inherent in any intellectual process. This fact must be kept in mind when dealing with any hardware or software transfer, whether it be for so-called "standard" application software, "custom" software, or system software.

HARDWARE CHARACTERISTICS

The different hardware elements of a computer system are often manufactured by different vendors. If a given computer system includes hardware and software supplied by different vendors, the problem is raised as to which vendor will establish certain standards as to how the hardware elements work together. The elements, such as peripheral storage, are said to be "plug compatible" when they merely have to be plugged into the CPU and will operate properly. This plug compatibility is possible because "interface standards" are published for the manner in which bits of data are grouped in the CPU to form a "byte"; for the way the data within a byte is coded with an address and for the rate at which data is transferred from one hardware element to another (expressed in megabits per second).

The CPU manufacturer will generally offer to grant to users licenses to use an operating system with its particular CPU. The operating system controls the operation of the CPU and other hardware so that the data is processed and then transferred between hardware elements according to the published interface standard. All of the hardware elements must be compatible, that is they must comply with the interface standard, e.g. be capable of processing the data fast enough or receiving data from another hardware element at the desired rate.

PROGRAM COMPATIBILITY

However, the CPU manufacturer may not develop any application programs or may only develop relatively few application programs. This leaves a large market for application software. This market is serviced by vendors that specialize in the development of application software that enables the computer system to perform such functions as airline reservations, book publishing, accounting and shipping.

Another aspect of compatibility is that all of the computer software for use with the hardware elements of a given computer system must be written in the same programming language, which in turn must be compatible with the hardware interface standard.

The use of a computer system can be arranged for in a variety of ways. First, one can enter into a contract with a service organization. This organization receives the customer's data, uses the organization's own computer facilities (hardware and software) to process the data and returns to the customer the desired output, which may be in the form of reports. Alternatively, one can enter into a contract to share the use of another company's computer, which is referred to as "time sharing." This is less costly and enables the user to work directly with the computer system. One can also purchase or lease a computer system. This could involve purchasing or leasing a "turnkey" system. This involves taking a license to use all necessary system and application programs with the leased or purchased hardware. The vendor represents that the turnkey system is capable of performing a specific function, such as maintaining all payroll records. Finally, one can purchase or lease hardware and buy or enter into a license to use separate systems and applicable programs that are available from a number of vendors. There are many combinations of hardware and software licenses and ownership, which result in substantial complexities in terms of who is responsible for the overall operation of the computer system and what rights the user has with respect to the computer programs. The rest of this article focuses on these subjects.

LICENSING GUIDELINES

The following questions are proposed as a way of analyzing a number of the common software licensing situations. They emphasize how the subject matter of the license — computers and software — adds to the factors that are normally considered in licensing other types of technology.

1. Is the program described in a complete specification published by the vendor?

If not, what will be referred to by the parties to define what operating system is compatible with, what functions the program will perform (and how fast) and what forms of input and output are used? The specification is one of the most essential parts of a computer software transaction.

First, the program descriptions in the specification will induce the user to enter into the license, such that the specification must be in or incorporated in the agreement.

Second, it serves as a standard against which operation of the program is tested before the user pays the license fee to the vendor.

Third, it serves as a continuing standard against which the vendor must maintain the program from a warranty or maintenance obligation standpoint.

Fourth, it defines all elements of hardware and software compatibility, such as the interface standard, with which the program is guaranteed to operate.

2. What is known about the quality of the computer program?

This is one of the most crucial questions to ask before signing a software license. Many vendors do not permit a trial period during which the user may use the program without obligation.

If a trial period is allowed under the terms of the agreement, the user should make sure (1) that the software is tested during this period, and (2) if the user does not give notice at the end of the trial period the agreement will automatically terminate rather than automatically continue. Also, it is essential to read the agreement *before* signing it to make sure that it in fact provides for a trial period (without an obligation) instead of a binding license under the software.

If no trial period is allowed, it is recommended that the user not sign the license unless it does further study. For example, if there is a group of users of the program, the members of the group should be contacted to learn what their experiences have been. If no user group exists, the potential licensee should ask the vendor for the names of current users and contact them. If others contacted are satisfied that the program operates in accordance with specifications, the user will have some comfort when it executes the license agreement and obligates itself to make payments.

3. What general type of computer program is the subject of the license, in what form will it be delivered, and who is the vendor?

The choices include operating systems, utilities, application, assembler, and compiler. The licensing executive must exert the necessary effort to determine the full details as to these factors.

If the program is an operating system, the specification should be checked to determine whether all required utilities are provided by the program, including adequate security measures. The specification should also indicate whether there are special features that may complicate the license agreement.

For example, a standard AT&T license for the UNIX System III operating system normally grants the right to use "Object Software," which is defined as the object code version of "Licensed Software," both as defined in the agreement. Query reveals that the object code has the capability of supporting a large number of simultaneous "users" (via time-sharing terminals).

Separate Agreement

In a separate agreement, AT&T also grants the right to produce "customer software." Customer software is the object code that has been modified to be able to support a limited number of users. The customer software may be used within the user company or may be distributed to third parties. In both cases, a fee is payable to AT&T based on the "capacity" of the customer software — i.e. based on the number of "users" that a given item of customer software can support.

For well know application programs, the user generally only needs object code in a language compatible with the operating system and hardware. However, if the user needs to modify the application software it should also receive the source code. For systems software, the user would normally require object code only, written in the language that the user's programmer knows or that is used with relative ease by a large number of its programmers.

Assembler Compiler

Assembler/compiler (translater) software is required if the user is doing its own programming. As noted above, a translater enables the user to translate source code into object code; the assembler being used for low level language and the compiler being used for high level language. The translater program must run on the CPU that the user will use for translation purposes and must be designed to assemble or compile the language in which the source code is written.

The user should generally own and have an unrestricted right to use the programs that are translated by the assembler or compiler. As an example of necessary inquiry, in one license for a Pascal-2 cross-compiler and related utilities, it was critical to determine that the cross-compiler was compatible with a PDP-11 CPU and *not* with the user's IBM 360/370 CPU. The user thus had to negotiate the additional right to convert the compiler code for use on the IBM 360/370 CPU.

It is also helpful to identify the vendor. For example, if the program is an IBM-licensed program, the user will know that it will be much more difficult to negotiate the terms of the license agreement than if the user is dealing with a small vendor.

4. What will the computer program be used for?

This refers to the "right to use" or "license" clause. The license must give the user the necessary rights to use the program. For example, unless the intended use is for strictly internal business purposes (end-user), the user should not accept any limitation on the purpose for which it can use the program. We believe that the usual limitation that the program only be used on one CPU protects the vendor enough. Thus, even if the user does not expect to time-share with an external third party, a commercial licensee may want to allow students to use the program via a link between the licensee's CPU and a university.

If it is clear that third parties must have the right to access and use the program by a communications network (time-sharing), then the license should clearly state that use by a third party of the licensee's CPU and the licensed program is acceptable.

If the computer program will be included in a product that the user sells or licenses, then special sublicensing grants are necessary. In this sense the user would be an original equipment manufacturer (OEM), which is an entity that purchases hardware and possibly licenses software from one or more vendors, adds value to the hardware and/or the software and then sells a complete or more complete system to an end-user.

Value-Added

The OEM could, for example, buy standard microprocessors and design them into a value-added microprocessor-controlled product that is sold to a third-party customer. In this case the OEM generally needs to grant to its customers licenses to use an operating system and application programs. While the OEM would generally write its own application programs, it may use a standard operating system licensed from the microprocessor vendor. If so, the "right-to-use" clause in the operating system license must give the OEM the right to sublicense the operating system to its customers.

Where the vendor licenses a utility, the utility software could be licensed to the OEM in the form of a separate load module apart from the vendor's usual operating system. The vendor will arrange with the OEM so as to allow the OEM to write its operating system to permit the OEM's end-user customer to access the utility only by way of the OEM's operating system. Thus, the vendor would not allow the OEM to license the utility to end-users separately in competition with the vendor.

Sublicensing raises a series of additional factors, such as:

- The code is normally sub-licensed by the OEM to the customer in object code form only.
- The vendor attempts to control some of the terms of the sublicense, generally by including provisions in the main license and requiring that the OEM include such provisions in the sublicense.
- The vendor may require the payment of royalty as a condition of sublicensing, but this is generally a question of whether the vendor received a large initial lump-sum payment or only a small lump sum on the expectation of receiving further sublicensing royalties.
- The vendor's obligation to provide maintenance for the sublicensed programs used by the OEM's customers. Here the vendor will generally obligate the OEM to gather all of the maintenance questions from its customers and direct them to the vendor.

5. With what CPU and site will the computer program be used?

The vendor's interest is generally to license a single copy of the program for use with a single CPU and to receive the full fee for each such licensed CPU. The vendor uses this approach even for the use of the same program on multiple CPUs by the same customer. In any event, if a single CPU license is intended, the CPU should be specifically identified. If the user thinks it is paying for use of a program on all CPUs in its computer room or at a given site, the license should clearly state such use.

In one situation, the user's address was stated in the license and the "Computer Model" was listed as "Amdahl V-6," without stating a serial number or any specific identification. The adequacy of the granting clause came into issue when the user installed a second V-6 CPU and wanted to install the software on the second CPU without entering into another license or paying another fee.

6. Is the computer program to be used on more than one CPU in the user company?

If so, is the right to use limited to use with a particular CPU? A key factor is whether the vendor will reduce the per-CPU charge for use on additional CPUs. If the program is currently used elsewhere by the user, and the license in question is for an additional copy of the program, or if the use is required on two new CPUs, it may be possible via a master license to obtain a reduced rate for the license in question and for any successive licenses of the same program. Here, the user becomes a "distributor" to all of its subsidiaries or divisions, assumes a duty to distribute additional copies to other sites and funnels all maintenance questions to the vendor.

Other effects of site licensing and master licenses are that the vendor is in effect establishing a form of distributorship, which can adversely impact the more traditional distributors of the vendor's software. From the user's standpoint, however, site licensing not only reduces software costs, but eliminates problems which would otherwise arise when its computer users "share" programs, which amounts to a form of software piracy by violating the terms of single-CPU licenses.

7. Is the user's acceptance of the program related to the conformance of the program with the specifications that are stated in the agreement?

This concept assumes that detailed specifications are a part of the license and that the acceptance of the program by the user (and the user's obligation to pay for the program) are conditioned upon the conformance of the program with the specifications. This also relates to when the warranty period and the vendor's maintenance obligations begin.

8. Is the program warranted to operate in conformance with the specifications? What are the vendor's duties to fix the program? (Warranty duration, response time).

Warranties of fitness for a particular purpose or of merchantability may be difficult to prove and more difficult to negotiate. The warranty clause should, however, at least clearly state that the vendor must keep the program operating in conformance with the specifications for a given time period. It should also define the amount of time (following receipt of the user's notice) that the vendor has in which to notify the user of a fix or correction to the program. The clause should also state the conditions under which the vendor must visit the user's installation to effect the correction. Preferably, the warranty period would not commence until formal acceptance of the program by the user.

9. CPU, site or master license?

If the vendor has not set up a master license program, you can expect difficulties in negotiating a master license. However, if the user's anticipated use is extensive, it is worth an attempt to reduce the cost of multiple licenses by negotiating a master license. The master license will provide that its terms and conditions can be rendered effective for a specific program and site by an addendum that is entered into by the parties as necessary.

One difficulty that arises within the user's organization is that it is more complex (hence time consuming) to negotiate a master license than the vendor's standard single-CPU license. It helps if all users within the user company agree to cooperate in an attempt to reach agreement on the master license. Some of the practicalities involved are whether the various user sites (for which the program would be licensed) are compatible. If not, the vendor must be able to supply a version of the software for each site. This weakens the user's argument for a reduction of the license fee. Also, under a master license the user should expect to centralize its request to the vendor for warranty service and maintenance.

10. What form of code is licensed?

In general, a software vendor is very reluctant to license the source code. If only the object code is licensed, the user should at least have the right to obtain the source code if certain events occur, such as the vendor going out of business or refusing to perform its maintenance obligations. If the user has access to the source code in these situations, then it will be much easier for it to do its own maintenance.

The issues in negotiating an "escrow" clause can be simplified if the user is reasonable in defining the events that give it access to the source code. One legal issue is whether a user can enforce an escrow clause if the vendor owns the escrowed code and goes bankrupt. Present bankruptcy laws provide an automatic stay against taking action against the bankrupt estate after the petition has been filed. One suggestion is that the license provide for a present sale of a copy of the source code to the user, with possession of the copy being withheld by an agent of the user until the user proves that the event (bankruptcy) occurred. This has not been tested, however, to the authors' knowledge.

Another issue is whether the vendor must periodically update the escrowed source code so that it will be current when the event occurs.

11. What are the user's rights to modify and copy?

The statement of the right to use should be reviewed carefully. It is generally very difficult to reverse compile or reverse assemble the object code, such that if only object code is licensed, the right to modify is generally unnecessary. Further, some licenses, such as the standard license for IBM-licensed programs, specifically prevent reverse assembly and reverse compilation. In the IBM case, it allows modifications of object code in machine-readable form for limited purposes. If the user has the right to modify, the effect of same on the vendor's maintenance obligations should be carefully reviewed.

If the user has the right to modify, the question arises as to who owns the modified program. This issue is similar to any grant-back or license-back issue in transferring other types of technology.

The right to "copy" relates to whether the user only has the right to copy the machine readable form of the program for backup purposes and whether the licensee has the right to make multiple copies of any documentation that is furnished with the program.

12. Are there ways to reduce cost?

In many cases the user will have a central data processing group that can be the focal point for the vendor's maintenance, such that the user's data processing group would distribute fixes and updates to the programs and thus reduce the cost of the license.

13. What are the definitions of charges and payment?

The license should include a clear statement of what charges will be payable and when. Especially if a one-time payment is made for a perpetual license, the user should not be obligated to make the payment until it has accepted the program following an initial trial period and following determination that the program conforms to the specifications. Because maintenance charges generally cost 10 to 15% per year based on the one-time payment, the user's obligation to make the payments should be clearly related to the vendor's obligations to maintain the program and to provide updates. Vendors customarily try to collect additional charges for installation and upgrades that substantially improve the performance of the program.

14. What are the vendor's obligations to maintain the program?

The maintenance clause should in effect extend the warranty period by obligating the vendor to keep the program operating in accordance with the specifications during the term of maintenance. The maintenance obligations are generally conditioned on the user refraining from modifying the program without the vendor's permission. As in the warranty situation, the vendor should agree to respond in a maximum time period to user's request for maintenance.

15. What are the user's obligations of confidence?

Vendors often loosely draft the clause that obligates the user to maintain all of the program materials in confidence. This is considered unfair to the user's personnel unless the obligation of confidence is limited to documentation and other materials in tangible form that are clearly marked with the vendor's confidential legend. The user should have no obligation of confidence with respect to published, copyrighted materials, for example. Other exclusions from the obligations of confidence include those normally excluded under standard confidential disclosure agreements.

16. Is the license assignable?

The sale by the user of its computer system raises the question as to whether the user has the right to assign the license for the software in the system. If it is at all likely that the user might sell the computer system with which the licensed software is used, it may be vital to be able to assign the license with the system. This benefits the user in that the future use of the system will not be in question. Because the purchaser will have the assurance that the computer system can still be used, the purchaser may be willing to pay more for the system.

THINGS THE COMPUTER AND SOFTWARE USER SHOULD DO

1. Define who is responsible for complying with the contract before signing.
2. Use separate employee agreement to emphasize the need for contract compliance at or immediately after the contract signing.
3. Enter licensed program into data processing resource database when license is consumated. Note status of acceptance or modification in the database.
4. Establish a way to prevent improper copying and to police based on written procedures.
5. Define who is responsible for the acceptance/maintenance of the licensed program, such as the MIS Manager.

MANAGEMENT OF LICENSES

1. Establish a basis for determining what and how much administration and legal staff review is necessary for various types/costs of computer programs. Should consider this for both *in* and *out* licensing.
2. Coordinate the administrative, legal and user groups during negotiation and post-signing phases. This is of principal importance when licensing *in*.
3. Establish database of data processing resources owned/used by the *user* company (licensee). This can also be useful to licensor if customer information is available. Marketing value of such information is obvious. Also, vendors should realize that they are possibly licensees and that identification of their owned and licensed *in* resources is necessary for ensuring that the vendor's organization practices what it expects its licensees to practice.

COMPUTER SUBJECT GLOSSARY

ALGORITHM—Set of well-defined rules or processes for solving a problem in a finite number of steps.
ASSEMBLE—To put together subroutines and routines into a main program. To prepare a machine language program from a symbolic language program by substituting absolute operational codes or addresses for symbolic codes or address.
ASSEMBLY LANGUAGE—Grouped alphabet characters called MNEMONICS to replace numberic instructions of machine language.
ASSEMBLY PROGRAM—Operates on symbolic input data to produce machine instructions.
BAUD— Unit of signalling speed in data transmission—an expression of bits/second.
BINARY—A numbering system using 2 as its base and using only digits 1 and 0.
BIT—Short for Binary Digit; the smallest unit of information having value of 1 or 0.
BYTE—A sequence of adjacent bits operated as a unit and shorter than a computer word. (A word, in some computers, occupies one storage location and is treated as a unit of information and an entity.)
CHARACTER—One of a set of elementary symbols or letters or numerals.
CODE—A system of characters and rules for representing information, usually in a language that can be understood by a computer. (See object and source codes.)
CODING—The act of preparing a list of instructions for successive computer operations in either machine or non-machine language.
COMPILER—A computer program using symbolic input data to produce machine instructions; considered more powerful or a higher level than an assembler.
COMPUTER—A device that accepts data, in acceptable forms; manipulates data according to programmed instructions;

take manipulated data and provides output data or commands.

Typically has an input means, storage or memory means, arithmetic/logic means or units, a central control means or unit and an output means.

Mostly a digital operating device but may also be an analog device using electrical analogs of physical magnitudes sometimes without conversion to digital form.

CPU—Short for Central Processing Unit; a group of components containing an internal memory needed to support the tasks of the arithmetic/logic functions and an internal control means. Sometimes called the Mainframe.

DIGIT—A single symbol or character representing quantity.

EMULATE—To imitate one system by another so that data is handled to achieve the same result; not necessarily simulation.

FIRMWARE—Part of a computer program incorporated as machine hardware; may be temporary or permanent providing specific character to a general purpose computer device. Example is a "ROM" or Read Only Memory.

LANGUAGE—Defined set of characters used to form symbols and words and the rules by which they are combined into meaningful communications.

MACHINE LANGUAGE—A language used directly by the computer without need for translation; can also be considered as "Object" language.

MEMORY—A device into which a unit of information can be placed or copied, which will hold such information until needed and obtained at a later time for use.

MICROPROCESSOR—Usually a microchip performing the function of a CPU.

MICROPROGRAM—A special-purpose program stored in a fixed memory, containing several instructions needed to execute a main program instruction. Can be a firmware item.

MINICOMPUTER—General purpose digital computer, of moderate cost, operating on word lengths of 12-16 BITS.

MNEMONIC—A symbol assisting human memory such as CLR for Clear.

MODEM—Modulator-Demodulator used in data transmission converting one form of signal to another; usually taking digital 1 or 0 and converting to tones of different frequencies representing a 1 or 0 state; makes data communications over a telephone line possible.

MULTIPROCESING—Done by simultaneous execution of two or more computer programs in a computer network.

NUMERIC CONTROL—Control of machine/process by commands in numeric coded form. Controls discrete operations.

OBJECT PROGRAM (OR CODES)—Program output from an automatic coding routine; a program in machine-compatible language supplied by software licensor or vendor to users usually on disc or tape.

PERIPHERALS—Equipment distinct from the CPU such as printers, readers, external memories.

PROCESS CONTROL—A system of control providing automation of continuous operations, not discrete operations.

PROGRAM—Sequential steps executed by a computer to solve a problem, perform a task, run a process.

PROM—Programmable Read Only Memory—can be programmed by a computer user, usually one or more times.

RAM—Random Access Memory which stores BITS that can be retrieved as quickly as any other BIT in the memory.

READ—To acquire/interpret data from a storage device/memory.

READ-WRITE MEMORY—Contents of which can be changed quickly during system operation; sometimes scratch-pad memory.

REAL TIME—Data processing done in synchronism with a physical process or business procedure, such that results of such processing is useful to the process or procedure.

ROM—Read Only Memory can be programmed usually only once; a device which will allow multiple readings from memory without degradation of information stored therein.

SERIAL PROCESSING—In a digital computer where all programs are handled in sequential fashion.

SIMULATOR—Software routine executed by one computer which imitates operations of another computer; not necessarily emulation.

SOFTWARE—Set of programs, procedures, rules and associated documentation for the system, such as compiler, library routines, user or maintenance information.

SOFTWARE (APPLICATION)—Works with system software to perform intended operations; usually licensed for specific uses.

SOFTWARE (SYSTEMS)—For operating systems, controls hardware of the computer; usually licensed to buyers, leasors or users of the hardware.

SOURCE PROGRAM (OR CODE)—Written originally in a symbolic source language to enable use of application software. This is usually proprietary to software vendor.

STATEMENT—A programming expression or generalized instruction in a source language.

TIMESHARING—Use of a CPU for two or more purposes during a specific time interval, usually interspercing timely actions of peripherals and a CPU.

UPWARD COMPATIBILITY—A feature of a program written in lower-level language allowing its use in a higher-level system.

VIRTUAL MEMORY—Addressable memory space appearing to user as real storage, where instructions are mapped into real storage. Can be limited by the addressing scheme of the system and by amount of auxillary storage devices available for use.

VOLATILE MEMORY—Contents of which can be lost when operating power is removed.

WORD—A group of characters which may occupy a single storage location in a computer. It is treated as an instruction by the CPU and a quantity by the arithmetic/logic unit.

WRITE—To impart or introduce information into a storage device such as a tape or disc.