Robot Manipulator : Past, Present and Future

A K M Azad[1] and Farruk Ahmed[2]

1. Introduction :

For many years robots have excited people's imaginations – whether in books, in films or inside fairgrounds or amusement arcades. But it is only quite recently that robots have left the realms and fictions. They are becoming useful to men and women in all shorts of ways. How this changes have come about and how it will progress over the next few years, is the subject of this article.

Even now hundreds of thousands of robots are at work around the world. Their activities range from delicate construction work of handling huge loads, to precision assembling in factories and nuclear power stations to outer space. In the future robots many become an even greater part of our everyday lives. With the help of neural network (Acting as human brain) robots may become as clever as people. The could take over a wide variety of jobs-in offices, in factories or in the home. Robots could also work in particularly unpleasant and unreachable places where complex, vital work needs to be done, such as mines or the bottom of the ocean.

2. What is Robot :

The word 'robot' originated from the Czech word robota, meaning work; In dictionary robot is defined as "an automatic device that performs functions ordinarily ascribed to human beings". With this definition, washing machines may be considered as robots. But the more acceptable definition used by the Robot Institute of America gives a more precise description of robots: "A Robot is a reprogrammable multi-functional manipulator designed to move materials, parts, tools, or specialized devices, through variable programmed motions for the performance of a variety of tasks." With this definition, a robot must possess intelligence, which is normally due to computer programme associated with its control and sensing system.

Over the years, robot makers have been especially interested in designing robots which, in some way, look like men or women. It is this which has given robots their superhuman reputation. People regard them with a mixture of fascination and fear. For many years engineers tried to find jobs for robots to do. But, outside amusement halls or the world of films, this proved very difficult. The reason was straight forward. Engineers lacked the skills, and the essential tools, to make robots reliable enough to do jobs in real file.

Two developments changed all this. First, engineers set their sights lower and decided that robot should do no more than imitate people. They need not necessarily look like people at all. To engineers, robots have simply become electro-mechanical device that imitate the actions of human arms, and are controlled by computers. The second development is that engineering skills, particularly in electronics and computers, have progressed at a great pace. The world has become better at making machines that fit the new definition of what a robot is. The result is that, almost overnight, robots have left the Kindergarden stage. They have started to go to work.

3. Anatomy of Robot :

Dynamically the human body mainly contains two sets of components, one is the brain and nervous system and the second one is the limbs and the other parts of the body that move. But the average robot at work today is nothing like as complicated as a human being. A robot in its general form is computer controlled manipulator consisting of several rigid links connected in series by revolve or prismatic joints. One end of the chain is attached to a supporting base, while the other end is free and equipped with a tool manipulate objects or perform assembly tasks. The motion of the joints results in relative motion of the links. Mechanically, a robot is composed of an arm (or metal frame) and a wrist subassembly plus a tool. It is designed to reach a workpiece located within its work volume. The work volume is sphere of influence of a robot (Figure-1) whose arm can deliver the wrist subassembly unit to any point within the sphere. The arm subassembly generally can move with three degrees of freedom. The combination of the movements positions the wrist unit at the workpice. The wrist subassembly unit usually consists of three rotary motions. The combination of these motions orients the tool according to the configuration of the object for ease in pickup. The last three motions are often called pitch, yaw and roll. Hence, for a six-joined robot, the arm subassembly is the positioning mechanism, while the wrist subassembly is the orientation mechanism. These concepts are illustrated by the unimation Cincinnati Milacron T^3 robot arm in figure-2.

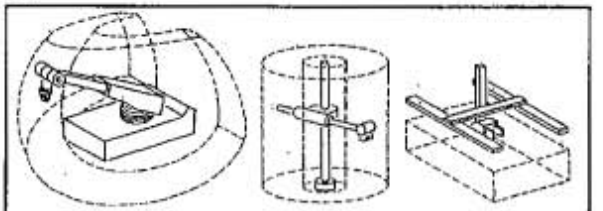

Figure 1 : Work volume for different types of robot manipulator.

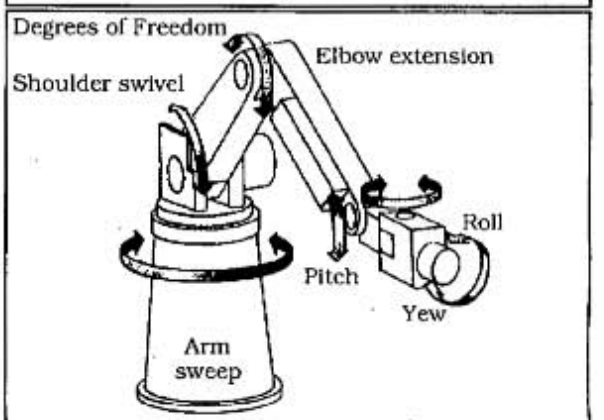

Figure 2 : Movement capability of a Cincinnati Milacron T^3 robot manipualtor

[1]Department of Automatic Control and System Engineering, University of Sheffield, UK.
[2]Department of Applied Physics & Electronics, University of Dhaka, Bangladesh.

4. Historical Development :

Early work leading to today's industrial robots can be traced to the period immediately following world war II. During 1946 Oak Ridge and Argonne National Labora tories developed a remotely controlled master-slave manipulator for handling radioactive materials, which was then improved in 1950's by a more sophisticated programmable device. Further development of this concept by Devol and Jesoph led to the first industrial robot, introduced by Unimation Inc. in 1959. The key to this device was the use of a computer that could be programmed to carry out variety of tasks automatically. Early 1960's Ernst (Ernst, 1962) reported the development of a computer controlled manipulator with tactile sensor, which could feel blocks and use this information to control the hand without operator's assistance. The work is one of the first examples of a robot capable of adaptive behavior. During the same period, Tomovice and Bono (1962) developed a manipulator with pressure sensor. This pressure sensor provides information about object size and weight to the computer. In 1963, the American Machine and Foundry (AMF) company introduced the VERSATRAN commercial robot. Starting in this same year, various arm designs for manipulators were developed, such as Roehampton arm and Edinburgh arm.

In the late 1960's McCarthy (1968) and his colleagues at the Stanford Artificial Intelligence laboratory reported a development of a computer with hands, eyes and ears (i.e. manipulators, TV cameras and microphones). During this period, Pieper (1968) studies the kinematics problem of a computer controlled manipulator while Kahn and Roth (1971) analyzed the dynamics and control of a restricted arm using bang-bang control.

One of the more unusual developments in robots occurred in 1969, when an experimental walking truck was developed by the General Electric Company for the U.S. Army. In the same year, the Boston arm was developed, and in the following year the Stanford arm was developed, which was equipped with a camera and computer controller. Some of the most serious work in robotics began as these arms were used as robot manipulators. In 1974, Cincinnati Milacron introduced its first computer controlled industrial robot, called the "The Tomorrow Tool", or T^3. It could lift over 100 lb as well as track moving objects on and assembly line.

During 1970's a gread deal of research work focused on the used of external sensors to facilitate manipulative operations. At Stanford, Bollers and Paul (1973), using both visual and force feedback, demonstrated a computer controlled Stanford arm connected to a PDP-10 computer for assembling automotive water pumps. At about the same time, Will and Groomsman (1975) at IBM developed a computer controlled manipulator with touch and force sensors to perform mechanical assembly of a 20-part typewriter. At the Draper Laboratory Nevins et al. (1974) investigated sensing techniques based on compliance. This work developed into the instrumentation of a passive compliance device called remote center compliance (RCC) close partsmating assembly. Bejczy (1974), at the Jet Propulsion Laboratory, implemented a computer based torque control technique on his extended Stanford arm for space exploration projects. Since then, various control methods have been proposed for servoing mechanical manipulators.

Today, robotics is viewed as a much broader field of work than we did just a few years ago, dealing with research and development in a number of inter disciplinary areas, including kinematics, dynamics, planning systems, control, sensing, programming languages and machine intelligence.

5. Robots in Operation :

There are around 400,000 robots existing throughout the world today. Of these probably 90% work in factories. Average robot at work today is nothing like as complicated as human being. An industrial robot is a general purpose computer controlled manipulator consisting of several rigid links connected in series by revolute or prismatic joints. One end of the chain is attached to a supporting base, while the other end is free and equipped with a tool to manipulate objects or perform assembly tasks. The motion of the joints results in relative motion of the links. Mechanically, a robot is composed of an arm (mainframe) and a wrist subassembly plus a tool. It is designed to reach a work volume. The work volume is the sphere of influence of a robot whose arm can deliver the wrist subassembly unit to any point within the sphere. The arm subassembly generally can move with three degree fo freedom. The combination of the movements positions the wrist unit at the workpiece.

Most of the todays industrial robots, though controlled by mini or microcomputers, are basically simple position machines. The execute a given task by playing back prerecorded or preprogrammed sequences of motions that have been previously guided or taught by a user with a hand held control teach box. Moreover, these robots are equipped with little or no external sensors for obtaining the information vital to its working environment. As a result, robots are used mainly in relatively simple, repetitive tasks. More research effort is being directed toward improving the overall performance of the manipulator systems.

6. Robot Diversify

Robots are already starting to move out of factories and into a variety of other places. One job which has always been dangerous for human workers is the inspection and maintenance of nuclear power stations. Robots are now beginning to take over this work. These work involves replacement of uranium rods, inspection of the rods for faults once they are inside the core, emergency welds on the interior or the core's wall. Taylor Hitec, and English company developed a robot which can take apart the highly radioactive core of a power station when it is no longer required. This is too difficult and dangerous job for human.

A company in the United States has a design for robots which can shoot highperformance water in clean up the rocket motors of the space shuttle. This would allow the motor to be reused. Also in United States, a US Navy robots takes out damaged rivets from the wings of aircraft. The US Navy has also designed a robot that crawls over the surface of large vessels, cleaning barnacles off them.

7. Sensitive Robot :

All the robots we have looked so far have been totally dependent on operator or programmer. If something happens during the job that the robot has not programmed to expect, it is powerless to react. The use of external sensing mechanisms allows a robot to interact with its environment in a flexible manner. Although the latter is by far the most predominant from of operation of present industrial robots, the use of sensing technology to endow machines with a greater degree of intelligence in dealing with their environment is indeed an active topic of research and development in the robotics field.

The function of robot sensors may be divided into two principal categories: Internal state and external state. Internal state sensors deal with the detection of variables such as arm joint position, arm vibration, which are used for robot control. External sensors on the other hand, deal with the detection of variables such as range proximity and touch. Although proximity, touch and force sensing play a significant role in the improvement of robot performance but vision is recognized as the most powerful of robot sensory capabilities. Robot vision may be defined as the process of extracting, characterizing, and interpreting

information from images of a three dimensional world. This process, also commonly referred to as machine or computer vision, may be subdivided into six principal areas: (1) sensing, (b) preprocessing, (c) segmentation description (e) recognition, and (f) interpretation.

Vision and touch sensitive robots demonstrate the important idea of 'feedback' control. 'Feedback' is the way in which the actions of robot may be altered or changed by outside events. Unfortunately for engineers, however, equipping robots with this ability is not easy. It is very difficult to code information from the outside world in a way that makes sense to the robot computer, and then to make sure it processes the data quickly enough for the robot to respond. At present, probably only 5% of the world's robot's have sense. They are known as 2nd generation robots.

8. Intelligence Robot :

None of the robots we have looked at so far matches up to humans in one importants respect–thinking. Enven 'second generation' robots, come a long way behind people in their ability to make sense of this information and act accordingly. They simply react as a reflex to what is taking place around them. The ability to weight up the situation, perhaps comparing it with pervious experience, is beyond them. This brings up to an important area of research called artificial intelligence, or AI. Artificial intelligence is the computer's ability (through program) to do some assignment which is done by human would be said to require intelligence. With AI techniques, robots can become far more powerful and versatile than they are today. The new, AI assisted robot are the bread of third generation robots. In these AI robot's memory is divided into to areas, one stores programs and data relevant to the job in hand and the other would store the great mass of 'knowledge base'. Linking up the 'knowledge base' with the main processing unit is the key factor of this system. However, to do certain activities (visual recognition) robot requires vast amount of memory, and in some cases it become impossible. To overcome the problem completely different type of idea called 'neural network computing' is introduced. Researchers are already working on this problem.

9. Space Industrialization :

Scientists played around with these ideas for years but then in 1980, researchers employed by the NASA in in the United States produced an engineering blueprint how this system could operate. Basic types of processes which could benefited from this development includes solidification of melts without convection or sedimentation, processing of molten samples without containers, diffusion in liquids and vapors, and electrophoretic separation of biological substances and exploitation of valuable raw materials on other planets. Once the particulars of a promising process are worked out, and the pressure for commercialization mounty, automated instrumentation for remote control will see development. Investment in a single space process may well amount to tens of millions of dollars.

Implementation of this idea will require a wider spectrum of robotics than needed for today's applications. Development of space robot and automation technology on a broad scale need large antennas, processing and manufacturing stations in Earth orbit to lunar bases, manned space stations, and vast satellite power system. Systems such as large space antennas, satellite power systems, and space stations will require large and complex construction facilities in space. The structural elements will be handled by teleoperated or automatic cranes and manipulators that emplace components and join them. Free flying robots can transport structural pieces between the Shuttle or the fabrication site and the assembly site and connect them. Free Flyer robots or robots attached to the system can service it. The functions such a service robot should be able to perform include calibration, checkout, data retrieval, resupply, maintenance, replacement of parts and other repair work, cargo and crew transfer, and recovery of spacecraft. During and after construction, a major space system should have a rescue robot standing by. Others might be held in readiness on the ground. Rescue robots could deliver expendable life support systems and lend first aid or encapsulate the astronaut in a life support environment for return to a Shuttle, space station or Earth.

Another phase of space industrialization envisions a lunar base. A largely automated system might launch the base-building project. After site survey by robotic rovers, an automated mini-factory could be placed on the Moon. It would collect solar energy and use it to extract volatiles, oxygen, metals, and glass from lunar soil delivered by rovers. It would automatically produce stockpiles in preparation for the main construction operations. The Base would be built using automated equipment and robots, in part drawing on stockpiles, and in part performing construction operations similar to those used in Earth orbit. After construction, general purpose robots would do maintenance and repair. Once established, the lunar base would use many type of industrial automation in its work. In summary, a new level or robotics and atuomation should catalyze some missions, but the technology will mainly be developed and applied to reduce operational costs for all missions.

10 Social Implications :

At the beginning of the 18th century about 80% of the Britain's population worked on the land. By the middle of the 19th century half the country's working population had jobs in factories. From the beginning of the 20th century this proportion has steadily fallen to about 35% today. Other industrialized nations throughout the world bave followed similar patterns.

The widespread use of automated machinery, computers and robots will continue this trend. Fewer and fewer jobs in manufacturing industries will be available to people wanting work. Many will find jobs in other industries that are growing, such as the service industries–catering, local government, banking and transport. Robots taking over jobs in factories could worsen the high unemployment rate that the industrialized world already suffers from. Although new industries are being created, they require few workers. Many people are worried that not enough jobs will be created to absorb the people no longer needed in factories.

The development of robots will have another, perhaps deeper, effect on society. Robots will make their presence felt not just in factories but in many other areas, in the home, in hospitals and other service jobs. People have had little trouble in getting used to new forms of electronic gadgetry such as washing machines, home computers and video recorders. But they may find the new forms of robots more difficult to accept easily. The intelligent robots discussed earlier many produce mixed feelings among people who work with or control them. Some writers argue that people will welcome machines they can get on with easily. There would be no need to address the new breed of robot with special computer languages. Most probable people would command them simply by taking to them. But, on the other hand, men and women may feel uneasy when confronted by machines that are as clever as, or cleverer than, themselves.

Other things is sure that the robots that wander around homes and factories during the next century will not be metal-clad, cumbersome monsters of past science fiction films. They will be specially designed to fit in with their surroundings. The robots of the future will probably be sleek and smooth, and look like ultraefficient vacuum cleaners. Robots will gradually become as familiar as the electric motors that abound in homes and factories today.□

Infamous Internal Errors in Clipper Programming

Dr. Khan Monzoor-E-Khoda

The failure of an internal assertion causes the base of an internal error. It is all environmentally related and does not branch to Clipper internal Error block whenever it occurs. In this paper, a set of unexplained internal errors are investigated and their casual factors are carefully examined.

1. Introduction

Every developer gets a runtime error when they are involved with their new development, but in fact, none of them can accept or tolerate any of these unwanted errors after the product has been released. Although most Clipper Errors branch to the current error block, some internal errors do not.

In a very recent paper by Key, S (Ref. 4.), an incomplete subset of Internal Clipper Errors were documented. We hope, in this paper, to both augment Key's work by further examining other less widely understood internal errors, and re-examine those errors listed by Key in his original paper. Error characteristics will be analysed, with an emphasis on root causes.

2. What are Internal Errors

In Clipper, runtime errors are dependent totally on the error handler and the subsystem that generates the error. Errors generally occur either because of mistakes in the program code (i.e., type mismatch, divided by zero) or because of some condition of the environment (i.e., out of file handles, file sharing violations, memory low). The error handler communicates with the subsystem by returning a value. This value indicates what the subsystem should attempt to do to recover from the error. The legal values are generally determined by the values handler. These type of errors, in fact, can be trapped in the error system so that the application which has been affected does not necessarily terminate. However, sometimes it is not possible to trap a runtime recoverable error due to an invalid value returning from the error handler to the subsystem (see Ref. 1).

On the other hand, whenever a program code is executed in the error system, *ErrorSys()* function, before invoking the *ErrorBlock()* function, an error handler is unavailable to the *ErrorSys()* and unrecoverable error is caused. This usually happens because such errors can not branch to the error block so that the *ErrorSys()* is unable to execute the error block. These errors are, in general, reported with the message

'Unrecoverable error <error type>'

Other errors occur when an internal assertion fails. These are simply reported with *'Internal Error'*. All of such errors are more of less related to the environment (i.e., out of memory, error reading code to execute from disk). An in-depth analysis of error handling strategies can be found in Ref. 2.

3. Highlighting Internal Errors

An outline of the major infamous Internal Errors are listed here, whereas the standard list can be found in either Ref. 3 or Ref. 1. The characteristics together with the causes of each error are examined.

Internal error 16

This error occurs at the end of a large block database operation (i.e., ***APPEND, JOIN, UPDATE or TOTAL*** command) and when one of the database files used in the operation is no longer open. The suggested solution is to check the userdefind function used within the erroneous statement for any statements that close the database files.

Internal errors 17, 18 and 19

These errors exist in both Clipper Summer '87 and 5.x releases. They occur when an index has become corrupted at the time of attempting to update an index page. The suggested solution is to delete and re-create the index.

Unrecoverable error 22: Ntx file key type error

It is caused only if either a seek expression in an index file (i.e., .ntx file) evaluates to a logical or the system runs out of memory whenever evaluating the key expression.

Unrecoverable error 24: write error

The Clipper internal function that does file writes returns the number of bytes that were successfully written. If the number is less than expected, this error occurs. The most likely cause is lack of disk space. In other words, it is caused if the Clipper application is unable to write a database file or index file to the disk.

To avoid the incident of this type of internal error, make sure that either sufficient disk space and directory entries are available or the file is not marked read-only. In a network environment, make sure the application has the necessary rights to write to the file.

Unrecoverable error 92: Sort/Index error

The system is unable to create a temporary file during a 'sort' or 'index' operation. It can happen only in one of the following situations

- No disk space.
- Disk is write-protected.
- No more directory entries.
- The file already exists and is read-only.

Unrecoverable error 331 and 332: string/array memory overflow

Clipper 5's Release Notes Norton Guide (see Ref. 3) contains some cryptic notes about the Segmented Virtual Object Store (SVOS). Basically, it's what Clipper uses to manage all of its strings, arrays, and codeblocks. These two errors occur when the SVOS tables have overflowed, which is caused by the creation of large, nested arrays.

Unrecoverable error 349: too many extend locks

This error caused when an extend system function locks pointers to more than 16 parameters. Whenever someone uses *_parc()* to retrieve a pointer to a string in an extend system function, Clipper has to lock it in memory. A maximum of 16 parameters can be locked at a time.

To solve this, design the calling function so that fewer than 16 parameters need to be locked at a time, and call the *_xunlock()* function to free them. *_xunlock()*, which release all current locks, is described in Clipper 5.01's documentation, but it was omitted from the documentation in Clipper 5.2.

Unrecoverable error 415: cannot open overlay file

This error occurs when Clipper can not open its

[1] Present Address : Analyst, Management Information System, The Jonik Hospitality Group Ltd., 980 Yonge Street, Suite 200, Ontario, M4W 2J5, Canada. (Email : Khan.Khoda@canrem.com)

overlay file. Usually, The file in question is the executable one, but it can also be an external overlay or a prelinked library created with RTLink. The most common cause for this error is a shortage of file handles.

Several errors occur due to a lack of file handles. The solution to this is to increase the number of available handles. There are three places where one could increase the file handle count:

- The F parameter in the Clipper environment variable.
- The CONFIG.SYS file.
- The network config file (provided the environment is attached with a network).

Unercoverable error 416: read error on overlay file

This error occurs when the overlay manager fails to read from the overlay file. Some possible causes:

- The file is no longer available. For example, a user can run a Clipper application from a floppy drive and then switch floppies.
- something has inadvertently closed the overlay file. Calling on an incorrect handle can cause this.

Unrecoverable error 520: Attempt to get value for an invalid field type

This error occurs when a database (i.e., .Dbf) file is found to be corrupted. The solution is to fix the database.

Unrecoverable error 650: processor stack fault

These error is caused when out of stack space is reached. The suggested solution is to use */STACK* command to increase the stack space in the time of linking with RTLink.

Internal error 666

This error occurs only when a C function has attempted to free an invalid pointer. The best bet is to check the application where the C function is being called.

Unrecoverable error 667, 668 and 669: eval stack fault

These errors are all related. Clipper uses two different stacks: the CPU stack and the Eval stack. The CPU stack holds local variables and parameters for C functions and can be controlled by using the */STACK* command when the link is performed. The Eval stack is similar, but it is used to hold Clipper's parameters and local variables. To understand the causes behind these errors and their possible solutions, one needs to understand how Clipper's memory is organized. Most programming languages store local data in an area of memory called DGROUP, which is limited to 64K. In Clipper, this area contains:

- C static data
- The CPU stack
- The Eval stack
- Clipper static data

After the C static data, one has the CPU stack, which grows downward. Next is the Eval stack, which grows upward. Finally, there is the Clipper static area that starts at the end of DGROUP and grows downward. These errors occur when one of the expandable areas tries to grow into the space of one of the others.

Specifically, the errors occur under the following circumstances:

Error	Circumstances
650	The CPU stack has overflowed.
667	The Eval stack has bumped into the Clipper static area.
668	The Eval stack has bumped into a locked Virtual Memory (VM, in short) segment in the space between the Eval stack and the Clipper static area.
669	The Clipper static area has bumped into a locked VM segment in the space between itself and the Eval stack.

Generally, problems that cause one of these errors could cause any of them, which is why we have listed them together.

Stack fault errors have several possible causes:

- Unintended or too deep recursion. If a function is called recursively, one of the stacks may run out of space. The most common cause for unintended recursion is an error in an error handler.
- Too much C static data. If someone has his own extend system code, the static data goes to the C static's area by default. If the program uses too much static data, one of the other areas may run short of space. Third party libraries can also be guilty of this. Later versions of C compilers allow one to specify a data threshold that permits one to move some of this data elsewhere.
- Too large a CPU stack. Clipper defaults to a 4k CPU stack, which is sufficient in most cases. However, some third-party libraries cause additional stack space to be allocated. In addition, the user can control the size with the *STACKS* command to the linker or with the Blinker's *BLINKER PROCEDURE DEPTH* command. To find out how much stack space the user has allocated, generate a map file and examine the size of the segment *STACK*. If it is too large, explicitly limit it with a link option.
- Too many static variables. Each static variable that is created requires 14 bytes in the Clipper static area. If one has quite a few, it adds up. Storing multiple values in a static array can help alleviate this problem because the 14-byte entry is used for the entire array.
- Too many ITEMs. ITEMs created with the ITEM API in Clipper 5.2 also go in the Clipper static's area. Every item created must be released, and failing to do so may cause one of the Eval stack faults as the static's area increases.
- Too large a load size. Normally, the area between the Eval stack and the Clipper static's area is unoccupied, but in low memory conditions, VM segments can get swapped there. If these segments are there and locked, errors 668 and 669 can occur. Generally, reducing load size can help alleviate the problem.

Unrecoverable error 670: memory initialization error

This error is occurred during the initialization or reinitialization of the memory system. It, in fact, indicates an extremely low memory condition at startup, or that an application using *RUN* command from within the Clipper allocated DOS memory without freeing it.

The suggested solution is to make more conventional memory available for the application if it is caused at startup time. If it is happened immediately following the *RUN* command, the application that was run should be eliminated to solve the problem. This error also occurs with an improperly linked or corrupted application.

Internal Error 999

Internal error 999 occurs when Clipper tries to call an internal function that was not linked in. This error

can be caused by one of two actions. One is a bad incremental link. If deleting and re-creating the executable corrects the error, this was probably the cause. The other is calling a Clipper function indirectly, usually through a macro, and not referring to it explicitly. The solution is to add an EXTERNAL declaration for it so that the linker will include it.

Internal error 1210

This error occurs when a record is not found in an index. Typically, it is caused when a file is updated and the index is not open. In the original release of Clipper 5.2, there was a bug that caused this error to appear even though the record was there.

Internal error 4424: Temporary file creation error

Error 4424 is caused by a failure when creating a temporary file for indexing or sorting. It is generally caused by either not enough file handles or invalid *TEMPPATH* setting for Clipper environment or a lack of file creation rights on a network for the directory where the temporary files are created. Notice that temporary files are created in the current directory if no *TEMPPATH* is specified.

The appropriate solution is to correct the environment. If there are not enough file handles, the solution is similar to that of error 415. If insufficient network rights is the problem, either give the user the necessary rights or redirects the temporary file to a more appropriate directory.

Unrecoverable error 5311: Unable to create VM swap file

This error is caused when the VM subsystem is unable to create a VM swap file on the disk. It is generally caused for several reasons, which includes

- The target disk or directory is full.
- Insufficient file handles are available.
- An invalid path is specified in the *SWAPPATH* parameter of the Clipper environment variables or on the command line used to start the application.
- The user has not enough rights on a network drive to create the swap file.

It, in fact, indicates that the Clipper application no longer has sufficient VM available to proceed. Specifically, this error occurs whenever the VM system needs to swap a VM segment out of conventional memory, and it has used all expanded memory and disk space that has been made available to it.

The suggested solution is that provide more VM available to the application. This can be don either by providing more expanded memory available with the aid of increasing the value of the *E* setting of the Clipper environment variable, or by providing more disk spaces available with the aid of increasing the *SWAPK* setting of the Clipper environment variable.

The problem caused by this error can also be resolved by reducing the size and/or number of strings and arrays that are active at anyone time. Probably, the most common cause of this error is the declaration of extremely large arrays. (for explanation, we refer to Ref.3).

Unrecoverable error 5300: not enough memory

This error happens when there is not enough conventional memory and the VM system can not initialize. The solution is to make more memory available.

Unrecoverable error 5302, 5304 5305, 5306: conventional memory exhausted

These errors are caused by a lack of conventional memory at various places in the VM system. Specifically, 5302, 5305 and 5306 occur when a VM segment can not be swapped in. When an attempt to allocate fixed memory with *_xgrab()* or an internal function fails, error 5304 occurs. The solution in all cases is to make more memory available, either by decreasing load size, increasing available conventional memory, or decreasing the amount of memory that is locked at any one time. Another solution is to move the application to protected mode.

Unrecoverable error 5312: VM swap file overallocated

Error 5312 occurs when Clipper's VM swap file becomes full and has no more space to receive swapped items. This occurs when the size of the swap file is limited by using the *SWAPK* setting of the Clipper environment variable.

Unrecoverable error 5313: out of space for VM swap file

This occurs when the VM swap file can not be expanded. Typically, it means you are out of disk space. It is similar to error 24 but it occurs specifically in the VM system.

Internal error 5320 and 5321:

These errors occur if there is a problem reading or writing to EMS. Try using the BADCACHE setting in the Clipper environment verbal. If that does not work, try to eliminate Clipper's usage of EMS by setting the E parameter of the Clipper environment variable to zero.

Internal Error 5333:

Colloquially named the internal error from hell, it was renamed in Clipper 5.2 to "VM Integrity Failure". It is probably the single most misunderstood Clipper internal error. It happens when the VM system is instructed to swap in a VM segment but the segment is empty. The vast majority of the time, this error has nothing to do with the VM system itself, but rather indicates a bug in another system that is either using the VM system incorrectly or trashing memory. Unfortunately, there is no easy way to track these errors down. However, moving the application to the protected mode in many cases makes these errors go away and in other cases, it identifies the problem with a general protection fault.

4. Conclusion

We conclude that the existing version of Clipper has major drawbacks because of its unexplained Internal Errors. It has been seen that is not a complete listing but it includes the most common internal errors and these that frequently occur in the development cycle. We investigated the cause for each one of the errors included in our set. We believe this will definitely help every Clipper developer while they are in their development stage.

Finally, we strongly wish for a better error handler to be incorporated into the future Clipper 5.3 release by refining and reducing its drawbacks.

5. References

1. CA-Clipper : Error Message and Appendices Guide for DOS, Version 5.2, Computer Associates, 1992.
2. CA-Clipper : Programming and Utilities Guide for DOS, Version 5.2, Computer Associates, 1992.
3. CA-Clipper 5.2 : Internal Error Listing, Databases for Norton Guide, Computer Associates, 1992.
4. Key, S., *"25 Internal Errors I've Witnessed"*, Clipper Reference, pp. 9-11, December 1993.

6. Acknowledgment

It is great pleasure to thank Greg Mezo, who helped me with all aspects of my investigation and research.