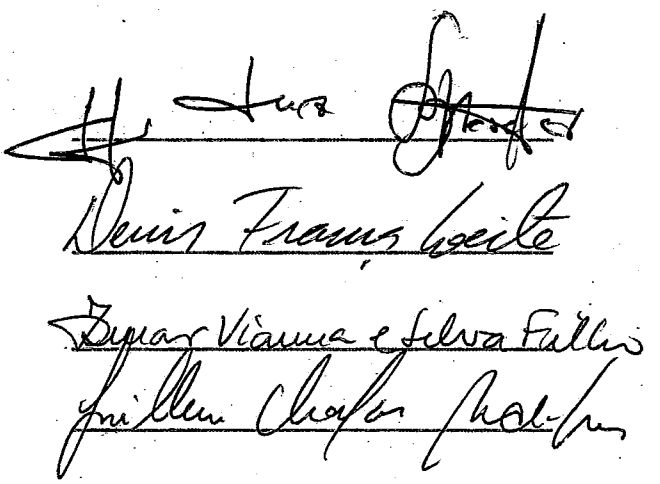
MACLAN - METODO DE ACCESO CONVERSACIONAL PARA LENGUAJES DE ALTO NIVEL

APLICACION A UN SISTEMA GENERAL DE CONSISTENCIA ON-LINE

Jose Manuel Glicberg Blum

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DE PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIA (M.Sc.).

Aprovada por



RIO DE JANEIRO

ESTADO DA GUANABARA - BRASIL

AGÔSTO DE 1972

A Fanny y Andrea.

A mis padres.

Orientador:

Prof. Jayme Luiz Szwarcfiter.

AGRADECIMIENTOS

Al profesor Jayme Luiz Szwarcfiter por la orientación general de la tesis.

Al profesor Guilherme Chagas Rodrigues por su aporte, a través de los constantes diálogos, para la afirmación de nues - tros conceptos.

Al Director del DSO Sr. Paulo Bianchi, al jefe del SSS Sr. Luiz Couceiro, al programador Manoel Pedro da Frota Moreira, por su asesoramiento para la implantación de los sistemas presentados.

A todos los funcionarios del NCE y en especial a su director Prof. Denis França Leite por todas las facilidades que nos dispensaron para la elaboración de este trabajo.

RESUMEN

El sistema MACLAN desarrollado en la primera sección de este estudio es un Método de Acceso para aplicaciones conversacionales de Teleprocesamiento en condiciones de memoria fuertemente restrictivas.

Consiste de un conjunto de rutinas que pueden ser llamadas desde programas escritos en cualquiera de los lenguajes más comunes de alto nivel.

Tales rutinas proveen recursos para edición de mensajes, corrección de errores y definición de códigos. Como aplicación de este sistema, desarrollamos un Sistema General de Consistencia de datos.

Este es un sistema para entrada de datos con controles simultáneos de consistencia que permite al usuario elegir el tipo de procesamiento operacional, es decir, "Batch" o "real time".

En "real time" puede procesar simultáneamente datos con formatos diferentes generando además los correspondientes archivos de informaciones válidas e informes de controles.

Con el propósito de facilitar la programación de lo que llamamos "Módulo de Consistencia" presentamos además una serie de ideas para desarrollar un lenguaje especial, con los recursos suficientes para alcanzar tales objetivos.

ABSTRACT

The MACLAN system developed in the first Section of this study is an Access Method for teleprocessing in conversational applications.
It consists of a set of routines that may be called from programs written in any of the most common high level languages.
These routines provide facilities for editing, error correction an code definitions.
As an application of this system we developed a General Data - Validation System.
This is a system for data-entry with simultaneous consistency-tests that gives the users the option of selecting the operational processing mode, that is, batch mode or real-time mode.
In the real-time mode it can process simultaneously different data formats, generating the corresponding files of valid data and control reports.
We also presented, for simplifying the programming of the Validation Module, an initial idea for defining a special language with the necessary resources for reaching this objective.

I N D I C E

CAPÍTULO 4: IMPLANTACION DEL SISTEMA

CAPÍTULO 5: CONCLUSIONES.

2a. SECCION: SISTEMA GENERAL DE CONSISTENCIA DE DATOS.

CAPITULO 6: INFORMACION GENERAL.

6.1 - Observaciones iniciales.
6.2 - Preparación y control inicial de datos.
6.3 - Configuración final de los datos.
6.4 - Preparación de datos en un sistema de "real time".
6.5 - Aspectos comunes en los sistemas sobre el procesador.

CAPÍTULO 7: FILOSOFIA DEL SISTEMA DE CONSISTENCIA.

7.1 - Definiciones y objetivos.
7.2 - Condiciones del Sistema.
7.2.1- Confiabilidad para todas las aplicaciones.
7.2.3- Velocidad de procesamiento y Gestión.
7.3 - Lenguajes de programación.

CAPÍTULO 8: ESTRUCTURA DEL SISTEMA

8.1 - Sumario
8.2 - Estructura de los datos de entrada.
8.3 - Opciones en los controles de validez de las informaciones de entrada.
8.3.1- Tipos de controles de validez.
8.4 - Deteccion de errores y mensajes resultantes.
8.5 - Almacenamiento de la información válida.
8.6 - Estructura básica del Sistema.

CAPÍTULO 10: CONCLUSIONES

10.1 - Integración y eficiencia.

CONCLUSIONES GENERALES.

APENDICE A: Macro-Instruccion TML 2741 generadora del sistema de acceso implantado.

APENDICE B: Mensajes de errores en el sistema TML 2741.

APENDICE C: Macro-instrucciones correspondientes al conjunto - de consistencias implantadas.

BIBLIOGRAFIA

INTRODUCCION

Durante los últimos años se está desarrollando un fenómeno sumamente significativo, la conjución de dos importante tecnologías; computación y comunicaciones.

La tecnología moderna de computadores tiene aproximadamente 20 años.

En seus inicios, los computadores actuaban como aparatos autosuficientes, con aplicaciones ocasionales que hacían necesaria la comunicación entre sistemas.

Dichas aplicaciones se fueron extendiendo, lo cual exigió una adaptación tecnológica para su desenvolvimiento y esto a su vez produjo una abertura en el campo de aplicaciones possibles.

El proceso de mezcla de tecnologías que lleva a la formación de sistemas complejos se ha ido desarrollando así, en forma casi exponencial.

Ya actualmente, los computadores están entremezclados - con sistemas de comunicaciones y las previsiones indican que esos sistemas integrados computación/comunicaciones van a crecer, desde los relativamente simples, a otros mayores, mas complejos, y que corresponden realmente a redes de transmisión, que interconectan subsistemas, los que a su vez podrían tener una estructura semejante a aquella a la cual están subordinados.

Los núcleos elementales en esta estructura, serían aquellos periféricos que estarían recibiendo y/o transmitiendo informaciones con el exterior, y a vez estarían sometidos al control de uno o más sistemas, a los cuales estarían subordinados.

Como ejemplo de dichos núcleos, podemos citar los periféricos normales del computador; lectoras de tarjetas, impresoras, unidades de disco, de cinta, etc., y aquellos más modernos conocidos como terminales de teleprocesamiento.

Estos sistemas complejos tendrían en su nivel más alto uno o mas sistemas que no estarían a su vez subordinados a ningún otro, sino que apenas interligados entre sí, y que podrían distribuir sus tareas de acuerdo al estado de la carga de trabajo circunstancial.

Estos sistemas que llamaremos Sistema Base tendrán subordinados a su vez sistemas satélites con su misma estructura, y así sucesivamente. Evidentemente tanto los sistemas bases como los demás subsistemas podrán y en general tendrán, como satélites, los núcleos elementales ya citados. Vemos en la Figura 1 la representación gráfica que tendría un sistema cualquiera, del tipo de que hemos idealizado.

Veamos cuales serían las vantajas que presentaría un sistema de este tipo y como deberían fijarse las relaciones de interdependencia.

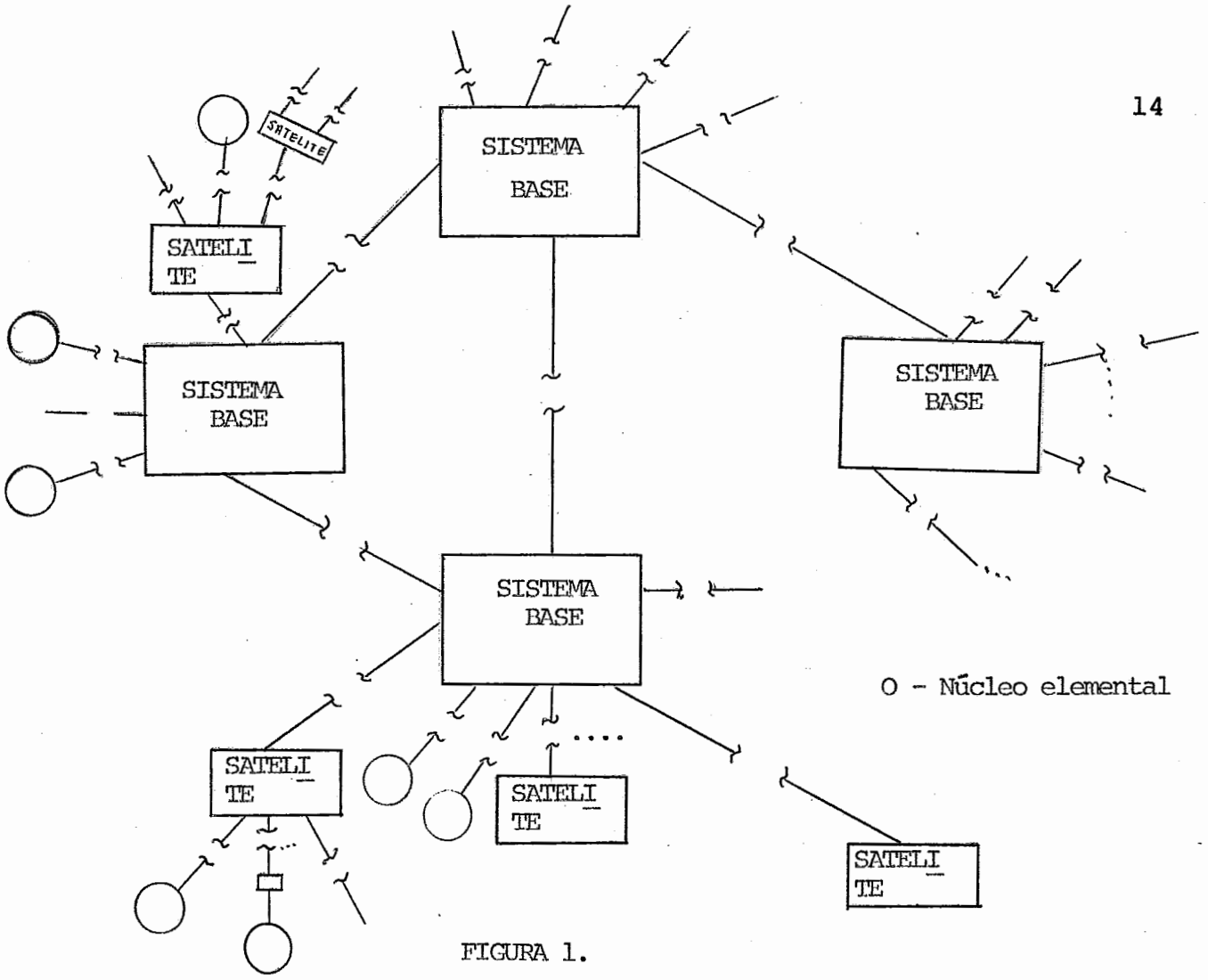


FIGURA 1.

Evidentemente tal clase de sistema permitiría, con una distribución geográfica adecuada y mediante un dimensionamiento - correcto de sus modulos, obtener a través de un mínimo de recur - sos el dar servicio a cualquier distribuicón de solicitaciones por irregular que ella se presente.

En efecto, los módulos periféricos cuya función sería el servir totalmente la demanda de tareas en su área geográfica de influencia podrían ser dimensionados para sus funciones de régimen normal siendo sus estados de sobrecarga solucionados por los sistemas a los cuales estarían subordinados.

Este concepto de sobrecarga que hemos presentado puede tanto referirse, en términos cuantitativos, al volúmen de tareas a realizar o en términos cualitativos, al carácter específico de ciertas tareas especiales que, en el caso de ser servidas por el propio módulo, obligaría a un sobredimensionamiento del mismo, - tanto en el aspecto de su hardware o de su software, que desmentiría la filosofía adoptada para el sistema global.

La desición acerca de si un módulo deberá sobredimensionarse o no para servir ciertas variaciones en el tipo y volúmen de la demanda corresponderá a un análise aconómico que deberá medir la inicidencia que tendría tal sobredimensionamiento sobre - el resto del sistema.

En el caso real ya se dispone de una cierta cantidad de recursos, en equipamiento, en software y en material humano que deben ser aprovechados.

Una primeira etapa debe ser en consecuencia realizar un levantamiento del total de recursos existentes, sus condiciones de funcionamiento y el volúmen y tipo de demanda que sirve.

Pero para la reavaluación de los equipamientos resulta sumamente impórtante tomar en cuenta que el Hardware disponible actualmente da mayores posibilidades para realizar una serie de aplicaciones hasta ahora nada convencionales.

Muchas veces estas mejoras de hardware pueden ser fácilmente acrecentadas, a costos relativamente razonables, a los equipamientos ya existentes.

Esto implica un reexamen de los recursos disponibles de procesamiento lo cual implica también un estudio del software existente y de su aplicabilidad en las condiciones de configuración y de demanda del computador en cuestión.

Otro aspecto que debe ser estudiado, pues puede influir tanto en la eficiencia del procesamiento como en el aprovechamiento de los recursos humanos disponibles para dar suporte a tal procesamiento es si existe coherencia, integración, entre los diversos sistemas que componen la estructura básica de servicios que los diferentes centros deben o deberán satisfacer.

En efecto, no tiene sentido el realizar un esfuerzo importante para aumentar la eficiencia de cada módulo, mediante el agregado de equipamientos más modernos y/o adaptando el software de soporte de los sistemas de procesamiento, sin examinar si éstos últimos a pesar de poder contar (como realmente acontece en la mayoría de los casos) con subsistemas de procedimientos y metas de gran analogía entre sí fueron elaborados como si se tratase de compartimientos estancos y este aspecto repetitivo del procesamiento que sobrecarga tanto los recursos en equipamientos como humanos, tiene una incidencia importante en la relación costo-performance del centro en particular.

En consecuencia concluimos que, aunque el llegar a desenvolver sistemas de alta sofisticación, que permitan un aprovechamiento intensivo de todos los recursos disponibles, es una meta absolutamente válida, aplicable inclusive en términos globales sin tomar en cuenta fronteras nacionales, resulta absolutamente imposterga - ble el procurar inicialmente desarrollar con um máximo de eficiencia los centros unitarios ya existentes con sus recursos y necesidades actuales.

Esta podría ser considerada como una primera etapa hacia los sistemas complejos que permitirían alcanzar las condiciones óptimas a que finalmente se desea llegar. Esto resulta aún más importante en el área latinoamericana en la cual el punto de estrangulamiento que impide un desarrolo más rápido justamente estriba en la disponibilidad de recursos de capital e humanos, que deben necesariamente acompañar tal proceso.

Siguiendo esta filosofía, tentamos completar en este trabajo un doble objetivo, por una parte hacer viable la utilización de ciertos adelantos de hardware que estaban a muestra disposición, pero que no disponían de un software adecuado para las condiciones de configuración actuales en el Núcleo de Computación Electrónica de la Universidad Federal de Río de Janeiro y por la otra

tentamos establecer la estructura de un sistema que aplique y exhiba la necesidad de usar técnicas integradas para dar asistencia a múltiples problemas comunes.

Un subconjunto de este último sistema fue implantado, en conjunto con la primeira parte, solamente a título de ejemplo para dar una idea inicial de la potencialidad que se podria alcanzar aplicando los criterios de integración expuestos.

La primeira sección de muestro trabajo consiste en la estructuración e implantación de un sistema de acceso para los terminales IBM-2741 mientras que la segunda fue el estructurar un sistema general de consistencia de datos ON-LINE finalizando con una pequeña aplicación de este último sistema usando el acceso de la primera parte.

SECCION 1

SISTEMA DE ACCESO PARA LOS TERMINALES 2741

CAPITULO 1

INFORMACION GENERAL

1.1 El Sistema Base.

El Núcleo de Computación Eletrônica de la Universidad Federal de Río de Janeiro tiene un computador IBM 360 MOD 40 con 256K bytes de memoria trabajando con el sistema OS/360 en MTF.

En lo que se refiere a su configuración de teleprocesamiento tiene una unidad de control 2702 y 10 terminales 2741 con sus correspodientes 10 líneas privadas ("nonswitched").

1.2 Software básico Disponible.

Sistema BTAM (BASIC TELECOMMUNICATIONS ACCESS METHOD)

Sistema QTAM (QUEUED TELECOMMUNICATIONS ACCESS METHOD)

Sistema TCAM (TELECOMMUNICATIONS ACCESS METHOD)

1.3 Sistema BTAM

El sistema BTAM es, de acuerdo a la nomenclatura del sistema/360, un método de acceso, lo cual significa que es un procedimiento para transferir datos entre la memoria y un aparato de Entrada/Salida.

Dentro de la filosofía del "Software" del sistema/360 los métodos de acceso se pueden clasificar en dos tipos fundamentalmente; el "BASIC" y el "QUEUED".

El tipo "BASIC" ofrece al programador el control de transferencia más próximo al nivel de máquina con excepción de lo que podríamos llamar el nivel "EXCP"(a través de la macroinstrucción

EXCP,EXECUTE CHANNEL PROGRAM). Permite y exige, a quien lo usa, decidir cuando desea recibir el control, si antes o después de estar completada la operación de Entrada/Salida (nivel READ/WRITE de macroinstrucciones de E/S), debe además verificar si dicha operación se realizó con o sin éxito y tomar las providencias necesarias para cada caso y aunque dispone de rutinas de asignación, automática de "Buffers", el programador debe realizar todas las tareas de control de filas de las diversas operaciones de E/S que se van sucediendo.

El tipo "Queued" ofrece todas las facilidades descritas para el tipo "Basic" y además controla todas las operaciones de E/S, liberando al programador de tal responsabilidad, crea y controla filas y además da la opción de recibir el control una vez que dichas operaciones ya han sido acabadas(nivel GET/PUT de Macro-Instrucciones de E/S.

El sistema BTAM,como lo expresa su nombre,corresponde al tipo "BASIC" y como tal, da una amplia liberdad al programador de elegir las opciones que más se adaptan a sus condiciones particulares de aplicación, por lo que se constituye en una buena herramienta para la construción de sistemas de acceso más sofisticados.

Este sistema BTAM, por otra parte, soporta todos los tipos de terminales remotos disponibles para ser usados en conjunción con el sistema/360.

Las exigencias de memoria del sistema BTAM son relativamente aceptables, sobre todo tomando en cuenta que de acuerdo a las necesidades particulares puede o no optarse por la utilización de sus rutinas padronizadas.

Más adelante, y ya en un caso particular, veremos la carga de memoria que significa el uso del sistema BTAM con las varias opciones que están disponibles.

1.4 Sistema QTAM

El sistema QTAM que corresponde a los métodos de acceso de tipo "QUEUED" tiene, para nuestro problema particular, un aspecto que lo elimina totalmente: no da soporte a los terminales 2741.

Sin embargo estimamos importante dar algunas referencias sobre su filosofía de funcionamiento y carga, en términos de memoria, pues consideramos que aún para otros terminales que no sean los 2741 citados y en ciertas condiciones de configuración se hace necesario, para un uso extendido de las aplicaciones de teleprocesamiento, el crear un software especial de soporte para aquellos terminales de que se disponde.

En lo que se refiere a la filosofía de funcionamiento, las rutinas de control de mensajes del Sistema QTAM actuan en forma asincrónica con los programas que habrán de procesar dichos mensajes.

Los mensajes son recibidos y almacenados en un archivo en disco, del cual serán retirados ante las solicitaciones del programa que habrá de usarlos.

Esto tiene una analogía clarísima con los conocidos sistemas de "Spooling" que tratan de evitar de esa forma la influencia negativa que tiene, con respecto al procesamiento, la baja velocidad de transferencia de informaciones de ciertos tipos de periféricos del computador(v.g.lectora de tarjetas e impresora).

Sin embargo esta filosofía, que aparentemente sigue una línea comprobada y testada de aumento de eficiencia, choca con una caracteristica esencial de las aplicaciones de teleprocesamiento y que incluso es quizás, el factor básico en su amplia y rápida expansión: el aspecto conversacional.

En una gran mayoría, las aplicaciones de teleprocesamiento exigen una consistencia inmediata de los datos de entrada y una respuesta de acuerdo a tal control y al procesamiento de esos datos, que permitan continuidad y confiabilidad en tiempo real del flujo de imformaciones que se suceden.

Esto implica recepción y transmisión sucesivas de mensajes (los cuales en algunos casos pueden reducirse simplemente a caracteres de control).

Desde el punto de vista de un sistema como el QTAM lo expuesto se traduce en la siguiente secuencia de operaciones:

- Recepción del mensaje
- Almacenamiento en la unidad de acceso directo.
- Retiro de tal unidad.
- Mensaje de disposición del programa que lo habrá de procesar.
- Procesamiento del mensaje.
- Mensaje a transmitir a disposición de la rutina del acceso.
- Almacenamiento en la unidad de acceso directo.
- Retiro de tal unidad.
- Transmisión del mensaje.

Tal sucesión de operaciones para una aplicación conversacional, corresponde en tiempo real a la suma de los tiempos de cada una de las tareas elementales citadas.

Del punto de vista de eficiencia del sistema, en tales condiciones, todas las etapas de transferencias transitorias y de control de esas transferencias resultan totalmente innecesarias.

Por otra parte la filosofia de un sistema como el presentado se basa en una transferencia ininterrumpida de datos entre el periférico de baja velocidad y la unidad de acceso directo, en ambos sentidos, de tal forma que la transferencia desde

y hacia el programa que procesa los mensajes, que se efectuará a través del disco, simule un periférico de velocidad mucho mayor.

Si en cambio el periférico debe esperar una respuesta para continuar la transferencia, el programa estará afectado por la velocidad de tal periférico y todas las etapas intermediarias de procesamiento y almacenamiento en disco contribuirán a quitarle eficiencia al sistema.

La ventaja que presenta el sistema QTAM frente a uno que no mantenga filas en disco es que el primero resultará mucho más eficiente en aplicaciones como Message Switching, Data Collection, con un ingreso de grandes masas de datos, y otras aplicaciones similares.

Sin embargo, dado que el aspecto conversacional representa una característica básica en la grande mayoría de las aplicaciones de teleprocesamiento, concluimos que el sistema QTAM, aún cuando es sumamente general, debe ser postergado ante tal clase de aplicaciones.

Nótese que llegamos a tal conclusión sin tomar en cuenta otro aspecto que debe evidentemente pesar en la toma de decisiones a que estamos haciendo referencia y que es el de la carga para el sistema en términos de memoria que significa la utilización del metodo de acceso en cuestión.

Dado que el sistema QTAM no da soporte a los terminales 2741 nos resulta imposible presentar la carga de memória que correspondería a una configuración como la que disponemos.

En consecuencia, y dado que nuestro mayor interés consistirá en establecer una comparación entre los diferentes "software" disponibles usaremos los ejemplos presentados en el manual de Storage Estimates del sistema/360

1.5 Sistema TCAM

Previamente a las consideraciones de memoria veremos en términos generales algunos conceptos del sistema TCAM.

El sistema TCAM permite, a través de una serie de macroinstrucciones, generar un programa de control para transferencia de mensajes entre las diversas unidades que conforman una configuración de Teleprocesamiento.

Dicho programa de control, conocido como MCP (Message Control Program), al ser generado dispone de una serie de opciones que le permiten adaptarse con bastante eficiencia a las condiciones particulares de aplicación.

En efecto, para aplicaciones de Message Switching con Mensajes de gran tamaño, permite y dispone de rutinas para mantener filas en disco y manipular dichas filas, en cambio para

aplicaciones de tipo conversacional permite optar por que dichas filas se mantengan en la memoria, evitando así los pasos inefi - cientes que citamos en el sistema QTAM.

Para la comunicación entre el MCP generado y los diversos programas de aplicación, dentro del sistema de teleprocesamiento TCAM exige y da facilidades para generar rutinas de interface que permitirán que programas con diferentes estructuras usen el mismo MCP.

La importancia del sistema TCAM surge naturalmente del hecho de que soporta todos los terminales del sistema/360, así como de su aplicabilidad en términos correctos a una variedad muy grande de problemas particulares.

Sim embargo es justamente en esta generalidad que esta su talón de Aquiles, toda la potencialidad expuesta exige una utilización importante de la memoria, lo cual no resulta compatible con ciertas configuraciones, como en el caso del Modelo 40.

Este modelo, cuya configuración máxima corresponde en términos de memoria a 256K bytes, trabajando con el OS/360 debe tener un módulo residente del orden de los 60K bytes, la "writter" residente, del orden de 16K bytes, y en caso particular del NCE tiene un programa de pasaje de lectora de tarjetas para cinta magnética, con la finalidad de evitar el congestionamiento en la recepción de programas, que ocupa 8K bytes residentes.

Con esta distribución, se tiene la posibilidad de trabajar con 2 particiones de 90K y 82K bytes respectivamente.

En tal configuración resulta sumamente dificil aumentar el modulo residente en una cantidad mayor a los 10K bytes pues se obtendrían particiones inoperativa para multiples aplicaciones.

En definitiva, eliminado el programa lector y exprimiendo a un máximo las posibilidades de disminución del módulo residente, se podría alcanzar hasta 20K bytes disponibles para ser usados en las aplicaciones de teleprocesamiento.

Problemas similares aparecen en otras configuraciones en las cuales un aumento sustancial de la parte residente puede afectar seriamente la performance del sistema.

1.6 Necesidades de Memoria

El manual de Storage Estimates del sistema/360 referenciando justamente BTAM,QTAM y TCAM, presenta ejemplos de necesidades de memoria, en ciertas configuraciones particulares, que aunque no resultan coherentes entre sí, están lo suficientemente próximas como para das ideas de magnitud y de relaciones entre tales métodos de acceso.

BTAM- 9,3K bytes

QTAM- 36,5K bytes

TCAM- 42,1K bytes

Corresponde hacer la observación de que en términos práticos, resulta totalmente imposible efectuar cualquier aplicación de teleprocesamiento, con los sistemas QTAM e TCAM, usando un volúmen de memoria inferior a los 100K bytes.

Como puede observarse en una configuración como la que disponemos, el uso de los sistemas QTAM y/o TCAM resultan absolutamente prohibitivos y en consecuencia la unica opción para poder alcanzar un uso generalizado en aplicaciones de teleprocesamiento, es el de crear, un sistema de acceso aplicado a la configuración disponible usando como herramienta ya sea el sistema BTAM o, sin usar ningún sistema disponible especial, mediante el uso de la macro-instrucción EXCP ya citada en el paragrafo 1.4.

En el capítulo siguiente tentaremos recorrer explicitamente todas las alternativas que se nos presentaron durante el desarrollo de este trabajo y las decisiones que tomamos en cada caso.

CAPITULO 2

FILOSOFIA DEL SISTEMA

2.1 Definiciones

Nuestra meta, en consecuencia de lo visto anteriormente será construir un sistema de acceso para terminales de teleprocesamiento, en aplicaciones de tipo conversacional, dentro de una estructura con grandes limitaciones en cuanto al volumen de memoria disponible y con un "software" que no se adapta a las condiciones particulares de configuración.

Planteado el problema, surge de imediato la necesidad de establecer cuales habrán de ser los objetivos y exigencias que tal sistema deberá satisfacer.

Las condiciones citadas dependerán de todas aquellas estructuras de las que el sistema de acceso formará parte y de las caracteristicas particulares en que pasará a participar.

Logicamente tal diversificación de estructuras con metas y necesidades diferentes y hasta posiblemente, en algunos casos, conflictuantes, exige el establecer criterios de prioridades, en los puntos en disputa, visando los objetivos del sistema global.

2.2 Objetivos del Sistema

Veamos a continuación una a una, cuales son las metas que se alcanzar y los estados de conflicto que aparecen si se quieren aplicar en forma simultánea:

A- Velocidad máxima de procesamiento.

B- Mínimo uso de la Memoria

C- Generalidad de aplicaciones

D- Uso sencillo.

E- Alta performance del sistema.

2.3 Memoria, Velocidad y Performance

Los puntos A, B representan, aparentemente en conjunto, el objetivo que denominamos E.

Esto, que en parte resulta cierto, no se ajusta totalmente a la realidad, pues aunque se debe llegar a una solución de compromiso de los puntos A y B(dado que resultan conflictuantes) si se desea obtener una buena performance del sistema de acceso, en si mismo, existen otros factores que inciden tanto sobre la aplicación particular como sobre la eficiencia de funcionamiento del sistema global.

Como ejemplo característico conviene citar el de los estados de espera del sistema (WAIT).

Supongamos inicialmente un procesamiento en monoprogramación. Evidentemente cuanto menor sea el uso de memoria por parte del acceso, mayor será la parte disponible para el programa de aplicación lo cual disminuirá la probabilidad de necesitar

procesar algunos casos en "OVERLAY" que resulta costoso en términos de velocidad.

En esta etapa de nuestro razonamiento surge ya la competencia memoria-velocidad pues en la mayor parte de las aplicaciones, si se desea procesamiento rápido, se debe pagar en términos del volumen de memoria utilizada.

En el ejemplo planteado, el minimizar la memoria del acceso significa hacer más lento el procesamiento de todas las aplicaciones evitando que parte de aquellas que exigen grandes volumenes de memoria sean procesadas, en OVERLAY.

La toma de decisión corresponderá a un estudio de la distribución de la carga de aplicaciones, en cuanto al volumen de memoria necesaria. Si se tratara de aplicaciones pequeñas se debería optimizar velocidad, en el caso contrario(aplicaciones grandes) se optimizaría memoria y en circunstancias medias se debe optar por una solución intermediaria, de compromiso, entre ambas alternativas extremas.

En todo este desarrolo excluimos propositadamente la consideración de los estados de espera(WAIT).

La performance de un sistema no mejora si disponemos de un procesamiento de alta velocidad que debe ser interrumpido durante un espacio de tiempo apreciable esperando por la finalización de una operación de Entrada/Salida. Este es justamente

nuestro caso pues los terminales de tipo conversacional son de baja velocidad de transferencia.

Otro aspecto ya visto en las aplicaciones de tipo conversacional es que el procesamiento condiciona y modifica en tiempo real las operaciones de Entrada/Salida y el mismo depende de la operación inmediata anterior lo cual impide que el procesamiento asincrónico (CPU-versus-E/S) que aumentaría la eficiencia global sea importante dentro de la sistemática del acceso.

Veremos más adelante que tal procesamiento asincrónico es estimulado dentro de los programas de aplicación e inclusive podría ser forzado.

En consecuencia y con referencia al ejemplo visto, la velocidad de procesamiento, el volumen de memoria usado, la velocidad de los periféricos de E/S, el número de tales periféricos que simultáneamente pueden transmitir datos al o desde el sistema, y la performance, son conceptos que interactúan entre sí de tal modo que para tomar decisiones, en cuanto a los criterios de funcionamiento de un sistema, se deben conocer y tomar muy en cuenta tales ligaciones.

Lo expuesto resulta coherente con un procesamiento en multiprogramación, el minimizar memoria corresponde a permitir un procesamiento más rapido en aplicaciones mayores o una coexistencia simultánea de más programas en la memoria a los efectos de competir por el control en el procesamiento por multiprogramación, el maximizar velocidad,(exige un mayor uso de memoria) en lo que se refiere a la performance del acceso, mantiene la restricción anotada para el caso de monoprogramación, de la incoherencia de una gran velocidad en procesar los controles de la operación de E/S seguidos por ésta con una velocidad de transferencia extremadamente baja. Sin embargo, para el sistema global, cuanto más rapido sea el procesamiento, mayor será el tiempo disponible para las otras aplicaciones, que reciben el control cuando las de mayor prioridad esperan por la finalización de una E/S y aparentemente mayor sería la performance obtenida. Esto resulta solamente aparente pues, dado que el tiempo para la operación de E/S, en terminal, es grande, eso permite que durante ese intervalo, el control se vaya alternando entre los restantes programas que coexisten en la memoria disputándoselo.

Si el número de estos programas resulta pequeño en virtud de la propia configuración y del uso de memoria del acceso (mayor por tratarse, en este ejemplo, de un sistema veloz) es

súmamente probable que los mismo pasen rapidamente a ejecutar operaciones de E/S que superpuestas van a dejar el sistema en estado de espera (WAIT) lo cual nos llevaría a iguales conclusiones que en el caso de monoprogramación.

En definitiva los puntos citados como A,B y E deberán ser cuidadosamente sopesados en cada caso particular: de configuración, aplicaciones, sistema supervisor, etc. No corresponderá a esta parte, en que tentamos establecer criterios generales para nuestro sistema, el estudar a partir de las condiciones particulares en que éste será implantado cuales será las características óptimas en cuanto a la performance del sistema global incluyéndose en ésta el uso de memoria, la velocidad del procesamiento, la minimización de estados de espera, etc.

2.4 Simplicidad vs. Generalidad

Los aspectos más importantes que aún restan por considerar son los citados como C y D.

Por un lado es sumamente importante que el acceso soporte una gran generalidad de aplicaciones particulares y por otra parte resulta vital que el sistema sea sencillo de ser usado por un programador medio.

Esto último exige que las técnicas y conceptos necesarios de se aplicados en las aplicaciones de teleprocesamiento

soportadas por el acceso, que deseamos construir, sean análogas a aquellas que participan de la mayor parte de las aplicaciones - procesadas en "batch".

He aqui una crítica para los sistemas QTAM y TCAM, pues éstos aunque facilitan la programación de las rutinas de aplica - ción exigen, para la generación de los MCP (Message Control Program) a través de las macroinstrucciones disponibles, ya sea en uno u otro método de acceso, un conocimiento vasto y profundo de estos dos sistemas.

Si agregamos el hecho de que es bastante limitada la generalidad de aplicaciones que pueden ser soportadas por cada MCP en condiciones de relativa eficiencia, concluímos que el número de MCP's que deberán ser generados crece en forma correlativa al número de aplicaciones esencialmente diferentes.

Considerando además que las características de las macro instrucciones que componen dichos sistemas son sumamente diferentes en cuanto a los conceptos que se deben manipular, a las que se acostumbra usar en el procesamiento normal, pues exigen criterios estrictamente de teleprocesamiento combinados con la filosofía y estructura de los propios accesos (que son sumamente complicados incluso para personas con un conocimento relativamente general, en ciencias de computación) llegamos a la conclusión que puede incluso resultar menos complicado y por supuesto bastante más eficiente,

el programar accesos para las citadas aplicaciones usando niveles de programación mucho más cercanos a las caracteristicas propias de los equipamientos (por ej. BTAM o incluso a nivel de EXCP).

En definitiva, en nuestro concepto, los sistemas como QTAM y TCAM aún cuando presentan la ventaja de dar soporte a una variedad muy grande de terminales de teleprocesamiento diferentes no resultan apropiados para serusados por parte de una configuración particular dado que exigen manipular conceptos tanto o más complicados que aquellos que justamente tienen como objetivo evitar, los de transmisión y control de mensajes entre equipos pertencientes a una estructura de teleprocesamiento.

Por otra parte la ventaja anotada es en especial inportante para el fabricante pues una única unidad de "software" le permite asistir a una variedad grande de clientes con configuraciones totalmente dispares pero a éstos, que disponen de una variedad de terminales pequeño, siendo incluso deseable el limitar al máximo tal variedad para evitar una manutención y soporte en términos de personal y materiales evidentemente onerosa, tal generalidad en su soporte de software, baja la eficiencia operativa (en términos de memoria y velocidad) y lo que es más grave, encarece y dificulta los servicios de Análisis y Programación.

Este paréntesis en el desarrollo que estábamos realizando, respecto de la doble necesidad, que como veremos resultará conflictiva, de dar generalidad y a su vez facilitar el uso del sistema, tuvo como finalidad demostrar cuan importante puede resultar un aspecto aparentemente secundario como lo es el de simplificar la programación, que incluso como concluimos en los casos vistos, puede llegar a efectar seriamente la aplicabilidad del sistema considerado.

Resulta evidente que generalidad y simplicidad son opciones conflictivas pues cuanto más general es un sistema, más detalles deben ser considerados para llegar a una solución particular.

En consecuencia, concluimos que se deben estudiar las aplicaciones más variadas posibles con las disponibilidades de hardware de forma de construir un sistema que a partir de tales limitaciones físicas permita una mayor generalidad sin dejar de considerar, el facilitar la programación, por ejemplo, manteniendo y en ciertas condiciones, generalizando los conceptos clásicos usados en "batch" para las aplicaciones de teleprocesamiento.

2.5 Lenguages de Programación

Un aspecto particularmente importante, en terminos de facilidades de utilización, es el de permitir que los programas de aplicación que habrán de usar el acceso pueden ser

escritos en cualquier lenguaje de alto nivel. En caso contrario, como sucede con los sistemas TCAM y QTAM, el lenguaje sería un limitador al desarrollo de las aplicaciones en cuanto a su volumen complexidad, exigiendo además la interacción con un subconjunto selecto del total de usuarios posibles.

Si tomamos en cuenta además de que la tendencia actual es la de generalizar el uso de los lenguajes de alto nivel, tanto para aplicaciones como para ciertos programas de base (como ser compiladores) dejándose los lenguajes simbólicos (a nivel de máquina) para aquellas aplicaciones en que resultan estrictamente necessarios, concluimos que la disponibilidad de personal capacitado para desenvolver volúmenes importantes de aplicaciones de cualquier complexidad con lenguajes a nivel de maquina será limitado y por lo tanto sumamente oneroso, este tipo de operación.

2.6 Tiempo de respuesta

Finalmente, debemos considerar un aspecto que no está incluido, por lo menos explícitamente, en los items ya planteados, el tiempo de repuesta.

Tiempo de respuesta sería, por definición, el período de tiempo entre la transmisión del último caracter de un mensaje de entrada y la recepción por parte del terminal del ler carác

ter de un mensaje de salida.

Esta es una definición que sirve a nuestros propósitos aunque difiere de varias de las definiciones dadas, por los diversos autores, a ese término.

Lo que realmente interesa, a los efectos de nuestro trabajo, es que se ha establecido, con base estadística, un tiempo de respuesta máximo aceptable para un funcionamiento correcto de una aplicación conversacional.

Este tiempo corresponde al máximo que el operador del terminal permanece esperando la respuesta sin perder la calma pues en caso contrario se ha verificado que su tendencia es tentar actuar sobre el terminal para verificar si aconteció algún defecto técnico en el mismo.

Tal cosa podría afectar, dependiendo de las características físicas del terminal, y del software que lo soporta, la continuidad del procesamiento en forma más o menos grave de acuerdo a cada circunstancia particular.

En consecuencia nuestro sistema deberá ser concebido de forma de minimizar tal tiempo de respuesta en las condiciones de mayor carga, tanto de aplicaciones de teleprocesamiento como de batch, que coexisten con aquellas en una estructura de multiprogramación, tentando no superar aquel máximo cotado. El alcanzar tal objetivo podría resultar prohibitivo en las

condiciones de configuración en que el sistema sería aplicado por lo cual en circunstancias particulares tal máximo resultaría sobrepasado.

En éste último caso será de primordial importancia crear una protección de Software que, aprovechando y en base a las características del hardware, impida que el operador del terminal afecte el proseguimiento de la operación.

La implantación de un sistema con un tiempo de respuesta corto implica considerar los siguientes aspectos independientes:

a) Procesamiento rápido de las operaciones de control de transferencia de mensajes de forma de dar más rapidamente el control al programa de aplicación.

b) Crear las condiciones necesarias para que las rutinas de aplicación pueden iniciar las operaciones de E/S con los terminales y continuar en forma simultánea su propio procesamiento. De esa forma se consigue reducir al máximo el número de operaciones a procesar entre cada operación de E/S, siendo aún más importante, de ser posible, el evitar realizar operaciones de E/S con otros periféricos durante ese periodo intermediario, postergándolos para después de iniciar la transferencia con el terminal.

c) Dentro de las limitaciones existentes, dar un máximo de prioridad de procesamiento, tanto al sistema de acceso como a los programas de aplicación que con él interactúan, en contraposición a los programas en "batch".

De ser posible, para alcanzar un máximo de rapidez de respuesta, el sistema global de teleprocesamiento (Sistema de acceso + rutinas de aplicación) solo debería ceder el control al procesamiento "batch" cuando todas las rutinas que lo componen estuvieran en estado de espera (WAIT).

2.7 Conclusiones

En resumen, en vista de todas las consideraciones hechas respecto, a los criterios que deberán ser empleados en la elaboración de un sistema de acceso como el que deseamos, partiremos de los siguientes postulados:

1 - La configuración del sistema en términos de memoria es sumamente restricta y este aspecto limitará la amplitud del desarrollo de los postulados que siguen.

2 - Los programas delaplicación deberán poder ser escritos en lenguajes de alto nivel.

3 - Los conceptos a ser manipulados, para generar el sistema y para programar las aplicaciones, deberán ser a lo sumo generalizaciones de aquellos ya usados, en el procesamiento tradiconal, en "batch".

4 - Deberá servir a la gran generalidad de aplicaciones posibles con la configuración hardware de que se dispone.

5 - Menor tiempo de respuesta posible aplicando como - postulados los items A.B Y C del parágrafo anterior.

6 - Complementar la protección "hardware", mediante un Software" que impida que el operador del terminal a fecte el proseguimiento de la aplicación mediante acciones, imprevistas por el programa.

CAPITULO 3

ESTRUCTURA DEL SISTEMA

3.1 CONFIGURACION BASE

Las condiciones en que debemos localizar nuestro sistema de acceso, como ya las hemos definido en el capitulo anterior, corresponden a una instalación de tiempo real con terminales de caracteristicas intrínsecas conversacionales.

Una instalación se dice de tiempo real si es tal que recibe datos de terminales remotos, los procesa y devuelve los resultados con suficiente rapidez de manera de afectar las condiciones en que está operando.

De acuerdo a esta definición , un sistema funcionando en forma conversacional es un caso particular de una instalación en tiempo real.

La figura 2 ilustra la configuración típica en la que se deberá construir el acceso.

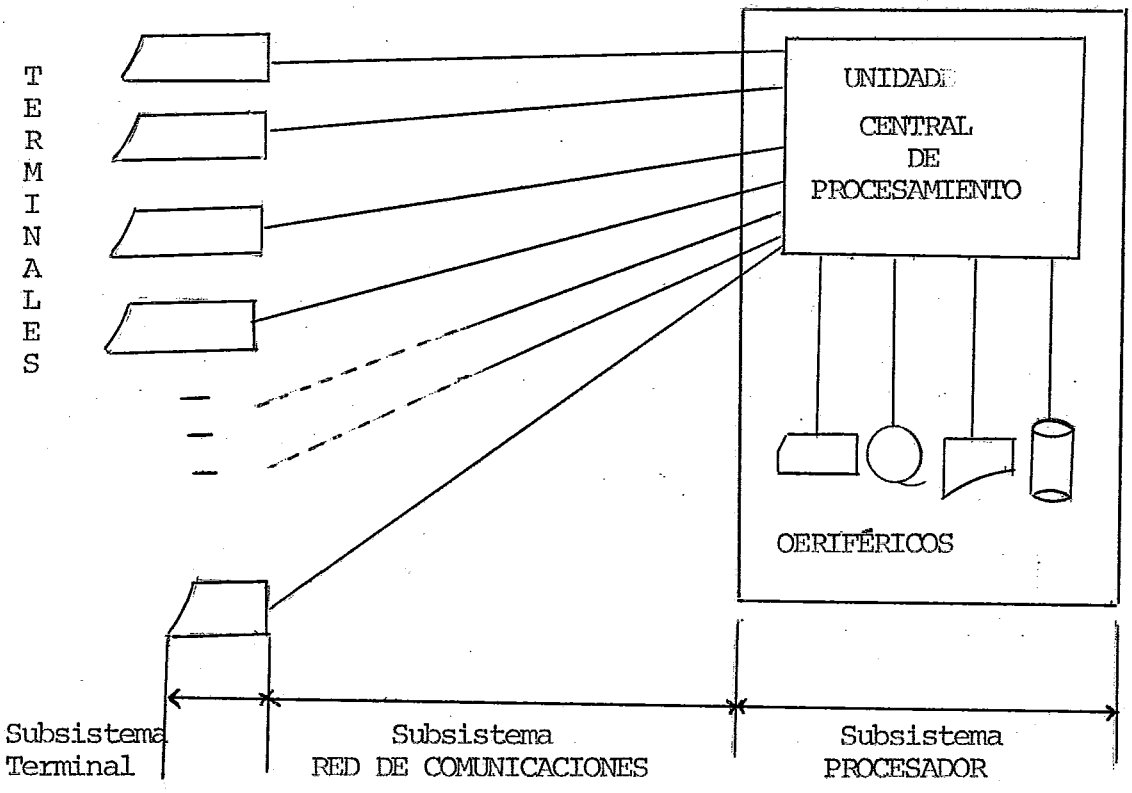


FIGURA 2.

3.2 Subsistemas- Relaciones

Como se puede observar hemos dividido la configuración de tiempo real en tres subsistemas, terminal, red de comunicaciones y procesador.

El subsistema "terminal" incluye los terminales y las solicitaciones sobre ellos que corresponderán a aplicaciones conversacionales.

El subsistema "red de Comunicaciones" incluye todos los canales de comunicaciones y equipos asociados como ser los "MODEM", los conmutadores telefónicos, etc.

Finalmente, el subsistema "procesador" incluye el procesador propiamente dicho, los equipos periféricos clásicos, el "Software" y los programas de aplicación.

Existe una relación de correspondencia entre los terminales, y los programas de aplicación cuando estos están en ejecución. A cada terminal le corresponde un único programa de aplicación y a cada programa de aplicación le puede corresponder cero, uno o más terminales.

Lo que no podrá suceder en nuestro sistema es que dos o más programas de aplicación diferentes se correspondan con un mismo terminal aún cuando más de dos terminales diferentes pueden corresponder perfectamente a un mismo programa de aplicación.

Para estructurar el sistema de acceso consideraremos solamente los subsistemas "Terminal" y Procesador" haciendo total abstracción del subsistema "red de comunicaciones."

Esto se debe a que éste último subsistema no influirá sobre la estructura del "Software" propiamente dicho sino sobre los controles de transmisión que deberán ser realizados, es decir que solamente influirá, en la fase de implantación, sobre los comandos que deberán ser utilizados para aquellos fines.

Es sin embargo muy importante la consideración de la red de comunicaciones para determinar por ejemplo, el tiempo de respuesta, la confiabilidad de la transmisión, la relación costo / performance de la comunicación y su influencia sobre el resto de la configuración, etc.

3.3 Interface entre Programas y Acceso.

En definitiva deberemos estructurar un sistema que permita una interrelación firme entre un conjunto de programas de aplicación, que pueden estar escritos en cualquer lenguaje ya sea de alto nivel o no, y los terminales de teleprocesamiento.

Esto planeta un primer problema, dado que por un lado deberemos tener un núcleo común de acceso, el cual podrá estar integrado por una o varias rutinas que realizarán todas las operaciones de transmisión y control para permitir la comunicación entre programas y terminales, y por otro lado esos programas escritos en lenguajes diferentes.

Tendremos en consecuencia que crear un interface entre tal núcleo común y los diversos programas de aplicación.

Tal interface no debe necesariamente ser una rutina que transforme todas las caracteristicas divergentes de los diferentes languajes y que podrían influir en la cohesión de los programas con el núcleo de acceso, llevándolas a un formato común, sino que eventualmente podría corresponder, en la fase de implantación, a una transformación de los diversos traductores, mediante la introducción de nuevos comandos y/o generalizando otros ya existentes, para interligar los programas al núcleo en forma directa,

Tal alternativa será considerada en el próximo capítulo - que corresponde a la implantación del sistema.

En consecuencia nuestra estructura básica corresponderá en esencia a lo representado en la FIGURA 3.

Puede verse en el citado esquema lo que llamamos AREA DE COMUNICACION, que servirá para transferencia de datos. Lógica - mente debe existir comunicación de informaciones entre el programa de aplicación y el núcleo que habrá de procesar el acceso.

Tales informaciones corresponderán a la necesidad de cuatro tipos diferentes de datos:

1) Datos transmitidos o recibidos de los terminales (mensajes)
2) Informaciones sobre diversas caracteristicas del mensaje(número de caracteres,codigo de transmisión, etc,).

3) Códigos indicando operación de Entrada/Salida efectuada o por efectuar.

4) Códigos indicando el resultado de dichas operaciones de Entrada/Salida. (Realizadas con suceso o si se verificó error).

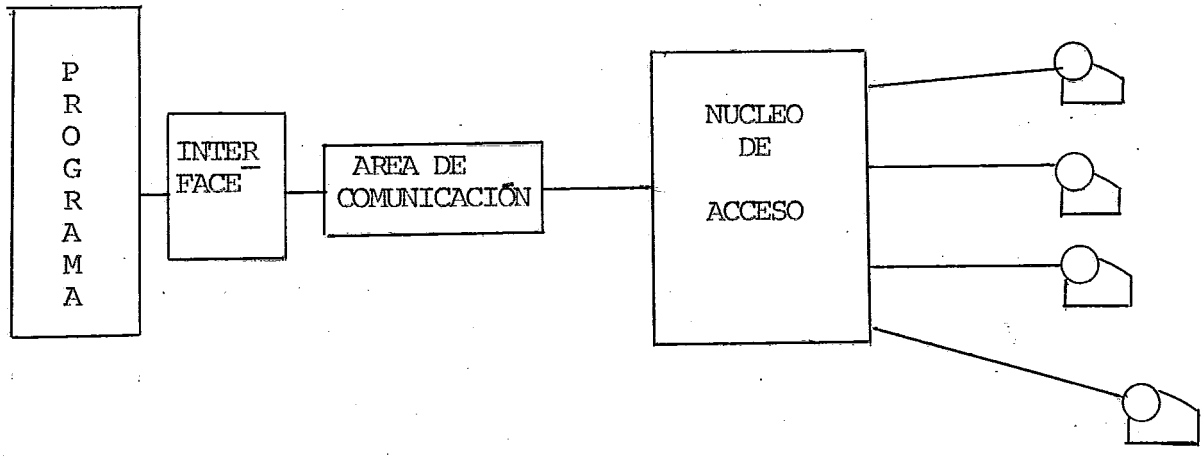


FIGURA 3.

3.4 Estructura del núcleo de Acceso

Ya hemos establecido en términos generales cuales habrán de ser las informaciones que el núcleo principal deberá utilizar.

Hay un aspecto que sobresale en cuanto a las características de esas informaciones y que es la repetición de la palabra código.

Por una parte tendremos el código interno del computador que permite el procesamiento de los mensajes, también los códigos para facilitar y uniformizar el resto de las informaciones, algunas de las cuales serán procesadas y otras simplemente indicadoras de estado, y finalmente tendremos los códigos propios de las unidades de teleprocesamiento, los que corresponden a la transmisíon propiamente dicha.

En consecuencia, en nuestro sistema, deberemos tener estructuras codificadoras y decodificadoras.

Como vimos, esas necesidades de codificar y decodificar corresponden a elementos distintos del procesamiento y por tal causa la definición del alcance, ubicación e incluso de la existencia de tales módulos CODEC(codificadores-decodificadores) serán motivo de un análisis más minucioso de su influencia sobre el sistema total y que veremos una vez que ya tengamos al mismo aproximadamente estructurado.

Volviendo ahora al núcleo principal del acceso, éste logicamente estará integrado por una o más rutinas y sus correspondientes áreas de trabajo.

Resulta importante hacer notar que, dadas las restricciones de memoria que por hipótesis habrán de limitar la extensión tanto del acceso como de las aplicaciones, deberemos estructurar nuestro sistema en forma modular, tentando que esos módulos sean lo suficientemente elementales (Yen consecuencia pequeños en términos de memoria) como para dar una gran versatilidad en la elección de aquella estructura que más se adapte a las condiciones particulares de configuración y aplicación.

Siguiendo tal criterio tentamos disecar las operaciones elementales, que habrán de ser ejecutadas por el acceso, para definir las estructuras unitarias que podrán o deberán integrar el sistema frente a ciertas necesidades particulares:

A) MODULOS CODEC DE COMANDOS DE ENTRADA/SALIDA.

B) MODULOS DE EJECUCION DE OPERACIONES ELEMENTALES DE ENTRADA/SALIDA.

C) MODULOS DE ESPERA DE FINALIZACION DE OPERACIONES DE ENTRADA/SALIDA (WAIT).

D) MODULOS DE EDICION DE MENSAJES DE ENTRADA/SALIDA

E) MODULOS DE RECUPERACION DE ERRORES.

F) MODULOS DE INTERPRETACION DE TRANSCEDENCIA DE ERRORES.

G) MODULOS DE INICIALIZACION.

H) MODULO DE COMUNICACION CON PROGRAMAS DE APLICACION.

I) MODULO DE DETECCION DE ERRORES Y CODIFICACION DE LOS MISMOS.

3.5 MODULOS BASICOS Y OPCIONALES

Observando esta lista concluimos que tenemos modulos de dos tipos: básicos y opcionales.

Los basicos resultarán naturalmente obligatorios dentro del acceso y corresponden a la ejecución de las operaciones fundamentales citadas en la lista como A,B,C,H,I.

Los módulos opcionales D,E y F implican decisiones sobre medidas a tomar frente a las alternativas del procesamiento.

Si los objetivos del programa de aplicación resultan coherentes con dichas decisiones, el usuário podrá optar por dejar al acceso la ejecución de las tareas citadas y en caso contrario deberán ser realizadas por él mismo, de acuerdo a sus necesidades.

Esta filosofía de que el sistema sea, en su estructura básica, independiente de las aplicaciones que deberá satisfacer mientras que a su vez disponga de una estructura opcional claramente dependiente y dirigida hacia aquellas necesidades que integran la gran mayoría de las aplicaciones, es común en los sistemas modernos de procesamiento de datos.

Esto tiene una doble ventaja pues una parte permite una gran generalidad de aplicaciones (dentro de las limitaciones - del hardware) con el minimo de memoria posible(acumulando los módulos basicos solamente) y por otra parte, para un grande número de aplicaciones, permite el uso de opciones del sistema - que facilitan grandemente la programación, evitando que el usuario deba ocuparse de los errores, de los códigos de transmisión y de otras tareas que como las de edición exigen una programación delicada.

En consecuencia tendremos un módulo básico, común a todas las aplicaciones y una serie de módulos opcionales, dependientes del programa de aplicación particular.

Si analizamos las características que deberían tener tales módulos en términos de memoria, lógicamente aquellos generales a todas las aplicaciones ya sean residentes o transitorios deberán ser asignados como sobrecarga del supervisor, mientras que los restantes, siempre transientes, deberán ser acumulados a los programas de aplicación dentro de los volúmenes de memoria asignados a éstos.

3.6 Estructura General

Finalmente, si redistribuimos los diferentes módulos de nuestro sistema, tendremos la estructura de la FIG 4 en la cual

tendremos dos blocks separados, block de aplicación y block de acceso, siendo éste último integrado exclusivamente por el núcleo básico mientras que el primero, de configuración y estructura variable, estará formado por el programa de aplicación, el núcleo interface(pudiendo éste estar separado o integrado al programa) y los diversos módulos opcionales, vinculados al acceso.

Otra característica que puede ser observada en la Figura 4 es que el Área de comunicación, que definimos en el parágrafo 3.3, fué considerada integrando el block de aplicación. Esto se debe al hecho prático de que deberemos tener un área de comunicación por cada programa de aplicación y dado que el número de éstos es indeterminado, la presente es la única asignación posible.

La filosofía expuesta de asignar, y en consecuencia cargar, en términos de memoria, al programa de aplicación con las rutinas y áreas correspondientes a sus necesidades particulares tiene también implicancias con las áreas de buffers y de control de transmisión asignadas a cada terminal. Esto surge como consecuencia directa del hecho de que, dada la configuración cualitativa y cuantitativa del sistema de teleprocesamiento, el programa de aplicación puede optar por usar aquellos terminales necesarios a sus necesidades.

En caso de integrar todas las áreas citadas al núcleo básico, éstas habrán de pesar igualmente sobre todos los programas sin discriminar aquellos que harán mayor uso de la configuración, de aquellos otros que lo harán en menor grado.

En consecuencia, y de ser posible, en condiciones de implantación, también se deberán asignar las areas de Buffers y de control de cada terminal al programa que lo utilice.

De acuerdo a los postulados establecidos en el parágrafo 2.7 del capítulo 2 restaría considerar en nuestra estructura el postulado Nº 6, el cual, aunque implica una medida esencial, no puede ser incluido como parte básica de nuestro sistema y si opcional, pues, aún cuando, en términos generales se pueden definir las medidas a tomar en cada caso, éstas habrán de limitar la libertad del programador de decidir, en base a sus necesidades, sus propias acciones.

La labor del sistema, en su aspecto básico, deberá ser la de comunicar al programa en forma codificada cual ha sido la acción del operador del terminal permitiendo a través de comandos, que el programador adopte su medida particular.

Por otra parte deberán existir módulos transitorios que habrán de tomar acciones predefinidas por el sistema y de una aplicación sumamente general.

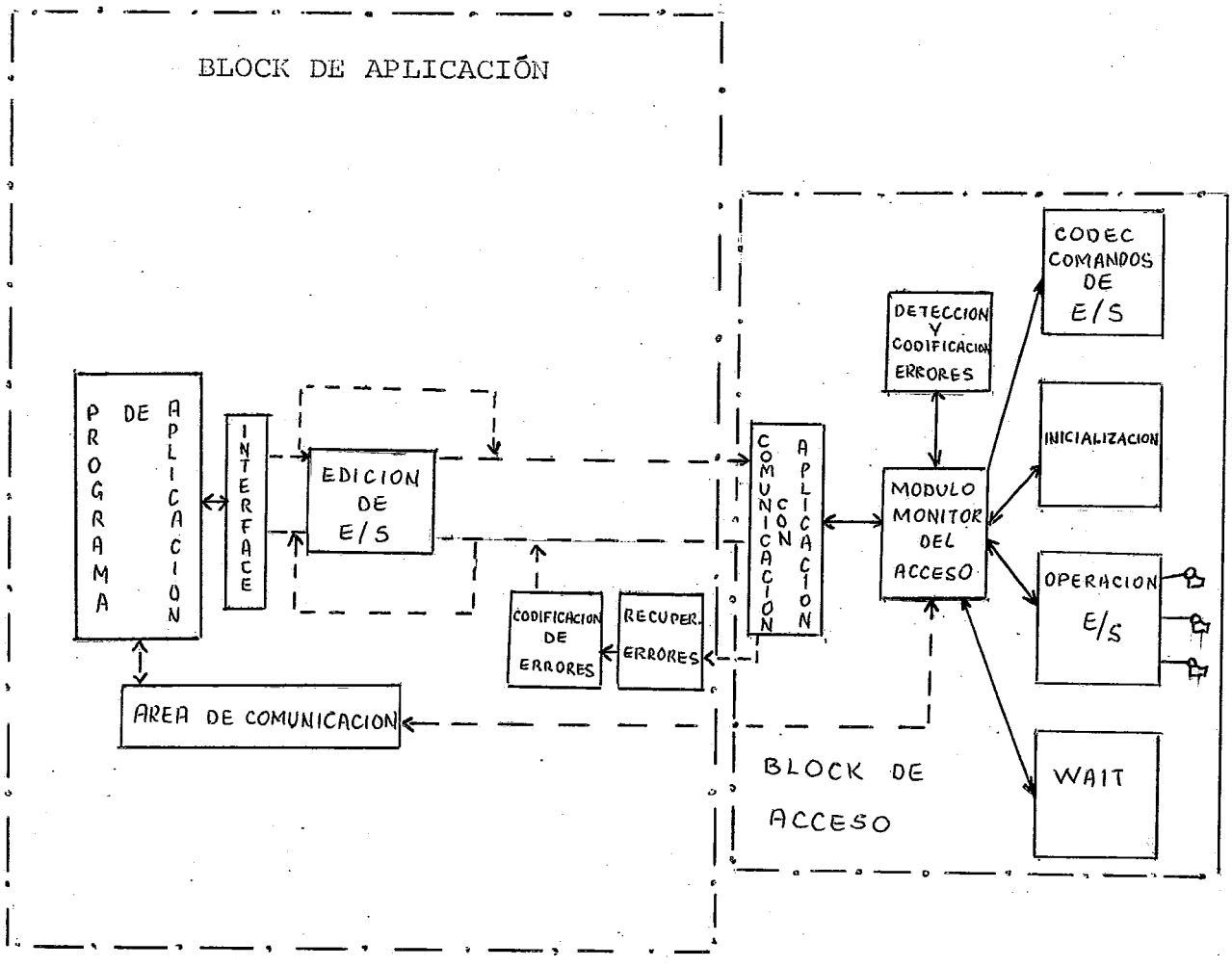


FIGURA 4.

Finalmente es importante hacer notar la necesidad de la existencia de un módulo que supervise el acceso para coordinar el funcionamiento coherente del resto de los módulos, y que aparece en la FIGURA 4 como MODULO MONITOR.

En el próximo capitulo completaremos la definición del acceso ya en la fase de implantación y aún cuando ésta será dirigida especialmente al sistema/360 consideramos que los criterios usados pueden generalizarse para otros sistemas tomando en cuenta, lógicamente, las características diferentes, tanto de hardware como de software.

CAPITULO 4

IMPLANTACION DEL SISTEMA

4.1 Observaciones.

Las consideraciones que haremos en este capítulo corresponderán a un sistema/360 trabajando en OS con la opción MFT y con un volúmen pequeño de memoria.

Esto es sin embargo muy fácilmente adaptable para la opción MVT e incluso (con mayor dificultad) para el sistema DOS (DISK OPERATION SYSTEM) valiendo para los mismos todas las consideraciones ya hechas.

A los efectos de la determinación del sistema, deberemos examinarlo desde dos puntos de vista diferentes y las conclusiones que surjam serán las que posteriormente permitirán crear el acceso final:

A - Análisis del acceso desde el punto de vista del usuario.

B - Análisis del acceso desde al punto de vista del subsistema procesador.

4.2 El acceso desde el punto de vista del usuario

Para el usuario el sistema deberá permitirle una gran generalidad de aplicaciones así como facilidades de programación.

Como ya fue expresado, la generalidad de aplicaciones resulta conflictiva con la facilidad de programación y en consecuencia adoptamos el criterio de crear una serie de opciones coherentes en sus criterios con la grán mayoría de las aplicaciones y que darán una evidente facilidad al programador evitando que este deba entrar en una serie de detalles del control de la ENTRADA/SALIDA, de los códigos internos de transmisión, de la detección y corrección de errores, etc. En los casos por supuesto minimos, en porcentaje de aplicaciones, en que tales opciones no se adaptan a las necesidades particulares, la programación se hará necesariamente difícil y engorrosa, aún cuando tendrá la ventaja de un menor volúmende memoria utilizado.

Deberemos en consecuencia determinar como es que el programador se comunicará con el método de acceso, por un lado, para solicitar la ejecución de las operaciones de Entrada/Salida y por el otro para definir por cuales de las opciones que el sistema pone a su disposición habrá de optar.

4.3 Comandos de ENTRADA/SALIDA

Veamos inicialmente cuales deberán ser los comandos de que deberá disponer el programador para procesar las diferentes operaciones de E/S.

A tales efectos seguimos el criterio establecido en el item 3 del paragrafo 2.7 de tentar mantener o generalizar los conceptos y/o comandos corrientes en las aplicaciones en "batch".

Debimos también considerar la necesidad de permitir e incluso incentivar al programador para procesar un máximo de operaciones en su propio programa, simultáneamente con el desarrollo de la Entrada/Salida, para lo cual debíamos poner a su disposición comandos que solamente den inicio a tales operaciones y otros que soliciten la finalización de la transferencia.

En consecuencia definimos los seguientes comandos:

INITP - Inicializa todos los terminales ligados al programa del usuario en lectura (TERMINALES 2741) y devuelve el control sin haber finalizado la operación.

INITPW- Igual que el anterior con la única diferencia de que devuelve el control solamente una vez que uno cualqueira de los terminales inicializados acaba la lectura.

READTP- Inicia la lectura de un mesaje(de una o más lineas de largo) desde un terminal particular y retorna inmediatamente el control, sin completar la transmisión.

GETP - Igual que el anterior con la única salvedad de que devuelve el control una vez que la lectura del mensaje de ese terminal, fue completada.

WRITP - Inicia la escritura de un mensaje en un cierto terminal, que debe ser indicado, y devuelve inmediatamente el control al programa del usuario.

PUTP - Igual que el anterior con la excepción de que devuelve el control cuando acabó la escritura.

WAITP- Tiene un doble significado según se indique o no un terminal en especial. En el primer caso devolverá el control solamente al finalizar la operación de E/S pendiente en el terminal indicado y en el último caso devolverá el control cuando el primeiro de todos los terminales con operaciones de E/S pedientes, la complete.

WAITPTY-Corresponde a dar prioridad absoluta a la finalización de la operación de Entrada/Salida en un terminal en particular a expensas del resto de los terminales.
Devuelve el control una vez que tal terminal acabe la operación de E/S.
La diferencia con los otros comandos de WAITP es que en aquellos al ser aplicados a un terminal particular o no lo que efectivamente se hace es establecer el punto de retorno del control y una vez que cierto terminal completa su operación de E/S, siempre el control es concedido a la instrucción inmediata siguiente al último WAITP que lo referenció.

(Ya sea éste general o particular para ese terminal). En el caso do esta operación de WAIT con prioridad aún cuando el resto de los terminales completen sus operaciones de E/S pendientes, el control solamente será devuelto al programa del usuario una vez que el terminal referenciado, por esta operación, haya finalizando su operación.

CLOSETP- Esta instrucción corresponde a dejar todos los terminales fuera del control del programa, en iguales condiciones que antes de iniciar el procesamiento. Si el sistema en este caso detecta que hay alguna operación de E/S pendiente, por solicitaciones anteriores del programa, debe marcar el error correspondiente.
Si se desea reiniciar las transmisiones desde yhacia los terminales será necesario recomenzar mediante los comandos de inicialización:
INITP o INITPW.

4.4 Criterios de elección de comandos

Nuestra intención al proyectar la serie de comandos presentados era como ya lo hemos señalado, por un lado el permitir operar los terminales con la mayor libertad admisible en las condiciones del "hardware" y por otro el uso por parte del programador de los mismos criterios y técnicas empleados en el procesa -

miento normal en"batch" permitiendo a lo sumo una generalización de tales criterios y procedimientos. En la grande mayoria de las aplicaciones de uso de terminales en forma conversacional , todos los terminales aunque puedan transmitir y/o recibir mensajes casi que simultaneamente, desde el punto de vista del operador del terminal, él está interactuando con el programa en forma aislada. En el caso de la mayoría de las aplicaciones en "batch" el programa interactúa desde el punto de vista del programador, a cada instante, con uno solo periférico, siendo este además el caso de todas las aplicaciones programadas en lenguajes de alto nivel.

En estos casos el procesamiento se efectúa en forma secuencial, es decir que solamente se procesa un nuevo registro cuando el anterior ya ha sido procesado. Esta caracteristica no puede existir en el caso de las aplicaciones de teleprocesamiento pues los terminales transmiten los mensajes asincrónicamente entre sí y en caso de ser atendidos uno a uno por un cierto orden de prioridades, todos estarían limitados por el operador de menor rendimento.

Por lo antes expuesto, estimamos, fundamentalmente el lograr que nuestro sistema simule el procesamiento en batch, es decir que, dentro de las limitaciones logicas,permita que en un grande número de las aplicaciones posibles el programador pueda

razonar suponiendo que va atendiendo secuencialmente a cada terminal, cuando en realidad, como veremos más adelante, esto no va a acontecer sino escepcionalmente.

Pero por otra parte, dado que no debíamos de ningún modo limitar la generalidad a menos que el costo de la misma, en términos de eficiencia global, fuese prohibitivo de acuerdo a los criterios que establecimos en el Capítulo 2, incluímos una instrucción que como WAITPTY implica un procesamiento concebido y dirigido hacia cada terminal en particular.

Además, la inclusión de este comando que entendemos, abre un amplio campo de aplicaciones, incomunes sí, pero no por ello menos válidas, no sobrecarga, ni al programador como resulta evidente, ni al acceso propiamente dicho tal cual veremos en el sistema que hemos implantado.

Otro aspecto que tomamos en cuenta, al definir nuestros comandos fue el aspecto mneumónico que aunque lógicamente no resulta básico, tiene una real importancia para el programador, así se puede observar que tentamos combinar, el nombre de la operación, con el sufijo TP que representa "teleprocesamiento".

En el caso de WAITPTY no respetamos esta regla pues estimamos más importante el diferenciarla de WAITP a través del sufijo PTY que significa "PRIORITY" y que, como podrán observar tiene un claro sentido mneumónico.

En lo que respecta al resto de los comandos usamos los pares READ, GET y WRITE, PUT, que en los diferentes accesos del sistema/360 tienen como diferencia marcante justamente el momento en que el control es devuelvo al programa (si antes o depués de completada la operación de E/S).

Respecto a CLOSETP consideramos que sobran los comentários, mientras que para INITP e INITPW debemos hacer dos acotaciones, la primera, que no usamos analógia con OPEN debido a que este último no implica ninguna operación de E/S mientras que aquellos INITP significaban una lectura inicial y eso podría llevar a configuración y la segunda, que el sufijo W, (WAIT) especifica la espera por la finalización de la operación para alguno de los terminales envueltos en el comando.

4.5 -Parámetros de Transferencia

Con escepción del comando CLOSETP el resto precisa de ser complementado, como mínimo, con la información que identifica el terminal de referencia y/o el mensaje propiamente dicho.

Pa evitar aumentar el procesamiento tanto por parte del programa de aplicación como del acceso y para facilitar el manipuleo de los mensajes en lo que se refiere a la comunicación entre ambos programas, estimamos conveniente acrecentar una

información: la longitud en número de caracteres (ya sean de control o de impresión) del mensaje a ser transmitido al terminal o recibido de este.

Este criterio, por otra parte, se adapta muy bien a la programación en lenguajes de alto nivel, pues si el mensaje fuese tratado como una cadena(STRING) de caracteres de longitud variable esa cifra correspondería en cada instante, al largo de la cadena. Corresponde la observación de que en toda instrucción que implica lectura, como ser INITP, INITPW, READTP y GETP,es necesario para el acceso recibir como información las seguientes direcciones de memoria en que: el programa de aplicación desea sea guardado el mensaje, sea indicado el terminal desde el cual fué obtenida la transmisíon, sea establecida la longitud del mensaje transmitido.

En aquellos casos en que acontezca algun hecho anormal que deba ser comunicado al programa principal podremos perfectamente codificarlo en alguna de las áreas citadas anteriormente.

4.6 El programa de aplicación y su correspondencia con el procesamiento real.

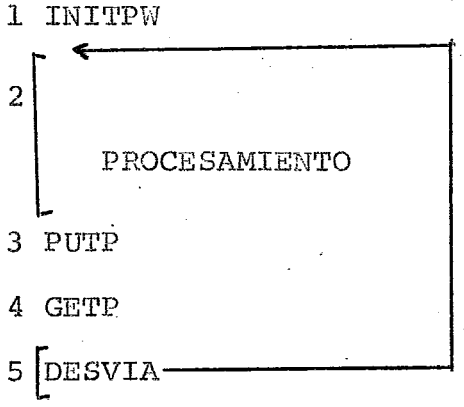
Como establecíamos en el parágrafo 4.4 tentamos a través de los comandos elegidos el permitir que, para un número grande

de aplicaciones, el programador pueda ver el procesamiento de los mensajes de cada terminal como sucediéndose sucuencial e independientemente.

A tales efectos, veremos un ejemplo demonstrativo de la diferencia que existe entre la secuencia lógica en la programación y la real en el procesamiento propiamente dichó.

Supongamos un programa que va a leer simultáneamente de tres terminales y de acuerdo a un cierto procesamiento de los mensajes, transmitirá respuesta y así sucesivamente.

El programa, en forma esquemática, corespondería, por ejemplo, a la combinación de los seguientes comandos:



La secuencia lógica corresponde a :

1. Lee de un terminal.
2. Procesa el mensaje.
3. Escribe en el terminal.

4. Lee del terminal
5. Retorna a 2.

La secuencia física podría corresponder a los siguientes eventos:

1. Lee de todos los terminales.
2. Recibe el mensaje del terminal 1 y lo procesa.
3. Escribe respuesta en el terminal 3.
4. Recibe el control luego de escribir en el terminal 1 y lee de éste.
2. Recibe el mensaje del terminal 2 y lo procesa.
3. Escribe respuesta en el terminal 2
5. Recibe el mensage del terminal 1 y desvía para 2.
2. Procesa el mensaje.

... etc.

Como podemos observar la secuencia real es totalmente impredecible pues toda vez que hay una operación de WAIT general (en el ejemplo ligadas a lectura y escritura en INITPW, GETP y PUTP) el siguiente comando a recibir el control dependerá de la operación pendiente que sea completada en primer lugar.

A pesar de lo señalado y desde el punto de vista del programador, todo acontece como si un sólo terminal estuviese interactuando con el programa.

4.7 Implantación de los comandos y rutina de Interface.

Para implantar los comandos anteriores se nos plantea la siguiente alternativa:

1 - Modificar los compiladores de los diversos lenguajes de alto nivel introduciendo esos comandos, padronizándolos dentro del total de comandos disponibles.

2 - Cada comando deberá ser codificado como un llamado a una subrutina que hará el interface con el método de acceso.

La alternativa 1, aunque realizable, presenta problemas de implantación sumamente dificiles. El modificar compiladores, dependiendo de la estructura de éstos, representa la doble dificultad de exigir un análisis minucioso del compilador y de efectivamente modificarlo sin afectar el resto de su estructura.

Por otra parte, dado que existe una variedad grande de compiladores de cada lenguaje, para hacer viable el uso del sistema, sería necesario el modificarlos todos, lo cual representa un volúmén de tareas sólamente compatibles con los objetivos de un fabricante que desee uniformizar su "SOFTWARE" para el uso de tal sistema de teleprocesamiento.

En nuestro caso la única posibilidad está representada por la alternativa 2, pero realizando la tal condificación

dentro de una rutina de interface que de todos modos resulta imprescindible como veremos, y en la que muy fácil y eficientemente se alcanzan los objetivos deseados.

Esta rutina deberá codificar los comandos que posteriormente serán decodificados por el módulo CODEC del Block de Acceso (Figura 4) para su posterior ejecución, será preciso también que los interprete, para así poder, de acuerdo a cada caso, guardar las direcciones de retorno del programa principal pues a ella le corresponderá devolver el control.

Otras tareas de la citada rutina serán: homogeneizar los formatos, tanto de transferencia como de datos, de los diferentes lenguajes de alto nivel, para llevarlos al formato - padrón establecido en el acceso y además el coordinar la interrelación de todos los núcleos transitorios del acceso por los cuales haya optado el programador.

La llamada Rutina de Interface estará integrada en consecuencia por lo que en la FIGURA 4 llamamos Núcleo de Interface, los diversos módulos opcionales y el Area de Comunicación.

Definiremos ahora con la mayor precisión posible tales módulos para finalmente conseguir generar la Rutina de Interface que en conjunto con el programa principal formará el Block de Aplicación.

4.7.1 - Módulo Editor de Mensajes de Entrada/Salida.

Este Módulo en realidad va a corresponder a dos módulos separados: un módulo para editar los mensajes de entrada al programa(Transmitidos por el terminal) y otro para la edición de los mensajes de salida (a ser transmitidos hacia el terminal).

El módulo Editor de entrada deberá tener a su vez las siguientes opciones:

- Detectar, en el mensaje, los caracteres de control de edición como ser "Backspace", Tabulator", etc. y tomar las acciones, correspondientes.
- Realizar las conversiones de códigos necesarias para que el programa principal pueda manipular fácilmente los caracteres del mensaje (de código de transmisión a EBCDIC por ejemplo).
- Depurar en el mensaje los caracteres de control de transmisión, como ser por ejemplo NL(NEW LINE), EOT(END OF TRANSMISSION) etc.
- Interpretar el significado de los caracteres de control y tomar las medidas correspondientes.

Por ejemplo, a los efectos de simplificar la tarea del operador del terminal definimos que:
1 caracter EOT significa que la línea ha sido transmitida con errores y que debe ser retransmitida.

1 caracter NL seguido de EOT significa que la línea está correcta y el mensaje incompleto, si no se ha especificado un caracter EOM (End of Messaje).

1 caracter EOM(definido por el programador) seguido de NL y EOT significa que la línea está correcta y el mensaje completo.

El editor deberá verificar estas opciones y tomar las iniciativas que hemos señalado.

4.7.2- Módulo de Recuperación y Calificación de errores.

Hemos fundido en uno sólo los dos módulos que con tales nombres aparecen en la figura 4 pues dadas sus correlaciones, deberán ser generados como una única opción.

Lo que en realidad sucede es que si la opción es de recuperar los errores, esto implica una calificación de los mismos como recuperables (de transmisión) e irrecuperables(de programa, como ser el tentar una operación de E/S, en un terminal, con otra aún pendiente).

En el caso de los recuperables se toman iniciativas predeterminadas y que en consecuencia pueden no adaptarse a todas las aplicaciones y en el caso de los irrecuperables existirían dos opciones, una de ellas es dar el control al programa con el

código correspondiente y la otra es finalizar sumariamente el procesamiento, con el mensaje del error correspondiente.

Si la opción es de no tentar recuperar los errores sino - dar-le el control al programa principal, para que éste tome la iniciativa que más se adapte a sus propiasnecesidades, no tiene sentido calificar los errores con un criterio que no será considerado por el programa de aplicación.

En definitiva, frente a la acción del núcleo básico que detectará y codificará los errores, el módulo de recuperación y calificación, en caso de existir, tomará iniciativas predefinidas para los casos recuperables y para los no recuperables el programador dispondrá de la opción de definir si recibirá o no el control, finalizando, en este último caso, el programa.

4.7.3 - Area de Comunicación

El Area de comunicación servirá para transferencia de informaciones entre el Block de Acceso y el Block de Aplicación. Como ya hemos indicado, los datos a transferir serán 3: la dirección del área que contiene el mensaje o dónde éste deberá ser guardado, la dirección de un área que contiene o contendrá la longitud en número de caracteres, del mensaje, y finalmente la dirección de otra área que contendrá la indentificación del terminal envuelto en la operación.

Los contenidos y estructura de estasáreas sufrirán modi ficaciones durante el procesamiento de la Rútina de Interface para adaptarlos a las exigencias tanto del programa principal como del acceso.

Así por ejemplo, un programa en lenguaje Fortran para poder manipular caracteres estos deberán, por lo menos, ocupar 1/2 palabra cada uno, mientras que el acceso manejará los caracteres compactos en 1 byte.

La Rutina de Interface deberá hacer las conversiones correspondientes para cada transferencia entre los programas.

Otro ejemplo, el complilador PL/1 tiene una estructura totalmente diferente a la de los otros lenguajes en lo que se refiere al formato de transferencia de cadenas de caracteres para subrutinas (usa un "Dop Vector" para dar información adicional sobre el "String" transferido).

Dado que padronizamos, para el acceso, la estructura - más general, correspondiente a los lenguajes FORTRAN, COBOL y a la macroinstrucción CALL de Assembler/360, cada vez que se produzca una transferencia, la Rútina de Interface habrá de reestructurar el formato de la misma.

4.7.4- Areas de Trabajo

La Rutina de Interface hará uso de una serie de áreas y tablas de códigos para el procesamiento de sus diversos módulos.

Acumuladas a éstas estarán también las áreas de "buffers" y de control de operaciones de Entrada/Salida correspondientes a cada uno de los terminales asignados al programa de aplicación (tal cual concluímos en el parágrafo 3.6).

La longitud de algunas de las áreas de control citadas, está fijada por las convenciones del sistema operacional (OS-360) y otras pueden ser fijadas por nosotros al determinar las informaciones que usará nuestro método de acceso, pero todas ellas serán fijas para el programador.

Lo que éste deberá determinar es la longitud de sus áreas de "buffers"(el numero será determinado por la cantidad de terminales) que dependerá de los mensajes máximos a ser transmitidos.

4.8 Generación de la Rutina de Interface.

Tenemos dentro del sistema/360, básicamente, dos alternativas.

1 - Disponer de todos los módulos factibles de ser integrados, acrecentar un módulo que supervise el procesamiento del resto

y montar todos los módulos, que se deseen usar, en conjunto con el programa de aplicación.

2 - Programar una macro-instrucción que tenga, como parámetro, todas las opciones posibles y que al ser expandida dé como resultado la Rutina de Interface, con todos los módulos formando parte de ella en forma integral.

La desventaja de la primer alternativa es que el número de módulos, que corresponden a todas las combinaciones de operaciones posibles, es alto y además que, la programación del núcleo que los supervise en cualquiera de esas combinaciones no resulta nada trivial, por el contrario, es muy probable que sea necesario disponer de un núcleo por configuración diferente.

La desventaja de la alternativa (2) es el tiempo para montar la rutina en macro-assembler que puede resultar porcentualmente importante dependiendo del consumido en la aplicación.

Sin embargo, en esta última alternativa, también se pueden conservar, en código objeto, las diversas combinaciones a medida que sean necesarias, sin presentar la necesidad de un núcleo especial de supervisión como en la primemera.

Concluímos, finalmente, en la conveniencia de genera la Rutina de Interface a través de una macro-instrucción con las diferentes opciones como operandos, presentando además, esta

última solución, la ventaja de facilitar el manejo de las citadas opciones por parte del programador.

4.9 El acceso desde el punto de vista del subsistema procesador

Como ya hemos señalado, partimos de la hipótesis de disponer de un volúmen pequeño de memoria para ser utilizado por el sistema de acceso sin afectar seriamente la continuidad cualitativa y cuantitativa del procesamiento tradicional en "batch".

Esto, en el caso de suponer el acceso, en su parte básica integrando el núcleo de rutinas residentes.

Si por el contrario, el tal módulo básico del acceso, tendrá caracter transitorio y pasará a ocupar la memoria disponible en la partición correspondiente al programa que lo llamó en primer lugar, no afectará al resto de las aplicaciones, en las restantes particiones, escepto en la medida que lo haría cualquier otra aplicación.

Sin embargo, efectaría seriamente al propio programa de aplicación de TP que lo acogería en su partición pues el uso de memoria del acceso básico más la Rutina Interface (ésta estaría de todos modos) dejaría un volúmen de memoria más limitado para el programa principal. Esto obligaría, en fase de operación, a cargar los programas de aplicación en forma controlada para conseguir que el primero disponga de volúmen sobrando en su partición para permitir la ubicación del Acceso Básico.

Nótese que las alternativas son siempre para un único núcleo de acceso, ya sea residente en el área del supervisor o transitorio en la partición del usuário, sin considerar la posibilidad de que cada programa cargue su propio acceso. Esto último resulta claramente ineficiente dado que limitaría innecesariamente, en términos de memoria, la amplitud de todas las aplicaciones, sin escepción.

4.9.1 El Sistema ECAM

Aunque, aparentemente, las opciones presentadas configuran todas las posibles, esto no es totalmente cierto.

En efecto, en el volúmen 37 de AFIPS(1970) se hace referencial sistema ECAM(Extended Comunications Access Method) el cual, valiéndose del sistema QTAM y futuramente también del sistema TCAM, tenta optimizar la eficiencia del acceso.

Para alcanzar tal objetivo ataca dos de las características más negativas de los citados sistemas: el consumo importante en volúmen de memoria y la falta de soporte para programas escritos en lenguajes de alto nivel.

En lo que se refiere a la primer característica, el sistema ECAM, que fué proyectado en la opción MVT (Multiprogramming with a Variable number of Tasks) del os/360, pero que podría también ser adaptado a la opción MFT con "Subtasking", aprovecha las facilidades de las estructurasarbóreas de "Tasks"

y "Subtasks" para minimizar las necesidades de memoria de un sistema con un cierto número, prefijado, de aplicaciones similares de teleprocesamiento.

ECAM consta de una task de máxima prioridad (15) llamada OIT (Operator Interface Task) la cual procesará las comunicaciones con el operador de la consola, la "Task" ECAM propiamente dicha, que será supervisora de todo el procesamiento y que tendrá prioridad 14, "Tasks"intermediarias que únicamente servirán para cargar las "Tasks" de aplicación (una por cada "Task" intermediaria), y finalmente estas últimas que tendrán prioridades que podrán variar de 1 a 13.

Las "Tasks" intermediarias fueron creadas con el único propósito de evitar que, al cargar el sistema las "Tasks" de aplicación y en caso de que no existan condiciones de memoria para tal carga, no se produzca un final anormal de la "Task" que tentó la carga y de todas sus "subtasks", y el proprio sistema lleguará a un fin anormal, en conjunto con todas las rutinas de aplicación.

Esta estructura permite que los programas de Teleprocesamiento, que no serán otra cosa que las "Tasks" de aplicación, siendo escritos en un mismo lenguaje de alto nivel(que puede ser cualquiera) hagan uso de las filas ("queues") creadas por

un mismo MCP (Message Control Program) de alguno de los sistemas QTAM ó TCAM.

Esta estructura permite que las "tasks" de aplicación puedan tener el atributo opcional de ser residentes o transitorias, siendo en el primer caso independientes de que efectivamente haya un flujo de mensajes a ser procesados por ellas, mientras que en el segundo serán cargadas solamente cuando existan tales mensajes y retiradas cuando otras con atributo transitorio deban entrar.

Las principales ventajas de este sistema son por una parte el permitir procesar un numero considerable de aplicaciones afines simultáneamente, compartiendo un volúmen de memoria mucho menor que el que correspondería a todos en conjunto y por otra, el disponer del control para las aplicaciones de TP (asignándolas prioridades altas) de forma tal que las aplicaciones de "batch" o las de TP de menor prioridad sólo reciban el control una vez que aquellas (que corresponderían a las que exigen menor tiempo de respuesta) no tengan mensajes para procesar.

Inconvenientes de ECAM:

Al compartir varias aplicaciones una misma región o partición, no estarán protegidas, una de la otra, por lo cual se hace necesario tomar providencias especiales para la

depuración. Por otra parte, y esto es válido en cualquier sistema de TP, la depuración es un factor que puede llegar a ser crítico pues en configuraciones en las cuales las terminales pueden ser remotos, tal procesamiento aparte de costoso y lento puede incluir problemas de accesibilidad (física e incluso lógica, por ejemplo por razones de seguridad,etc.).

Otro inconveniente surge del propio uso de los sistemas TCAM y QTAM por las razones ya expuestas en capítulos anteriores.

4.9.2 Observaciones.

Existe una diferencia fundamental de filosofía entre el sistema que hemos idealizado y los sistemas QTAM y TCAM, que consiste en que estos últimos a través del MCP tienen el control total sobre los terminales que le fueron asignados y de esa forma el programa de control recibe los mensajes de esos terminales, almacenándolos en filas (quenes), sin que sea necesario que algún programa en particular haya solicitado tales mensajes, mientras que en el sistema MACLAN toda vez que hay una operación de E/S, ésta corresponde a un comando del programa de aplicación que es quien recibe directamente los mensajes una vez que la transmisión ha sido completada.

Esta caracteristica es consecuencia directa del aspecto conversacional de MACLAN y de la generalidad de tipos de terminales a los que los sistemas QTAM y TCAM dan soporte.

La implicancia directa de lo expuesto es que, mientras en el sistema MACLAN habrá una relación exclusiva entre un grupo de terminales y un cierto programa de aplicación, en los sistemas QTAM y TCAM la relación será entre los mensajes y el programa de aplicación de tal forma que un mismo terminal puede estar transmitiendo, durante un cierto periodo de tiempo, mensajes a ser procesadas por varios programas de aplicación.

La citada característica de independencia entre el MCP y los programas de aplicación exige la existencia de filas y son éstas las que presentan la alternativa de estar en la memoria (sobrecargan la misma) o estar en un medio de almacenamiento periférico (por ejemplo disco) lo cual en una aplicación conversacional actúa negativamente sobre el tiempo de respuesta (esto especialmente en virtud del tiempo de acceso a la información).

Si en éste último caso tomamos en cuenta que en computadores de porte medio y pequeño, el número de canales selectores (para transferencia de datos a alta velocidad) es en general pequeño (2 en el modelo 40 del NCE) y dado que su uso por parte del sistema operacional es sumamente amplio, la existencia de

un sistema que exija un uso extensivo de tales canales deberá forzosamente afectar la performance total de una configuración que funcione en multiprogramación.

Dado que nuestro sistema está principalmente inclinado hacia configuraciones con esas caracteristicas, es decir con poca disponibilidad de memoria en computadores de porte menor, debemos evitar en lo posible un uso extenso de memoria y de filas en disco dado que ésto último afectaría tanto el tiempo de respuesta como la eficiencia global del subsistema procesador, como fué definido en el parágrafo 3.2.

Sin embargo resultaría importante el disponer de la posibilidad de que simultáneamente un grupo de terminales interactúen indiscriminadamente con varios programas de aplicación, y esto es posible de ser programado por la aplicación propriamente dicha, recibiendo los mensajes y distribuyéndolos para su procesamiento por varias rutinas, gerenciando el "overlay" de las mismas en la memória o por medio de "tasks" en forma similar al sistema ECAM.

En definitiva, en este caso, la rutina principal recibiría los mensajes, los interpretaría y los conduciría a aquellas rutinas apropiadas para su procesamiento. Algunas de tales rutinas podrían estar residentes y otras, menos prioritarias,(en virtud de recibir menor numero de mensajes), podrían compartir una misma area de memoria en "overlay".

Esta estructura es claramente más rígida que la del sistema ECAM pero también es aplicable a la opción MFT sin "Subtasting" cosa que no sucede con el citado sistema, siendo además que dicha opción es en mayor grado compatible con las condiciones de configuración que hemos considerado.

Consideramos importante de todos modos y, estimamos, queda abierto el camino para estructurar un esquema que aplique la misma filosofía del ECAM en cuanto a la correlación - con las diferentes aplicaciones mientras que por otro lado esté directamente ligado con el acceso propriamente dicho, de los terminales.

En cuanto a la implantación de tal sistema, presenta la facilidad, en nuestro caso, de poder ser programado en lenguaje de alto nivel, como por ejemplo PL/1, dado que éste permite el manipuleo de "tasks", cosa que no sucede con los otros lenguajes.

En la figura 5 mostramos esquemáticamente la estructura de tal sistema, que por otra parte, dentro del conjunto de módulos presentados, en la figura 4 ocuparía el lugar del programa de Aplicación.

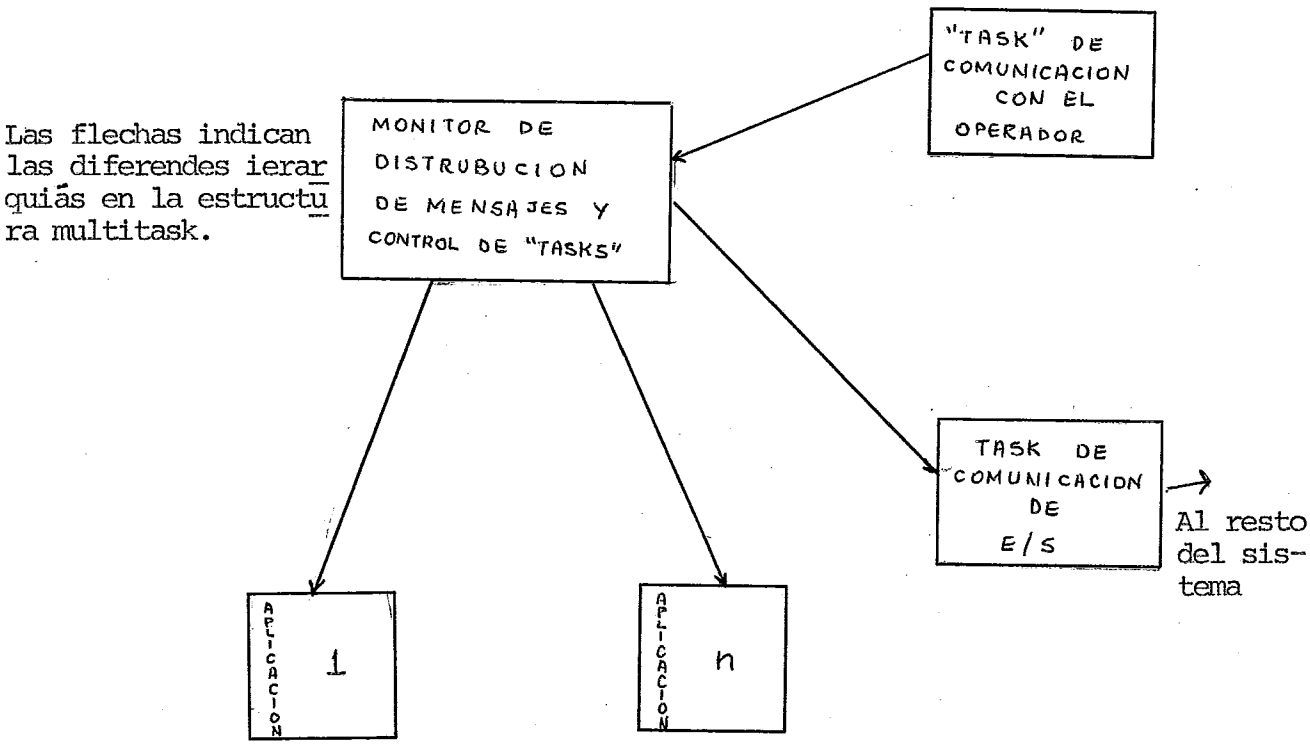


FIGURA 5.

En consecuencia, nuestro sistema tendrá una mayor generalidad ya que permitirá compartir un grupo de terminales por parte de un conjunto de programas de aplicación o por el contrario el uso exclusivo de los terminales por un único programa de aplicación pudiendo éste último optar o no por ejecutar por sí mismo una serie de tareas como ser: edición de texto, codificación de mensajes, control de líneas de transmisión, decisión ante errores, etc.

En lo que se refiere al caso de un grupo de programas de aplicación compartiendo cierto número de terminales y dada la posición que le hemos asignado al módulo que supervisará tal sistema en la estructura de la FIGURA 4 (la del Programa de Aplicación) todas las rutinas diferentes que habrán de procesar los mensajes estarán sujetas al mismo módulo opcional del acceso y deberán por lo tanto ser coherentes con las mismas opciones tomadas en el sistema.

Esta, que logicamente constituye una limitación importante, es consecuencia directa de las restricciones que nos hemos impuesto en lo que a uso de la memoria se refiere, pues en caso contrario cada rutina de aplicación debería cargar consigo su propio acceso opcional, con el acrésimo de volúmen correspondiente.

4.10 Módulos Básicos del Acceso

Estos Módulos serán aquellos comunes a todas las aplicaciones y en consecuencia deberá existir un sólo block de este tipo interactuando con los programas de TP.

Los módulos que deberán participar del block básico de acceso serán:

1) Módulo Monitor del Acceso.

2) Módulo de Inicialización.

3) Módulo de Operaciones de Entrada/Salida.

4) Módulo de Detección y Codificación de errores.

5) Módulo de Wait.

En esta estructura incluímos dentro del Módulo Monitor los módulos de comunicación con las aplicaciones y de Codec (Codificación-Decodificación) de los comandos de Entrada/Salida como aparecen en la Figura 4.

El funcionamiento será el seguiente:

Llegada una solicitación desde el interface de un cierto programa de aplicación, es recibida por el módulo que monitora - el proceso, éste, luego de codificar el comando requerido, transfiere el control hacia el módulo correspondiente que podrá ser de:

Inicialización, Operación de Entrada/Salida o Wait. Supongamos que sea el de E/S, éste módulo inicia la operación y devuelve el control al Módulo Monitor el cual, según cual sea la solicitación, devuelve el control al Programa de Aplicación o al Módulo de Wait (a espera de fin de la Operación de E/S).

Una vez que el Monitor recibe el control del Wait, da el control al Módulo de Detección y Codificación de errores y finalmente devuelve el control al Interface del programa de aplicación correspondiente.

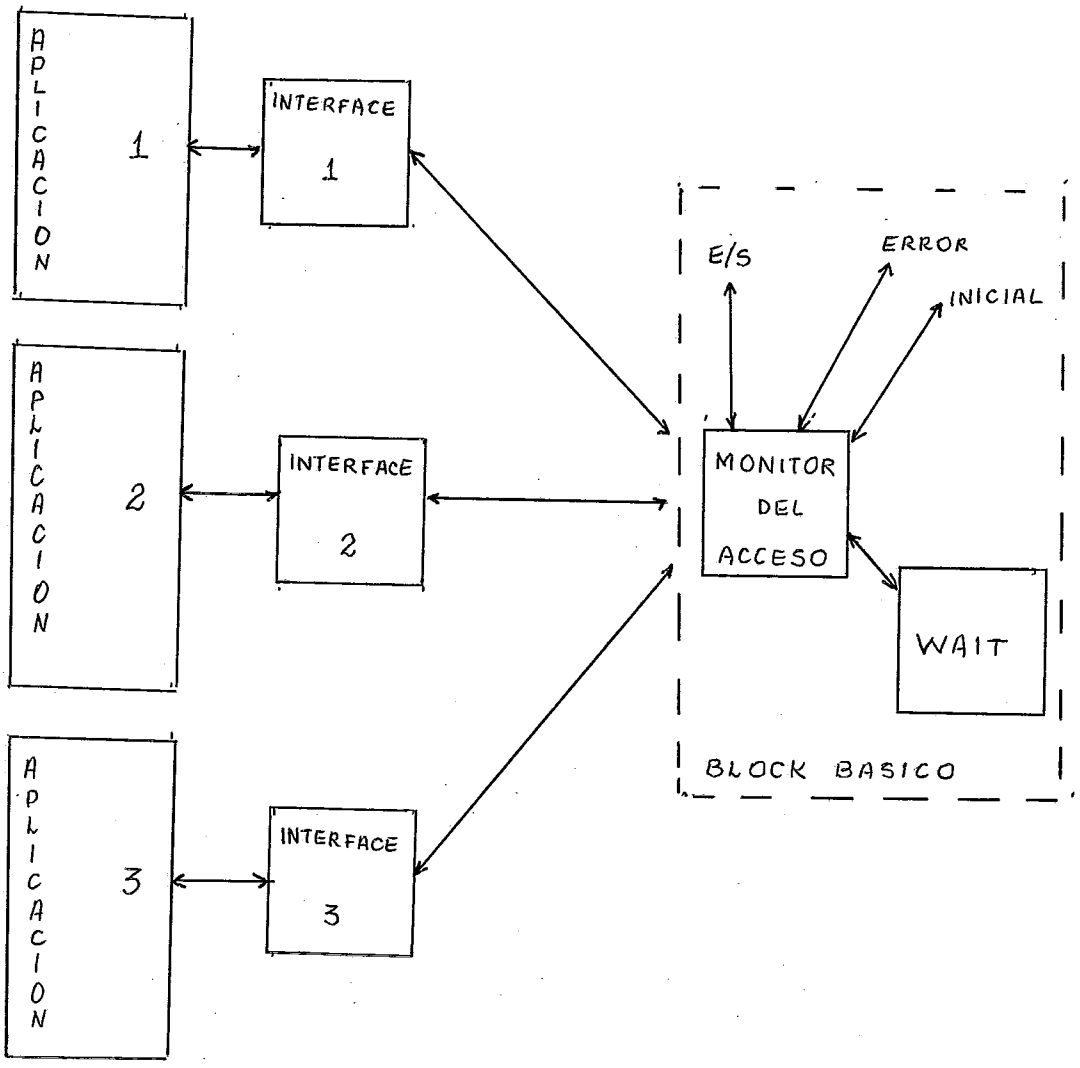


FIGURA 6.

Supongamos, como en la Figura 6, que tenemos 3 programas de aplicación, cada uno con su correspondiente Interface que, suponemos, incluyen todas las opciones.

Veamos un conjunto de eventos posibles:

1 - Aplicación 1 recibe el control.

2 - Solicita un GETP.

3 - El monitor del acceso luego de iniciar la lectura, con el módulo correspondiente, pasa a estado de espera (WAIT).

4 - Aplicación 2 recibe el control.

5 - Solicita un PUTP.

6 - El monitor del acceso luego de iniciar la escritura pasa a estado de "WAIT".

7 - Aplicación 3 recibe el control.

8 - Es interrumpida su ejecución por fin de la lectura del programa 1.

9 - Monitor del acceso recibe el control y lo devuelve a la Aplicación 1.

10- Solicita un PUTP.

11- Monitor del acceso luego de iniciar la escritura pasa a estado de "WAIT".

12 - Recibe el control la Aplicación 3 dónde había sido interrumpida.

13 - Solicita un GETP

14 - Monitor del acceso recibe el control y es interrumpido por fin de escritura de la aplicación 2.

15 - El fin del Wait da el control al Monitor en otro punto que el que estaba ejecutando y éste devuelve el control a la Aplicación 2.

16 - Aplicación 2 entra en estado de espera por razones independientes del procesamiento de TP.

17 - El Monitor recibe el control donde había sido interrompido inicia la operación de lectura solicitada por la Aplicación 3 y entra en estado de Wait.

Como se puede observar el block básico del acceso deberá ser reentrante y cada vez que reciba el control de algún programa de aplicación deberá reservar un área para guardar los registrado - res y la dirección de retorno, así como otras áreas de control.

También será necesario salvaguardar su propias áreas de trabajo y los registros que representan su propio estado para que pueda ceder el control a los módulos que lo integran en forma independiente para cada programa de aplicación (el proprio Monitor puede ser interrumpido como vimos en el evento 14).

Si deseamos hacer el block básico, independiente del número de programas de aplicación que pueden simultaneamente interac - tuar con él, deberemos asignarle una región ilimitada para hacer uso de las áreas citadas de acuerdo a sus necesidades. Como tal cosa resulta imposible o por lo menos claramente indeseable y dado que el fijarle un área extra, en forma arbitraria,

estará sujeto al riesgo doble de resultar insuficiente o excesivo, siendo esto último incompatible con las restricciones de memoria que nos habíamos impuesto, la alternativa más justa es, nuevamente, que la totalidad de estas áreas sean asignadas a la propia región de cada programa de aplicación.

De tal forma, cada programa, al solicitar los servicios del acceso, no sólo deberá darle área para salvar sus registradores sino también la necesaria para salvar los registradores del proprio acceso y para realizar las tareas que sea preciso.

Otro aspecto importante y que ya supusimos en la serie de eventos vistos es que los módulos básicos del acceso deben ser asignados con alta prioridad de tal modo que, si acaba un evento a la espera del cual estaba el Acceso, éste reciba inmediatamente el control de modo de poder procesarlo.

De igual modo, a los programas de aplicación de Teleprocesamiento se les debe asignar mayor prioridad que los "batch" de modo de recibir el control lo más rapidamente posible después de haber terminado las condiciones que los mantenían en espera.

En el sistema/360 la solución consiste en poner los módulos del acceso básico en estado de Supervisor y los programas de aplicación en las regiones de más alta prioridad.

4.10.1 Tópicos especiales en la implantación.

Como resulta claro de las consideraciones que hicimos en el Capitulo 1, las únicas bases del "Softaware" disponible que podíamos usar para implantar nuestro sistema, lo eran, el sistema BTAM y, a un nivel aún más cercano a la máquina, la macroinstrucción EXCP(Execute Channel Program).

El sistema BTAM pone a nuestra disposición una serie de macroinstrucciónes: de Entrada/Salida, de Control, de armando de tablas de códigos, de traducción de tales codigos a EBCDIC, etc. Y de rutinas de soporte de esas macroinstrucciónes, así como rutinas de recuperación de errores, etc. que resultan sumamente útiles y significan un ahorro importante en el tiempo y en el esfuerzo necesario para elaborar un sistema como el que deseamos.

Sin embargo, el sistema BTAM da soporte a una variedad, sumamente extensa, de terminales y circuitos diferentes y en consecuencia todas las rutinas citadas, que acompañan tal sistema, presentan una generalidad de propósitos, que los hacen relativamente ineficiente, con referencia a configuraciones particulares como la que estamos considerando.

Por otra parte, todas las áreas de control que acompañan el funcionamiento del sistema, como ser las DCB(Data Control Block), DECB(Data Event Control Block), etc., están sobredimensionadas para poder, de ese modo, servir las caracteristicas específicas de todos y cada uno de los terminales.

Si tomamos en cuenta que parte de éstas áreas corresponden al control de cada terminal individualmente, concluímos que, a medida aumenta el No. de terminales, aumentará el volumen de memoria usada inútilmente.

A través del uso de la macroinstrucción EXCP evitamos estos factores de ineficiencia, pues a través de ella, tenemos acceso a los propios programas de canal, y a los controles directos que estos implican.

Naturalmente la programación, en tal nivel primario, significa un costo relativamente elevado, en esfuerzo, depuración, etc.

Finalmente, dadas las condiciones restrictivas del uso de memoria que estamos considerando y si suponemos una carga importante de aplicaciones de teleprocesamiento, puede resultar, incluso económico, el programar los módulos del acceso básico a nivel de EXCP.

Cuando se programa en forma independiente de los métodos de acceso existentes, es necesario realizar tareas que corresponden al módulo que llamamos de inicialización y que consisten en interligar y crear el conjunto de áreas de control que por exigencias del OS./360 deben seguir un formato y reglas generales preestablecidas. (Tareas éstas que en otros accesos corresponden a la macroinstrucción Open).

Esta será en esencia la única parte de nuestro acceso que será dependiente de la opción del OS/360 por la cual optamos.

Felizmente las variaciones de reglas y formatos entre las diversas opciones (MTF,MVT,etc.) son de menor importancia y por tanto fácilmente adaptables).

Por otra parte, en virtud de deberse programar, también el acceso básico, en forma de macroinstrucción, con operandos que indiquen las características específicas de cada configuración, en que habrá de ser implantado, como ser: número de terminales, opción del OS/360 (solucionando así el problema de la dependencia del Módulo de Inicialización con respecto a tal opción), el acceso ya será generado en las condiciones más apropiadas a las circunstancias.

Así por ejemplo, en la opción MFT, el programa de Interface de la Aplicación, con el Módulo de Inicialización,deberán compartir y en consecuencia dividir la tarea de crear e interrelacionar áreas , como ser las ya existentes TIOT (Task Input/Output Table) que apunta para el UCB (Unit Control Block), tarea ésta realizada por el Initiator cuando el comienzo de la "job Step", o áreas que deben ser creadas e interligadas como las: DEB QUEUE (DATA EXTENT BLOCK) que constituyen un grupo organizado en fila y que apuntan para las ya vistas UCB y para las DCB(Data Control

Block) las que a su vez apuntan para el IOB (Input/Output Block) que contendrá varias informaciones básicas para el control de la operación de Entrada/Salida y la dirección del propio programa de canal (CCW- Channel Command Word).

Las citadas áreas y sus interrelaciones son las representada en la Figura 7.

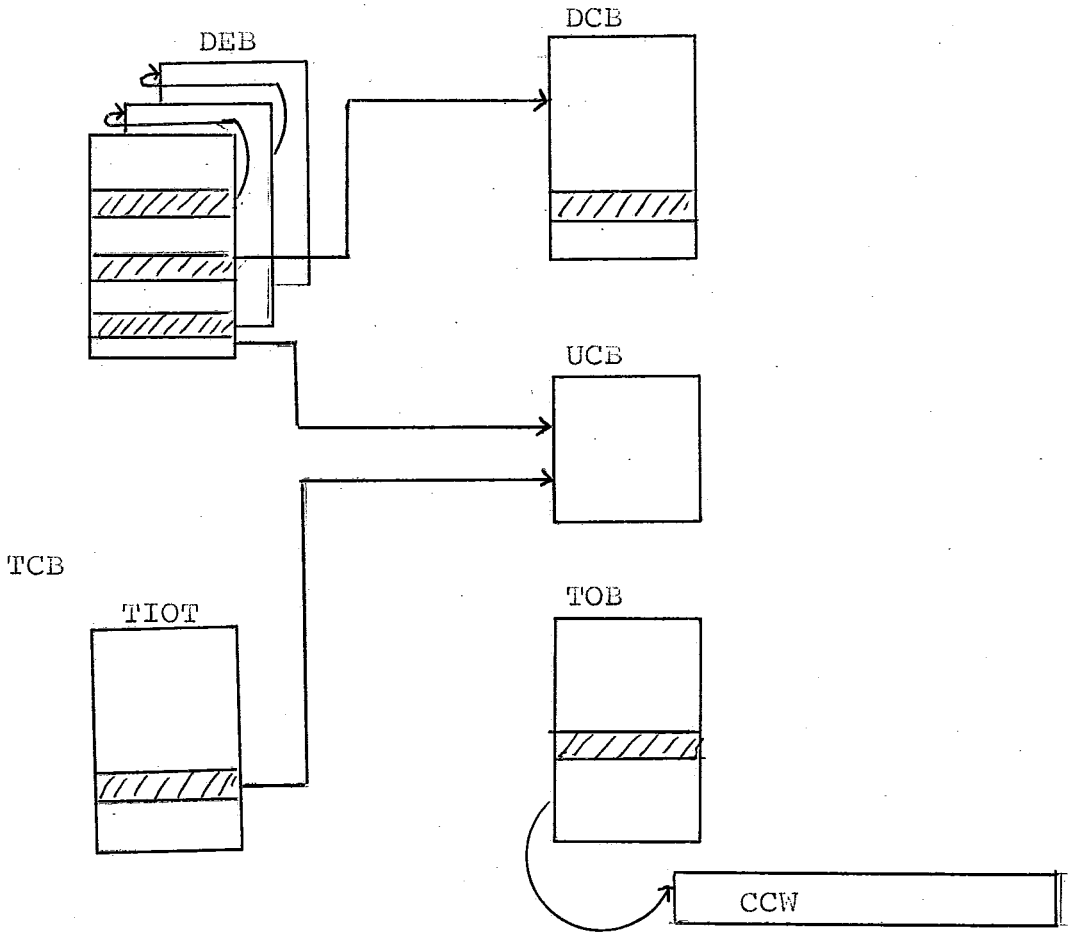


FIGURA 7.

Lógicamente las áreas deberán ser creadas por el programa de Interface en su propia región, así como éste deberá obtener la dirección de la TIOT a través de la macroinstrucción EXTRACT (pues esta macro devuelve la TIOT de la "Task" Activa) y hacer las ligaciones: TCB (Task Control Block) DEB, DEB DCB, y DEB UCB.

El Módulo de Inicialización del Acceso deberá siplemente ligar la DCB a la IOB y a éstas últimas entre sí.

Todas estas tareas corresponderán a la ejecución de parte de los comandos que llamamos INITP e INITPW.

En lo que se refiere a los programas de canal, que deberán ser implantados, éstos serán totalmente similares a aquellas que forman parte de las macroinstrucciones "READ INITIAL", "READ CONTINUE", y "WRITE CONTINUE" del sistema BTAM, para los terminales 2741 en circuito "nonswitched", acrecentados con los que serán de aplicación específica en los circuitos de tipo "Switched".

Cuando la configuración de teleprocesamiento es de éste último tipo, no sólo habrá variantes en los controles necesarios para las operaciones de Entrada y Salida, sino que, pueden existir diferencias en cuanto al total de estados diferentes, del equipamiento, posibles de ser detectados.

Así por ejemplo, en una configuración "Switched" es posibles de detectar si un terminal 2741 fué desligado, mientras que en una "nonswitched" tal cosa resulta imposible.

En consecuencia, para configuraciones de tipos diferentes ("Switched" o "nonswitched") no sólo será necesario un conjunto distinto de programas de canal, en el Módulo de Operaciones de Entrada/ Salida sino que también sufrirá modificaciones el Módulo de Detección y Codificación de errores.

Finalmente, la definición, de la configuración, en lo que se refiere a si es de tipo "switched","nonswitched" o incluso mixta, es decir con algunos terminales organizados en circuitos del primer tipo y el resto en circuitos del segundo tipo, deberá ser dada como parámetro de la macrodefinición del block básico del acceso y en el caso de ser mixta, el programa de interface deberá comunicar, en forma codificada, al acceso, de cual tipo se trata en cada caso.

4.11 Formatos de las macrodefiniciones del Acceso.

Como ya hemos visto el sistema de acceso se subdivid en una parte opcional, que será montada y ligada al programa de aplicación de acuerdo a las necesidades particulares del mismo y otra parte básica, común a todos los programas de aplicación, y dependiente exclusivamente de la configuración física de Tele-

procesamiento en la cual el Sistema es implantado.

Las dos subdivisiones citadas, que serán generadas mediante macrodefiniciones, como ya se ha establecido, serán montadas en forma totalmente diferente, pues mientras que la opcional lo será toda vez que un programa de aplicación sea implantado y bajo la exclusiva responsabilidad del programador, la básica deberá ser montada en conjunto con el resto del software del sistema y bajo control de los responsables por la implantación del propio Sistema Operacional.

Esperamos que todas las consideraciones presentadas durante los pasados cuatro capítulos hayan justificado plenamente el nombre que le hemos atribuído a este sistema: MACLAN que significa Método de Acceso Conversacional para Lenguajes de Alto Nivel.

Consecuentemente con todos los criterios utilizados, dimos el nombre de MACLAN a la propia macroinstrucción que deberá ser usada por el programador para definir el Acceso Opcional que será montado en conjunto con su programa y el de OSMACLAN a aquella que servirá para definir el Acceso básico, soporte de todos los programas de aplicación.

Los formatos de las citadas macroinstrucciones serán:

OSMACLAN NUMTER= número de terminales a utilizar por el Sistema,
CIRCUITO= { SWITCHED / NONSWITCHED / MIXTO }

MACLAN NUMTER = numero de terminales a utilizar por el programa[,

EDITOR = {SI / NO} [,] BACKSP = {SI / NO}] [, TABUL = {SI / NO}] [,

COLINIC = COLUMNA física del terminal en que se fija el comienzo de la línea] [

TABCOL = Lista de columnas tabuladas] ,

MAXMES = Tamaños máximos de los mensajes a ser procesados en E/S[,

CODIGOS = Códigos para decodificar mensajes a la entrada y a la salida] [,

OCUPLINE = {SI / NO}] [, BLOCK = {SI / NO}] [,

FINBLK = Representación hexadecimal en codigo de transmisión del caracter indicador de fin de block

DECOD = {SI / NO}] , SYSTER = número de terminales en uso por el sistema [,

TRANERR = {SI / NO} , DDNAME = Lista de los nombres dados a las tarjetas "DD" referentes a los terminales] ,

IDIOMA = {ASM / COB / FORT / PL/1}

Mientras los operandos de la macroinstrucción OSMACLAN son de interpretación inmediata, los de la MACLAN exigen explicaciones individuales:

EDITOR - El sistema presenta una edición opcional de formato fijo para los mensajes a ser transmitidos a los terminales Esta edición fija el número de caracteres por línea y organiza el texto, combinado con los caracteres de control necesarios para la transmisón. Si se opta por "NO" el programador deberá ocuparse de todos los controles citados.

BACLSP - Corresponde a editar o no, en los mensajes de entrada, tomando en cuenta el caracter de control "BACKSPACE".

TABUL - Corresponde a editar o no, en los mensajes de entrada, con tabulación, es decir tomando en cuenta el carácter de control "Tabulator".

TABCOL - Lista de columnas físicas en la línea del terminal que fueron tabuladas, esta opción está ligada a la anterior.

DECOD - Corresponde a traducir o no los mensajes de código de la transmisión al código de procesamiento por parte del programa de aplicación y viceversa.

CODIGOS- Lista de los dos códigos de traducción para mensajes de entrada y de salida. El sistema dispondrá de códigos - propios entre los cuales el programador deberá optar.

OCUPLINE - Si la línea de transmisión estuviera ocupada al tentar realizar una operación de E/S, el programador puede optar por recibir la comunicación del acontecimiento de tal evento y continuar con el control o que el sistema considere tal hecho como un error grave y provoque un fin anormal ("ABEND") del programa de aplicación.

BLOCK - Corresponde a la opción de que el Acceso pase el control al programa de aplicación toda vez que reciba una línea ("NO") o solamente después de recibir un mensaje (de una o varias líneas) cuyo fin será detectado por un carácter especial. ("SI")

TRANERR - Opción entre recibir la transferencia codificada de todos los eventos anormales acontecidos durante la transmisión (Si") o permitir que el sistema tente recuperarlos según reglas prefijadas.

4.12 - Ejemplo de Programación .

A los efectos de verificar si los criterios que fueron usados en la construcción del Acceso eran afines con las necesidades y exigencias de un sistema integrado, usuario-procesador, dentro de las condiciones de configuración presentadas, programamos un acceso con conceptos idénticos a los empleados en el

sistema MACLAN siendo las únicas diferencias, aquellas que resultan de las medidas tomadas para simplificar la programación.

A tales efectos la parte básica y la opcional en un único módulo que deberá ser definido y montado por el programador en la partición donde habrá de entrar su programa de aplicación.

A los efectos de todas las operaciones de Entrada/Salida, traducción de códigos, detección de errores, etc., hicimos uso de las facilidades otorgadas por el sistema BTAM.

Por otra parte, en la consideración de lo que en el sistema MACLAN serían las variables de la configuración física, adoptamos como fijas, en el ejemplo, las condiciones particulares del Sistema/360 Modelo 40 con 10 terminales 2741 en un circuito "Nonswitched" de que disponemos en el Núcleo de Computación Electrónica.

En consecuencia no tendremos las opciones planteadas en la macrodefinición OSMACLAN: NUMTER y CIRCUITO.

Todo el resto de las opciones presentadas en la macrodefinición MACLAN del parágrafo 4.11 existirán también en el sistema ejemplo.

En consecuencia, a los efectos del programador, el Sistema MACLAN y el Sistema ejemplo (a cuya macrodefinición dimos el nombre de TML 2741) serán absolutamente idénticos.

Las diferencias sustanciales estarán radicadas en: el área de memoria ocupada, relaciones entre núcleos del acceso, - performance del sistema, etc.

No tendrá sentido, por lo tanto, examinar las características relacionadas con la eficiencia, tanto del Acceso como del Sistema Global(por ejemplo tiempo de respuesta a los terminales, etc.), pues no se podrá concluir nada firme con referencia al sistema MACLAN, escepto el considerar esos resultados como cotas superiores con respecto a las reales.

En el apéndice A presentamos el listado de la macroinstrucción TML 2741 mediante el cual es posible verificar las aseveraciones anteriores, con referencia a criterios y técnicas usadas.

El ejemplo presentado resultó válido en el sentido de mostrar la viabilidad de dar soporte a una estructura física de teleprocesamiento que no tenia software adecuado en las condiciones de configuración y que los comandos seleccionados para comunicación con el acceso cumplieron sus propósitos de facilitar la programación de la mayoría de las aplicaciones y de permitir una gran generalidad de las mismas.

CAPITULO 5

CONCLUSIONES.

En nuestro concepto, el principal objetivo que nos llevó a estructurar un sistema de Acceso para condiciones particulares de configuración, tanto física como de programas, fue alcanzado.

En efecto, nos propusimos mostrar la necesidad y posibilidad de adaptar y crear las condiciones del "software", de soporte, compatibles con la configuración disponible y con el tipo de aplicaciones que se desea desenvolver y no, como acontece geralmente, que dados los programas de base, suministrados por el fabricante, que lógicamente se adaptan sólo parcialmente a las condiciones particulares del caso, de desarrollan aquellas aplicaciones compatibles con éstos.

Tal objetivo lo encaminamos hacia una realidad actual, el procesamiento haciendo uso de las facilidades tecnológicas de las telecomunicaciones se ha ido extendiendo exponencialmente dentro de los procedimientos de computación, abriendo opciones hasta ahora insospechadas en el campo de tales aplicaciones.

Sin embargo, en los sistemas existentes, aún cuando físicamente podrían ser, y efectivamente lo son, adaptables para interactuar con unidades de teleprocesamiento, el software para soportar el uso de esos equipos es sumamente limitado, e impide el desarrollo de las aplicaciones.

En definitiva, hemos planteado un modelo típico para tentar desarrollar nuestros criterios y con un campo extenso para definir conceptos y métodos, que sin ser nuevos, son diferentes de los tradicionales en el procesamiento en "batch" y sobre los cuales, dada su novedad, no existe una literatura coherente y unificada.

De ese modo debimos adoptar criterios sobre la estructura que deberían tener los programas de aplicación de teleprocesamiento y en base a tales criterios, crear el lenguaje de comunicación entre dichos programas y el Sistema de Acceso, estructurando así las bases de lo que sería el Método de Acceso.

Asimismo, al tentar estructurar e implantar el sistema, debimos considerar técnicas tradicionales y/o recientes, a partir de los conceptos especiales que se deben manipular en teleprocesamiento, hasta obtener la integración del nuevo núcleo dentro del block ya existente, sin crear incompatibilidades y tentando un máximo de eficiencia global.

Finalmente, el sistema ejemplo que fue realmente implantado, demostró claramente, pensamos, la factibilidad de realizar el sistema idealizado, con los conceptos y metodologías en él empleados, sin por ello pretender que son éstos los más eficientes aunque sí, que son apropiados, en las condiciones para las cuales fueron concebidos.

SECCION 2

SISTEMA GENERAL DE CONSISTENCIA DE DATOS

CAPITULO 6

INFORMACION GENERAL

6.1 Observaciones Iniciales

Ha sido una constante en el desarrollo moderno de las técnicas de procesamiento de datos la búsqueda de una mayor coherencia entre la propia entrada de tales datos y el procesamiento de los mismos.

Esta es una consecuencia directa tanto de la naturaleza - diferente de los equipos que hacen la lectura de los datos y de aquellos que los procesan, como del hecho de que estos últimos se han desarrollo mucho más, en términos de velocidad que los anteriores, durante los periodos recientes, aumentando aún más la brecha existente entre ellos.

Tal búsqueda llevó al desarrollo de técnicas como las de "Spooling", multiprogramación, etc., mediante las cuales fué posible atenúar los efectos de amortiguación, causados por la lentitud relativa de la entrada de datos, en el procesamiento general, Las citadas técnicas resuelven el aspecto performance del sistema global pero no necesariamente implican eficiencia de las aplicaciones individuales.

En efecto, las dificultades subsisten en aquellos sistemas particulares que para su procesamiento utilizan un volúmen importante de datos.

Por ejemplo, en el procesamiento "batch" convencional, con las caracteristicas indicadas, coexisten dos modos diferentes de preparación de los mismos, que resultan imprescindibles a los efectos de darles las condiciones para poder ser ulteriormente procesados, por los programas que integran el sistema en cuestión, con eficiencia y confiabilidad:

a) Fase "OFF LINE" de preparación y control inicial de datos.

b) Fase "ON LINE" de control y configuración final de datos.

La parte A consiste de una etapa de perforación de los datos en tarjetas y otra de verificación de que las mismas corresponden exactamente a lo que figura en los formularios de informaciónes.

Esto se hace, habitualmente, asignado a operadores diferentes, la realización de ambas tareas pues pueden existir tendencias a la repetición de ciertos errores por parte de algunas personas.

A pesar de las precauciones indicadas, igualmente se deslizan errores, en esta primeira fase, los cuales se pueden estimar en un volúmen del orden de 2 tarjetas equivocadas por cada 100 perforadas.

En razón de tales errores y de aquellos que pudieron haberse originado en la propia generación de los formularios iniciales es que se hace necesario darle una mayor complejidad a la segunda fase (B).

En caso de saberse con total certeza que las informaciones perforadas, en las tarjetas, se ajustan a los formatos exigidos, por los programas que habrán de procesar tales datos, bastaría con ejecutar, en la fase B un, simple traslado de las informaciones, de las tarjetas, a algún medio de acceso más rápido, como ser cinta magnética o disco.

Esto es en consideración de la necesidad de evitar que el procesamiento ulterior sea limitado por la baja velocidad de la lectora de tarjetas, dado que estamos suponiendo una interacción, de los programas, con volúmenes apreciables de datos.

Sin embargo, como hemos señalado, aún después de la primera fase subsisten errores, dentro del volúmen total de tarjetas, que clasificaremos en dos grandes grupos:

1) Errores de consistencia formal.

2) Errores de consistencia real.

En ambos grupos hacemos total abstracción del origen de tales errores, en cuanto nos interesan solamente las consecuencias operativas que los mismo tendrán sobre el sistema que los habrá de procesar, debiéndose además hacer notar que el

orden en que los clasificamos no es accidental sino indicador de prioridades entre ellos.

En efecto, el orden de detección y atendimiento de los errores es el representado, dándose prioridad a los de carácter - formal y dejándose para una segunda etapa los que llamamos"reales".

Errores de consistencia formal corresponden a aque - llos que no cumplen con la padronización preestablecida para los mismos (por ejemplo: formato de fechas, nombres, etc.).

Errores de consistencia real serían aquellos que aún cuando siguen fielmente los formatos exigidos no corresponden a la realidad que deberían representar (ejemplo: nombre equivocado, números equivocados, etc) siendo estos últimos factibles en ciertas circunstâncias de ser detectados por programa a través de subtotales u otros procesos.

La consecuencia operacional que tales tipos de errores provocan, es que mientras los primeros pueden ser totalmente detectados por programa, los últimos podrían exigir una operación de verificación visual tomando como referencia las propias fuentes de información.

A tales efectos la fase B deberá constar de aquellos programas necesarios: para controlar la consistencia formal y la parte real de los datos de entrada, para crear, en un medio de

almacenamiento de acceso más rapido, una imagen de los datos de entrada más o menos algunos caracteres de control (Al conjunto de datos resultantes de esta operación le daremos el nombre de Archivo Movimento de acuerdo a la referencia bibliografica (6) y finalmente para imprimir todos los datos leídos de manera de permitir la verificación visual antes indicada.

6.2 Preparación y control inicial de datos.

En la descripción de esta fase del trabajo quedaron implícitos una serie de inconvenientes y otros por explicitar que la misma implica.

Por una parte, la tarjeta perforada constituye un vehículo caro e incómodo de manipular, con el agravado de exigir un código de perforación que no permite una detección rápida de inconvenientes en el equipamento como también la imposibilidad de que el operador logre visualizar, simultáneamente con la perforación, los caracteres que son impresos. Esta última particularidad implica que el operador dispone de una percepción limitada de los errores que va cometiendo lo cual impide una corrección inmediata de la mayoría de ellos y aún en aquellos casos en que consigue percibirlos, la operación de corrección es engorrosa y lenta. En efecto, debe suspender la carga automática de

tarjetas, duplicar la parte correcta, continuar la perforación hasta completarla y finalmente retirar la tarjeta equivocada - para poder así dar a la operación la continuidad debida.

Otra característica que produce una baja en el rendimiento de perforación, en el aspecto velocidad, es que en general los diferentes campos en las tarjetas son una imagen casi perfecta, en tamaño, de los que serán implantados en el Archivo Movimiento para su posterior procesamiento. Esto implica que aún cuando la información no ocupe toda la capacidad del campo la misma será completada, sin perforación, respetando el tamaño real del campo al que pertence. Para realizar esta tarea, eficientemente, es necesario el uso de tabuladores que por otra parte, están a disposición del operador (las máquinas perforadoras deben ser programables para este tipo de tareas). El problema radica justamente en que en general un registro corresponde a varias tarjetas consecutivas, todas con campos de formatos diferentes y, en consecuencia, tabuladores diferentes para cada una. Dado que las tarjetas serán perforadas secuencialmente, la multiplicidad de tabuladores servirá para hacer más lenta la operación (pues el operador deberá controlar las columnas) y aumentará la probabilidad de errores.

Existe otra opción, que veremos en el próximo parágrafo, de disponer en las tarjetas, los campos con largo variable en función de las informaciones a ellos asignadas. Esta opción presentará una serie de ventajas e inconvenientes que será oportunamente consideradas.

La perforación es seguida por una etapa de verificación que consiste en simular, por parte del operador (este, como ya fué indicado, debe ser otro que el que procesó la primera parte), el perforar las mismas tarjetas que van pasando por la máquina verificadora y comparadas en esta operación. En caso de detectarse una diferencia deberá hacerse una comparación visual con el formulario de datos que permitirá verificar cual perforación fue la correcta.

Como puede observarse todas estas operaciones implican un manipuleo costoso y sumamente demorado para finalmente obtenerse como resultado un conjunto de datos que aún conservan un porcentaje de errores formales que en volumen absoluto resultan importantes, y que hacen necesaria una etapa posterior para depurarlos totalmente.

6.3 Configuración final de los datos

Las tarjetas perforadas resultantes de las dos etapas citadas son usadas como entrada en lo que llamamos la fase "B"

de preparación de datos.

En esta fase son leídas secuencialmente todas las tarjetas, controladas las consistencias de las mismas, individualmente de ser posible o en el grupo que corresponde al registro lógico, si es necesario, y con aquellas que pasan correctamente el control, es generado el Archivo Movimento, mientras que las equivocadas, son impresas con los mensajes correspondientes, para su correción y vuelta a la misma fase hasta su definitiva implantación en el sistema.

En consecuencia habrá una recirculación entre las fases A y B, disminuyendo cada vez el volúmen de datos que se deberán movilizar.

A pesar de ello tal recirculación implica costo altos en procesamiento, impresión de listas y lo que es más grave en la procura mediante los impresos, de aquellas tarjetas, dentro del conjunto total, que contienen los campos detectados como errados. Tales tarjetas deberán pasar el proceso de la fase A y luego retornar al control de consistencia iniciado.

Hemos dejado de considerar aquí la necesidad de listar las tarjetas, que pasan con suceso por el control de consistencia a los efectos, si es necesario, de realizar la verificación visual para detección de los errores de tipo "2" (parágrafo 6.1). Esto se debe a la posible necesidad de realizar tal operación cualquiera sea el sistema que habremos de considerar y por lo tanto

ésta no tendrá ninguna influencia para una posible alternativa de elección.

En el parágrafo 6.2 citamos la existencia de una opción diferente de aquella que establece que los campos de las tarjetas son, aproximadamente, una imagen de los que serán implantados en el Archivo Movimento, por la cual los campos serán reducidos, en las tarjetas solamente, a su minima expresión, es decir a contener pura y exclusivamente la información atribuída a tales campos.

En consecuencia, las tarjetas constiturían algo similar a la cinta de papel, es decir una línea sin límite, sin discontinuidades de especie alguna, y que necesariamente deberán tener caracteres indicadores de fin de campo y de fin de registro.

Esta técnica, que tiene como ventajas el menor uso que hará tanto de las tarjetas como de la lectora de las mismas, así como evitar el uso de tabuladores, control de columnas por parte del operador, etc., tiene como desventajas, que va a exigir una programación más sofisticada, como veremos más adelante, y lo que resulta más importante e incluso, en algunos casos, eliminatorio, la dificultad que significará la localización de los campos equivocados, dentro de todas aquellas tarjetas con formatos absolutamente libres, a partir de los impresos resultantes de la consistencia.

Esta última, asimismo, exigirá una programación más depurada previendo entre otras, la posibilidad de "OVERFLOW" de campos y/o incluso de registros, en base a la falta de alguno de los caracteres de control citados y por supuesto, un programa editor, que transforme ese formato compacto en el formato final de los datos definidos para el sistema.

Por otra parte esta opción debería ser puesta, incluso en la fase de operación, para tentar verificar si se confirma lo previsto inicialmente o si la continuidad absoluta que implica el proceso de perforación, en este caso, puede afectar la velocidad o el propio coeficiente de confiabilidad, normal en un operador.

6.4 Preparación de datos en un sistema de"Real-Time"

Todo el proceso de preparacion de datos, presentado en los parágrafos anteriores y que implica una carga sumamente importante, dentro del contexto total, en un sistema que deba interactuar con grandes volúmenes de datos, ha sufrido un proceso de mejoría progresiva con la aplicación de las técnicas, métodos y potencialidad de los sistemas de tiempo real.

Mediante estos procedimientos resulta posible alcanzar una preparación de datos finales en una única fase de procesamiento, por supuesto toda ella "on line".

Lo más importante radica en la continua interconexión que existe entre la consistencia propiamente dicha y el operador - (ahora de terminal) que permite a éste último hacer verificaciones rápidas, dado que dispone de todas las informaciones necesarias a su disposición, y en consecuencia proceder en mayor grado a correcciones acertadas.

Así se evitan todas las recirculaciones entre las fases(A) ("off Line") y (B)("On Line") que como vimos deben existir en el procesamiento en "batch", con el consiguiente ahorro de listados, procesamiento, y lo que es más importante, de búsqueda, a partir de la impresión de los campos equivocados, de las tarjetas que corresponden a los mismos.

6.4.1 - Descripción del Funcionamiento

El operador del terminal transmite los registros que son inmediatamente controlados en su consistencia y en función del resultado de esta operación se almacena el registro en el correspondiente Archivo Movimiento y se libera la continuación del seguiente registro por parte del operador o en caso de detectarse un error habría un mensaje inmediato al terminal para que tal hecho sea subsanado.

La secuencia de procesos indicada implica una serie de apciones tanto operativas como funcionales.

Las operativas corresponden, por ejemplo, a decisiones respecto al tipo y número de terminales a ser usados, archivo o archivos a asociarse al sistema, generación o no de estadísticas para control de rendimientos de operadores, compactación o no de los campos en la entrada, formato de los mensajes de errores para permitir máxima eficiencia en la recuperación de los mismos, etc.

Las opciones funcionales corresponden por ejemplo, al tipo de consistencias a ser procesadas, lenguajes a emplearse, etc.

La conclusión que surge inmediatamente es que mientras, en el procesamiento común, los errores de perforación implican - multiplicar la carga de procesamiento, los impresos de control el manipuleo engorroso de las tarjetas, etc. en "real-Time" implica una repetición mínima del procesamiento, sin necesidad de listados especiales correspondientes a los errores, a menos que los mismos sean confeccionados a efectos contables, para controlar el rendimiento de los operadores.

6.5 - Aspectos comunes en los Sistemas de Consistencia:

Fundamentalmente en cualquier sistema de consistencia habrá dos aspectos claramente diferenciados, uno corresponde al tipo y modo de contacto con los datos de base a ser controla - dos y el segundo, al conjunto de rutinas que disponiendo del

registro, realizan la serie de controles necesarios para tes - tar la consistencia del mismo.

El primero será el encargado de encaminar los datos a los mensajes de errores, hacia donde corresponda (diferente según se trate de procesamiento en "batch" o en "tiempo real") según cual sea el resultado de los controles, realizados por la o las, rutinas correspondientes. Tales rutinas, que aceptarán o rechazarán un cierto campo o un registro, en función de la compatibilidad entre los datos, los sistemas que habrán de procesarlos y la realidad exterior que los originó, serán esencialmente idénticas, ya sean las informaciones transmitidas por tarjetas o por terminales, interactuando en tiempo real. Decimos "esencialmente" y no "totalmente" idénticas en virtud de que como veremos, los formatos que tales datos iniciales habrán de tener puede variar, en función del tipo del medio físico empleado para su transmisión, lo cual habrá de afectar, aún cuando lo sea en forma leve, el manipuleo de la información - por parte de aquellas rutinas.

6.6. Influencia sobre el Procesador

Hasta el momento hemos presentado una serie de ventajes y prácticamente ninguna desvantaja, del procedimento de preparación de datos en tiempo real frente al convencional en - "batch".

Sin embargo existe un factor, que afecta especialmente la performance del sistema procesador, y, que puede resultar en definitiva básico en la opción por uno o otro procedimiento.

En efecto, mientras al perforar las tarjetas realizamos tareas "off-Line", es decir totalmente ajenas, a la operación del computador e independientes de éste, en la preparación por terminales el procedimiento es "On-Line" es decir permanentemente ligado al computador e interactuando alternadamente con éste.

Este proceso, que exigirá la residencia en la memoria de todas o gran parte de las rutinas, tanto de consistencia como de comunicación, durante todo el tiempo que llevará la preparación de los datos, habrá de afectar (y podría serlo seriamente) la performance del resto del sistema mereciendo, en consecuencia, este aspecto, un análisis detenido y minucioso.

Por otra parte, el tiempo de preparación de los datos, dependerá de la calidad y cantidad de equipos disponibles para tales tareas, y en definitiva también la inversión de capital en tales rubros, deberá tomarse, seriamente, en cuenta para alcanzar un sistema lo más apropiado posible a las circunstancias.

Como puede observarse el problema no es trivial y aunque existen muchos factores que favorecen los procedimientos en tiempo real pueden existir situaciones que, de facto, descarten la posibilidad de usar tal opción.

Sin embargo, no debemos en nuestro razonamiento limitar las palicaciones a las opciones de los sistemas existentes, sino estimular, a través de una definición detallada de los modos operativos más eficientes, un dimensionamiento correcto y actualizado de los nuevos sistemas de procesadores a ser implantados.

Esto no impide, igualmente, el tentar conciliar estas situaciones conflictivas, mediante sistema que se adaptan con relativa eficiencia a ambos tipos de operación.

CAPITULO 7

FILOSOFIA DEL SISTEMA DE CONSISTENCIA

7.1 Definiciones y objetivos.

El problema que nos hemos planteado es el de preparación de datos a los efectos de que puedan ser procesados por toda una serie de rutinas, dependientes de los objetivos que se desean alcanzar, en condiciones, máximas posibles, de confiabilidad.

Cualquiera sea la solución dada a ese problema, deberán existir tres tareas claramente diferenciadas, en el sistema resultante:

1) Preparación física de los datos.

2) Controles de consistencia de los mismos.

3) Comunicación entre los subsistemas anteriores.

La primera resulta, evidentemente, dependiente del medio físico usado para la preparación y dado que existe una interacción contínua entre las tareas 1 y 3, ésta última resultará también dependiente de las características intrínsecas de aquel medio.

Tal dependencia implica modos, tipos y formatos de operación afectando todos y cada uno de los sectores, en la organización, de la que tal sistema formará parte. Estos factores son los que en definitiva, dada su significación económica ,

habrán de servir para la definición de la configuración de equipamientos que integrarán el sistema y que, esencialmente, fijarán las estructuras de las tareas 1 y 3.

La tarea (2) será, por el contrario, "esencialmente" independiente (ver parágrafo 6.5), a los efectos estructurales, de la configuración física del sistema, dependiendo básicamente de las características particulares de cada aplicación.

Este aspecto ha tenido, históricamente, una influencia sumamente negativa sobre los sistemas que han sido estructurados para la satisfacción de las necesidades especiales que cada caso implica, pues incentivó la tendencia a la creación de sistemas diferentes para aplicaciones diferentes.

Tal característica fragmentaria de los diferentes sistemas de computación, que por otra parte ha sido un legado directo de los sistemas convencionales de procesamiento, significa una sobrecarga en los esfuerzos y, en consecuencia, en los costos, tanto de programación como de procesamiento (repetición en la depuración y listados de programas generalmente bastante complejos).

Nuestros objetivos consistirán en tentar organizar las diferentes opciones posibles de formatos y consistencias de datos, de forma tal que sea factible para el programador definir, a partir de los elementos básicos que deberemos crear, su propio subsistema de consistencia, apropiado a sus necesidades particulares.

Para que tal generalidad sea compatible, no sólo con la mayoría de aplicaciones posibles sino también, con las diferentes condiciones de configuración que, como ya dijimos, fijan los medios físicos de preparación de datos y de comunicación con los controles de los mismos, será necesario, también, analizar las diferentes opciones que existirán en este aspecto y crear los medios para conectar eficientemente el subsistema de consistencia con aquellos subsistemas que procesan las citadas tareas.

7.2 Condiciones del sistema.

En función de los objetivos planteados y considerando la necesidad de que el sistema actúe eficientemente, tanto en lo que se refiere a sus propias tareas como en sus relaciones con el resto de los sistemas que comparten el procesador , deberá cumplir al máximo, los siguientes requisitos:

A) Confiabilidad de las informaciones resultantes.

B) Satisfacer la demanda de todas las aplicaciones posibles.

C) Simplicidad para el programador.

D) Velocidad de Procesamiento y gestión.

E) Mínimo uso de memoria.

7.2.1 - Confiabilidad de Informaciones.

Esta condición resulta básica, para el sistema, dado que constituye la própia razón de su existencia. En consecuencia deberá tener prioridad máxima, frente al resto de los requisitos presentados, es decir que, aún cuando exista conflicto entre el mismo y alguna de las restantes condiciones, éste será resuelto indiscutiblemente asignándole absoluta predominancia a la confiabilidad de los datos en desmedro de los otros requisitos.

Para alcanzar este objetivo, de que los datos de salida del sistema sean esencialmente confiables para los programas que habrán de procesarlos, será necesario analizar exaustivamente los tipos de errores que pueden ser cometidos y las clases de controles que deberán ser establecidos, por programa, para evitar al máximo que algunos puedan deslizarse sin ser detectados y en consecuencia, corregidos.

Como fué visto en el parágrafo 6.1 los errores pueden clasificarse en dos grupos principales: errores de consistencia formal y errores de consistencia real.

Los primeros corresponden a los que no coinciden con la padronización establecida, tanto cuantitativa (numero de caracteres en un campo, etc.) como cualitativa (tal campo con caracteres numericos, tal otro alfabético y parte de un conjunto de

símbolos conocidos, etc.).

Los errores de consistencia real serían aquellos que, aun cuando superen todos los controles establecidos para el primer grupo, no representan la información real a que corresponden.

En este caso, la detección resulta mucho más dificil y costosa dado que generalmente, si el deslizamiento de esos errores implica alta gravedad, la única alternativa es la verificación visual de los datos con todos los inconvenientes que esta operación implica.

En algunos casos particulares existen soluciones especiales, que aún cuando no implican un 100% de confiabilidad, la probabilidad de errores es tan baja y el costo tanto menor, que la verificación visual, que según cual sea la aplicación puede resultar conveniente correr el riesgo.

Un ejemplo típico es el de campos numéricos representando cantidades, los que pueden ser controlados por lotes, a través de subtotales, y finalmente éstos en un control global, siendo la probabilidad de haber, dentro de un lote, errores que se equilibren, sumamente baja (esta es la única posibilidad de no detección) y además decreciente a medida que decrece el tamaño del lote.

7.2.2 - Generalidad para todas las aplicaciones.

Este requisito implica una labor imposible: definidas un conjunto de opciones poder afirmar que estas son "todas" las opciones.

En función de esta caracteristica, la única alternativa es la de analizar y procurar, extensamente, todas las opciones posibles de ser naturalmente halladas y permitir, dándole una estructura modular al sistema, que las opciones sean acrecentadas al mismo a medida que vayan apareciendo.

Finalmente, la consecuencia principal extraída de la necesidad de generalidad del sistema es que se le deberá dar una estructura "Modular".

Logicamente habrá conflicto entre la Multiplicidad de opciones y la simplicidad de programación, pues a medida que aumenta el numero de opciones, aún cuando aumenta la potencialidad del sistema, también aumentan las dificultades para que el programador las conozca y las coordine con eficacia.

Este conflicto tiene suma importancia, la que se puede detectar recordando que uno de los argumentos presentados para justificar la necesidad de crear un sistema como éste era justamente el esfuerzo de programación para implantar cada sistema en particular.

Sin embargo este conflicto puede ser fácilmente obviado dado que las diferentes opciones pueden ser clasificadas en grupo reducidos de tareas distintas.

Así el programador, cuando debe efectuar cierta operación de control, debe solamente consultar un subconjunto de opciones y elegir aquella que mejor se adapte a su caso particular.

El aspecto simplicidad también habrá de influir sobre la propia estructura del lenguaje, que deberá ser diseñado para generar las rutinas de consistencia y en especial sobre los comandos de ese lenguaje que como veremos tenderán a englobar en sí mismo varias opciones de control.

7.2.3 - Velocidad de procesamiento y gestión.

Lógicamente el programa de consistencia generado con los comandos del sistema deberá ser lo más veloz posible pero lo que, en definitiva, habrá de interesar es si el sistema completo de preparación de datos resulta eficiente tanto en lo que se refiere a su funcionamiento propio como en relación a los sistemas restantes.

A los efectos de atender estos objetivos debemos dividir el análisis según que la operación se efectúe en "real time" o en "batch" pues los aspectos de eficiencia varían según se trate de uno u otro tipo.

La parte de consistencia pura, común a ambas clases de operación, tendrá las consideraciones normales de eficiencia, comunes a cualquier procesamiento, en lo que puede y deberá existir diferencia es, sobre todo, en el subsistema de comunicación como veremos en el próximo capitulo.

Asimismo puede presentarse conflicto entre el aspecto velocidad y el uso de memoria estando ambos conceptos lo suficientemente interligados como para merecer consideraciones comunes.

7.3 - Lenguajes de Programación.

Hemos citado parcialmente la necesidad de crear comandos en un cierto lenguaje especial a los efectos de permitir la programación facilitada que se pretende alcanzar.

Como ya lo hemos presentado, los lenguajes existentes, ya sean de alto nivel o no, exigen esfuerzos considerables en la programación y depuración de tales sistemas aplicados a casos particulares.

Los citados comandos, específicos para consistencia, comunicación, etc, deberán permitir una programación más difícil y en consecuencia más confiable.

La elección del o los lenguajes a usarse, para decodificar esos comandos y tornarles ejecutables, será posible de ser efectuada disponiendo de las conclusiones que tomaremos respecto a

la estructura del sistema y las condiciones de implantación del mismo. (capítulos 8 y 9).

En estos capítulos corresponderá también disponer que tipo de traductor será apropiado para ser aplicado a un programa con comandos del sistema, para hacerlo ejecutable.

7.4 Conclusiones.

En resumen, para la elaboración del sistema general de consistencia de datos deberemos basarnos en los siguientes principios:

1) La etapa de consistencia pura será independiente del modo de operación del sistema ("real Time" o "batch").
2) La confiabilidad de los datos resultantes de la consistencia aplicada tendrá la máxima prioridad en la elaboración del sistema.
3) El sistema será generado por medio de comandos especiales siendo que, los orientados al control de consistencia, deberán corresponder a varias opciones afines de test.
4) El sistema deberá ser totalmente "modular" en el sentido de que deberá admitir fácilmente cualquier variación en el número de comandos factibles de ser usados.
5) La programación por medio de tales comandos deberá ser esencialmente simple.

6) El sistema y sus comandos deberán admitir modos operacionales diversos.

7) Los factores de eficiencia del sistema, memoria y velocidad, serán considerados en función del modo particular de operación para la correspondiente configuración.

8) Las condiciones físicas, donde el sistema será usado, fijarán el caracter operacional del mismo, no su estructura basica de programa.

CAPITULO 8

ESTRUCTURA DEL SISTEMA

8.1 Sumario

La observación del carácter fragmentario que, generalmente, tienen los procedimentos de entrada de datos en las diversas estructuras de procesamiento de los mismos fué la que nos indujo a tentar armar un esquema que, aprovechando todos los aspectos comunes en tal tipo de operación, permita encuadrar todas las aplicaciones bajo un único denominador.

Este denominador común, al cual asignamos el nombre, en cierta forma pomposo (pero justificado en función de sus metas y esperamos que también lo sea por sus logros), de Sistema General de Consistencia de Datos, tenta no sólo eliminar las deficiencias inherentes a la citada fragmentación (ediciones repetidas de programas, para ser depuradas, con la carga consecuente sobre el programador y sobre el procesador, etc) sino también, a través del análisis detallado de todas las opciones posibles, conocidas, de control de validez de datos e implantadas en el sistema, dar al programador una gran potencialidad para poder obtener un máximo posible de confiabilidad de sus datos.

Pero la generalidad que deseamos dar e ese sistema no se reduce solamente servir para cualquier clase de datos sino también, permitir que sea aplicado en condiciones operacionales diferentes, especificamente, tanto en "batch" como en "real time".

En consecuencia, a través de este sistema general, todos los datos primarios son factibles de ser procesados y pueden ser leídos de cualquier aparato convencional de entrada e incluso de terminales remotos (pudiéndose, en este último caso, operar en tiempo real).

De acuerdo a los principios establecidos en el parágrafo 7.4 el prôpio Archivo Movimiento resultante del sistema está comprometido en la eficiencia del mismo dado que sus "buffers" afectarán su volúmen de memoria.

En la operación "batch" no hay dudas que el Archivo Movimiento deberá ser único, pero en la operación en tiempo real, en la que lógicamente varios terminales transmitirán simultáneamente informaciones al sistema, podría plantearse la posibilidad de procesar, a continuación de los controles de consistencia, una etapa de verificación totalmente análoga a la que se realiza en la operación "off line". Em ese caso se deberá manterner una relación entre una cierta secuencia de formularios y un archivo, que lógicamente será, del punto de vista práctico, una relación

entre un archivo y la identificación del terminal que transmitió esos datos.

Aún en este caso se podría generar un único archivo de salida de la etapa de consistencia, manteniéndose un indicador del terminal, transmisor del mensaje, que generó el registro, para en una etapa posterior proceder a la división del archivo unitario en una serie de archivos que permitirían procesar la verificación deseada.

De allí, todos los datos de entrada serían reunificados en un único archivo magnético el que sería usado para actualizar un archivo de Base que es el que en definitiva sería operado por los sistemas de procesamiento de esos datos.

Para lograr estructurar el sistema general debemos identificar cada área que envuelve el problema y si dejamos para considerar más adelante los aspectos de formación de los datos y comunicación con los diversos aparatos de transmisión de informaciones, queda por analizar los puntos más importantes del control de validez de datos, que son: Estructura de los datos de entrada, detección de errores y mensajes consecuentes y almacenamiento de la información válida.

8.2 Estructura de los datos de entrada.

Un procedimiento sumamente extendido y que ya ha mostrado su eficacia y que esencialmente puede ser aplicado tanto para entrada de datos por tarjetas como por terminal, en tiempo real, para transferencia de informaciones a los controles de validez es que dichas informaciones estén organizadas en "batches" de registros afines.

En esta estructura, cada "batch" debería estar precedido por un registro de encabezamiento el cual tendrá, aun cuando podrá tener mas dependiendo de la aplicación, esencialmente tres funciones:

1) Identificación del tipo de datos que serán leídas en el "batch" que lo sigue.
2) Todo tipo de información que figurará repetidamente en todos los registros del "batch" lográndose de ese modo transmitir registros más cortos y menos sujetos a errores.
3) Vehículo de informaciones de control como ser cuenta de registros, subtotales acumulados en el batch, etc.

Mientras las funciones 2 y 3 resultan absolutamente, transparentes, la función 1, por el contrario, exige un análisis especial y detenido, pues en esencia, ella implica el núcleo central de todos los controles de validez.

En efecto, al identificar el tipo de datos que se deberán controlar se están implicitamente considerando todas las opciones posibles de que el sistema deberá disponer para poder así cumplir eficazmente sus cometidos.

8.3 Opciones en los controles de validez de las informaciones de entrada.

Existen dos alternativas básicas:

1) Las informaciones son auto-suficiente.
2) Las informaciones están relacionadas (y limitadas en función de estas relaciones).

En el primer caso basta con examinar las informaciones elementales, que integran el conjunto general de datos (usaremos el nombre de "archivo" para representar este conjunto), en forma independiente unas de las otras, mientras que en el segundo caso deberemos también combinar todas las informaciones relacionadas a los efectos de controlar además la validez en ése aspecto.

Las relaciones pueden existir entre informaciones de archivos diferentes, de registros (unidades lógicas de información en cada archivo) en un mismo archivo y finalmente de campos (unidades básicas de información de los registros) en un mismo registro.

8.3.1 - Tipos de controles de validez.

a) Control de caracteres

Este control se hace, en general, referenciando un campo en el que los caracteres pueden ser subconjunto de un grupo de ellos como ser alfabéticos, numéricos, especiales, etc., o algunos caracteres específicos como ser blanco, A. ∅, etc.

Resulta importante mostrar que eventualmente el subconjunto podría ser cualquiera y no predefinido como secede con el alfabético, numérico y especial por lo cual será necesario que el programador disponga, en el sistema, del potencial suficiente para poder, por sí propio, definir tal subconjunto.

Otro aspecto a considerar es el test de campo numérico en el cual puede aparecer el signo como carácter a los efectos de impresión o simplemente incluído en el propia representación del número como lo que sucede por ejemplo en el formato zonado del sistema/360.

b) Control de relación con otro/s campo/s.

El campo que está siendo testado forma parte de una cierta relación que puede ser tanto lógica como aritmética y por supuesto mezcla de ambas.

En tal relación pueden también ser referidos campos constantes (por ejemplo números o vectores de valores conocidos, fija

dos a priori o residentes en tablas, etc.) o campos variables que deberán forzosamente formar parte del conjunto de informaciones que están siendo controladas.

Los operadores que indican las relaciones pueden ser todos los conocidos, es decir:

Aritméticos:

Relacionales:

Logicos:

En consecuencia podremos tener relaciones entre:

b 1) El campo base (que está siendo controlado) con respecto a un campo constante dado.

b 2) El campo base con respecto a un vector de campos constantes dados. Aquí podrían aparecer relaciones que podrían ser definidas especialmente, como ser:
El campo base igual a uno de los campos del vector, el campo base menor que todos los del vector, etc.

b 3) El campo base con respecto a otro u otros campos del mismo registro dentro del archivo de entrada.

b 4) El campo base con respecto a otro u otros campos de registro diferentes en el mismo archivo de entrada.

b 5) El campo base con respecto a otro u otros campos de registros (iguales o diferentes) en archivos diferentes.

Veremos que estas relaciones presentan una gran generalidad y que el dar potencialidad al sistema para soportar tal amplitud dificultará en gran forma la programación a pesar de que grande parte de las aplicaciones usarán apenas una parte mínúscula de las opciones de relación presentadas. Este aspecto justifica un análisis más detallado, de su influencia sobre el sistema, el cual será expuesto en el capítulo de implantación - (capitulo 9).

A continuación mencionaremos algunos de los controles más comunes y que eventualmente podrían merecer atención especial, en la estructuración del sistema, por no estar específicamente incluídos en los vistos.

Control de Secuencia. Eventualmente los registros de entrada de informaciones podrían tener que estar en una cierta secuencia para permitir su procesamiento posterior. Tales controles de secuencia puede ser también a nivel de campos y en el caso de registros podrían ser totales o parciales, es decir, secuencia en todos los registros del archivo de entrada o secuencias entre grupos limitados de registros (lotes).

Control de Razonabilidad. Este control también podría se llamado de Control de Tendencia.

Esencialmente consiste en efectuar comparaciones entre

una cierta información y otras informaciones que deberían tener una cierta afinidad con aquella.

Tales informaciones de comparación podrían ser resultado del propio conjunto de datos entrados, de valores probables - establecidos por el programador o de procesos más sofisticados como ser estadísticas de valores y tendencias historicas, etc.

Como se puede observar, del punto de vista del control propiamente dicho, este test puede, perfectamente, ser clasificado en el tipo b antes citado.

El interés que presenta se refiere, especialmente, a la clase de acción quese deberá tomar en función de sus resultados.

Como lo pretende expresar el nombre que le hemos asignado (Razonabilidad o Tendencia) un resultado negativo no representa necesariamente un error y en consecuencia un rechazo, sino - simplemente un toque de atención para que la etapa siguiente de conferencia visual verifique si la aparente incongruencia refleja un error o una simple escepción en la tendencia lógica.

En consecuencia, la medida a ser tomada será de aceptar el o los registros en tales condiciones dejando una constancia de la ocurrencia del fenómeno descripto.

Control por dígito verificador. Una técnica sumamente usada para detección de errores en informaciones con representación numérica es la de acrecentar a la propia información uno o más

dígitos que cumplan una cierta relación con respecto a aquellos que serán propiamente procesados.

Estas relaciones siguen una serie de técnicas basadas en la capacidad del codigo para detectar la mayor parte de los errores posibles.

Aún cuando lo ideal sería usar un sistema que detecte la mayor cantidad de errores esto no es siempre posible por motivos que pueden ser de orden económica y/o técnica.

En consecuencia, dadas las limitaciones establecidas, la elección del sistema a ser empleado, entre el conjunto de los posibles, será el resultado de analizar por una parte la distribución de frecuencias de errores de ciertos tipos en las condiciones particulares de operación y de información (variará según las características físicas de formación del dato numérico, del formato y longitud del propio número, etc.) y del poder de detección del sistema de dígito verificador para los tipos de errores que presenten mayores frecuencias.

Una vez elegido el sistema, el control corresponderá al tipo b, y las obligaciones que esto implica para el sistema de consistencia se reducen a dar al usuario la posibilidad de optar por un conjunto variado de métodos de detección.

Controles por Grupo. En éstos casos puede haber controles efectivos o simplemente descriptivos, dados como informes pasivos del procesamiento de la consistencia.

Este último caso corresponde a informar, por ejemplo, el número total de registros controlados en cada grupo, cuántos fueron aceptados y cuántos rechazados y se podría informar también cuántos de ellos presentan una cierta característica especial, - etc.

Como controles efectivos tendríamos el de secuencia ya citado, el de comparación con datos de subtotales previamente conocidos, etc.

Este último control junto con el de tendencia y otros similares corresponden a una tentativa de detectar aquellos registros que, aún cuando su formato sea absolutamente aceptable, no representan la información que deberían.

Estos controles asumen una gran importancia en todo sistema de preparación de datos pues evitan un procedimiento que, como el de verificación visual, resulta enormemente costoso para el sistema.

El vehículo para transmitir las informaciones de control necesarias para el procesamiento de estos testes será el registro de encabezamiento citado en el parágrafo 8.2.

8.4 - Detección de errores y mensajes resultantes.

Lógicamente, el valor de un sistema de preparación de datos depende fundamentalmente de la extensión en la cual puede detectar los errores factibles de ser cometidos.

Sin embargo, no resulta menos trascendente que la potencialidad de detección, para el sistema, la forma y contenido de los informes de errores.

Teniendo en cuenta que esos informes (mensajes, en tiempo real) serán usados para la corrección de los errores que han sido detectados, y que la eficiencia del sistema dependerá, en alto grado, de la velocidad y acierto con que esta tarea será realizada, su valor depende de su claridad y nivel de detalle.

Aún cuando lo expuesto es común tanto a la operación "batch" como "real time", el hecho de que en la primera la corrección es posterior al procesamiento de los controles de todos los registros y en la segunda es consecutiva a cada control interactuando en forma conversacional con el operador, habrá matices diferentes, en ese aspecto, en ambas operaciones que justifican un análisis por separado.

Mientras en la operación "batch" todo se reduce a listar todos los registros que intervienen en el proceso con indicaciones bien visibles de la ocurrencia de errores y con explicaciones extensas de cuales fueron esos errores, así como separación

por grupos, con los indicadores e informaciones correspondientes a cada grupo, en la operación en tiempo real se deberá separar - una parte de mensajes y otra de informes.

Los mensajes deberán solicitar la retransmisión del registro (o campo) en función de haberse detectado la presencia de un error. Resulta inconveniente dar explicaciones sobre el punto donde tal error fue localizado y las características que lo hacen incompatible pues se debe partir de la base de que el operador no tiene porqué tener formación e informaciones suficientes para tomar decisiones que le permitan por sí mismo cambiar la estructura y formato de los datos. Otro argumento que reafirma este criterio es que la necesidad de observar e interpretar mensajes tendrá efectos de amortiguación sobre su velocidad de transmisión.

Esto se refiere a la entrada de datos por terminal, en operación en tiempo real, pero análoga a "batch" en el sentido de - transmitirse un conjunto grande de informaciones a la vez.

Sin embargo, existe la posibilidad (muy importante), en tiempo real, de transmitir la información a medida que se va produciendo siendo por lo tanto, lógicamente, el operador el que va a dar a los datos, los formatos necesarios para la transmisión. En ese caso la detección del error deberá ser seguida de un mensaje detallando sus caracteristicas, de forma de permitir al operador corregir su equivocación.

Los listados e informes en la operación en tiempo real se limitarán a los resultados de los controles, por grupo y totales y de los controles de tendencia.

8.5 Almacenamiento de la información válida

Una vez que los datos fueron considerados como válidos deben ser almacenados para su posterior procesamiento.

Tal procesamiento podría estar precedido por una etapa de edición en caso que los mensajes tuviesen caracteres de control correspondientes por ejemplo a una compactación o haber sido transmitidos en registros variables, etc.

El sistema general no podrá prever todas las infinitas variantes que podrían presentar tales formatos, en función de lo cual, va a presentar, estructuralmente, dos alternativas:

1) El usuario deberá definir el archivo sequencial donde quiere que sean gravados los registros válidos, imagen perfecta de las informaciones transmitidas (le daría - mos el nombre de Archivo Imagen).
2) El usuario deberá dar una rutina por el concebida y programada, bajo su total responsabilidad, que procesará la edición, actualizará o generará su Archivo de Base, etc. y luego devolverá el control al sistema.

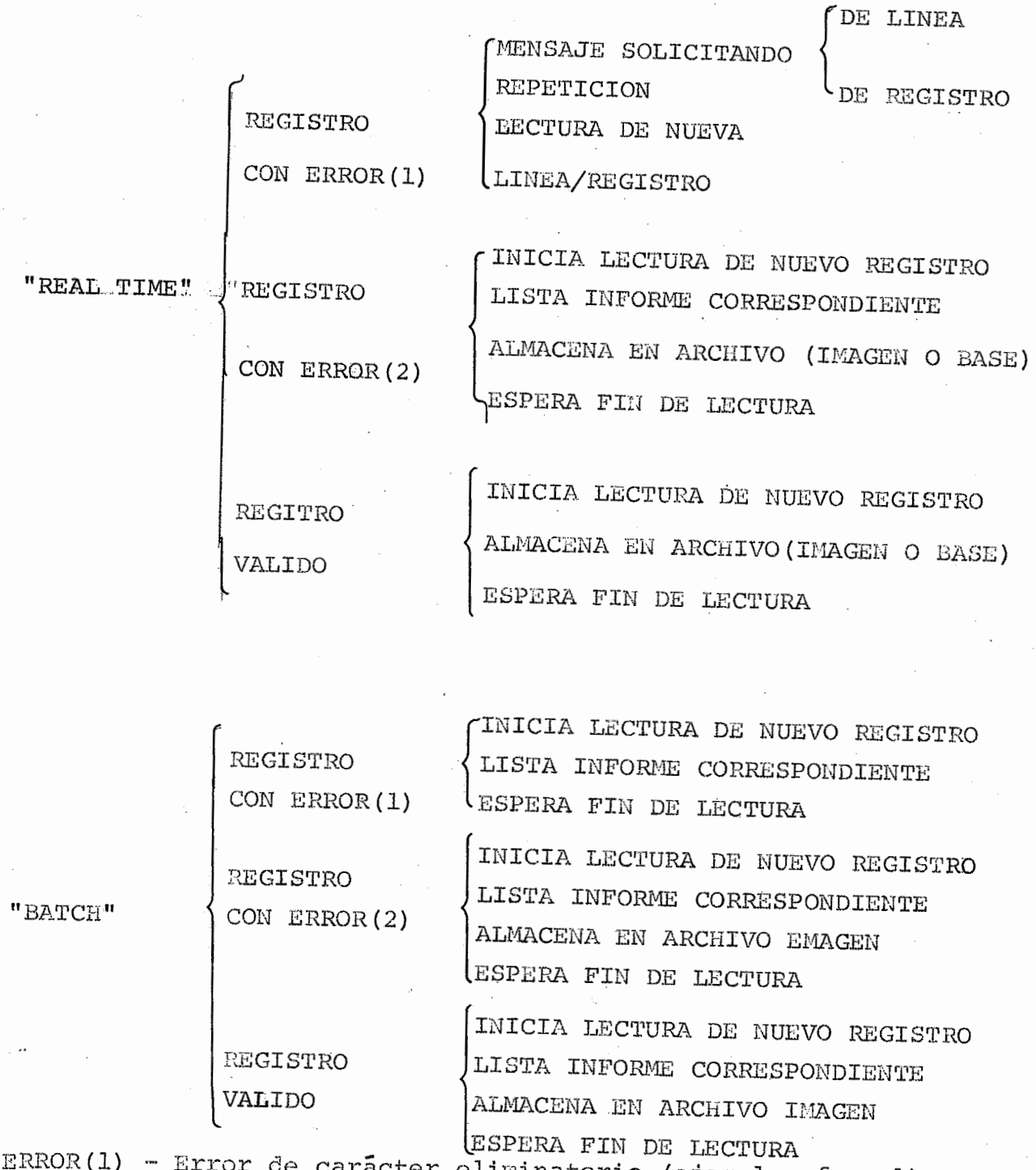
En el caso en que la transmisión de datos se efectúa con volúmenes importantes simultáneamente (ya sea en "batch" o en tiempo real) la opción más correcta es la primera pues a continuación del procesamiento de la consistencia se podría proceder a las actualizaciones, etc., sin efectar el funcionamiento del esquema global.

Por el contrario, si la información es transmitida a medida que es producida, el sistema seguramente, tendrá una continuidad que exigiría una actualización simultánea de los archivos. En ese caso la opción 2 es la más apropiada.

8.6 - Estructura básica del Sistema.

Antes de estructurar el sistema veremos una descripción, evento por evento, de cómo debería funcionar el sistema tanto en la operación "Batch" como "real time".

En el seguiente diagrama hemos representado esos eventos posibles y las acciones que el sistema deberá tomar para cada uno de ellos:



ERROR(1) - Error de carácter eliminatorio (ejemplo: formal)

ERROR(2) - Error de grupo o de tendencia.

En el caso de tiempo real el sistema deberá armar, uniendo las diferentes lineas transmitidas, el registro cuya consistencia debe controlar y si mantiene una referencia de los límites que, corresponden a cada línea transmitida podrá tener la opción, si el error está localizado, de solicitar - la repetición de una de las líneas del registro y en caso contrário solicita la repetición de todo el registro.

En batch el sistema debe armar el registro como conjunto de tarjetas pero esto solamente tiene importancia en el sentido de que los informes listados deben mantener el formato de la propia tarjeta para facilitar la verificación visual.

Un formato apropiado de edición de informes de errores - sería el poner una columna contadora de tarjetas, una columna indicando el tipo de la tarjeta (si las hay de varios tipos), una indicación de si la tarjeta fue o no grabada, la propia tarjeta separando los campos, aprovechando y distribuyendo las columnas sobrantes de la línea de impresión, y si aconteció un error en la línea siguiente, marcar con números las columnas de localización del error.

Cada uno de esos números correspondería a un tipo de error, definido por el programador al armar su núcleo de control de consistencia.

Cuando los errores referencian un grupo de registros el programador debe definir el mensaje y un código de tal for-

ma que cuando la rutina de impresión reciba ese codigo, listará el mensaje tal cual lo ha recibido.

En el caso de "real time" los informes impresos serán fundamentalmente de éste último tipo. Sin embargo podría especificarse un número fijo de tentativas (es necesario hacerlo) de repetición de un registro o línea que se detecte como errada, después de lo cual el sistema solicitaría al operador que pase al proximo registro.

La necesidad de imprimir esos registros dependerá de las características físicas del terminal en lo referente a si deja o no impresión de todas sus transacciones.

De acuerdo a lo expuesto nuestro sistema deberá tener la estructura indicada en la figura 8 con un pequeño supervisor - que distribuya las diferentes tareas e informaciones entre los diversos módulos.

Los módulos que llamamos "Supervisor, " Impresor" y "RECEPTOR" serán dadospor el sistema, permitiendo apenas al programador definir algunas opciones que ellos presentan.

El único módulo que, dada su complejidad y grán variedad de opciones escapa a todas las tentativas de darle un caracter fijo, es el módulo de consistencia.

La única solución que vemos para éste problema es el de crear un lenguaje que permita al programador especificar fa

cilmente los controles que se deben aplicar a los registros de entrada.

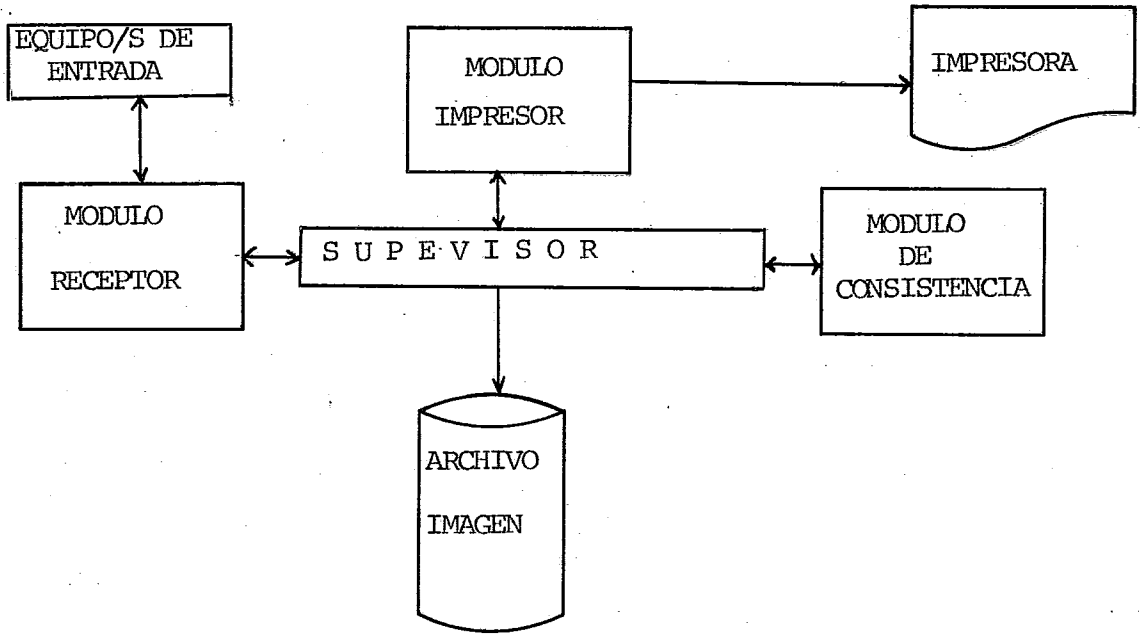


FIGURA 8.

En dónde este esquema varía es en la operación en tiempo real cuando la transmisión se desenvuelve a medida que se producen los datos.

En este caso se puede, y se debería, para aprovechar la capacidad de transmisión del terminal y de la línea, transmitir registros totalmente independientes y de características totalmente disímiles, con el único agregado de un campo de identifi-

cación, desde un mismo terminal y compartiendo el mismo período de tiempo.

El esquema para éste tipo de operación sería el de la Figura 9 donde se tendría un único Supervisor, un único IMPRESOR y un único RECEPTOR y vários módulos de Consistencia y Editores - (estos últimos no será dados por el sistema aunque podrían establecerse algunos padronizados, ver referencia en la alternativa 2 del parágrafo 8.5).

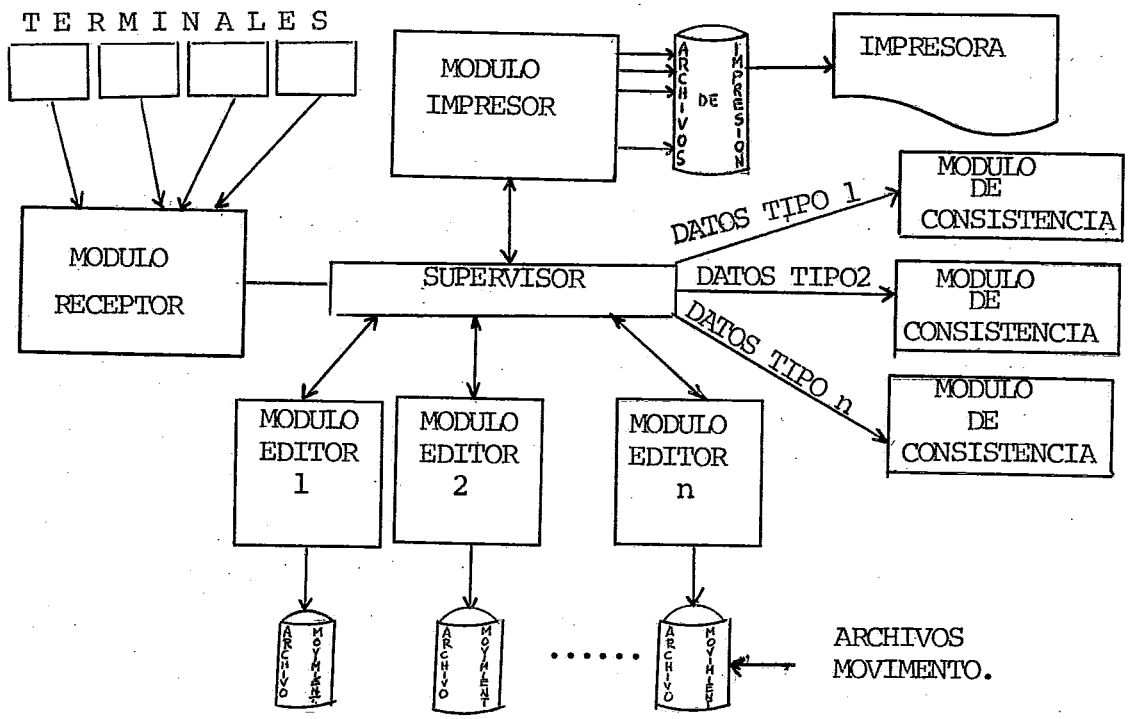


FIGURA 9.

Corresponde observar que el sistema representado en la figura 9 con muchos tipos de informaciones diferentes puede significar una carga importante en términos de memoria para el computador por lo cual sugerimos la posibilidad de aplicar una técnica similar a la del sistema ECAM. En efecto, los módulos de Consistencia y Edición correspondientes a tipos de datos - con alta frecuencia de transmisión podrían ser " subtasks" residentes del supervisor mientras que las restantes serían transitorias, siendo cargadas toda vez que aparezcan mensajes de ese tipo. Por supuesto, para ser entradas en la memoria, deberá hacerse uso de "tasks" intermediarias por iguales razones que las presentadas en el sistema.

CAPITULO 9

Implantación del Sistema.

9.1 - Necesidades y objetivos.

Una vez definida la estructura del sistema, no presenta mayor interés, del punto de vista académico, la implantación - de los diversos módulos, escepción hecha del Módulo de Consistencia que no quedó plenamente definido en virtud de que restaba especificar el lenguaje especial en que debía ser programado.

Sin embargo y en virtud del carácter típicamente conversacional del Sistema de Consistencia para operar en tiempo real éste se constituye en una aplicación característica del - Sistema de Acceso estructurado en la primera parte.

Por esa razón y con el único objetivo de realizar un test conjunto de ambos sistemas, deberemos hacer una implantación parcial de los módulos que resultaran, indispensables para concretizar el ejemplo de aplicaciones.

Asimismo, una vez definido el lenguaje que permitirá - programar el módulo de consistencia deberemos implantar un subconjunto del mismo a los efectos de poder efectuar el test citado.

9.2 - Origen de los comandos.

En el parágrafo 8.3.1 presentamos los tipos diferentes, más usados, de controle de validez.

Lógicamente, a través de un análisis exaustivo de sistemas existentes, de formatos de datos posibles, etc., se obtendría un conjunto de opciones sumamente completa y aplicable a una gran cantidad de casos particulares por disímiles que ellos sean. Sin embargo y a pesar de ello, nunca podremos asignarle una total generalidad al sistema, en tales condiciones y bastaría la aparición de un control no previsto y con amplia aplicación, para que el sistema pierda, en gran parte, su generalidad, para pasar a ser considerado como obsoleto. Por otra parte, el proceso de recolección de datos, publicaciones, experiencias, etc., representa una tarea tan difícil cuanto costosa en virtud de que no sabemos porqué razón, este tema no ha sido privilegiado(todo lo contrario) por las publicaciones existentes en el área de procesamiento de datos (con excepción de los fabricantes quienes, lógicamente, presentan un punto de vista extremadamente parcial).

En consecuencia, la alternativa que más se adapta a estas condiciones es la de incluir todas las opciones, a las que se tiene un acceso relativamente fácil, y se le da al lenguaje y lógicamente a las rutinas que lo sirven, una estructura formalmente modular, queriendo expresar de esa forma la capacidad de

ser indefinidamente ampliado mediante el agregado de módulos (que serán comandos elementales) diferentes entre sí.

Por lo tanto, habrá una correspondencia biunívoca entre los comandos del lenguaje que deseamos definir y las opciones de controles de validez básicos detectados por el análisis expuesto. Ante la aparición de nuevas opciones, nuevos comandos deberán ser implantados.

De donde proviene esa necesidad de que los comandos sean un reflejo de las propias opciones de control y no se considere la posibilidad de darles una estructura más elemental de forma de que combinándolos permitan construir tales opciones?

La respuesta a esta pergunta es una sola: simplicidad de programación. En la estructura establecida, el programador, para procesar cualquier control de validez, deberá simplemente coordinar, todos los comandos que corresponden a los tests de consistencia que deba realizar en su registro, con comandos elementales de ligazón que le permitan, según la lógica establecida, pasar de un control al que le deba seguir en secuencia, de acuerdo a los resultados obtenidos en cada uno de ellos. Por el contrario, si los comandos fuesen más elementales, el programador deberá en primer lugar crear, combinándolos, las estructuras básicas de control y de ahí partir para la organización del test global.

Esto presenta un doble inconveniente, por un lado el programador, deberá aprender un nuevo lenguaje que deberá constar

de un número elevado de comandos y que a menos que se llegue a adoptar un subconjunto de algún lenguaje de alto nível, lo que sería absurdo pues en ese caso bastaría con usar ese mismo lenguaje, con lo que volveríamos a los inconvenientes apuntados en el parágrafo 7.1, exigirá el obtener facilidad para manipular multiples comandos nada convencionales y por otro lado, tampoco estos comandos (a menos que sean muy elementales y volvemos a los lenguajes convencionales) podrán servir de base para generar cualquier tipo de control si es muy diferente del resto conocido.

Por consiguiente, la necesidad de modularidad se hace también presente en este caso y las consecuencias, anotadas en el mismo, del uso de comandos más elementales, reafirma nuestra decisión antes expresada.

9.3 - Lenguaje y Comandos.

En consecuencia de todo lo visto, a los efectos de la definición de los comandos de nuestro Lenguajes de Consistencia (podría también llamársele Lenguaje de Crítica) bastará con analizar los diferentes controles de validez, asimilarlos a un conjunto afín de necesidades y por medio de parámetros (operandos) permitir al comando correspondiente, definir aquellos aspectos que pueden ser variables para el control citado.

A estos comandos básicos se les deben agregar todos aquellos necesarios, para dar coherencia lógica al programa, para definir y reservar áreas a ser usadas, para comunicarse con el módulo supervisor, etc.

En éste trabajo definiremos apenas los comandos más representativos de los criterios que hemos expuesto, agregando también aquellos que nos permitan estructurar un ejemplo simple de aplicación:

1) Comando CARCTROL. Corresponde al tipo (a) del parágrafo 8.3.1 es decir al control de caracteres.

Deberá constar de los seguientes operandos:

Localización del campo: Si los campos, en la información de entrada, son fijos, bastará con dar, como información, el dislocamiento del primer caracter del campo con respecto al origen del registro y la longitud del campo, todo por supuesto a través de los correspondientes subparámetros.

Si existen campos variables se deberá dividir el registro de entrada en sectores, separados por los caracteres indicadores de fin de campo. Para localizar un campo

se deberá definir el sector a que pertenece, el dislocamiento de su primer carácter con respecto al origen del sector y su longi - tud o en su defecto una indicación de que el campo será variable.

Tipo de Control: Indicará la clase de test a ser realizado con todos los caracteres del campo:

ALFABETICO: Verificará si cada uno de los caracteres está entre A y Z.

NUMERICO _ Verificará si cada uno de los caracteres está entre 0 y 9.

NUMERICO CON SIGNO: Verificará si el signo incluído en la representación numérica es válido y además realizará el test numérico puro. En esencia verificará si la zona correspondiente al último dígito corresponde a los códigos de signo admitidos y si los caracteres anteriores son efectivamente numéricos.

ESPECIAL : Verificará si los caracteres corresponden a los conocidos como especiales.

APRTICULAR : Verificará si el o los caracteres del campo son iguales a un cierto carácter indicado en el operando (por ejemplo Z o ∅ , etc.).

Si se especifican varias de estas opciones simultáneamente esto habrá de equivaler a una opción de "OR" entre los diversos tests indicados, es decir que para ser validado ese campo, alcanza con que uno cualquiera de los controles dé un resultado - correcto.

Localización del resultado. Se especifica la dirección de un área en la memoria donde se informará al Módulo Supervisor de los resultados del control de validez realizado. Estos resultados incluirán un código indicador de"verdadero" o "Falso" y en este último caso, las columnas del campo donde se constataron las anomalías.

Código de Error: El programador deberá definir un número que representará la detección del error ("falso") por parte de este comando. ("Verdadero" debe ser común a todos).

A los restantes comandos también se les deberá asignar un código de error para luego permitir su manejo por parte del Módulo Supervisor.

2) Comando Contrel.Corresponde a los controles de relaciones entre campos, es decir, a los de tipo (b) citados en el Parágrafo 8.3.1. Sin embargo, no incluye el tipo (b5) que, dado que alcanza una mayor complejidad al establecer relaciones entre archivos diferentes, merecerá una atención particular y un comando especial.

Este comando, CONTREL, procesará simplemente una cierta relación, que debe especificarse en sus operandos, entre el campo referenciado y una o varias áreas de la memoria.

Será de responsabilidad del programador, cargar en dichas áreas los segundos operandos citados en las opciones (b1),(b2), (b3) y (b4) a medida que se consideren disponibles, es decir,en cuanto ya haya sido verificada la validez intrínseca de los mismos.

Como puede observarse, a través de estos comandos se logra tanto una programación simple cuanto al atendimiento de multiples aplicaciones diferentes en condiciones de geheralidad aceptables.

Puede además deducirse de lo expuesto que para conseguir estructurar un control de validez particular del tipo citado será necesario definir comandos para cargar el contenido de un campo en una cierta área, para procesar algunas operaciones aritméticas que pueden estar incluídas en la expresión lógica que se desea

controlar, para procesar operaciones lógicas (AND, OR, etc.),etc.

El comando CONTREL deberá constar de los siguientes operandos:

Localización del campo: Sigue idénticas especificaciones a las indicadas para el mismo operando en CARCTROL.

Localización de los operandos en la memoria: La relación será entre el campo definido por el operando anterior y el contenido de una o varias áreas en la memoria. En este operando se definen la localización y tamaño de tales áreas (el tamaño puede solamente ser especificado en comparación numérica, pues al comparar caracteres, ambos operandos deberán tener igual tamaño).

Operadores de relación: Especifican las relaciones a ser controladas y son:

Localización del resultado: Se reduce a un indicador de "verdadero" o "Falso" que podría ser transmitido por Registrador.

3) Comandos EXTREL - Corresponden al relacionamiento de un campo con otro, u otros campos, en un archivo exterior del que está siendo controlado, es decir al tipo (b5) citado en el Parágrafo 8.3.1.

Dependerán del tipo de organización del archivo que deberá ser consultado y así tendremos los comandos:

EXTRELSS: Para organización secuencial.

EXTRELIS: Para organización secuencial con índice (Indexed secuential).

EXTRELRM: Para organización aleatoria (Random).

La finalidad de estos comandos no es específicamente la de controlar la relación entre los campos pertencientes a registros de archivos diferentes sino la de constatar la existencia del registro referido y eventualmente cargar tal registro en un área indicada por el comando. Una vez cargado, se le puede relacionar con cualquier campo del registro de datos a través del comando CONTREL ya visto.

La aplicación más común, sin embargo, es la de verificar si el campo a controlar es, por ejemplo, la palabra "llave" - (Key Word), de un registro de un cierto archivo.

Será, en consecuencia, de exclusiva responsabilidad del programador el dar la descripción de ese archivo en la etapa de inicialización del programa, a los efectos de que el mismo pueda efectuar todas las tareas previas indispensables para lograr acceder a sus registros.

Los comandos EXTREL deberán presentar los siguientes parámetros.

Localización de la llave del registro: Deben darse la dirección en la memoria y el tamaño de dicha llave.

Indicador de tarea: Debe indicarse si el registro leído deberá ser o no, almacenado en la memoria.

Localización del registro: En caso de ser almacenado, éste parámetro, deberá indicar la dirección del área donde será movido.

Localización del resultado: Idem a CONTREL.

Corresponde hacer la observación que aún cuando la organización de archivo secuencial debe ser considerada, su aplicación para un procesamiento como el que nos ocupa resultaría extremadamente ineficiente y en consecuencia desaconsejable.

Otros comandos que deberían ser implantados son los que corresponden a una serie de aplicaciones comunes y que fueron parcialmente enumerados en el Parágrafo 8.3.1-

Aún cuando algunos de los mismos podrían ser estructurados mediante la combinación de los ya definidos, su amplio uso y la necesidad de dar facilidad de programación al sistema son argumentos suficientes para su implantación en particular.

El caso de control por dígito verificador es un caso especial pues existe una enorme cantidad de técnicas diferentes y, la implantación de todas ellas serán un esfuerzo excesivo en función de que, generalmente, cada instalación hace uso de un número muy reducido de ellas.

En consecuencia y dado que pretendemos darle una estructura modular al sistema deberá corresponder a cada instalación

el implantar los comandos de control por dígito verificador que le sean necesarios.

El resto de los comandos deberán ser de ligazón y creación de áreas y serán entre otros:

IFTRUE - Si el test anterior resultó "verdadero" desvía para la rutina indicada por su único operando, en caso contrario continúa en la secuencia.

IFALSE - Análogo y simétrico del anterior.

GOTO - Desvío incondicional.

DEFSTOR- Reserva áreas de memoria.

DEFCONST -Define y almacena constantes.

STORE - Almacena un campo del registro de datos de entrada, en una cierta área de la memoria.

OR - Realiza una operación lógica de "OR" entre los dos últimos test realizados.

AND - Idem pero con la operación lógica "AND".

EXOR - Idem pero con la operación lógica "Exclusive OR".

INICIO - Primer comando del programa que implica inicializarlo, salvar los registradores, abrir los archivos, etc., y deberá recibir informaciones al respecto desde sus operandos.

RETORNA- Fin lógico del programa, cerrará los archivos y retornará el control al Módulo Supervisor.

A éstos comandos deben agregarse aquellos que procesarán operaciones aritméticas y que analizaremos en el próximo parágrafo.

9.4 - El programa objeto.

En el sistema/360 disponemos de tres opciones para traducir un programa a lenguaje objeto:

1) Interpretador.

2) Compilador.

3) Macro-Generador.

Mientras las dos primeras implican el contruir rutinas que decodifiquen los comandos del lenguaje hasta darles condiciones para ser ejecutadas, la última exige el asignarle a cada comando, siguiendo una serie de reglas establecidas por el Macro-Generador disponible, las instrucciones en lenguaje Assembler que le corresponderán.

Tal Macro-Generador no es una constante en todos los sistemas y generalmente no está disponible en los Minicomputadores.

Hacemos mención de éstos, pues son sumamente usados para aplicaciones como las que nos ocupa (Entrada de datos) y eventualmente podría implantarse en ellos un sistema análogo al presentado en éste trabajo.

En lo que se refiere a las tres alternativas ya podemos eliminar la primera, es decir, el Interpretador.

En efecto, éste resulta totalmente incompatible con el hecho de que el programa de consistencia será ejecutado tantas veces cuantos sean los datos a controlar, pues, cada vez que sería ejecutado, debería también ser interpretado, lo cual para volúmenes grandes de datos como los que suponíamos resulta totalmente ineficiente.

En nuestro caso nos restan solamente dos últimas opciones, mientras que, para los minicomputadores, la única opción es la de Compilación. Comparando ambas posibilidades surgen dos características que hacen más conveniente el uso del Macro generador:

1) El Macro-Generador tiene modularidad automática de comandos. Si se desea agregar un comando basta con definir la macroinstrucción correspondiente.
   En el compilador, el agregado de nuevos comandos afecta casi toda si estructura y exige por lo tanto un conocimiento exaustivo de su estructura y funcionamiento.
2) El programa Assembler, que es el resultado de la Macrogeneración, puede ser, sin mayores dificultades, ligado a cualquier programa, en cualquier lenguaje, mientras que, para un lenguaje especial, compilado, el compilador deberá seguir una serie de requisitos, extensos y comple

jos, para poder dar, al programa objeto resultante, condiciones similares al Assembler. Este hecho tiene real importancia tomando en cuenta que, aun cuando nosotros hemos incluido el Módulo de Consistencia como parte esencial en nuestro sistema, nada impide que, para cualquier aplicación, un programa use el Módulo de Consistencia como subrutina para controlar la validez de sus datos.

El hecho de que ese programa pueda ser escrito en cualquier lenguaje aumenta la posibilidad de aplicación de esa estructura.

En consecuencia, debemos definir nuestros comandos como Macro-Instrucciones y esto es lo que fue presentado, en el Apéndice B de este trabajo, para un subconjunto de ellos.

Corresponde el respecto del Macro-Assembler hacer la seguiente observación:

Las estructuras fijadas para las macro-instrucciones corresponden a un lenguaje de tipo Regular y en consecuencia solamente serán posibles de ser representadas de esa forma las expresiones lingüísticas de ese tipo. Pero he aquí que las expresiones lógicas y/o aritméticas no corresponden a un lenguaje Regular sino de Contexto libre y esto implica que tales expresiones no podrán ser representadas por macro-instrucciones.

Sin embargo, si la exigencia de generalidad en ese tipo de expresiones no resulta crítica, se las puede convertir en regulares limitando el número máximo de operandos y hacerlas así representables por Macro-Instrucciones. Ese es justamente el criterio usado en nuestro sistema, como se podrá constatar por los comandos que procesan operaciones aritméticas que figuran en el Apéndice B.

CAPITULO 10

CONCLUSIONES

10.1 - Integración y eficiencia

Con el sistema que acabamos de presentar tentamos realizar una integración de una serie de técnicas comunes, que aparecían dispersadas en una gran multiplicidad de sistemas de Entrada de Datos.

No podemos saber hasta no haberlo verificado, durante un funcionamiento extendido del sistema, si los objetivos que nos trazamos fueron realmente alcanzados.

Creemos, y el test presentado lo confirma, que nuestro sistema puede funcionar, y la duda solo se plantea respecto a su eficiencia comparada con otros sistemas similares.

Sin embargo, lo que realmente importa no es si este o aquel sistema será más o menos eficiente, sino que la diversificación de sistemas con aplicaciones afines resulta sumamente ineficiente y que las técnicas de integración deben ser extendidas en el campo de procesamiento de datos, que en virtud de sus condiciones actuales se constituye en un terreno fértil para ello.

En lo que se refiere al propio sistema, presenta como ventajas no sólo el disponer de un sistema general de entrada

de datos, sino también que, a través del significado de sus comandos, se establece una verdadera metodología para controlar la validez de las informaciones entradas.

Otro concepto que estimamos importante, es el de "modularidad" en lenguajes aplicados, como lo es el de consistencia presentado en este trabajo. En efecto, por aparente que sea la generalidad, resulta imposible el prever todas las variantes posibles y es factible el sugimiento más o menos frecuentemente, de la necesidad de acrecentar comandos al sistema con los incovenientes que ello implica.

Si, por el contrario, se estructura el sistema partiendo de que, tales cosas deberán forzósamente acontecer y se le habilita con las condiciones favorables para ello, se obtendrá una excelente eficiencia, del sistema para cualquier instalación, dado que será esta la que habrá de definir sus propios comandos particulares a ser agregados a los comandos de base, de aplicación general.

CONCLUSONES GENERALES

Al leer el presente trabajo seguramente habrá, en el lector, una fuerte tendencia a considerarlo, como dos estudios totalmente separados, con objetivos y metodologías a tal punto diferentes, que difícilmente se acepte cualquier justificativa respecto a su inclusión simultánea en el mismo.

Este aspecto se agravará, posiblemente más aún, cuando tal justificativa se establece en base a la necesidad de realizar un test exigente del Método de Acceso estructurado en la primera Sección, pues tal test no justifica el tipo de análisis y la profundidad que se tentó da al estudio del problema de consistencia.

Sin embargo, esta aparente inconsistencia tiene como explicación, que estimamos valedera en el propio camino que debimos recorrer en la tentativa de resolver las dificultades que fueron apareciendo.

En efecto, no existe ninguna relación entre el orden establecido en este trabajo y el orden seguido para su elaboración, pues el problema a que nos abocamos en primer lugar fue el de procurar integrar los métodos para controlar la consistencia de los datos de entrada y durante su consideración nos vimos enfrentados a la posibilidad de operarlos en tiempo real. Esto nos llevó a estudiar los sistemas existentes para acceder a los equipos de teleprocesamiento y nos encontramos con la

desagradable sorpresa de no existir nada que se adaptara, tanto a nuestrasnecesidades como a nuestra configuración.

Enfrentados a esta realidad decidimos implantar un Acceso para los terminales de que disponíamos y tomando en cuenta únicamente el tipo de exigencias operativas que presentaba el sistema de consistencia.

Sin embargo no obtuvimos éxito, una serie de personas interesadas en potencialidades del procesamiento para sus aplicaciones particulares nos procuraron para tentar usar el sistema que estábamos estructurando.

Y así, de acuerdo a las hecesidades de cada uno de los solicitantes fuimos tomando conciencia del tipo de criterios, conceptos, etc., que debían regir un sistema de Acceso de aplicación general.

Una vez completado el sistema de Acceso retornamos al Sistema de Consistencia y lo reanalizamos en razón de los conceptos adquiridos.

El orden dado en el trabajo corresponde, entendemos, al que resulta más coherente para el lector, pues éste no habrá de aplicar los terminales en el sistema de consistencia, sin saber antes cómo se accede a ellos. Aplicamos, en consecuencia en la redación lo que resulta elemental en cualquier área de trabajo: antes de proceder a la aplicación debe crearse la infraestructuta.

Como punto final de nuestro trabajo debemos decir que no hemos inventado técnicas nuevas, simplemente nos hemos sumergido en un mundo apasionante de múltiples opciones y multiples soluciones, para salir de él con nuevas ideas y algunas conclusiones que no sabemos si son total o parcialmente acertadas pero sí sabemos que son parte del camino para alcanzar el objetivo de crear.-

APENDICE A

Macro-Instruccion TML 2741 generadora del sistema de acceso implantado.

```
         MACRO
         TML2741    &NUMTER=1,&EDITOR=SI,&BACKSP=SI,&TABUL=SI,          *
               &COLINIC=1,&TABCOL=,&MAXMES=(130,120),&CODIGOS=(RF40,    *
               SD40),&OCUPLIN=NO,&BLOCK=SI,&FINBLK=AO,&DECOD=SI,        *
               &SYSTER=10,&TRANERR=NO,&DDNAME=,&IDIOMA=ASM
         LCLC  &LABEL
         LCLA  &TERNUM
         LCLA  &BUFLEC
         LCLA  &BUFESC
         LCLA  &BUF
         LCLA  &LECT
         LCLA  &NOBLOCK
         LCLA  &AUX
         LCLA  &IMPLINE
         LCLA  &IMP
         LCLA  &NSTAB
         LCLA  &PARTAB
         LCLA  &ERROR
         LCLA  &TAM
         LCLA  &A
         LCLA  &B
&AUX     SETA  0
&NSTAB   SETA  N'&TABCOL
&ERROR   SETA  0
         AIF   (&NUMTER GT 0 AND &NUMTER LE &SYSTER).SEQ1
&ERROR   SETA  1
         MNOTE 1,'*** NUMTER CON VALOR INVALIDO ***'
.SEQ1    AIF   ('&EDITOR' EQ 'SI' OR '&EDITOR' EQ 'NO').SEQ2
&ERROR   SETA  1
         MNOTE 2,'*** EDITOR CON OPCION INVALIDA ***'
.SEQ2    AIF   ('&BACKSP' EQ 'SI' OR '&BACKSP' EQ 'NO').SEQ3
&ERROR   SETA  1
         MNOTE 2,'*** BACKSP CON OPCION INVALIDA ***'
.SEQ3    AIF   ('&TABUL' EQ 'SI' OR '&TABUL' EQ 'NO').SEQ4
&ERROR   SETA  1
         MNOTE 2,'*** TABUL CON OPCION INVALIDA ***'
.SEQ4    AIF   (&COLINIC GE 1 AND &COLINIC LT 131).SEQ5
&ERROR   SETA  1
         MNOTE 3,'*** COLINIC CON VALOR INVALIDO ***'
.SEQ5    AIF   (&NSTAB EQ 0).SEQ6
&AUX     SETA  &AUX+1
         AIF   (&AUX GT &NSTAB).SEQ6
        AIF   (&TABCOL(&AUX) GE &COLINIC AND &TABCOL(&AUX) LT 130).SEQ5
&ERROR   SETA  1
         MNOTE 4,'*** PARAMETROS DE TABCOL CON VALORES INVALIDOS ***'
.SEQ6    AIF   (N'&MAXMES EQ 2 AND &MAXMES(1) GE 0 AND &MAXMES(2) GE 0)*
               .SEQ7
&ERROR   SETA  1
         MNOTE 5,'*** PARAMETROS DE MAXMES INVALIDOS ***'
.SEQ7    AIF   (N'&CODIGOS EQ 2).SEQ8
```

```
&ERROR   SETA  1
         MNOTE 6,'*** PARAMETROS DE OPERANDO CODIGOS INVALIDOS ***'
.SEQ8    AIF   ('&OCUPLIN' EQ 'SI' OR '&OCUPLIN' EQ 'NO').SEQ9
&ERROR   SETA  1
         MNOTE 2,'*** OCUPLIN CON OPCION INVALIDA ***'
.SEQ9    AIF   ('&BLOCK' EQ 'SI' OR '&BLOCK' EQ 'NO').SEQ10
&ERROR   SETA  1
         MNOTE 2,'*** BLOCK CON OPCION INVALIDA ***'
.SEQ10   AIF   (K'&FINBLK EQ 2).SEQ11
&ERROR   SETA  1
         MNOTE 7,'*** REPRESENTACION DE FINBLK INVALIDA ***'
.SEQ11   AIF   ('&DECOD' EQ 'SI' OR '&DECOD' EQ 'NO').SEQ12
&ERROR   SETA  1
         MNOTE 2,'*** DECOD CON OPCION INVALIDA ***'
.SEQ12   AIF   ('&TRANERR' EQ 'SI' OR '&TRANERR' EQ 'NO').SEQ13
&ERROR   SETA  1
         MNOTE 2,'*** TRANERR CON OPCION INVALIDA ***'
.SEQ13   AIF   ('&IDIOMA' EQ 'FORT' OR '&IDIOMA' EQ 'ASM' OR '&IDIOMA' *
               EQ 'PL1' OR '&IDIOMA' EQ 'COB').SEQ14
&ERROR   SETA  1
         MNOTE 2,'*** IDIOMA CON OPCION INVALIDA ***'
.SEQ14   AIF   (&ERROR EQ 0).SEQ15
         MEXIT
.SEQ15   AIF   ('&EDITOR' EQ 'SI').G01
&BUFESC  SETA  17+&MAXMES(2)
         AGO   .G02
.G01     ANOP
&AUX     SETA  &MAXMES(2)/(126-&COLINIC)*(126-&COLINIC)
         AIF   (&AUX EQ &MAXMES(2)).G03
&BUFESC  SETA  &MAXMES(2)/(126-&COLINIC)*16+17+&MAXMES(2)
.G02     AIF   ('&BLOCK' EQ 'SI').G04
&NOBLOCK SETA  170+&COLINIC
&BUFLEC  SETA  39+&MAXMES(1)
&LECT    SETA  &BUFLEC-17
         AIF   (&BUFLEC LE &NOBLOCK).G05
&BUFLEC  SETA  &NOBLOCK
&LECT    SETA  &BUFLEC-17
         AGO   .G05
.G04     ANOP
&LECT    SETA  154-&COLINIC
&BUFLEC  SETA  40+&MAXMES(1)
         AIF   (&LECT LE &BUFLEC-17).G05
&LECT    SETA  &BUFLEC-17
.G05     AIF   (&BUFESC GE &BUFLEC).G06
&BUF     SETA  &BUFLEC-17
         AGO   .G07
.G06     ANOP
&BUF     SETA  &BUFESC-17
.G07     ANOP
&IMPLINE SETA  126-&COLINIC
```

```
&IMP      SETA  &IMPLINE+16
&A        SETA  &LECT+1
&B        SETA  &LECT-1
INITPW    CSECT
          ENTRY INITP,WAITP,WAITPTY,CLOSETP,READTP,GETP,WRITP,PUTP
          STM   14,12,12(13)
          LR    12,15
          USING  INITPW,12
          MVC   CODAREA(4),=F'1'
&TERNUM   SETA  &NUMTER
.TA1      ANOP
          MVC   SAVE&TERNUM.(60),12(13)
          MVI   TONT&TERNUM+1,X'FF'
&TERNUM   SETA  &TERNUM-1
          AIF   (&TERNUM GT 0).TA1
          B     TRANSFER
          DS    0F
INITP     STM   14,12,12(13)
          BAL   12,12(0,15)
          DC    A(INITPW)
          L     12,0(0,12)
          MVC   CODAREA(4),=F'0'
          B     TRANSFER
          DS    0F
WAITPTY   STM   14,12,12(13)
          BAL   12,12(0,15)
          DC    A(INITPW)
          L     12,0(0,12)
          MVC   CODAREA(4),=F'6'
          L     2,0(0,1)
          L     3,0(0,2)
          C     3,=F'0'
          BNH   TRANSFER
          L     2,4(0,1)
          MVC     0(4,2),=F'0'
          BAL   8,STRET
          B     TRANSFER
          DS    0F
WAITP     STM   14,12,12(13)
          BAL   12,12(0,15)
          DC    A(INITPW)
          L     12,0(0,12)
          MVC   CODAREA(4),=F'6'
          L     2,0(0,1)
          L     3,0(0,2)
          L     2,4(0,1)
          MVC   0(4,2),=F'1'
          C     3,=F'0'
          BL    TRANSFER
          BE    PROCURA
```

```
          BAL     3,STRET
          B       TRANSFER
STRAT     C       3,=F'&NUMTER.'
          BH      TRANSFER
          S       3,=F'1'
          SR      2,2
          M       2,=F'2'
          LA      4,TONT1(3)
          MVI     1(4),X'00'
          BR      8
STRET     C       3,=F'&NUMTER.'
          BH      TRANSFER
          S       3,=F'1'
          SR      2,2
          M       2,=F'2'
          LA      4,TONT1(3)
          MVI     1(4),X'FF'
          M       2,=F'4'
          B       SALVA(3)
&LABEL    SETC    'SALVA'
.TA2      ANOP
&TERNUM   SETA    &TERNUM+1
&LABEL    MVC     SAVE&TERNUM.(60),12(13)
          BR      8
&LABEL    SETC    ''
          AIF     (&TERNUM LT &NUMTER).TA2
&LABEL    SETC    'PROCURA'
&TERNUM   SETA    1
.TA8      ANOP
&LABEL    CLI     TONT&TERNUM+1,X'FF'
          BNE     *+10
          MVC     SAVE&TERNUM.(60),12(13)
&LABEL    SETC    ''
&TERNUM   SETA    &TERNUM+1
          AIF     (&TERNUM LE &NUMTER).TA8
          B       TRANSFER
&TERNUM   SETA    &TERNUM-1
          DS      0F
CLOSETP   STM     14,12,12(13)
          BAL     12,12(0,15)
          DC      A(INITPN)
          L       12,0(0,12)
          MVC     CODAREA(4),=F'7'
.TA3      ANOP
          MVI     TONT&TERNUM+1,X'00'
          MVC     SAVE&TERNUM.(60),12(13)
&TERNUM   SETA    &TERNUM-1
          AIF     (&TERNUM GE 1).TA3
          B       TRANSFER
          DS      0F
```

```
READTP   STM   14,12,12(13)
         BAL   12,12(0,15)
         DC    A(INITPW)
         L     12,0(0,12)
         MVC   CODAREA(4),=F'2'
         L     2,0(0,1)
         L     3,0(0,2)
         BAL   8,STRAT
         B     TRANSFER
         DS    0F
WRITP    STM   14,12,12(13)
         BAL   12,12(0,15)
         DC    A(INITPW)
         L     12,0(0,12)
         MVC   CODAREA(4),=F'4'
         L     2,0(0,1)
         L     3,0(0,2)
         BAL   8,STRAT
         AIF   ('&IDIOMA' NE 'FORT').TA5
         B     STRING
         AGO   .TA6
.TA5     B     TRANSFER
.TA6     ANOP
         DS    0F
GETP     STM   14,12,12(13)
         BAL   12,12(0,15)
         DC    A(INITPW)
         L     12,0(0,12)
         MVC   CODAREA(4),=F'3'
         L     2,0(0,1)
         L     3,0(0,2)
         C     3,=F'0'
         BNH   TRANSFER
         BAL   8,STRET
         B     TRANSFER
         DS    0F
PUTP     STM   14,12,12(13)
         BAL   12,12(0,15)
         DC    A(INITPW)
         L     12,0(0,12)
         MVC   CODAREA(4),=F'5'
         L     2,0(0,1)
         L     3,0(0,2)
         C     3,=F'0'
         BNH   TRANSFER
         BAL   8,STRET
         AIF   ('&IDIOMA' NE 'FORT').TA4
STRING   L     2,4(0,1)
         L     3,0(0,2)
         S     3,=F'1'
```

```
         C      3,=F'0'
         BL     TRANSFER
         SR     2,2
         M      2,=F'2'
         L      2,3(0,1)
         LA     6,0(3,2)
         LA     7,BUF
GON      MVC    0(1,7),0(2)
         A      7,=F'1'
         A      2,=F'2'
         CR     2,6
         BNH    GON
.TA4     ANOP
TRANSFER ST     13,SAVEAR+4
         LR     11,13
         LA     13,SAVEAR
         ST     13,8(11)
         LA     5,CODAREA
         MVC    LISTA(4),0(1)
         MVC    LISTA+4(4),4(1)
         MVC    LISTA+8(4),8(1)
         AIF    ('&IDIOMA' NE 'PL1').TA15
         L      9,LISTA+8
         MVC    LISTA+9(3),1(9)
.TA15    AIF    ('&IDIOMA' NE 'FORT').TA7
         MVC    LISTA+9(3),DIR+1
.TA7     LR     6,1
         LA     1,LISTA
         L      15,DESVIO
         BALR   14,15
         AIF    ('&IDIOMA' NE 'FORT').TA9
         L      2,4(0,1)
         L      3,0(0,2)
         C      3,=F'0'
         BL     GOBACK
         CLC    CODAREA(4),=F'3'
         BNE    GOBACK
         L      4,8(0,6)
         L      7,8(0,1)
         AR     3,7
COMPAR   CR     7,3
         BE     GOBACK
         MVC    0(1,4),0(7)
         A      4,=F'2'
         A      7,=F'1'
         B      COMPAR
.TA9     ANOP
GOBACK   L      2,0(0,1)
         L      3,0(0,2)
         S      3,=F'1'
```

```
          M       2,=F'60'
          L       13,SAVEAR+4
          LA      12,SAVE1(3)
          LM      14,12,0(12)
          BR      14
          LTORG
SAVEAR    DS      18F
CODAREA   DS      F
LISTA     DS      3F
DESVIO    DC      V(TML2741)
&TERNUM   SETA    1
.TA10     ANOP
SAVE&TERNUM DS    15F
&TERNUM   SETA    &TERNUM+1
          AIF     (&TERNUM LE &NUMTER).TA10
&TERNUM   SETA    1
.TA11     ANOP
TONT&TERNUM DC    2X'00'
&TERNUM   SETA    &TERNUM+1
          AIF     (&TERNUM LE &NUMTER).TA11
.TA12     ANOP
          AIF     (&MAXMES(1) GT &MAXMES(2)).TA13
&TAM      SETA    &MAXMES(2)
          AGO     .TA14
.TA13     ANOP
&TAM      SETA    &MAXMES(1)
.TA14     ANOP
BUF       DC      &TAM.X'00'
DIR       DC      A(BUF)
.TA20               ANOP
TML2741   CSECT
          STM     14,12,12(13)
          LR      12,15
          USING   TML2741,12,9
          CNOP    0,4
          BAL     9,*+8
          DC      A(TML2741+4096)
          L       9,0(0,9)
          ST      13,SAVEAREA+4
          LR      11,13
          LA      13,SAVEAREA
          ST      13,8(11)
          LA      10,LINETP
          USING   IHADCB,10
          USING   IECTDECB,11
          CLI     LLAVE,X'01'
          BE      NOP
          MVI     LLAVE,X'01'
          ST      5,CODAD
          OPEN    (IMPRES,(OUTPUT))
```

```
          OPEN  LINETP
          TM    DCBOFLGS,X'10'
          BZ    ABRRO1
NOP       L     7,24(0,11)
          TM    0(7),X'80'
          BNZ   ERROR1
          TM    4(7),X'80'
          BNZ   ERROR1
          TM    8(7),X'80'
          BZ    ERROR1
          L     5,CODAD
          CLC   0(4,5),=F'1'
          BH    IOPCODE
&TERNUM   SETA  &NUMTER
.VA1      READ  DECB&TERNUM,TI,LINETP,AREA&TERNUM,&LECT,,&TERNUM
          LTR   15,15
          BNZ   RWERRO
          MVC   RLN&TERNUM+10(2),=H'1'
&TERNUM   SETA  &TERNUM-1
          AIF   (&TERNUM GT 0).VA1
          CLC   0(4,5),=F'1'
          BE    TWAIT
          MVC   0(4,5),=F'0'
RETORNA   L     13,SAVEAREA+4
          LM    14,12,12(13)
          BR    14
IOPCODE   CLC   0(4,5),=F'7'
          BE    FIN
          CLC   0(4,5),=F'6'
          BNE   SIGUE
          L     1,0(0,7)
          CLC   0(4,1),=F'0'
          BL    ERROR3
          BH    PRTY
&LABEL    SETC  'GRAL'
.VA3      ANOP
&TERNUM   SETA  &TERNUM+1
          AIF   (&TERNUM GT &NUMTER).VA2
&LABEL    CLC   RLN&TERNUM+10(2),=H'1'
          BE    TWAIT
&LABEL    SETC  ''
          AGO   .VA3
.VA2      B     ERROR5
RETURN    L     5,CODAD
          CLC   0(4,5),=F'3'
          BE    TWAIT
          CLC   0(4,5),=F'5'
          BE    TWAIT
          MVC   0(4,5),=F'0'
          B     RETORNA
```

```
PRTY     L      2,4(0,7)
         CLC    0(4,2),=F'0'
         BNE    GRAL
         L      3,0(0,1)
         C      3,=F'&NUMTER.'
         BH     ERROR3
         SR     2,2
         S      3,=F'1'
         M      2,=F'14'
         B      *+4(3)
&TERNUM  SETA   1
.BR31    ANOP
         CLC    RLN&TERNUM+10(2),=H'1'
         BE     WAIT&TERNUM
         B      ERROR5
&TERNUM  SETA   &TERNUM+1
         AIF    (&TERNUM LE &NUMTER).BR31
TWAIT    TWAIT  (8),ECBLIST=LISTEC8
         L      15,WTGLIST(15)
         BR     15
SIGUE    L      1,0(0,7)
         SR     2,2
         SR     3,3
         L      3,0(0,1)
         C      3,=F'&NUMTER.'
         BH     ERROR3
         S      3,=F'1'
         C      3,=F'0'
         BL     ERROR3
         M      2,=F'12'
         LA     6,RLN1(3)
         L      4,4(0,6)
         CLC    0(4,5),=F'3'
         BNH    0(0,4)
         L      1,4(0,7)
         SR     3,3
         L      3,0(0,1)
         C      3,=F'0'
         BNH    ERROR4
         L      5,8(0,7)
         L      4,0(0,6)
         BR     4
FIN      CLOSE  LINETP
         MVI    LLAVE,X'00'
&TERNUM  SETA   1
.BR30    CLC    RLN&TERNUM+10(2),=H'1'
         BE     MENS
&TERNUM  SETA   &TERNUM+1
         AIF    (&TERNUM LE &NUMTER).BR30
         CLOSE  IMPRES
```

```
         B       RETORNA
MENS     MVC     LINEA+1(62),=CL62'*** LINEA DE TP CERRADA CON OPERACIONE*
                 S DE E/S EN PROGRESO ***'
         PUT      IMPRES,LINEA
         XC      LINEA,LINEA
         CLOSE   IMPRES
         L       1,4(0,7)
         MVC     0(4,1),=F'-6'
         B       RETORNA
MESER    MVC     LINEA+1(82),=CL82'*** ERROR *** LA UNIDAD DE CONTROL O L*
                 A LINEA NO ESTAN EN CONDICIONES DE OPERACION'
         B       VUELVE
ABRRO1   MVC     LINEA+1(82),=CL82'*** ERROR *** PROBABLEMENTE ERROR DE C*
                 ODIFICACION EN LA DEFINICION DEL GRUPO DE TP'
VUELVE   PUT     IMPRES,LINEA
         CLOSE   (IMPRES,,LINETP)
         L      13,SAVEAREA+4
         L       13,4(13)
         L       13,4(13)
         LM      14,12,12(13)
         BR      14
ERROR1   MVC     LINEA+1(80),=CL80'*** ERROR *** PARAMETROS TRANSFERIDOS *
                 AL ACCESO FUERA DE LAS NORMAS ESTABLECIDAS'
         B       VUELVE
RWERRO   MVC     LINEA+1(50),=CL50'*** ERROR *** DEFECTO EN LA TCU - ERRO*
                 R DE MAQUINA'
         B       VUELVE
ERROR3   MVC     LINEA+1(38),=CL38'*** ERROR *** NUMERO DE LINEA INVALIDO*
                 '
         B       VUELVE
ERROR4   MVC     LINEA+1(39),=CL39'*** ERROR *** MENSAJE DE LARGO INVALID*
                 O'
         B       VUELVE
ERROR5   MVC     LINEA+1(69),=CL69'*** ERROR *** CALL WAITP SIN OPERACION*
                 ES DE ENTRADA/SALIDA PENDIENTES'
         B       VUELVE
.VA5     ANOP
&TERNUM  SETA    &TERNUM-1
NEWL&TERNUM MVC  RLN&TERNUM+8(2),=H'1'
         MVC     RLN&TERNUM+10(2),=H'1'
         MVC     0(4,5),=F'5'
         WRITE   DECB&TERNUM,TV,LINETP,AR&TERNUM,1,,&TERNUM,MF=E
         B       TEST&TERNUM
WTERL&TERNUM LA  4,AREAS&TERNUM
         BAL     8,EDITOR
         STH     3,DECB&TERNUM+6
         MVC     RLN&TERNUM+8(2),=H'0'
         MVC     RLN&TERNUM+10(2),=H'1'
         WRITE   DECB&TERNUM,TV,LINETP,AREAS&TERNUM,,,&TERNUM,MF=E
TEST&TERNUM LTR  15,15
```

```
          BZ    RETURN
          MVC   LINEA+1(30),=CL30'*** ERROR *** LINEA &TERNUM. OCUPADA '
          AIF   ('&OCUPLIN' EQ 'NO').BRANCH
          PUT   IMPRES,LINEA
          XC    LINEA,LINEA
          L     1,4(0,7)
          MVC   0(4,1),=F'-1'
          L     1,0(0,7)
          MVC   0(4,1),=F'&TERNUM.'
          B     RETORNA
          AGO   .BR1
.BRANCH   B     VUELVE
.BR1      ANOP
READTL&TERNUM L 6,AP&TERNUM
          A     6,=F'&LECT.'
          C     6,FINBUF&TERNUM
          BH    CONT3&TERNUM
          L     6,AP&TERNUM
          MVC   RLN&TERNUM+10(2),=H'1'
          READ  DECB&TERNUM,TT,LINETP,0(0,6),&LECT,,&TERNUM,MF=E
          B     TEST&TERNUM
CONT3&TERNUM LA 6,FINBUF&TERNUM
          S     6,AP&TERNUM
          STH   6,DECB&TERNUM+6
          L     6,AP&TERNUM
          MVC   RLN&TERNUM+10(2),=H'1'
          READ  DECB&TERNUM,TI,LINETP,0(0,6),,,&TERNUM,MF=E
          B     TEST&TERNUM
WAIT&TERNUM WAIT 1,ECB=DECB&TERNUM
WAITR&TERNUM CLI  DECB&TERNUM,X'7F'
          BNE   ERRO&TERNUM
          MVI   DECB&TERNUM,X'00'
          LA    11,DECB&TERNUM
          MVC   RLN&TERNUM+10(2),=H'0'
          CLI   DECTYPE+1,X'03'
          BE    LEYO&TERNUM
          CLI   DECTYPE+1,X'01'
          BE    LEYO&TERNUM
          CLC   RLN&TERNUM+8(2),=H'1'
          BE    WCTRL&TERNUM
          L     1,4(0,7)
          MVC   0(4,1),=F'0'
          L     1,0(0,7)
          MVC   0(4,1),=F'&TERNUM.'
          L     5,CODAD
          MVC   0(4,5),=F'5'
          B     RETORNA
WCTRL&TERNUM MVC  RLN&TERNUM+8(2),=H'0'
          MVI   MESSTL&TERNUM,X'5E'
          L     5,CODAD
```

```
          MVC    0(4,5),=F'3'
          B      READTL&TERNUM
LEYO&TERNUM SR   1,1
          L      6,AP&TERNUM
          AIF    ('&TABUL' EQ 'NO').VA6
          LA     4,TAB
.VA6      ANOP
TRYT&TERNUM TRT  0(&LECT,6),TABLA
          CLI    0(1),X'1F'
          BE     FINLIN&TERNUM
          AIF    ('&TABUL' EQ 'NO').VA7
          CLI    0(1),X'7A'
          BE     TABRUT&TERNUM
.VA7      AIF    ('&BACKSP' EQ 'NO').VA8
          CLI    0(1),X'60'
          BNE    RETR&TERNUM
          S      1,=F'1'
          LR     5,1
          A      5,=F'&A.'
          CL     5,FINBUF&TERNUM
          BNL    BIG&TERNUM
          CL     1,AP&TERNUM
          BL     EX&TERNUM
          MVC    0(&B,1),2(1)
          LR     6,1
          B      TRYT&TERNUM
EX&TERNUM A      1,=F'1'
          MVC    0(&A,1),1(1)
          LR     6,1
          B      TRYT&TERNUM
BIG&TERNUM L     5,FINBUF&TERNUM
          SR     5,1
          S      5,=F'2'
          STC    5,*+5
          MVC    0(1,1),2(1)
          LR     6,1
          B      TRYT&TERNUM
.VA8      ANOP
RETR&TERNUM MVC    RLN&TERNUM+10(2),=H'1'
          AIF    ('&TRANERR' EQ 'SI').VA11
          B      RLECT&TERNUM
          AGO    .VA21
.VA11     B      CERR3&TERNUM
.VA21     AIF    ('&TABUL' EQ 'NO').VA9
TABRUT&TERNUM LR 8,1
          S      8,AP&TERNUM
          TM     0(4),X'80'
          BNZ    NOTAB&TERNUM
          CH     8,2(0,4)
          BNL    SIGUE&TERNUM
```

```
SIGUE&TERNUM.) L  5,0(0,4)
          L     2,=F'&B'
          SR    2,5
          SR    5,8
          LA    11,0(5,1)
          MVC   AUXAREA(&B),1(1)
          MVI   0(1),X'01'
          S     5,=F'2'
          CL    5,=F'0'
          BL    DOS&TERNUM
          STC   5,*+5
          MVC   1(1,1),0(1)
          L     5,FINBUF&TERNUM
          SR    5,11
          CR    2,5
          BNL   CINCO&TERNUM
          STC   2,*+5
          MVC   0(1,11),AUXAREA
DOS&TERNUM L    6,AP&TERNUM
          A     6,0(0,4)
          A     4,=F'4'
          B     TRYT&TERNUM
CINCO&TERNUM S  5,=F'1'
          STC   5,*+5
          MVC   0(1,11),AUXAREA
          B     DOS&TERNUM
SIGUE&TERNUM A  4,=F'4'
          TM    0(4),X'80'
          BNZ   NOTAB&TERNUM
          CH    8,2(0,4)
          BNL   SIGUE&TERNUM
          B     SIGUE&TERNUM.1
NOTAB&TERNUM MVI 0(1),X'01'
          LA    6,1(1)
          B     TRYT&TERNUM
.VA9      ANOP
FINLIN&TERNUM S 1,=F'1'
          L     5,CODAD
          MVC   0(4,5),=F'3'
          CLC   0(2,1),=X'5B1F'
          BNE   NEWL&TERNUM
          AIF   ('&BLOCK' EQ 'NO').VA10
          S     1,=F'1'
          CLI   0(1),X'A0'
          BE    ARMAR&TERNUM
          MVC   1(2,1),=X'0101'
          A     1,=F'1'
          ST    1,AP&TERNUM
          B     READTL&TERNUM
.VA10     ANOP
```

```
ARMAR&TERNUM LA  8,AREA&TERNUM
          MVC   AP&TERNUM.(4),=A(AREA&TERNUM.)
          SR    1,8
          L     2,4(0,7)
          ST    1,0(0,2)
          L     2,0(0,7)
          MVC   0(4,2),=F'&TERNUM.'
          AIF   ('&DECOD' EQ 'NO').VA12
          LR    6,1
          LR    0,1
          CNOP  0,4
          BAL   1,*+16
          DC    A(LINETP)
          DC    A(IECT&CODIGOS(1))
          DC    A(AREA&TERNUM)
          L     15,*+8
          B     *+8
          DC    V(IECTTRNS)
          BALR  14,15
          LR    1,6
.VA12     S     1,=F'1'
          L     5,8(0,7)
CONT8&TERNUM CH 1,=H'255'
          BNH   CONT7&TERNUM
          MVC   0(256,5),0(8)
          A     8,=F'256'
          A     5,=F'256'
          S     1,=F'256'
          B     CONT8&TERNUM
CONT7&TERNUM STC 1,*+5
          MVC   0(1,5),0(8)
          B     RETORNA
ERRO&TERNUM MVI DEC8&TERNUM,X'00'
          MVC   LINEA+1(25),=CL25'*** WAIT COMP CODE=41 ***'
          PUT   IMPRES,LINEA
          XC    LINEA,LINEA
          LA    11,DEC8&TERNUM
          AIF   ('&TRANERR' EQ 'NO').VA32
          CLI   DECFLAGS,X'02'
          BNE   CERR1&TERNUM
          L     1,4(0,7)
          MVC   0(4,1),=F'-2'
          L     1,0(0,7)
          MVC   0(4,1),=F'&TERNUM.'
          B     RETORNA
CERR1&TERNUM CLI DECERRST,X'80'
          BE    MESER
          CLI   DECSENSO,X'08'  DATA CHECK
          BNE   CERR2&TERNUM
          L     1,4(0,7)
```

```
          MVC   0(4,1),=F'-3'
          L     1,0(0,7)
          MVC   0(4,1),=F'&TERNUM.'
          B     RETORNA
CERR2&TERNUM CLI DECSENSO,X'02'  LOST DATA
          BNE   CERR3&TERNUM
          L     1,4(0,7)
          MVC   0(4,1),=F'-4'
          L     1,0(0,7)
          MVC   0(4,1),=F'&TERNUM.'
          B     RETORNA
CERR3&TERNUM L  1,4(0,7)
          MVC   0(4,1),=F'-5'
          L     1,0(0,7)
          MVC   0(4,1),=F'&TERNUM.'
          B     RETORNA
          AGO   .VA33
.VA32     CLC   RLN&TERNUM+8(2),=H'1'
          BE    RLECT&TERNUM
          CLI   DECFLAGS,X'02'
          BE    RETCL&TERNUM
          CLI   DECTYPE+1,X'01'
          BE    LECT&TERNUM
          CLI   DECTYPE+1,X'03'
          BE    LECT&TERNUM
          WRITE DECB&TERNUM,TV,LINETP,AREAS&TERNUM,,,&TERNUM,MF=E
          L     5,CODAD
          MVC   0(4,5),=F'5'
          B     TEST&TERNUM
.VA33      ANOP
RLECT&TERNUM MVI MESSTL&TERNUM,X'5B'
LECT&TERNUM  MVC MESSTL&TERNUM+17(30),=CL30'*** REPITA LA ULTIMA LINEA *
               ***'
          AIF   ('&DECOD' EQ 'SI').VA13
          TRNSLATE LINETP,SD40,MESSTL&TERNUM+17,30
          AGO   .VA14
.VA13     TRNSLATE LINETP,&CODIGOS(2),MESSTL&TERNUM+17,30
.VA14     MVC   MESSTL&TERNUM+47(16),EXTR
          WRITE DECB&TERNUM,TV,LINETP,MESSTL&TERNUM,63,,&TERNUM,MF=E
          MVC   RLN&TERNUM+8(2),=H'1'
          L     5,CODAD
          MVC   0(4,5),=F'5'
          B     TEST&TERNUM
RETCL&TERNUM SR 2,2
          LH    2,DECB&TERNUM
          A     2,=F'1'
          STH   2,DECB&TERNUM
          WRITE DECB&TERNUM,TV,LINETP,AR&TERNUM,,,&TERNUM,MF=E
          L     5,CODAD
          MVC   0(4,5),=F'5'
```

```
         B       TEST&TERNUM
         AIF     (&TERNUM GT 1).VA5
EDITOR   CL      3,=F'&MAXMES(2).'
         BH      ERROR4
         AIF     ('&EDITOR' EQ 'NO').VA15
         LR      6,3
         A       3,=F'32'
CONT2    CL      6,=F'&IMPLINE.'
         BH      CONT1
         LR      0,6
         S       6,=F'1'
         STC     6,*+5
         MVC     16(1,4),0(5)
         CNOP    0,4
         BAL     1,*+16
         DC      A(LINETP)
         DC      A(IECT&CODIGOS(2))
         DC      A(0)
         A       4,=F'16'
         ST      4,*-8
         L       15,*+8
         B       *+8
         DC      V(IECTTRNS)
         BALR    14,15
         LA      4,1(6,4)
         MVC     0(16,4),EXTR
         BR      8
CONT1    MVC     16(&IMPLINE,4),0(5)
         MVC     &IMP.(16,4),EXIR
         LA      0,&IMPLINE
         CNOP    0,4
         BAL      1,*+16
         DC      A(LINETP)
         DC      A(IECT&CODIGOS(2).)
         DC      A(0)
         A       4,=F'16'
         ST      4,*-8
         L       15,*+8
         B       *+8
         DC      V(IECTTRNS)
         BALR    14,15
         A       3,=F'16'
         A       4,=F'&IMPLINE.'
         S       6,=F'&IMPLINE.'
         A       5,=F'&IMPLINE.'
         B       CONT2
         AGO     .VA16
.VA15    LR      6,3
         A       3,=F'16'
CONT2    CL      6,=F'256'
```

```
            BH      CONT1
            S       6,=F'1'
            STC     6,*+5
            MVC     16(1,4),0(5)
            BR      8
CONT1       MVC     16(256,4),0(5)
            A       4,=F'256'
            S       6,=F'256'
            A       5,=F'256'
            B       CONT2
.VA16       ANOP
SAVEAREA    DS      18F
TABLA       DS      0CL256
            DC      31X'00'
            DC      X'37'
            DC      61X'00'
            AIF     ('&BACKSP' EQ 'SI').VA17
            DC      X'00'
            AGO     .VA18
.VA17       DC      X'38'
.VA18       DC      28X'00'
            AIF     ('&TABUL' EQ 'NO').VA19
            DC      X'39'
            AGO     .VA20
.VA19       DC      X'00'
.VA20       DC      133X'00'
LINEA       DC      CL133'0'
AUXAREA     DS      CL&LECT
EXTR        DC      X'58'
            DC      15X'5E'
LLAVE       DC      X'00'
CODAD       DS      F
            AIF     ('&TABUL' EQ 'NO').VA24
TAB         DS      0F
&NSTAB      SETA    N'&TABCOL
&AUX        SETA    0
.VA23       AIF     (&AUX EQ &NSTAB).VA22
&AUX        SETA    &AUX+1
&PARTAB     SETA    &TABCOL(&AUX)-&COLINIC
            DC      A(&PARTAB.)
            AGO     .VA23
.VA22       DC      XL4'80000000'
.VA24       ANOP
RLN&TERNUM  DC      A(WTERL&TERNUM.)
            DC      A(READTL&TERNUM.)
            DC      F'0'
&TERNUM     SETA    &TERNUM+1
            AIF     (&TERNUM LE &NUMTER).VA24
.VA25       ANOP
&TERNUM     SETA    &TERNUM-1
```

```
AR&TERNUM DC    X'58'
AREAS&TERNUM DC 15X'5E'
          DC    X'7C'
AREA&TERNUM DC  &BUF.X'01'
          AIF   (&TERNUM GT 1).VA25
.VA26     ANOP
AP&TERNUM DC    A(APEA&TERNUM.)
FINBUF&TERNUM DC A(AREA&TERNUM+&BUF.)
MESSTL&TERNUM DC 16X'5E'
          DC    X'7C'
          DS    59X
&TERNUM   SETA  &TERNUM+1
          AIF   (&TERNUM LE &NUMTER).VA26
WTGLIST   DS    OF
&TERNUM   SETA  1
.VA27     ANOP
          DC    A(WAITR&TERNUM.)
&TERNUM   SETA  &TERNUM+1
          AIF   (&TERNUM LE &NUMTER).VA27
&TERNUM   SETA  1
LISTECB   DS    OF
.VA28     AIF   (&TERNUM EQ &NUMTER).VA29
          DC    A(DECB&TERNUM.)
&TERNUM   SETA  &TERNUM+1
          AGO   .VA28
.VA29     DC    X'80'
          DC    AL3(DECB&TERNUM.)
IMPRES    DCB   DDNAME=SYSACC,EROPT=ACC,LRECL=133,RECFM=FA,DSORG=PS,     *
                MACRF=(PM),BFTEK=S,BLKSIZE=133
LINETP    DCB   DSORG=CX,MACRF=(R,W),DDNAME=&DDNAME,EROPT=WT
          LTORG
          AIF   ('&DECOD' EQ 'NO').VA30
          ASMTRTAB &CODIGOS(1),&CODIGOS(2)
          AGO   .VA31
.VA30     ASMTRTAB SD40
.VA31     DCBD  DSORG=CX
          IECTDECB
          MEND
```

APENDICE B

Mensajes de errores en el sistema TML 2741.

En el sistema MACLAN, así como en el TML 2741 que fue implantado, el usuario tendrá la opción, en caso de errores y de acuerdo a la gravedad de los mismos, de elegir entre recibir en forma codificada la información correspondiente al acontecimiento de un error, inmediatamente después de ser detectado, o por el contrario, dejar esta labor al sistema. Este, en virtud del uso de rutinas especiales, tenta recuperarlo y en caso de no ser posible, provoca una finalización anormal del programa correspondiente, adjuntando el o los mensajes adecuados para hacer posible la interpretación, por parte del usuario, del fenómeno causante del error.

Con el objetivo de informar acerca de los diferentes tipos de errores, haremos a continuación una lista, con una explicación suscinta de las características que los provocaron acrecentando además los mensajes a ser impresos por el sistema al lograr su detección así como los códigos usados para comunicar al programa de aplicación el acontecimiento de los mismos:

a) Error detectado al recibir, el Acceso, un comando Closetp y verificar que restan operaciones de entrada/salida pendientes.
   En éste caso el sistema imprime el mensaje:
   *** LINEA DE TP CERRADA CON OPERACIONES DE E/S EN PROGRESO ***
   Codifica TAMAÑO=-6,Y devuelve el control al programa de aplicación una vez ejecutada la operación de cierre.

b) Error detectado en el funcionamiento de algun equipamiento de transmisión.
   El sistema imprime:
   *** ERROR *** LA UNIDAD DE CONTROL O LA LINEA NO ESTAN EN CONDICIONES DE OPERACIÓN.
   A continuación da fin a la ejecución del programa.

c) Detecta un error causado probablemente por definición equivocada, en el programa de aplicación, del grupo determinados afectados al mismo.
Imprime el mensaje:
*** ERROR *** PROBABLEMENTE ERROR DE CODIFICACION EN LA DEFINICION DEL GRUPO DE TP.
Y a continuación da fin al programa.

d) Detecta errores en los parámetros que le son transferidos por los diferentes comandos.
Imprime el mensaje:
*** ERROR *** PARAMETROS TRANSFERIDOS AL ACCESO FUERA DE LAS NORMAS ESTABLECIDAS.
Y a continuación da fin al programa.

e) Detecta defecto en la TCU (Transmission Control Unit)
Imprime el mensaje:
*** ERROR *** DEFECTO EN LA TCU-ERROR DE MAQUINA.
Y a continuación da fin al programa.

f) Recibe como información el RLN (Relative Line Number) que resulta inconsistente.
Imprime el mensaje:
*** ERROR *** NUMERO LINEA INVALIDO.
Y a continuación da fin al programa.

g) Recibe un mensaje de largo excesivo de acuerdo a las especificaciones recibidas.
Imprime el mensaje:
*** ERROR *** MENSAJE DE LARGO EXCESIVO.
Y a continuación da fin al programa.

h) Detecta que al recibir un comando WAITP no existen operaciones de entrada/salida pendientes.
Imprime el mensaje:
*** ERROR *** CALL WAITP SIN OPERACIONES DE ENTRADA/SALIDA PENDIENTES.
Y a continuación da fin al programa.

i) Recibe un comando, solicitando una operación de entrada/salida sobre una linea con otra operación pendiente.
El sistema imprime:
*** ERROR *** LINEA n OCUPADA.
Y a continuación según que la opción OCUPLIN especifique SI o NO, devuelve el control al programa de aplicación y codifica TAMAÑO=-1 o da fin al programa, respectivamente.

j) Detecta que el operador del terminal interrumpió el proceso de escritura en el mismo, mediante la tecla ATTN. Si la opción TRANERR fue establecida como SI codifica TAMAÑO=-2
Y da el control al programa de aplicación, si por el contrario es NO, el sistema retransmite el mensaje hacia el mismo terminal.

k) Detecta que ha ocurrido un error de transmisión por "Data - Check". Si la opción TRANERR es SI, codifica TAMAÑO=-3 y le transfiere el control al programa de aplicación, si por el contrario es NO, el sistema detecta si el error aconteció durante la transmisón o la recepción de datos, retransmitiendo en el primer caso el mensaje y en el segundo solicita la retransmisión enviando el mensaje:
*** REPITA LA ULTIMA LINEA ***

l) Detecta que ha ocurrido un error de transmisión por pérdida de datos (LOST DATA).
Procede igual que en el caso anterior, codificando TAMAÑO=-4.

m) Detecta que ha ocurrido un error de transmisión sin poder - identificar su causa.
En ese caso procede igual a los dos casos anteriores, codificando TAMAÑO=-5.

APENDICE C

Macro-instrucciones correspondientes al conjunto de consistencias implantadas.

```
*
          MACRO
&LABEL    CARCTROL  &CAMPO=/,&ORIG=,&LONG=,&AREA=,&TIPO=,&ERROR=,&CODIGO=
          GBLA   &TABLA
          LCLA   &NSLSTIP
          LCLA   &NSLSTSG
          LCLA   &NEWLONG
          LCLA   &SLSTNDX
          LCLA   &NORIG
          LCLC   &INLABEL
          LCLC   &OUTLABL
          LCLB   &LOGIC
          ACTR   80
&OUTLABL  SETC   '#&SYSNDX'
&INLABEL  SETC   '$&SYSNDX'
&NORIG    SETA   &ORIG-1
&NSLSTIP  SETA   N'&TIPO+1
&SLSTNDX  SETA   1
&NSLSTSG  SETA   &SLSTNDX+1
          AIF    (K'&ERROR GT 8 OR K'&AREA GT 8).ERROR1
          AIF    (&NORIG LT 0).ERROR2
          AIF    ('&AREA' EQ '').VA1
&LABEL    LA     4,&AREA
          A      4,=F'&NORIG.'
          LR     2,3
          L      3,=F'1'
          MVI    &ERROR,C'1'
          AGO    .VA2
.VA1      AIF    ('&CAMPO' EQ '/').VA3
          AIF    (N'&CAMPO NE 2).ERROR3
&LABEL    L      5,=F'&CAMPO.'
          LR     2,3
          L      3,=F'1'
          MVI    &ERROR,C'1'
          L      4,0(0,1)
CC&SYSNDX S      5,=F'1'
          C      5,=F'0'
          BE     PAK&SYSNDX
          LR     10,1
C&SYSNDX  TRT    0(256,4),#TABLA
          CLI    0(1),C'&CAMPO(1).'
          BNE    CON&SYSNDX
          LR     4,1
          A      4,=F'1'
          LR     1,10
          B      CC&SYSNDX
CON&SYSNDX A      4,=F'256'
          B      C&SYSNDX
PAK&SYSNDX A      4,=F'&NORIG.'
```

```
          AGO   .VA2
.VA3      ANOP
&LABEL    L     4,0(0,1)
          LR    2,3
          L     3,=F'1'
          MVI   &ERROR,C'1'
          A     4,=F'&NORIG.'
.VA2      ANOP
&LOGIC    SETB  ('&TIPO(&SLSTNDX)' NE 'ALF' AND '&TIPO(&SLSTNDX)' NE 'NU*
                M' AND '&TIPO(&SLSTNDX)' NE 'SIGN' AND '&TIPO(&SLSTNDX)'*-
                NE 'ESPEC' AND K'&TIPO(&SLSTNDX) NE 1)
          AIF   (&LOGIC AND '&TIPO(&SLSTNDX)' NE 'BL').ERROR4
          AIF   ('&LONG' EQ '/').VA4
          L     6,=F'&LONG.'
.VA4      AIF   ('&TIPO(&SLSTNDX)' NE 'ALF').SIGA1
&INLABEL&SLSTNDX CLI 0(4),C'A'
          BL    &INLABEL&NSLSTSG
          CLI   0(4),C'Z'
          BH    &INLABEL&NSLSTSG
          CLI   0(4),C'S'
          BNL   &OUTLABL
          CLI   0(4),C'R'
          BH    &INLABEL&NSLSTSG
          CLI   0(4),C'J'
          BNL   &OUTLABL
          CLI   0(4),C'I'
          BNH   &OUTLABL
&SLSTNDX  SETA  &SLSTNDX+1
          AIF   (&SLSTNDX EQ &NSLSTIP).END
.SIGA1    AIF   ('&TIPO(&SLSTNDX)' NE 'NUM').SIGA2
&NSLSTSG  SETA  &SLSTNDX+1
&INLABEL&SLSTNDX CLI  0(4),C'0'
          BL    &INLABEL&NSLSTSG
          CLI   0(4),C'9'
          BNH   &OUTLABL
&SLSTNDX  SETA  &SLSTNDX+1
          AIF   (&SLSTNDX EQ &NSLSTIP).END
.SIGA2    AIF   ('&TIPO(&SLSTNDX)' NE 'SIGN').VA6
&NSLSTSG  SETA  &SLSTNDX+1
&INLABEL&SLSTNDX B  GON&SYSNDX
Z&SYSNDX  DC    Z'0'
GON&SYSNDX MVC  Z&SYSNDX.(1),0(4)
          OI    Z&SYSNDX,X'F0'
          CLI   Z&SYSNDX,C'9'
          BH    &INLABEL&NSLSTSG
          MVC   Z&SYSNDX.(1),0(4)
          NI    Z&SYSNDX.(1),X'F0'
          CLI   Z&SYSNDX,X'F0'
          BE    &OUTLABL
          CLI   Z&SYSNDX,X'D0'
```

```
          BE    &OUTLABL
          CLI   Z&SYSNDX,X'C0'
          BE    &OUTLABL
          CLI   Z&SYSNDX,X'E0'
          BE    &OUTLABL
&SLSTNDX  SETA  &SLSTNDX+1
          AIF   (&SLSTNDX EQ &NSLSTIP).END
.VA6      AIF   ('&TIPO(&SLSTNDX)' NE '8L').SIGA3
&NSLSTSG  SETA  &SLSTNDX+1
&INLABEL&SLSTNDX CLI O(4),C' '
          BE    &OUTLABL
&SLSTNDX  SETA  &SLSTNDX+1
          AIF   (&SLSTNDX EQ &NSLSTIP).END
.SIGA3    AIF   (K'&TIPO(&SLSTNDX) NE 1).SIGA4
&NSLSTSG  SETA &SLSTNDX+1
&INLABEL&SLSTNDX CLI O(4),C'&TIPO(&SLSTNDX).'
          BE    &OUTLABL
&SLSTNDX  SETA  &SLSTNDX+1
          AIF   (&SLSTNDX NE &NSLSTIP).SIGA3
          AGO   .END
.SIGA4    AIF   ('&TIPO(&SLSTNDX)' NE 'ESPEC').BEGIN
&NSLSTSG  SETA  &SLSTNDX+1
&INLABEL&SLSTNDX CLI   O(4),X'4A'
          BL    &INLABEL&NSLSTSG
          CLI   O(4),X'7F'
          BH    &INLABEL&NSLSTSG
          CLI   O(4),X'7A'
          BNE   &OUTLABL
          CLI   O(4),X'6F'
          BH    &INLABEL&NSLSTSG
          CLI   O(4),X'6B'
          BNL   &OUTLABL
          CLI   O(4),X'61'
          BH    &INLABEL&NSLSTSG
          CLI   O(4),X'60'
          BE    &INLABEL&NSLSTSG
          CLI   O(4),X'5A'
          BNL   &OUTLABL
          CLI   O(4),X'50'
          BNH   &OUTLABL
&SLSTNDX  SETA  &SLSTNDX+1
          AIF   (&SLSTNDX NE &NSLSTIP).BEGIN
.END      ANOP
&INLABEL&SLSTNDX L  3,=F'0'
          MVI   &ERROR,C'0'
          AIF   ('&CODIGO' EQ '').SIGA5
          LR    5,4
          S     5,0(0,1)
          A     5,4(0,1)
          MVC   0(1,5),=X'&CODIGO.'
```

```
.SIGA6   ANOP
         AIF   ('&LONG' EQ '/').VA5
&OUTLABL A     4,=F'1'
         BCT   6,&INLABEL.1
.SIGA8   AIF   ('&CAMPO' EQ '/').SIGA7
         AIF   (&TABLA NE 0).SIGA7
         B     FIN&SYSNDX
#TABLA   DS    0CL256
         DC    (X'&CAMPO(1).')X'00'
         DC    X'&CAMPO.'
         DC    (256-X'&CAMPO.')X'00'
&TABLA   SETA  1
FIN&SYSNDX EQU *
.SIGA7   MEXIT
.SIGA5   B     &OUTLABL+8
         AGO   .SIGA6
.VA5     A     4,=F'1'
         CLC   0(1,4)=X'&CAMPO(1).'
         BNE   &INLABEL.1
         AGO   .SIGA8
.ERROR1  MNOTE 1,'*** ERROR O AREA CON SIMBOLOS INVALIDOS ***'
         MEXIT
.ERROR2  MNOTE 2,'*** ORIG CON VALOR INVALIDO ***'
         MEXIT
.ERROR3  MNOTE 3,'*** CAMPO CON PARAMETROS INVALIDOS ***'
         MEXIT
.ERROR4  MNOTE 4,'*** TIPO CON PARAMETROS INVALIDOS ***'
         MEND
*
*
         MACRO
         DEFSTOR  &NOMBRES=,&LONGS=
         LCLA  &INDEX
         AIF   (N'&NOMBRES NE N'&LONGS).ERROR1
&INDEX   SETA  1
         AIF   ('&NOMBRES' EQ '' OR '&LONGS' EQ '').ERROR2
.VA1     AIF   (K'&NOMBRES(&INDEX) GT 8).ERROR3
&NOMBRES(&INDEX) DS  CL&LONGS(&INDEX)
&INDEX   SETA  &INDEX+1
         AIF   (&INDEX LE N'&LONGS).VA1
         DS    0H
         MEXIT
.ERROR1  MNOTE 1,'*** NOMBRES Y LONGS CON NUMERO DIFERENTE DE SUBPARAME*
               TROS ***'
         MEXIT
.ERROR2  MNOTE 2,'*** NOMBRES Y LONGS SIN PARAMETROS ***'
         MEXIT
.ERROR3  MNOTE 3,'*** NOMBRES(&INDEX.) CON SIMBOLO INVALIDO ***'
         MEND
*
```

```
*
          MACRO
&LABEL    STORE &CAMPO=/,&ORIG=,&LONG=,&AREA=
          GBLA  &TABLA
          LCLA  &NORIG
          LCLA  &SALE
          AIF   ('&ORIG' EQ '' OR '&LONG' EQ '' OR '&AREA' EQ '').ERROR1
&NORIG    SETA  &ORIG-1
          AIF   (&NORIG LT 0).ERROR2
          AIF   (K'&AREA GT 8).ERROR3
          AIF   ('&CAMPO' EQ '/').VA1
          AIF   (N'&CAMPO NE 2).ERROR4
&LABEL    L     5,=F'&CAMPO(2).'
          L     4,0(0,1)
CC&SYSNDX S     5,=F'1'
          C     5,=F'0'
          BE    PAK&SYSNDX
          LR    10,1
C&SYSNDX  TRT   0(256,4),#TABLA
          CLI   0(1),C'&CAMPO(1).'
          BNE   CON&SYSNDX
          LR    4,1
          LR    1,10
          A     4,=F'1'
          B     CC&SYSNDX
CON&SYSNDX A    4,=F'256'
          B     C&SYSNDX
PAK&SYSNDX  A   4,=F'&NORIG'
          AGO   .VA2
.VA1      ANOP
&LABEL    L     4,0(0,1)
          A     4,=F'&NORIG.'
.VA2      MVC   0(&LONG.,4),&AREA
          B     ST&SYSNDX
&AREA     DS    CL&LONG
          AIF   ('&CAMPO' EQ '/').VA3
          AIF   (&TABLA NE 0).VA3
#TABLA    DS    0CL256
          DC    (X'&CAMPO(1).')X'00'
          DC    X'&CAMPO(1).'
          DC    (256-X'&CAMPO(1).')X'00'
&TABLA    SETA  1
.VA3      DS    0H
ST&SYSNDX  EQU  *
          MEXIT
.ERROR1   MNOTE 1,'*** ORIG,LONG O AREA SIN PARAMETROS ***'
          MEXIT
.ERROR2   MNOTE 2,'*** ORIG CON VALOR INVALIDO ***'
          MEXIT
.ERROR3   MNOTE 3,'*** AREA CON NOMBRE INVALIDO ***'
```

```
         MEXIT
.ERROR4   MNOTE 4,'*** CAMPO CON NUMERO INVALIDO DE SUBPARAMETROS ***'
         MEND
*
*
         MACRO
&LABEL   ADD    &AREAS=,&LONG=,&RESULT=,&RESLONG=
         LCLA   &RESPACK,&INDEX
         LCLC   &LAB,&SALE,&OP1,&OP2
         AIF    (N'&AREAS NE N'&LONG).ERRO1
&SALE    SETC   '$&SYSNDX'
&LAB     SETC   '&LABEL'
&OP1     SETC   '$$&SYSNDX'
&OP2     SETC   '##&SYSNDX'
&RESPACK SETA   &RESLONG/2+1
&INDEX   SETA   1
.VA      ANOP
&LAB     PACK   &OP2.(&RESPACK),&AREAS(&INDEX),(&LONG(&INDEX))
         AP     &OP1.(&RESPACK),&OP2.(&RESPACK)
&INDEX   SETA   &INDEX+1
&LAB     SETC   ''
         AIF    (&INDEX LE N'&AREAS).VA
         UNPK   &RESULT.(&RESLONG),&OP1.(&RESPACK)
         B      &SALE
&RESULT  DS     CL&RESLONG
&OP1     DC     PL&RESPACK'0'
&OP2     DS     PL&RESPACK
&SALE    DS     0H
         MEXIT
.ERRO1   MNOTE 1,'*** AREAS Y LONGS NO SE CORRESPONDEN ***'
         MEND
*
*
         MACRO
&LABEL   CONTREL  &CAMPO=/,&ORIG=,&LONG=,&AREA=,&RELAC=,&SEGOP=,&ERROR=*
               ,&CODIGO=
         GBLA   &TABLA
         LCLA   &NORIG
         AIF    (K'&ERROR GT 8 OR K'&AREA GT 8 OR K'&SEGOP GT 8).ERROR1
         AIF       ('&RELAC' NE 'H' AND '&RELAC' NE 'L' AND '&RELAC' NE *
               'E' AND '&RELAC' NE 'NH' AND '&RELAC' NE 'NL' AND '&RELA*
               C' NE 'NE').ERROR2
&NORIG   SETA   &ORIG-1
         AIF    ('&AREA' EQ '').VA1
&LABEL   LA     4,&AREA
         A      4,=F'&NORIG,'
         LR     2,3
         L      3,=F'1'
         MVI    &ERROR,C'1'
         AGO    .VA2
```

```
.VA1      AIF   ('&CAMPO(1)' EQ '/').VA3
          AIF   (N'&CAMPO NE 2).ERROR3
&LABEL    L     5,=F'&CAMPO(2).'
          LR    2,3
          L     3,=F'1'
          MVI   &ERROR,C'1'
          L     4,0(0,1)
CC&SYSNDX S     5,=F'1'
          C     5,=F'0'
          BE    PAK&SYSNDX
          LR    10,1
C&SYSNDX  TRT   0(256,4),#TABLA
          CLI   0(1),C'&CAMPO(1).'
          BNE   CON&SYSNDX
          LR    4,1
          LR    1,10
          A     4,=F'1'
          B     CC&SYSNDX
CON&SYSNDX A    4,=F'256'
          B     C&SYSNDX
PAK&SYSNDX A    4,=F'&NORIG.'
          AGO   .VA2
.VA3      ANOP
&LABEL    L     4,0(0,1)
          A     4,=F'&NORIG.'
          LR    2,3
          L     3,=F'1'
          MVI   &ERROR,C'1'
.VA2      ANOP
          CLC   0(&LONG.,4),&SEGOP
          B&RELAC FIN&SYSNDX
          MVI   &ERROR,C'0'
          L     3,=F'0'
          AIF   ('&CODIGO' EQ '').VA5
          MVI   &ERROR,+1,C'&CODIGO'
.VA5      AIF   ('&CAMPO' EQ '/').VA4
          AIF   (&TABLA NE 0).VA4
          B     FIN&SYSNDX
#TABLA    DS    0CL256
          DC    (X'&CAMPO(1).')X'00'
          DC    X'&CAMPO(1).'
          DC    (256-X'&CAMPO(1).')X'00'
&TABLA    SETA  1
.VA4      ANOP
FIN&SYSNDX EQU *
          MEXIT
.ERROR1   MNOTE  1,'*** AREA,SEGOP O ERROR CON PARAMETROS INVALIDOS ***'
          MEXIT
.ERROR2   MNOTE  2,'*** RELAC CON PARAMETRO INVALIDO ***'
          MEXIT
```

```
.ERROR3  MNOTE  3,'*** CAMPO CON PARAMETRO/S INVALIDO/S ***'
         MEND
*
*
         MACRO
         DEFCONS &NAMES=,&LONGS=,&DATOS=
         LCLA   &INDEX
         LCLA   &PAR
         AIF    (N'&NAMES NE N'&LONGS OR N'&NAMES NE N'&DATOS).ERROR1
&INDEX   SETA   1
         AIF    (N'&NAMES EQ 0).ERROR2
.VAL     ANOP
&PAR     SETA   &LONGS(&INDEX)
         AIF    (K'&DATOS(&INDEX) NE &PAR).ERROR3
&NAMES(&INDEX) DC  CL&LONGS(&INDEX).'&DATOS(&INDEX).'
&INDEX   SETA   &INDEX+1
         AIF    (&INDEX LE N'&NAMES).VAL
         DS     0H
         MEXIT
.ERROR1  MNOTE 1,'*** NAMES,LONGS Y DATOS CON NUMERO DIFERENTE DE SUBPA*
               RAMETROS ***'
         MEXIT
.ERROR2  MNOTE   2,'*** NAMES,LONGS Y DATOS SIN SUBPARAMETROS ***'
         MEXIT
.ERROR3  MNOTE   3,'*** DATOS(&INDEX.) NO CORRESPONDE A SU LONGITUD ****
               '
          MEND
*
*
         MACRO
&LABEL   IFTRUE   &BRANCH
&LABEL   CL     3,=F'1'
         BE     &BRANCH
         MEND
*
         MACRO
&LABEL   IFALSE   &BRANCH
&LABEL   CL     3,=F'0'
         BE     &BRANCH
         MEND
*
         MACRO
&LABEL   GOTO   &BRANCH
&LABEL   B      &BRANCH
         MEND
*
         MACRO
&LABEL   NOT
&LABEL   C      3,=F'0'
         BE     *+12
```

```
          L       3,=F'0'
          B       *+8
          L       3,=F'1'
          MEND
*
      MACRO
&LABEL    ORR
&LABEL    CR      2,3
          BNE     *+20
          C       2,=F'1'
          BE      *+12
          L       3,=F'0'
          B       *+8
          L       3,=F'1'
          MEND
*
          MACRO
&LABEL    AND
&LABEL    CR      2,3
          BNE     *+12
          C       3,=F'1'
          BE      *+8
          L       3,=F'0'
          MEND
*
          MACRO
&LABEL    EXOR
&LABEL    CR      2,3
          BE      *+12
          L       3,=F'1'
          B       *+8
          L       3,=F'0'
          MEND
*
          MACRO
          DEPAREAS &AREAS=,&LONGS=,&NUMFORM=
          LCLA    &INDEX
          LCLA    &TAMAN
          AIF     (N'&AREAS NE N'&LONGS).ERROR1
          AIF     ('&AREAS' EQ '').ERROR2
          AIF     (&NUMTER LE 0).ERROR3
&TAMAN    SETA    0
&INDEX    SETA    1
.VA       AIF     (K'&AREAS(&INDEX) GT 8).ERROR4
&TAMAN    SETA    &TAMAN+&LONGS(&INDEX)
&INDEX    SETA    &INDEX+1
          AIF     (&INDEX LE N'&AREAS).VA
&INDEX    SETA    1
          L       4,4(0,1)
          L       5,0(0,4)
```

```
          CL     5,#TER
          BE     FIN&SYSNDX
          LR     7,5
          S      5,=F'1'
          M      4,=F'&TAMAN.'
          LA     4,TPSAVE
          LA     6,0(5,4)
          L      5,#TER
          ST     7,#TER
          S      5,=F'1'
          M      4,=F'&TAMAN.'
          LA     7,0(5,4)
.VA1      MVC    0(&LONGS(&INDEX).,7),&AREAS(&INDEX)
          MVC    &AREAS(&INDEX).(&LONGS(&INDEX).),0(6)
          A      7,=F'&LONGS(&INDEX).'
          A      6,=F'&LONGS(&INDEX).'
&INDEX    SETA   &INDEX+1
          AIF    (&INDEX LE N'&AREAS).VA1
          B      FIN&SYSNDX
TPSAVE     DS   &NUMFORM.CL&TAMAN
#TER      DC     F'1'
FIN&SYSNDX DS    0H
          MEXIT
.ERROR1   MNOTE 1,'*** AREAS Y LONGS CON NUMEROS DE SUBPARAMETROS DIFERE*
               NTES ***'
          MEXIT
.ERROR2   MNOTE 2,'*** AREAS SIN PARAMETROS ***'
          MEXIT
.ERROR3   MNOTE 3,'*** NUMFORM CON VALOR INVALIDO ***'
          MEXIT
.ERROR4   MNOTE 4,'*** SIMBOLO DE AREAS INVALIDO ***'
          MEND
*
          MACRO
          INICIO  &CONSIST
          AIF    (K'&CONSIST GT 8).ERROR1
&CONSIST  CSECT
          STM    14,12,12(13)
          LR     12,15
          USING  &CONSIST,12
          ST     13,#SAVE+4
          LR     11,13
          LA     13,#SAVE
          ST     13,8(11)
          B      FIN&SYSNDX
#SAVE     DS     18F
FIN&SYSNDX EQU *
          MEXIT
.ERROR1   MNOTE 1,'*** PARAMETRO INVALIDO DE INICIO ***'
          MEND
```

```
*
          MACRO
&LABEL    RETORNA  &AREA
          AIF    (K'&AREA GT 8).ERROR1
&LABEL    L      13,#SAVE+4
          LA     1,&AREA
          LM     14,0,12(13)
          LM     2,12,28(13)
          BR     14
          MEXIT
.ERROR1   MNOTE 1,'*** AREA CON SIMBOLO INVALIDO ***'
          MEND
*
```

B I B L I O G R A F I A

MARTIN, JAMES - Programming real time computer systems. Englewood Cliffs, Prentice - Hall, 1965.

WITHINGTON, FREDERIC G.-The use of computer in business organizations.Reading, Addison - Wesley, 1966.

STIMLER SAUL. - Real-time data- processing systems. Mc Graw-Hill, 1969.

G.J.CLANCY Jr.- ECAM - Extended Communications Access Method. Afips, volume 37, pg. 581 , 1970.

P.P. SCHODERBEK. - Management Systems. John Wiley Inc., 1967.

SZWARCFITER J.L. - Uma sistematização do processamento de dados. Tese, COPPE, MSc, 1971.

IBM. CORP. - OS/360 BASIC telecommunications access method message control program (BTAM). Form. GC30-2004.

IBM. CORP. - OS/360 planning for the telecommunications access method (TCAM). Form GC30-2020.

IBM. CORP. - OS/360 queued telecommunications access method message control program (QTAM). Form GC30-2005.