UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES

"ARAGON"

17

"DISEÑO DE UNA INTERFAZ GRAFICA PARA UN SISTEMA DE CALIBRACION DE DATOS HIDROGRAFICOS RECOLECTADOS POR UNA SONDA OCEANOGRAFICA"

T E S I S

Que para obtener el Título de:

INGENIERO EN COMPUTACION

P r e s e n t a :

CARLOS ADRIAN DELGADO DIAZ

ASESOR: ING. JUAN GASTALDI PEREZ

San Juan de Aragón, Edo. de México 1996.

DEDICATORIAS

A mis Padres:

María de Los Ángeles Díaz Vargas
y
Pedro Delgado Ponce

Por los que siento un gran orgullo y respeto. Por los que estaré eternamente agradecido por haberme brindado un techo, una educación, un apoyo económico, pero sobre todo por haberme otorgado toda su confianza y cariño para salir adelante en mi vida como persona y como profesionista.

Gracias.... Muchas Gracias.

A mis hermanos:

Guillermina, Avelina, Andrés, Angélica, Laura, Pepe y Pamela; por los que siento una gran admiración y cariño, y por que hacen posible el sentirme cada día mas orgulloso de pertenecer a una familia que al igual de grande es sólida y sobre todo unida.

A mis sobrinos:

Cayo, Ariadna, Marthita, Martín, Arturo, Alejandra, Javier, Andrea, Pedro Miguel, Pamelita, Andresín y José . A los cuales deseo servir de apoyo para que sus metas y objetivos se cumplan satisfactoriamente

A mis cuñados, tíos y primos.

A mi abuelita Teresa.

GRACIAS A....

A los Doctores Francisco y Victor M. Vidal Lorandi por la confianza otorgada al aceptarme como becario del GEO, por su acertada dirección durante mi trabajo, y sobre todo por su siempre clara y abierta amistad.

Al M.C. Eustorgio Meza Conde y al M.C. José María Pérez Molero, por su constante esfuerzo para que esta tesis fuera de la mayor calidad posible, por sus consejos atinados y por su preocupación para que este trabajo fuera concluido y realmente utilizado en la práctica.

A mis asesores y revisores de tesis de la ENEP Ing. Juan Gastaldi Pérez, Ing Juan José Martínez Cosgalla, Ing. Francisco García Mora, Ing. Ernesto Peñaloza Romero, Ing. Lucio Domínguez Estrada, por su valiosa ayuda en la corrección de la tesis y sus consejos para mejorarla.

A la Universidad Nacional Autónoma de México, por haberme brindado una completa educación y convertirme en un profesionista con el compromiso eterno de llevar siempre en alto su nombre.

Al Instituto de Investigaciones Eléctricas y al Grupo de Estudios Oceanográficos, por el apoyo económico brindado durante la estancia de tesis, así como por la oportunidad de hacer uso de sus instalaciones y equipos sin restricción alguna.

A mis amigos del IIE, con quienes compartí un tiempo y un espacio de trabajo, quienes hicieron posible que mi estancia en el Instituto fuera mas placentera y agradable. Agradezco a Jou Luis, Maricela (uyuyuy), Enrique, Mike, Leticia, Nimrod, Pepe, Iván (chochoutla), Macrina, Blanca, Mª Elena, Mallida, Teresita, Dulce, Endi, Gonzalo, Jorge Coca, Margarita, Víctor Pérez, Irma, Fabiola.

A todos los Investigadores del UREN y en especial del GEO, que siempre estuvieron dispuestos a brindarme su amistad, apoyo y asesoría. Agradezco a Lorenzo, Abel, Toño, Zulma, Josué, Pedro, Rodolfo, Reinaldo A y F. Huesca.

A mis compañeros y amigos de generación en la ENEP Aragón.

Indice

ÍNDICE

Introducción

Es un requisito indispensable para el emplazamiento de centrales núcleo eléctricas por parte de la Comisión Federal de Electricidad (CFE) la realización de estudios oceanográficos costeros y regionales para la preparación del capitulo referente al impacto ambiental del emplazamiento de centrales nucleares en la costa del Golfo de México.

Para obtener el licenciamiento para la construcción y operación de centrales nucleares en las costas, es necesario cumplir con los requerimientos de la Guía Reguladora 4.2, Revisión 2, adoptada por la Comisión Nacional de Seguridad Nuclear y Salvaguardias (CNSNS). En esta se estipula básicamente que debe realizarse una caracterización detallada, dentro de un radio de 100 km, tomado como centro la central nuclear, de la naturaleza física, química, biológica y geológica de los cuerpos de agua contiguos a la central, susceptibles de ser afectados por la operación normal o accidental de la misma.

Con la finalidad de llevar a cabo las mediciones oceanográficas regionales de sitios costeros y con el fin de reunir una base informática que defina las características físico ambientales del mar territorial Mexicano del Golfo de México. CFE y el Grupo de Estudios Oceanográficos (GEO) firmaron el contrato 1926 cuyo nombre es "Estudios Oceanográficos regionales para la Selección de sitios y licenciamiento de centrales nucleares en el golfo de México".

Por otro lado el Consejo Nacional De Ciencia y Tecnología (CONACYT) tuvo el interés en que se recabara y publicara un banco de datos oceanográficos de la zona exclusiva económica Mexicana del Golfo, para lo cual contribuyó con las campañas oceanográficas "ARGOS" cediendo su tiempo de uso compartido del buque oceanográfico "JUSTO SIERRA". Gracias a lo anterior el GEO se dio a la tarea de realizar 8 cruceros oceanográficos que abarcaron la mayor parte del Golfo de México.

Los resultados de dichas campañas deberían de ser difundidos de una manera accesible para la comunidad científica abocada a estudios marinos y al sector productivo, mediante la publicación de atlas oceanográficos que incluirían un compendio de las investigaciones del GEO en el Golfo.

Como resultado de estos cruceros se recopilaron, por medio de una sonda hidrográfica (CTD), datos de presión, temperatura, conductividad y oxígeno disuelto. Además, también se cuantifican las concentraciones de otros parámetros químicos disueltos como son: nitratos, silicatos y fosfatos, los cuales se utilizan como trazadores del campo de circulación y sirven para definir la distribución y productividad pesquera de los diversos cuerpos de agua que integran la masa total de agua del Golfo de México. Estos datos son grabados digitalmente en cintas magnéticas, las cuales debido al gigantesco tamaño de la información recabada llegan a sumar entre 10 y 15 cintas por cada crucero.

La organización de esta información, de manera que su procesamiento sea expedito, no es un ejercicio trivial y demanda la estructuración de un sistema de procesamiento que facilite el análisis. Este consta de diferentes pasos, uno de los cuales es llevar un control sobre la calidad de los datos que está registrando el CTD. Para esto, durante los cruceros oceanográficos, se tomaron muestras de agua a profundidades predeterminadas y se les hizo un análisis químico para determinar su concentración de sal y de oxígeno. Estos datos se utilizan para calibrar la información recogida por el CTD. Esta calibración consiste en un análisis de regresión múltiple entre los datos del CTD y los datos obtenidos del análisis químico realizado a las muestras de agua.

Durante los cruceros se capturaron los datos de muestras de agua, así como las posiciones geográficas de las estaciones donde realizaron muestreos de datos. Con toda esta información se formó una base de datos que requirió un tratamiento matemático y estadístico para así poder ser representada en su forma final, es decir, gráficamente.

Por su complejidad se pretende crear un sistema de grandes proporciones (al cual se denominó ***CTDSIS***) que agrupe al conjunto de tareas necesarias para el análisis de los datos hidrográficos. La figura 1.1 presenta el modelo conceptual de este proceso, el cual se resume en las siguientes etapas:

a) ***Recolección de los datos hidrográficos.*** Esta actividad las realizó el GEO durante las campañas oceanográficas en el Golfo de México, almacenado todos los datos recabados en cintas magnéticas.

b) ***Sistema CTDSIS.***- El sistema en el que residen y dentro del cual interactuan los módulos encargados del procesamiento de los datos hidrográficos.

c) ***Módulo de Procesamiento de Datos.*** Este módulo se encarga de obtener los datos de calidad controlada a partir de la información recabada, así como de realizar la captura de archivos de datos indispensables para los siguientes módulos.

d) ***Módulo generador de Datos.*** Este módulo se encarga de consultar y procesar la información (ya calibrada) de la base de datos, creando archivos de datos que el siguiente módulo se encargará de graficar.

e) ***Módulo de representación gráfica.*** Es el responsable de representar gráficamente cada uno de los archivos que el anterior módulo haya generado

f) ***Base de Datos.***- Es el lugar físico donde residirán todos los archivos (de entrada y salida), que CTDSIS utilizará para el proceso.

g) ***Edición de los Atlas.*** Los documentos que se generen de CTDSIS (en forma de textos, tablas o gráficas) conformarán un Atlas Oceanográfico, siendo este el producto final de todo el procedimiento.

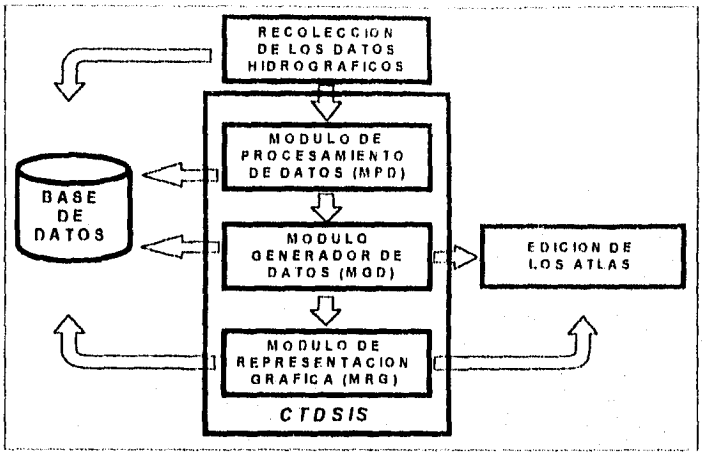


Figura 1.1 Modelo Conceptual del procesamiento de datos oceanográficos.

OBJETIVO DE LA TESIS.

El principal objetivo de esta tesis es precisamente la realización del módulo **MPD**, el cual deberá ser manejado a través de una ***interfaz gráfica***, encargada de obtener los datos hidrográficos de calidad controlada y capturar los archivos necesarios para que **MGD** opere.

LA INTERFAZ GRÁFICA

El desarrollo de interfaces hombre-máquina es fundamental en el desarrollo de cualquier tipo de software (comercial, industrial o de investigación). El funcionamiento de un sistema es juzgado principalmente por su interfaz; la interfaz es el medio de comunicación que hace saber al usuario lo que un sistema pude o no hacer.

Para guiar al diseñador en el desarrollo de interfaces, existen metodologías de desarrollo, modelos conceptuales y herramientas de desarrollo. Sin embargo, seguir o usar ninguno de ellos no garantiza la calidad de las interfaces que se desarrollan con ellos. La calidad de una interfaz es un concepto que depende de factores tales como, facilidad de aprendizaje y efectividad de comunicación con el usuario.

Separación entra la Interfaz y la Aplicación

En la actualidad se nota una separación conceptual y física entre la interfaz y la aplicación que antes no era aparente. Esta separación permite que la investigación sobre áreas relacionadas con el diseño de interfaces se pueda emprender independientemente de la aplicación, con lo que se obtiene una mayor especialización en el diseño de interfaces. el que un sistema fracase o tenga éxito depende en gran medida de la calidad de su interfaz; es por esto que hoy en día el diseño de interfaces es de gran importancia.

La separación no incluye únicamente la agrupación de entradas y salidas en algunos procedimientos o archivos separados del resto del programa. Esto es buena programación, mas no asegura la independencia de la interfaz (diálogo). La independencia del diálogo es generalmente entendida como la separación del sistema computacional en dos componentes:

a) Componente de diálogo, a través del cual toda la comunicación entre el usuario y el sistema toma lugar,
b) Componente computacional, que incluye mecanismos de procesamiento de la aplicación y con el cual el usuario no interactua directamente.

Las ventajas que se buscan con esta separación son:
a) Lograr interfaces de alta calidad. La interfaz no le debe esconder al usuario las funciones de la aplicación
b) Reutilización. Esta se promueve si se disminuye el conocimiento que tiene el usuario de la aplicación interna
c) Adaptar la interfaz a los factores humanos de los usuarios.
d) La comunicación entre la interfaz y el usuario puede ser cambiada de manera arbitraria sin alterar la comunicación entre la interfaz y la aplicación, y viceversa.
e) La interfaz de usuario es fácil de modificar y de mantener
f) El desarrollo de la interfaz y de la aplicación puede realizarse en paralelo.
g) Los cambios de un módulo no afectan necesariamente a otros

Una vez que se acepta la separación entra la interfaz de usuario y la aplicación, las definimos como sigue:

Interfaz de Usuario

Lo que el usuario ve o siente, debe entender y puede (o debe) hacer para obtener los resultados apropiados de un sistema. Las acciones que el usuario puede tomar, apoyado por su memoria y control sobre la serie de pasos requeridos para obtener los resultados, son influidas por los objetos que se le presenten.

Aplicación.

Los componentes de procesamiento que se encargan de realizar operaciones y cálculos sobre la tarea que realiza el sistema. El usuario no interactua directamente con ésta.

ESTRUCTURA DE ESTE TRABAJO DE TESIS.

Este trabajo esta compuesto por una Introducción, cuatro capítulos y cinco apéndices. A continuación se describe el contenido de cada uno.

En la introducción se describen brevemente los orígenes y requerimientos del sistema CTDSIS, desglosando cada una de sus partes y señalando cual será el propósito del subsistema MPD (integrante de CTDSIS). También se habla de los objetivos de las interfaces hombre-máquina y de la importancia de la separación de esta con la aplicación.

En el primer capítulo se realizan el planteamiento de los objetivos, generales y específicos de MPD; seguido del análisis, es aquí donde se define el problema, los antecedentes de MPD, se estudia la factibilidad de los objetivos y se listan cada uno de los requerimientos que MPD deberá poseer.

En el capítulo dos se define la arquitectura del sistema, analizando los conjuntos de información, el diseño de las bases de datos a utilizar y la estructura lógica de estas. Se identifican cada uno de los programas que conformarán a MPD y se analiza su comportamiento dinámico. Se muestran cada uno de los diagramas de estructura del sistema y se realiza una descripción de todos los procedimientos que se involucran en MPD.

En el capítulo tres se explica como es que se llevo a cabo la etapa de desarrollo del sistema, la cual involucra la codificación e integración. También se describe cada una de las secciones que conforman a la interfaz gráfica.

En el cuarto y último capítulo se presentan una serie de casos de prueba que nos ayudan a comprobar el correcto funcionamiento de cada uno de los módulos del sistema y de la interfaz gráfica.

Finalmente, en la conclusiones, se presenta un resumen de los resultados obtenidos, así como un análisis de las experiencias obtenidas al término de la tesis. También se listan una serie de recomendaciones que se exponen para el mejor funcionamiento de MPD.

Acompañan a este trabajo un total de cinco apéndices. El apéndice A muestra el archivo de control que lee y escribe la interfaz gráfica. El apéndice B muestra una introducción al sistema X-Window y a MOTIF. En el apéndice C se listan cada una de los procedimientos externos usados por MPD, En el apéndice D se muestran todas y cada una de las ventanas que conforman la interfaz gráfica. El apéndice E muestra el contenido de los archivos de Catálogo de crucero, de transectos y de nutrientes.

Capítulo 1

Fase de Definición de Requerimientos

No se puede hablar de un desarrollo sistémico de programas o sistemas de cómputo sin una definición precisa de objetivos y sin un planteamiento claro de requerimientos.

Los objetivos deben definir las funciones generales que se esperan del producto final, y en ellos deberá basarse la estructuración del sistema.

Los requerimientos definen con precisión las características de los programas de computadora por desarrollar y establecen los alcances del sistema de cómputo. Sirven de base para el diseño del sistema, de referencia para las pruebas del producto final y de fuente principal de información para utilizar el producto de programación. Los requerimientos permiten, además, controlar la evolución del sistema durante las etapas de su desarrollo.

La fase de "Definición de requerimientos" debe realizarse en tres etapas[1]:

1. Planteamiento de Objetivos.
2. Análisis de los objetivos.
3. Especificación de Requerimientos.

De acuedo con lo anterior, se estructuró este capitulo en tres secciones. En la primera mostraremos cada uno de los objetivos perseguidos por el sistema MPD, en la segunda parte señalaremos cual es la definición del problema, observando algunos antecedentes y estudiando la factibilidad se su solución. Y en la última sección mostraremos cada uno de los requerimientos planteados para el sistema .

[1] Victor Jerez, Mauricio Mier. "Desarrollo y administración de programas de computadora". IIE 1985

1.1 PLANTEAMIENTO DE OBJETIVOS

A continuación nos abocaremos a identificar y describir las necesidades que el GEO presenta, a fin de proponer un conjunto de objetivos, que de lograrse, implicarán la satisfacción de la mismas.

1.1.1 NECESIDADES

Los siguientes objetivos surgen de la necesidad que tiene el GEO de realizar de una manera rápida y confiable los procedimientos de calibración de los datos hidrográficos, tratando de dejar a un lado los tediosos y complicados procesos que esta tarea requiere y que obliga a la persona encargada de este procedimiento a poseer conocimientos específicos y exactos de cada una de los módulos encargados para ello, de los archivos de datos intermedios que se generan y de los archivos de comandos[2] que estos utilizan.

1.1.2 OBJETIVO GENERAL

El objetivo de este proyecto es realizar el rediseño del subsistema MPD, incorporando este a una interfaz gráfica amigable que manipulará el proceso de una forma mas efectiva y confiable, apoyándose siempre de los procedimientos ya existentes para ello.

1.1.3 OBJETIVOS ESPECÍFICOS

1.1.3.1 Dado un conjunto de datos recolectados en una campaña oceanográfica por una sonda hidrográfica (CTD[1]) organizados por lances según el sistema de archivos de Woods Hole Oceanographic Institution (WHOI[4]).

Se deberán realizar las capturas de los datos de laboratorio (salinidad, oxigeno, silicatos, fosfatos y nitratos) que se obtienen de las muestras de agua recopiladas por la sonda para formar el catalogo del crucero y con esto llevar a cabo la calibración de los datos hidrográficos (conductividad y oxigeno) de cada uno de los lances realizados en una campaña especifica.

1.1.3.2 Los datos de entrada de los lances deberán de estar disponibles en formato NBIS (Neil Brown Instruments System[5]) dentro de la estructura de directorios definida por WHOI.

1.1.3.3 Los datos de salida del proceso de calibración deberán de haber pasado satisfactoriamente por la secuencia de procedimientos que para dicha tarea se necesitan y que es establecida por el WHOI. (figura 1.1)

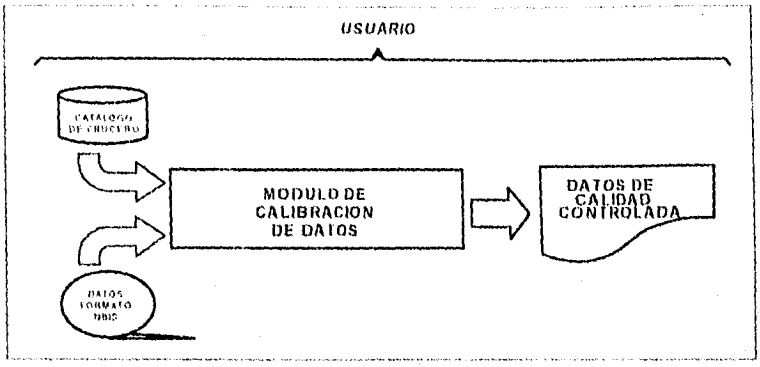
---

[2] Los archivos de comandos son archivos auxiliares que contienen todos los datos necesarios para que las rutinas de calibración se ejecuten. Ver apéndice D.

[3] A esta sonda se le conoce como CTD por sus siglas en inglés **C**onductivity, **T**emperature y **D**epth.

[4] Instituto de Investigaciones Oceanográficas en los Estados Unidos.

[5] Fabricante de los equipos de sondeo de datos hidrográficos.



1.1 Diagrama de Flujo de datos del sistema de calibración.

1.1.3.4 El sistema deberá validar la información de entrada para asegurar que esta sea correcta, de acuerdo a las condiciones del objetivo 1.1.3.1

1.1.3.5 El sistema deberá automatizar el proceso de calibración por medio de la actualización y el desarrollo de nuevos módulos, en un procesador 3100 VAX STATION.

1.1.3.6 La interacción con el usuario se deberá llevar a cabo mediante una interfaz gráfica (GUI[6]), que aproveche las capacidades de alta resolución de la estación de trabajo.

1.2 ANÁLISIS DE LOS OBJETIVOS

1.2.1 DEFINICIÓN DEL PROBLEMA

El problema puede formularse de la siguiente manera:

Actualmente el procedimiento de calibración de datos hidrográficos que requiere realizar el Grupo de Estudios Oceanográficos para la correcta interpretación de estos se realiza mediante un procedimiento cuya confiabilidad , así como la forma de ejecutarlo no representan precisamente una metodología segura ni amigable, ya que todos los submódulos que componen a este sistema se ejecutan a través de *archivos de comandos,* que contienen los datos de entrada necesarios para la correcta ejecución de cada submódulo.

[6] Graphical User Interface.

El principal inconveniente al utilizar este tipo de archivos es que para su actualización es necesario modificarlos en un editor de texto, realizar los cambios pertinentes "a mano" y salvarlos con el mismo o con un nuevo nombre. Todo lo anterior se refleja, por ejemplo, en la forma en que el usuario debe cambiar cada uno de los nombres de los archivos de salida o de entrada tecleándolos directamente, checar que la dirección física en disco donde se requiere que residan los archivos sea la correcta, corroborar que los datos fueron tecleados en un formato correcto, etc. lo que es , al fin y al cabo como ya lo hemos dicho, un procedimiento poco confiable.

A continuación se presenta un ejemplo de un archivo de comandos utilizado para un programa del Submódulo de Calibración de Conductividad. Como se puede observar es prácticamente imposible, si no se tienen los conocimientos lo suficientemente detallados de lo que cada línea significa, de describir el funcionamiento del programa que se ejecutará con este archivo.

```
MLTLDRS852.COM
$! calibracion de la conductividad
$! para el crucero argos852
$! realizado en JUNIO - julio de 1985
$! A bordo del B/O Justo Sierra
$!
$! WITH CTD#1 CALS. POST JS085
$!VERSION OF 11 MAY 1983
$SET VERIFY
$SHOW STATUS
$ASSIGN [etd.js085d002.smcc]mltldrs.out FOR006
$RUN MLTRG2d
19/ SET DEFAULTS
5/ SET ISSW ARRAY
6,0/ NO PLOT YET
7,,,,,,,,/ENTER EDIT BR
1,4000,4000/ EDIT PRS de 0 a 8000
22,,27,1/ READ .MLT FILE          ??????CHANGE FILE SPEC NEXT LINE
js852.mlt
12,7,8/ SWAP STA # & OX TMP
12,6,8/ SWAP OT & OXYGEN COL 8 & 6
YES DELETE SALTS OUT OF RANGE
20/
1,1,20.0,0,0/ CONVERT PRS TO DIG. UNITS COL 1
11,1/ RESCALE COL 1 TO CUBIC PRES
0.0,.05,0.0,0.0/ POST JS085 UP CALS senson para 6000 db
20/
2,2,2000,,0.0/ CONVERT TMP TO DIGITIZER COL 2
11,2/ RESCALE COL 2 (TMP) TO QUAD.
0.0,.0005,0.0,0.0/ POST JS085 TMP CALS. ??????????
14/ CREATE KCOMM.MLT
js852S.MLT
10,6,1,30/
18/ EXIT
$! pu [etd.js085d002.smcc]mltldrs.out
$! EXIT
```

Figura 1.2. Archivo de Comandos para el procedimiento de calibración de conductividad

Así como el ejemplo anterior, existen aproximadamente otras 20 archivos de comandos para cada uno de los programas que componen al sistema de calibración de datos.

I.2.2 ANTECEDENTES DE CADA UNO DE LOS MÓDULOS DEL MPD

a) Captura del Catálogo del Crucero .- Actualmente existe un programa llamado DATEST[3] que se encarga de realizar la captura del archivo de Catálogo de Crucero. Este es un programa poco flexible, ya que, una vez introducidos los datos, es imposible regresar a modificarlos o siquiera consultarlos. Cualquier modificación o consulta del archivo de Catálogo de Crucero hay que realizarla mediante un editor de textos, lo cual expone la integridad del archivo.

b) Archivos de muestras de agua.- Este módulo se ejecuta actualmente utilizando las rutinas del M.C. José M. Pérez Molero [3] programadas en lenguaje FORTRAN. El único inconveniente es que al correrse estas rutinas el usuario no sabe a ciencia cierta si el proceso se esta ejecutando o no; es decir no tienen disponible una interacción para con el usuario.

c) Captura del Archivo de Transectos.- No existe actualmente un programa que realice la tarea de capturar el archivo de transectos confiablemente, es decir, los archivos que en este momento están disponibles fueron capturados mediante un editor de texto, herramienta que también se utiliza en caso de querer darles mantenimiento. No existe un procedimiento de validación de las estaciones capturadas para cada transecto, por lo tanto cabe la posibilidad de incorporar identificadores de estaciones que probablemente ni siquiera existan.

d) Captura del Archivo de Nutrientes .- Actualmente los programas de captura de nutrientes no son capaces de darles mantenimiento a los archivos que generan, ni de validar que el número de niveles que muestra cada estación dentro del archivo de nutrientes sea el mismo que en el archivo de Catálogo de Crucero. Este último punto es importante, ya que el buen procesamiento de los datos de nutrientes depende precisamente de dicha validación.

e) Calibración de Datos Hidrográficos .- Actualmente el GEO cuenta con una serie de librerías encargadas de la calibración de los datos hidrográficos que se adquirieron de WHOI [2]. Estas librerías fueron programadas en lenguaje FORTRAN con una estructura de programación que obliga al usuario a ejecutarlas en una forma rígida y con una metodología muy específica. La interacción con estas rutinas es a base de peticiones de ramas, las cuales son porciones de código que realizan una acción determinada. El usuario debe insertar el identificador de la rama que se quiere ejecutar, para después incorporar los datos o valores con los cuales se quiere trabajar; en este momento el programa se encuentra en un ciclo de pregunta_respuesta que se rompe hasta cuando se solicita la rama de salida.

El ciclo principal de este tipo de programas viene determinado de la siguiente manera:

```
10    lee número_rama
20    Si  número_rama = 1 entonces ve a 100
Si  número_rama = 2 entonces ve a 200
Si  número_rama = 3 entonces ve a 300
100 Procesa algo,  ve a 10
200   Procesa algo,  ve a 10
300   Salida
```

Los archivos de datos de entrada o de salida que requieran los programas para ejecutarse deben ser asignados a una unidad lógica [4] desde el sistema operativo: *$ASSIGN [ctd.js085d002.snucc]mltldrs.out FOR006* Se asigna el archivo mltldrs.out a la unidad lógica 6
El programa de calibración deberá de hacer mención a esta unidad lógica:
write(6,);* Se manda a escribir en la unidad lógica 6.

Como resultado del análisis del sistema de calibración de datos hidrográficos con que cuenta actualmente el GEO, se encontró que su principal desventaja surge de la escasa documentación y del aislamiento existente entre sus diferentes etapas. Es por esto que para alcanzar los objetivos acordados, se realizará lo siguiente:

a) Construir nuestra interfaz de usuario bajo una técnica que permita una total independencia entre esta y las aplicaciones, pero que al mismo tiempo administre la interacción de estas aplicaciones.

b) Aprovechar los procedimiento ya existentes para el sistema de calibración tradicional e incrementar su documentación.

c) Incorporar procedimientos de validación de datos

1.2.3 FACTIBILIDAD DE OBJETIVOS

La factibilidad de todos los objetivos mencionados con anterioridad esta dada y tal aprobación se apoya tanto en términos técnicos como en términos operativos.

En cuanto a la *factibilidad técnica* se tiene que :

a) Existen herramientas suficientes e importantes para el desarrollo de programación, tanto para la interfaz como para los programas de aplicación tales como:

- Lenguaje de programación "C" para VAX
- Lenguaje de programación FORTRAN para VAX

- Herramientas para el desarrollo de interfaces gráficas DECWINDOWS, MOTIF.
- Emulador de una X-Terminal para equipo PC.

b) Se tiene disponible equipo de cómputo con terminales gráficas que permiten una perfecta Implementación del sistema a desarrollar.

- Vaxstation 3100.
- Equipo PC-AT con emulador X-Terminal

c) Se cuenta con la asesoría necesaria y suficiente de expertos en las ramas de desarrollo de software y en el manejo de los datos hidrográficos.

d) Existe bibliografía suficiente, es decir, artículos, informes y manuales relacionados con los procedimientos y herramientas a utilizar en el desarrollo.

Por otro lado se tiene que la *factibilidad operativa*, que se refiere al pronóstico de que el sistema se llegará a usar una vez instalado, esta también garantizada, ya que existe el interés real de los usuarios del GEO de contar con un sistema que sea mas eficaz y que cuente con mejores accesos.

1.2.4 MEDIOS PARA LOGRAR LOS OBJETIVOS

Para alcanzar los objetivos de usuario, se propone lo siguiente:

a) Diseñar la estructura del sistema de tal manera que su confiabilidad y robustez sean primordiales.

b) Implementación de la programación necesaria en lenguajes de vanguardia y con una vida útil significante.

1.3 ESPECIFICACIÓN DE REQUERIMIENTOS DEL SISTEMA

1.3.1 INTRODUCCIÓN

En esta sección estableceremos las características que el sistema de captura y calibración de parámetros hidrográficos deberá poseer y la descripción de los aspectos, las tareas o funciones que este deberá realizar.

I.3.2 IDENTIFICACIÓN Y PROPÓSITO

Este documento establece los requerimientos del sistema de captura y calibración de datos hidrográficos recolectados por una sonda hidrográfica, y sirve de medio de comunicación entre el usuario (GEO), y el analista y diseñador del sistema (su servidor). Además servirá de base para :

a) La arquitectura
b) El plan de pruebas

La descripción del sistema se presenta en dos partes. La primera consiste en la definición conceptual que ofrece un panorama general del sistema. La segunda parte presenta formalmente los requerimientos del sistema.

Se utiliza una sintaxis definida para identificar los requerimientos con el objeto de facilitar el mantenimiento de este documento, permitiendo rastrear los cambios que se registren durante el desarrollo del sistema.

Cada requerimiento es precedido de una estructura con la siguiente sintaxis:

REQUERIMIENTO X.Y:NOMBRE

Donde *X* indica el número secuencial del requerimiento; *Y* tendrá el valor inicial de "0" y se incrementará, en caso de ser necesario, con la inserción de los requerimientos adicionales relacionados conceptualmente con el requerimiento *X*, y *NOMBRE* será el nombre específico que se asigne a ese procedimiento.

I.3.3 DEFINICIÓN CONCEPTUAL DEL SISTEMA

I.3.3.1 DEFINICIÓN DEL SISTEMA MPD, SU FRONTERA Y SU ENTORNO

El sistema MPD es una herramienta que calibra los datos recopilados por el sensor de conductividad y del oxígeno disuelto, mediante procesos de regresión múltiple[7] (Millard 1982) para determinar los coeficientes que ajustan los datos obtenidos del CTD (datos almacenados en cintas magnéticas) con los valores de salinidad y oxígeno disuelto de las muestras de agua tomadas (Catálogo del Crucero). Para posteriormente formar los archivos de datos calibrados para cada estación.

[7] Los detalles completos de estos procedimientos se encuentran en [1]

También nos permite realizar la captura, tanto del Catálogo del Crucero, como de otros archivos que son utilizados por CTDSIS.

Conceptualmente el sistema MPD es integrado por las entidades de la figura 1.3.3.1, estas son:

a) La entidad *ARCHIVOS CTD* se refiere a los archivos cuyos datos ya fueron ajustados, es decir, aquí residen los datos calibrados.

b) *DATOS NBIS* representa los datos (en formato NBIS), que fueron recolectados por el CTD.

c) *ARCHIVOS DATOS* se refieren a los archivos, del catálogo del crucero, de transectos y de nutrientes.

d) *CONTROL DE CALIDAD* representa la entidad que, a partir de los archivos en código NBIS y del Catálogo del Crucero, obtiene los datos de calidad controlada.

e) *CAPTURA DE ARCHIVOS* es la entidad que genera y lee los archivos de datos del Catálogo del Crucero, de Transectos y de Nutrientes.

f) *INTERFAZ DE USUARIO* se refiere al medio de comunicación entre las entidades de proceso y el usuario

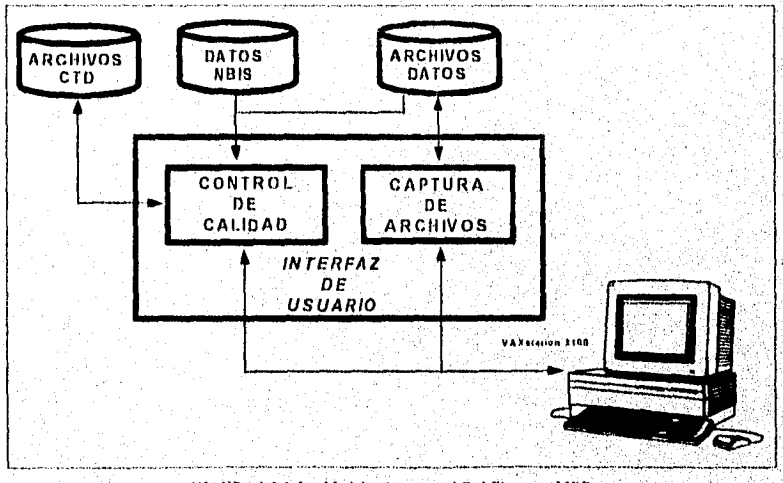


FIGURA 1.3.3.1 Modelo Conceptual Del Sistema MPD

1.3.3.2 RESEÑA DE LAS TAREAS DEL SISTEMA MPD

El sistema MPD realiza 8 funciones principales que son:

1. *Captura del archivo de Catálogo del Crucero [3]* .
   Se manipula un archivo de datos que consta de la posición geográfica donde se realizaron las estaciones; profundidades; valores de presión, temperatura, corriente en el sensor de oxigeno .etc. Permitiendo realizar operaciones de altas, bajas, modificaciones y consultas.

2. *Captura del archivo de Nutrientes* .
   Se maneja un Archivo de datos que contiene todos los parámetros de Silicatos, Fosfatos y Nitratos, recolectados durante una campaña oceanográfica. Permitiendo realizar operaciones de altas, bajas, modificaciones y consultas.

3. *Captura del archivo de Transectos*
   Manipula un archivo de datos con información de todos los transectos pertenecientes a una determinada campaña oceanográfica y de las estaciones correspondientes a cada transecto. Permitiendo realizar operaciones de altas, bajas, modificaciones y consultas.

4. *Creación de archivos de muestras de agua [3].*
   Se extraen del Archivo de Catálogo del Crucero los valores de salinidad y de oxigeno que se requieren para la calibración de datos del CTD.

5. *Transcripción de datos.*
   Se transcriben de los datos del CTD del formato en que son grabados por la Unidad Abordo de la sonda CTD (NBIS) al formato CTDVAX [1][2].

6. *Calibración de la celda de conductividad[1][2].*
   Corrige la conductividad del CTD por la deformación que sufre el sensor por la temperatura y la presión, para obtener después los datos de conductividad de las muestras de agua que se ajustarán mediante un polinomio de primer grado con los parámetros de conductividad del CTD.

7. *Obtención de los datos del CTD como series ordenadas de presión[1][2].*
   Se genera una serie de datos uniformes de temperatura, salinidad y oxigeno a intervalos de dos decibarios de presión. Toma los archivos en formato CTDVAX y genera archivos por estación oceanográfica en los cuales la presión a sido promediada y sorteada.

8. *Calibración del sensor de oxigeno[1][2].*
   Extrae del archivo de Catálogo de Crucero los datos de muestras de agua, y de los archivos extensión .ctd código M los valores de las observaciones del sensor CTD,

Se obtienen parámetros de ajuste que se aplican a los perfiles del CTD (archivos código M) generando archivos extensión .ctd código X[8].

El modelo conceptual de las tareas del MPD se presentan en la figura 1.3.3.2.

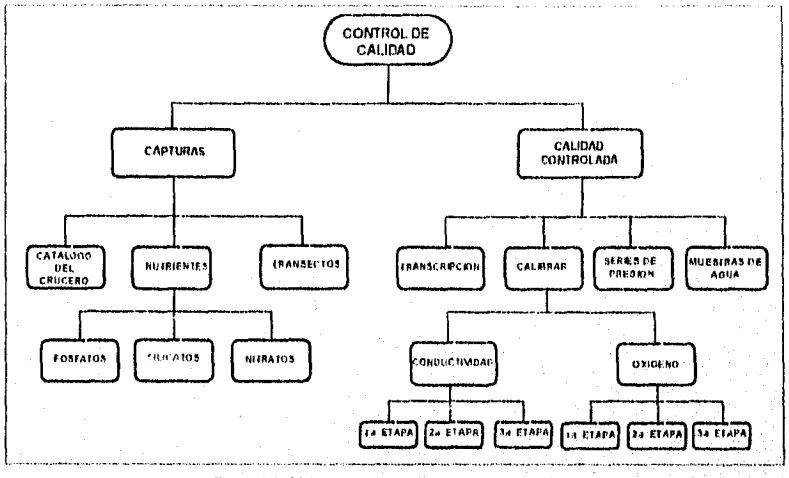


Figura I.3. Modelo Conceptual de las tareas del MPD

El modelo conceptual de los procesos de captura de archivos se muestra en la figura 1.4, y cada uno de sus elementos se describen a continuación.

a)*Bitácora De Abordo Datos De Niveles* : Son las planillas de los datos que registro el CTD en el momento de detenerse a muestrear.
b)*Datos Del Laboratorio Químico:* Son los datos de oxígeno y de salinidad calculados de las muestras de agua.
c)*Bitácora de Datos de Nutrientes:* Planillas con datos de absorbancias y demás parámetros necesarios para determinar concentraciones de nutrientes.
d)*Datos de Transectos* :Planillas con los identificadores de las estaciones pertenecientes a cada transecto.
e)*GEO Usuario* : Personal del GEO que se encargará de capturar los parámetros de los anteriores documentos.
f)*MPD(Captura de archivos):*Módulo de procesamiento de datos encargado de generar los archivos y extraer información de estos.
g)*Archivo de Catálogo del Crucero:*Archivo de datos que contiene valores de niveles, fechas, posiciones y parámetros químicos que fueron muestreados por el CTD.

[8] El formato de los archivos de datos de calidad controlada se encuentran disponibles en [3]

h)*Archivo de Nutrientes:* Archivo que contiene datos de concentraciones de Nitratos, Fosfatos y Silicatos.

i)*Archivo de Transectos:* Posee los datos de cada uno de los transectos de algún crucero en particular.

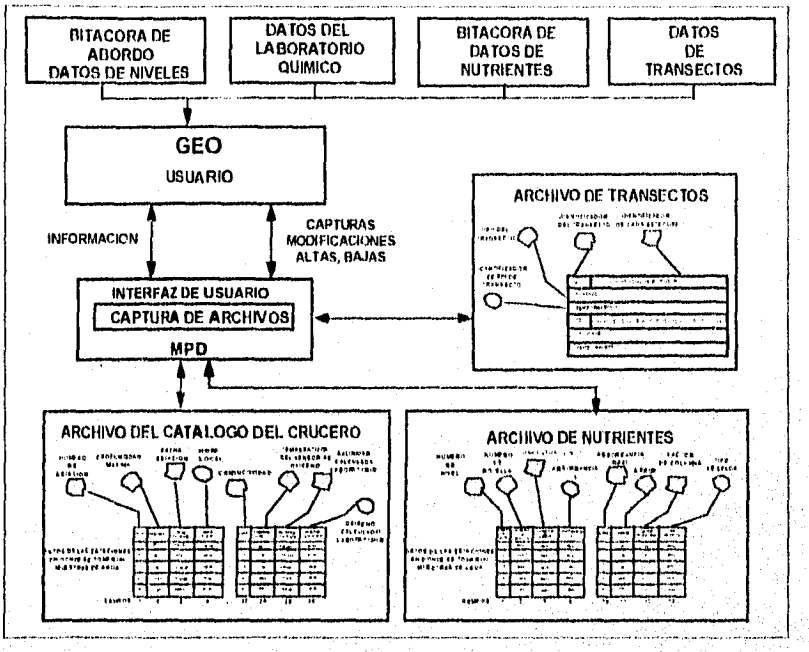


figura 1.4 Modelo conceptual de capturas de archivos de datos.

1.3.4 REQUERIMIENTOS DEL SISTEMA MPD

Se presentan a continuación los requerimientos del sistema MPD

1.3.4.1 *Requerimiento 1.0 ACTIVACIÓN.*

El sistema será accesado y activado únicamente dentro de otro sistema global, llamado CTDSIS. Nuestro sistema deberá ejecutarse a través de un objeto definido dentro del menú principal de CTDSIS y solo desde ahí podrán ser accesados todas las opciones que el MPD ofrezca (Altas, bajas, Modificaciones, Consultas y Calidad).

Los modos de operaciones (altas, bajas, modificaciones y consultas) de los programas de captura de archivos de Catálogo, de transectos y de nutrientes, serán activados dependiendo de la opción elegida en el menú principal de CTDSIS.

En cuanto a los nombres de los archivos que el sistema MPD utilice, deberán ser determinados automáticamente por los datos iniciales de entrada que se le suministren al sistema CTDSIS, estos datos de identificación son:

a) Nombre del Barco
b) Código del Crucero
c) Versión de Datos
d) Código del Proyecto

1.3.4.2 *Requerimiento 2.0 ALMACENAMIENTO DE DATOS.*

Todos los archivos que maneja el sistema MPD, tanto de entrada como de salida de datos, residirán en un sistema de directorios definido por la WHOI.

1.3.4.3 *Requerimiento 3.0 VALIDACIÓN DE DATOS.*

El sistema deberá validar la información de entrada, de tal modo que solo acepte datos del tipo requerido y dentro de los rangos específicos.

Cuando se haga referencia a identificadores de estaciones, se verificará que dicha estación exista en el archivo de Catálogo de Crucero.

En caso de presentarse una acción no válida, como cualquiera de las anteriores, la interfaz presentará un objeto que avise de la acción no permitida.

1.3.4.4 *Requerimiento 4.0 CATALOGO DEL CRUCERO.*

1.3.4.4.1 *Requerimiento 4.1 PROGRAMA DEL CATALOGO DEL CRUCERO.*

a) El programa deberá ser capaz de leer el archivo del Catálogo del Crucero en su formato original, es decir, en el formato que generaban los anteriores programas.
b) Se deberán realizar cuatro operaciones sobre el archivo de catálogo, que son: Altas, Bajas, Consultas y Modificaciones de estaciones.
c) En caso de que se realice una operación de "alta" de alguna estación hidrográfica, esta deberá ser incluida al final del archivo de Catálogo del crucero.

1.3.4.4.2 *Requerimiento 4.2 INTERFAZ DEL CATALOGO DEL CRUCERO.*

a) Se tratará a cada dato del catálogo del crucero como un objeto independiente dentro de la interfaz para su mejor visualización, validación y manejo.
b) La interfaz tendrá la capacidad de mostrar de una sola vez en pantalla todos los datos pertenecientes a una estación

hidrográfica determinada (elegida previamente), diferenciando los datos de cabecera de los datos propios de los niveles. Los datos de niveles se mostrarán sobre un objeto con apariencia de hoja de cálculo, con un numero de columnas fijo (determinado por el número de datos para cada nivel) y un total de registros igual al número de niveles de la estación.

c) Se requieren también de objetos individuales para la captura de datos específicos, como fechas, horas, latitudes, longitudes y comentarios.

d) Se debe tener los objetos para realizar operaciones de: Inserción, Borrado, Modificación y Movimiento de los registros que se muestran en la ventana.

1.3.4.5 *Requerimiento 5.0 NUTRIENTES.*

1.3.4.5.1 *Requerimiento 5.1 PROGRAMA DE NUTRIENTES*

a) El programa será capaz de leer el archivo de Nutrientes en su formato original, es decir, en el formato que generaban los anteriores programas.

b) El programa deberá tener la capacidad de realizar cuatro operaciones sobre el archivo de nutrientes, que son: Altas, Bajas, Consultas y Modificaciones de Nutrientes.

c) El programa mantendrá una consulta continua entre el archivo de Nutrientes y el archivo de Catálogo de crucero, ya que la captura de nutrientes deberá mostrar datos de cabecera (posiciones, horas, fechas, etc.) que son propios de este último archivo.

d) Aunque los datos de Nutrientes (Silicatos, Fosfatos y Nitratos) se capturan en un mismo archivo, se deberá elaborar un procedimiento individual de manejo para cada uno de ellos.

e) Existirá un procedimiento que forme archivos individuales de nutrientes por estación hidrográfica.

f) Para la determinación de la concentración de algún parámetro de nutriente, se utilizarán los algoritmos que para dicho fin ya existen.

1.3.4.5.2 *Requerimiento 5.2 INTERFAZ DE NUTRIENTES.*

a) Se tratará a cada dato del archivo de nutrientes como un objeto independiente dentro de la interfaz para su mejor visualización, validación y manejo.

b) Se construirá una interfaz de captura para cada tipo de nutriente (Silicato, Fosfato y Nitrato); estas deberán ser activadas desde el menú de inicio de CTDSIS.

c) La interfaz tendrá la capacidad de mostrar de una sola vez en pantalla todos los datos pertenecientes a una estación hidrográfica determinada (elegida previamente), diferenciando los datos de cabecera (horas, fechas, posiciones) de los datos propios de los nutrientes. Los datos de nutrientes deberán estar

montados sobre un objeto con apariencia de hoja de calculo, con un número de columnas fijo (uno por cada dato de nutriente que contiene el nivel) y un numero de registros igual al total de niveles de la estación.

d) La captura o modificación de datos de nutrientes se realizara a través de un objeto independiente que debe ser diseñado de tal manera que facilite la inserción y validación de datos.

e) Dependiendo del modo de operación (altas, baja, modificaciones o consultas) con el que se vaya a trabajar y mediante la habilitación o desabilitación de objetos, se deberá limitar al usuario a realizar solamente dicha operación.

Se definirá un objeto para ejecutar la formación de los archivos individuales por estación de nutrientes, y de otro objeto que indique que el proceso se está realizando.

1.3.4.6 *Requerimiento 6.0 TRANSECTOS.*

1.3.4.6.1 *Requerimiento 6.1 PROGRAMA DE TRANSECTOS.*

a) El programa será capaz de leer el archivo de Transectos en su formato original, .

b) El programa deberá tener la capacidad de realizar cuatro operaciones sobre el archivo de transectos, que son: Altas, Bajas, Consultas y Modificaciones de transectos; cada uno de estos modos de operación será activado dependiendo de la opción elegida en el menú principal de CTDSIS.

c) Las operaciones de altas y modificaciones de transectos deberán trabajar y hacer referencia única y exclusivamente a estaciones hidrográficas que existan en el correspondiente Catálogo de Crucero.

d) En caso de que se realice una operación de "Alta" de transecto, este deberá ser incluido al final del archivo de Transectos.

1.3.4.6.2 *Requerimiento 6.2 INTERFAZ DE TRANSECTOS.*

a) Dependiendo del modo de operación (altas, baja, modificaciones o consultas) con el que se vaya a trabajar y mediante la habilitación o desabilitación de objetos, se limitará al usuario para realizar solamente dicha acción.

b) La interfaz deberá contar con un objeto que sea capaz de desplegar todos los identificadores de los transectos para ese archivo, así como tener la capacidad de poder elegir el transecto con el que deseemos trabajar.

c) Existirá otro objeto que desplegará todos los números de estaciones pertenecientes al transecto elegido.

d) La captura o modificación de las estaciones para cada transecto, así como de los atributos correspondientes se realizará mediante un objeto independiente, que manipulará en forma individual cada uno de los datos o atributos.

1.3.4.7 *Requerimiento 7.0 MUESTRAS DE AGUA.*

a) Este programa utilizará las rutinas ya existentes (en FORTRAN) que generan los archivos de muestras de agua.

1.3.4.8 *Requerimiento 8.0 CALIBRACIÒN.*

1.3.4.8.1 *Requerimiento 8.1 PROGRAMAS DE CALIBRACIÒN..*

a) Se deberán emplear las rutinas de calibración con las que actualmente cuenta que el Grupo de Estudios Oceanográficos, modificándolas en forma mínima con el fin de poder asignar, desde estructuras en código "C" y mediante variables tipo *'common'* en lenguaje Fortran, los nombres de los archivos de datos de entrada y de salida necesarios para su ejecución.
b) Se podrán leer los archivos de comandos que se encuentren ya disponibles.
c) Una vez ejecutados los procedimientos se crearán los archivos de comandos necesarios para que la rutina de calibración en turno pueda ser procesada.
d) El archivo de comandos recién generado se borrará del espacio en disco inmediatamente después de haberse ejecutado la rutina de calibración.

1.3.4.8.2 *Requerimiento 8.2 INTERFAZ DE CALIBRACIÓN.*

a) Las interfaces serán diseñadas de tal forma que faciliten en forma importante la entrada de datos que cada rutina de calibración necesita, empleando para esto términos y títulos claros y precisos que no confundan en ningún momento al usuario.
b) Deberá de existir una agrupamiento dentro de la interfaz de los objetos que hagan referencia a un procedimiento o rama en especial de la rutina de calibración. Este grupo de objetos deberá de tener un título en particular y estará enmarcado para una mejor identificación.
c) Existen algunas ramas en las rutinas de calibración a las que se hace referencia mas de una ocasión para la ejecución de algún proceso, para este tipo de casos se establecerá una objeto que permita la visualización individual de los datos de las distintas referencias de las ramas.

d) Se designará un objeto que tendrá asociado un procedimiento que guardará los datos de la interfaz a las estructuras de datos de la misma.

1.3.4.9 *Requerimiento 9.0 ARCHIVO DE CONTROL.*

El sistema guardará en un archivo de control de interfaz todos los datos correspondientes de entradas, salidas, cambios, consultas y procesamientos de una campaña oceanográfica especifica. Cualquier recurso que sea invocado de la interfaz leerá sus datos de este archivo, así como cualquier dato de recurso que sea aceptado residirá también en este archivo.

1.3.4.10 *Requerimiento 10.0 LISTAS LIGADAS*

Los programas de capturas de archivos de datos (catálogo de crucero, nutrientes y transectos) emplearán estructuras de listas doblemente ligadas. Lo anterior obedece a que el uso de este tipo de estructuras facilitará las operaciones de inserción, borrado y modificación de datos.

Además se mejorarán los tiempos de respuesta del sistema, pues como se sabe, el acceso a memoria es mucho mas rápido, que el de un archivo grabado en disco.

Se generará una copia del archivo físico en memoria residente, formando lo que llamaremos lista principal , una vez disponible esta lista principal, se podrán extraer de ella los datos de alguna estación hidrográfica disponible, estos datos formará lo que llamaremos una lista de datos de estación, es con esta lista con la que se trabajará para realizar las capturas y modificaciones pertinentes.

Una vez que se haya procesado la lista de datos de estación, esta regresará a la lista principal, en el mismo lugar de donde fue extraida. A su vez la lista principal, una vez que termine su manejo, será copiada a un archivo físico dentro de la vaxstation (figura 1.3.4.10).

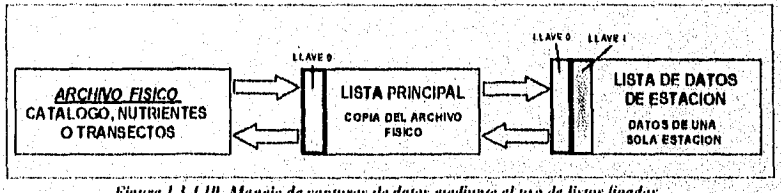


Figura 1.3.4.10 Manejo de capturas de datos mediante el uso de listas ligadas.

1.3.4.11 *Requerimiento 11.0 ENTORNO DE DESARROLLO DE MPD*

El sistema se desarrollará con las siguientes caracteristicas :

a) Se construirá sobre la plataforma de X-Window
b) El sistema deberá contemplar una separación entre la interfaz y la aplicación
c) Las aplicaciones se realizarán en lenguaje de programación "C"
d) Se utilizarán las librerías de desarrollo de interfaces de MOTIF apoyándose de las funciones XI-Intrinsics

e) Se programará la interfaz empleando funciones de alto nivel (XUI)
f) Se empleará una metodología específica para el nombramiento de cada objeto que forme parte de la interfaz

1.3.4.12 *Requerimiento 12.0 ARCHIVO DE COMANDOS*

Todos los programas referentes a la obtención de los datos de calidad controlada poseerá un procedimiento que generará su correspondiente "archivo de comandos", este archivo deberá tener siempre el mismo nombre, "*cld_mpd_camandos.com*" y solamente existirá una versión del mismo.

Capítulo 2

Diseño del Sistema.

El diseño es el segundo paso de la fase de desarrollo de cualquier producto o sistema de ingeniería. Puede definirse como : ".... el proceso de aplicar distintas técnicas y principios con el propósito de definir un dispositivo, proceso o sistema con los suficientes detalles como para permitir su realización física" [6].

Esta Capítulo se divide en tres secciones:

1. Arquitectura.
2. Diagramas de estructura.
3. Detalle de Módulos.

2.1 ARQUITECTURA DEL SISTEMA.

El lineamiento fundamental del diseño de programas de computadora recomienda subdividir los requerimientos en partes que resulten mas fáciles de manejar y de entender. Identificar las principales funciones que se encuentran implícitas en los requerimientos a fin de asociarlas a programas de computadora, es el objetivo principal de la etapa de "Arquitectura".

La arquitectura deberá incluir la definición de:

1. Los conjuntos de Información.
2. Los procesos o programas a desarrollar.
3. El comportamiento dinámico.
4. Las interacciones organizacionales.

2.1.1 CONJUNTOS DE INFORMACIÓN.

Los conjuntos de información se clasifican en tres tipos:

2.1.1.1 INFORMACIÓN DE ENTRADA.

En relación a las capturas de archivos (catálogo de crucero, nutrientes y transectos) el MPD requiere como datos de entrada, la información recopilada dentro de los siguientes documentos[1]:

a) Bitácora de abordo(datos de niveles).
b) Datos del Laboratorio químico.
c) Bitácora de datos de nutrientes.
d) Datos de transectos
e) Archivo de control

[1] El contenido de cada documento se explica en 1.3.3.2.

Para la obtención de los datos de calidad controlada, el MPD necesita que los archivos en formato NBIS, que son generados por la UACTD[2], se encuentren disponibles en los discos de la microvax II.

2.1.1.2 INFORMACIÓN DE SALIDA

El MPD generará como datos de salida los siguientes archivos:

a) Archivo del Catálogo del Crucero.
b) Archivo de nutrientes (individuales y generales)
c) Archivos de Transectos
d) Archivos de índices de estaciones.
e) Archivos de datos de calidad controlada (archivos CTD).
f) Archivo de control

2.1.1.3 BASES DE DATOS

2.1.1.3.1 DISEÑO CONCEPTUAL DE LA BASE DE DATOS:

Para cada uno de los archivos de salida listados arriba se establecen los siguientes datos:

a) *Archivo del Catálogo del Crucero.*

Número de la estación hidrográfica., profundidad máxima de la estación, fecha de la estación, hora local, hora DRT, hora GMT (línea 2), latitud (Línea 2), longitud (Línea 2), hora GMT (Línea 3), latitud (Línea 3), longitud (Línea 3), número de botella, profundidad del nivel de muestreo, presión (decibarios), temperatura del agua, conductividad, salinidad, valor de Sigma Teta, oxígeno disuelto, temperatura del sensor de oxígeno, corriente del sensor de oxígeno, número de botella de salinidad, número de botella de oxígeno, número de botellas de nutrientes, 3 números de botella de alcalinidad, salinidad calculada (Laboratorio), oxígeno calculado (Laboratorio).

b) *Archivo de Nutrientes.*

Número de estación hidrográfica, número de botella, Concentración de nutriente, absorbancia 1, absorbancia 2, absorbancia 3, absorbancia promedio, salinidad (laboratorio), absorbancia corregida, absorbancia real, error, factor de columna, tipo de celda.

c) *Archivo de Transectos.*

Identificación del transecto, número de estaciones para el transecto, identificador de cada una de las estaciones, tipo del transecto, dirección de transecto, posición del transecto.

[2] Unidad de Abordo del CTD.

ZAMAMY - GEO - JJE

2.1.1.3.2 DISEÑO FÍSICO DE LA BASE DE DATOS:

1. *Archivo de Catálogo del Crucero.*

El formato de grabación del archivo del Catálogo del Crucero deberá ser igual al que se generaba con los procedimientos anteriores. El formato se compone de:

-Dos renglones de datos de cabecera:

Campo	1	2	3	4	5
Nombre	Id de inicio de estación	Número de estación	Profundidad Máxima de la estación	Fecha de la estación	Hora Local
Formato	*	nnn	nnnn.n	nn/cc/nn	nn:nn

Campo	1	2	3	4	5	6	7
Nombre	Hora DRT	Hora GMT Línea 2	Latitud Línea 2	Longitud Línea 2	Hora GMT Línea 3	Latitud Línea 3	Longitud Línea 3
Formato	nn:nn	nn:nn	nn.nn.n	nn.nn.n	nn:nn	nn.nn.n	nn.nn.n

-Un registro de datos de nivel por cada nivel que contenga la estación , con formato:

Campo	1	2	3	4	5	6
Nombre	Número de Botella	Profundidad del nivel	Presión (decibares)	Temperatura del Agua	Conductividad	Salinidad
Formato	nnn	nnnn.n	nnnn.n	nn.nnn	nn.nnn	nn.nnn

Campo	7	8	9	10	11	12
Nombre	Sigma T.	Oxigeno Disuelto	Temperatura del sensor de Oxigeno	Corriente del sensor de Oxigeno	# de botella Salinidad	# de botella Oxigeno
Formato	nn.nnn	nn.nnn	nn.nnn	nn.nnn	nnn	nnn

Campo	13	14	15	16	17
Nombre	# de botella Nutrientes	# de botella Nutrientes	# de botella Alcalinidad	# de botella Alcalinidad	# de botella Alcalinidad
Formato	nnn	nnn	nnn	nnn	nnn

Campo	18	19
Nombre	Salinidad Calculada	Oxigeno Calculado
Formato	nn.nnn	nn.nnn

donde "*n*" es un carácter numérico, y "c" es un caracter alfanumérico

2. *Archivo de nutrientes.*

El formato de grabación del archivo de Nutrientes que se generará con los nuevos programas será distinto al que originalmente se grababa. Este archivo esta constituido por:
-Una linea de cabecera del tipo **Nnnn*. Donde:

*.- Es un identificador de inicio de datos de nutriente para una estación determinada.
N.- Es el tipo de nutriente que se este capturando. (Silicatos "S", Fosfatos "F", Nitratos "N")
nnn.- Es el número de estación.

-Un registro de datos por cada nivel que constituya a una determinada estación. Su formato es:

Campo	1	2	3	4	5	6	7
Nombre	Número de Nivel	Número de Botella	Concentración de Nutriente	Absorbancia 1	Absorbancia 2	Absorb. 3	Absorbancia Promedio
Formato	nn	nn	nn.nnn	n.nnnn	n.nnnn	n.nnnn	n.nnnn

Campo	8	9	10	11	12	13
Nombre	Salinidad de Laboratorio	Absorbancia Corregida	Absorbancia Real	Error	Factor de Columna	Tipo de Celda
Formato	nn.nnn	n.nnnn	n.nnnn	n.nnnn	n.nnnn	nn

3. *Archivo de Transectos.*

El formato de grabación del archivo físico de transectos deberá sufrir algunas modificaciones, quedando de la siguiente manera:

-Una linea de cabecera del tipo **nn*. Donde:

*.- Es un identificador de inicio de datos de transectos para una estación determinada.
nn.- Es el número de estación.

-Una linea donde se indica el número total de estaciones que conforman al transecto, y el identificador de cada una de esas estaciones; el formato es el siguiente:

Nombre	Número de Estaciones	Identificador de cada una de las estaciones
Formato	nn	n0,n1,n2,n3,n4,n5.....

- Después de los datos del transecto se incluirá como una línea nueva los atributos de latitud o longitud según el tipo que corresponda. El formato de esta nueva línea se deberá acoplar a la siguiente convención:

ZAMAM - GEO - JJE

Tipo_transecto Direccion_transecto Grados Minutos donde:

Tipo_transecto	Dirección_transecto	Grados	Minutos
"0" (Latitudinal)	"+" (Norte)	99	9999
"0" (Latitudinal)	"-" (Sur)	99	9999
"1" (Longitudinal)	"+" (Este)	99	9999
"1" (Longitudinal)	"-" (Oeste)	99	9999
"2" (Anómalo)	"+"	IUNDEF	IUNDEF

-Después de la línea de atributos se incluirá como un renglón adicional la cadena "FIN DE TRANSECTO", que representara el final de cada transecto , substituyendo asi la anterior nomenclatura para fin de transectos que era "1,4000,0,2,12".

2.1.1.3.3 DISEÑO LÓGICO DE LA BASE DE DATOS.

En respuesta al requerimiento 10.0, se presentan a continuación las estructuras de datos que conformarán las listas doblemente ligadas, necesarias para mantener residentes en memoria los datos de los archivos del catálogo del crucero, y de nutrientes.

Para los tres casos de archivos existirá una lista principal, la cual representará una copia del archivo físico en memoria, este tipo de lista contiene solamente dos campos: un identificador de registro o llave, y el otro contendrá todo un renglón del archivo. El total de nodos en esta lista será el total de renglones que contenga nuestro archivo físico.

Existirá también otra lista, llamada de estación, que contendrá todos los datos de los niveles (catálogo de crucero o nutrientes) correspondientes a una determinada estación hidrográfica. Este tipo de lista contendrá un número de campos igual al total de diferentes datos que caracterizan a un determinado nivel de muestreo.(pej. para el caso del archivo del catalogo del crucero se tendrán 19 campos, mientras que para el archivo de nutrientes necesitarán 13 campos). Esta lista tendrá un nodo por cada nivel de muestreo de la estación.

La lista de datos contendrá dos campos llave, uno será la liga entre esta y la lista principal, y la otra llave será exclusiva para su manejo. Como se muestra en la figura 2.1.1.3.3 la relación que existe entre la lista principal y la lista de datos viene determinada por sus correspondientes llaves primarias, la cual mantiene la posición dentro de la lista principal que la lista de datos deberá guardar.

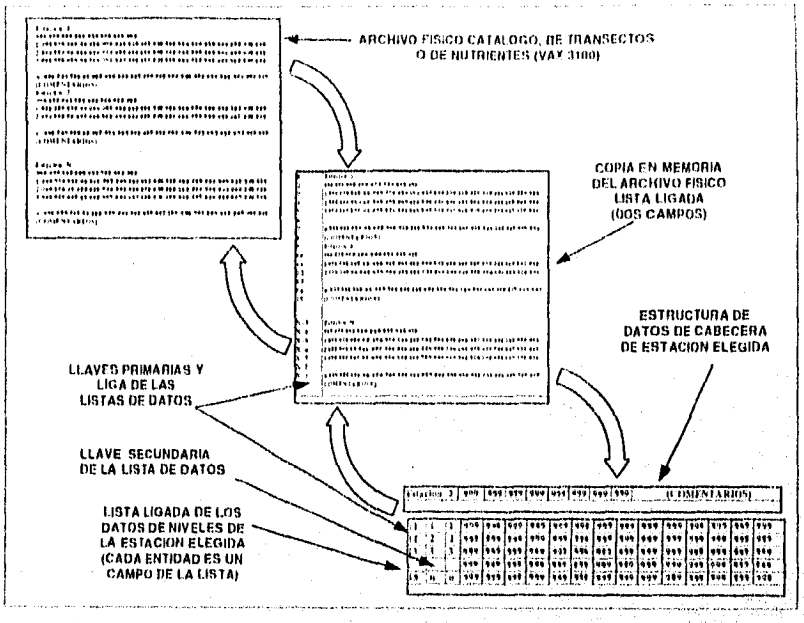


Figura 2.1.1.3.3 Relación lógica de la lista principal, de datos y del archivo físico

Para el caso del archivo de transectos, así como de los datos de cabecera del catálogo del crucero, se utilizaran estructuras sencillas.

La estructura de los registros que conforman a la lista principal se muestra a continuación:

```
typedef struct {
    long  id_prin;                  (llave de la lista principal)
    char  buffer[150];              (Campo buffer que contiene  toda un renglón de un archivo físico)
} REC_PRIN;
```

Tabla 2.1.1.3.3.b estructura de los registros que conforman a la lista principal

2.1.1.3.3.1.- CATÁLOGO DEL CRUCERO.

Estructura que contiene datos de cabecera de las estaciones hidrográficas.

```
struct cac_cabecera
    {
        int   i_cac_estacion;
        int   i_cac_prolmax;
        char i_cac_fecha_esta[10];
        char i_cac_hora_local[6];
```

```
    char i_cac_hora_dn[6];
    char i_cac_hora_gmt[6];
    char i_cac_lat_2[8];
    char i_cac_long_2[6];
    char i_cac_hora_gmt_2[6];
    char i_cac_lat_3[8];
    char i_cac_long_3[8]; }
```

Tabla 2.1.1.3.3.1.a.- Estructura de datos de cabecera de las estaciones hidrograficas

Estructura de los registros que conforman la lista de datos de la estación :

```
typedef struct {
    long  id_esta;
    long  nivel;
    float  i_cac_oxig_lab;
    float  i_cac_cable;
    float  i_cac_pres;
    float  i_cac_temp;
    float  i_cac_conduc;
    float  i_cac_salini;
    float  i_cac_sigma;
    float  i_cac_oxidis;
    float  i_cac_temoxi;
    float  i_cac_coroxi;
    float  i_cac_bot_salini;
    float  i_cac_bot_oxig;
    float  i_cac_nut;
    float  i_cac_nut2;
    float  i_cac_alca;
    float  i_cac_alca2;
    float  i_cac_salini_lab;
    } REC_ESTA;
```

Tabla 2.1.1.3.3.1.b. Estructura de los registros que conforman la lista de datos de la estación

2.1.1.3.3.2.- ARCHIVO DE NUTRIENTES;

Estructura de los registros que conforman a la lista de datos de nutrientes.

```
typedef struct {
    long  id_nut_nivel_esta;
    long  i_nut_botella;
    float  i_nut_abs1;
    float  i_nut_abs2;
    float  i_nut_abs3;
    float  i_nut_abs_prom;
    float  i_nut_salinidad;
    float  i_nut_abs_corr;
    float  i_nut_abs_real;
    float  i_nut_error;
    float  i_nut_factor;
    int    i_nut_celda;
    float  i_nut_concentra;
    } REC_NUT_LISTA_ESTA;
```

Tabla 2.1.1.3.3.2. Estructura de los registros que conforman a la lista de datos de nutrientes

2.1.1.3.3.3.- ARCHIVO DE TRANSECTOS.

Estructura de datos para el archivo de transectos.

```
struct tran_output
{
 int tran_transecto;
 int tran_max_estacion;
 int tran_array_estacion[100];
 int tran_grados;
 int tran_minutos;
 int tran_latlong;
 char tran_nseo[5];
};
```

Tabla. 2.1.1.3.3.3. Estructura de datos para el archivo de transectos

2.1.1.3.3.4. PROGRAMAS DE CALIBRACIÓN

Para el caso de los programas que realizan la calibración de los datos hidrográficos, se tendrá una o mas estructuras de datos por cada procedimiento que se involuera en dicha calibración, estas estructuras se derivan del archivo de control del apéndice A, quedando de la siguiente manera:

2.1.1.3.3.4.1 CALIBRACIÓN DE CONDUCTIVIDAD (Primera Etapa)

a) Rama 5 Del Programa mltrg2d.for

```
struct set_switches {
  char switch_3[10];
  char switch_4[10];
  char switch_6[10];
  char switch_10[10];
  char switch_11[10];
  char switch_12[10];
  char switch_15[10];
  char switch_16[10]; }
```

Tabla 2.1.1.3.3.4.1.a. Estructura de datos para la rama 5 del Programa mltrg2d.for

b) Rama 7 del programa mltrg2d.for

```
struct dato_limites
  {
  char variable[15];
  char valor[10];
   char rango[10];
   };
```

Tabla 2.1.1.3.3.4.1.b. Estructura de datos de la rama 7 del programa mltrg2d.for

c) Rama 22 del programa mltrg2d.for

```
struct archivo_entradas
 {
 char dato[5];
 char estacion[5];
 char variable[15];
 char archivo[60];
 };
```

Tabla 2.1.1.3.3.4.1.c. Estructura de datos de la rama 22 del programa mltrg2d.for

d) Rama 21 del programa mltrg2.for

```
struct tabla_datos
{
 char variable[15];
 char valor[10];
 char rango[10];
 char rechaza[5];
}
```

Tabla 2.1.1.3.3.4.1.d. Estructura de datos de la rama 21 del programa mltrg2d.for

e) Rama 25 del programa mltrg2d.for

```
struct escribe_mlts
 {
 char archivo[60];
 };
```

Tabla 2.1.1.3.3.4.1.e. Estructura de datos de la rama 25 del programa mltrg2d.for

f) Rama 10 del programa mltrg2d.for

```
struct lista_tablas
{
  char unidad[3];
  char primer[5];
  char ultimo[5];
}
```

Tabla 2.1.1.3.3.4.1.f. Estructura de datos de la rama 10 del programa mltrg2d.for

g) Rama 12 del programa mltrg2d.for

```
typedef struct
    {
     long  id_smcc1a_inter;
     char  variable1[15];
     char  variable2[15];
    }
     SMCC1A_INTER;
```

Tabla 2.1.1.3.3.4.1.g. Estructura de datos de la rama 12 del programa mltrg2d.for

h) Rama 20 del programa mltrg2d.for

```
typedef struct {
    long  id_smcc1a_regre;    /* campo llave de busqueda 0 */
    char  variable[15];
    char  primer[10];
    char  ultimo[10];
    char  abs2[5];
    char  abs3[5];
    } SMCC1A_REGRE;
```

Tabla 2.1.1.3.3.4.1.h. Estructura de datos de la rama 20 del programa mltrg2d.for

i) Rama 11 del programa mltrg2d.for

```
typedef struct {
    long  id_smcc1a_rescala;  /*campo llave de busqueda 0*/
    char  variable[15];
    char  a0[10];
    char  a1[10];
    cha   a2[10];
    char  a3[5];
    } SMCC1A_RESCALA;
```

Tabla 2.1.1.3.3.4.1.i. Estructura de datos de la rama 11 del programa mltrg2d.for

j) Estructura de datos general para la etapa de calibración de conductividad en su primera etapa

```
struct estructura_smcc1a {
  struct  set_switches          set_switchs;
  struct  dato_limites          dato_limite;
  struct  archivo_entradas      archivo_entrada;
  struct  tabla_datos           tabla_dato;
  struct  escribe_mlts          escribe_mlt;
  struct  lista_tablas                  lista_tabla;
  SMCC1A_INTER                  *rec_smcc1a_inter;
  SMCC1A_REGRE                  *rec_smcc1a_regre;
  SMCC1A_RESCALA                *rec_smcc1a_rescala;
 };
struct estructura_smcc1a estructura_smcc1a_nodo;
```

tabla 2.1.1.3.3.4.1.j. Estructura de datos general para el programa de calibración de conductividad en su primera etapa

2.1.1.3.3.4.2 CALIBRACIÓN DE CONDUCTIVIDAD (Segunda Etapa)

a) Rama 23 del programa mltrg2d.for

```
struct set_comentarios
 {
  char comentario[50];
 };
```

Tabla 2.1.1.3.3.4.2.a. Estructura de datos de la rama 23 del programa mltrg2d.for

b) Rama 21 del programa mltrg2d.for

```
typedef struct {
 long id_smcc2a_tabla;        /* campo llave de busqueda 0 */
 char variable[15];
 char valor[10];
 char rango[10];
 char rechaza[5];
 }SMCC2A_TABLA;
```

Tabla 2.1.1.3.3.4.2.b. Estructura de datos de la rama 21 del programa mltrg2d.for

c) Rama 3 del programa mltrg2d.for

```
typedef struct {
 long id_smcc2a_terminos;   /*campo llave de busqueda 0 */
 char coeff1[10];
 char coeff2[10];
 char coeff3[10];
char coeff4[10];
 }SMCC2A_TERMINOS;
```

Tabla 2.1.1.3.3.4.2.c. Estructura de datos de la rama 3 del programa mltrg2d.for

d) Rama 21 del programa mltrg2d.for

```
typedef struct {
 long id_smcc2a_edita;        /* campo llave de busqueda 0 */
 char rechaza[4];
 }SMCC2A_EDITA;
```

Tabla 2.1.1.3.3.4.2.d. Estructura de datos de la rama 21 del programa mltrg2d.for

e) Rama 8 del programa mltrg2d.for

```
typedef struct {
 long id_smcc2a_promedia;  /* campo llave de busqueda 0 */
 char variable[15];
 char cambia_factor[4];
 char factor[10];
 }SMCC2A_PROMEDIA;
```

Tabla 2.1.1.3.3.4.2.e. Estructura de datos de la rama 8 del programa mltrg2d.for

f) Rama 4 del programa mltrg2d.for

```
typedef struct {
 long id_smcc2a_histograma; /*campo llave de busqueda 0 */
 char valor[10];
 char rango[10];
 char rechaza[4];
 char apantalla[4];
 }SMCC2A_HISTOGRAMA;
```

Tabla 2.1.1.3.3.4.2.f. Estructura de datos de la rama 4 del programa mltrg2d.for

g) Estructura de datos general para la etapa de calibración de conductividad en su primera etapa.

```
struct estructura_smcc2a
 {
  struct set_switches          set_switche;
  struct set_comentarios       set_comentario;
  SMCC2A_TABLA                 *rec_smcc2a_tabla;
  SMCC2A_TERMINOS              *rec_smcc2a_terminos;
  SMCC2A_EDITA                 *rec_smcc2a_edita;
  SMCC2A_PROMEDIA              *rec_smcc2a_promedia;
  SMCC2A_HISTOGRAMA            *rec_smcc2a_histograma;
 };
struct estructura_smcc2a estructura_smcc2a_nodo;
```

tabla 2.1.1.3.3.4.2.g. Estructura de datos general para la segunda etapa de la calibración de conductividad

2.1.1.3.3.4.3 CALIBRACIÓN DE CONDUCTIVIDAD (Tercera Etapa)

a) Rama 00 del programa mltrg2d.fur

```
struct elige_funciones
 {
  char funcion[50];
  char aplica[5];
 };
```

Tabla 2.1.1.3.3.4.3.a. Estructura de datos de la rama 00 del programa mltrg2d.fur

b) Estructura de datos general para la etapa de calibración de conductividad en su primera etapa

```
struct estructura_smcc3a
 {
  struct sel_switches          set_switche;
  struct elige_funciones       elige_funcion;
  SMCC1A_INTER                 *rec_smcc3a_inter;
 };
struct estructura_smcc3a estructura_smcc3a_nodo;
```

tabla . 2.1.1.3.3.4.3.b Estructura de datos general para la tercera etapa de la calibración de conductividad

2.1.1.3.3.4.4 CALIBRACIÓN DE OXIGENO (Primera Etapa)

a) Estructura que guarda el nombre del archivo de muestras de agua a abrir

```
struct muestras_de_aguas
 {
  char nombre_archivo[38];
 };
```

Tabla 2.1.1.3.3.1.4.a. Estructura que guarda el nombre del archivo de muestras de agua a abrir

b) Estructura de datos para determinar modo de operación

```
struct rango_estaciones
  {
   char modo[15];
   char rango_min[4];
   char rango_max[4];
  };
```

Tabla 2.1.1.3.3.4.b. Estructura de datos para determinar modo de operacion de programa mltoxdwn

c) Rama 12 del programa mltoxdwn.for

```
typedef struct
 {      /* example record structure */
 long  id_smco1a_estacion; /*campo llave de busqueda 0*/
 char  estacion[5];
 }SMCO1A_ESTACION;
```

Tabla 2.1.1.3.3.4.c. Estructura de datos de la rama 12 del progcamo mltoxdwn.for

d) Estructura de datos general para la etapa de calibración de conductividad en su primera etapa

```
struct estructura_smco1a
 {
 struct muestras_de_aguas        muestras_de_agua;
 struct rango_estaciones         rango_estacion;
 SMCO1A_ESTACION                 *rec_smco1a_estacion;
 };
struct estructura_smco1a estructura_smco1a_nodo;
```

tabla 2.1.1.3.3.4.d. Estructura de datos general para la primera etapa de la calibración de oxigeno.

2.1.1.3.3.4.5 CALIBRACIÓN DE OXIGENO (Segunda Etapa)

a) Estructura para datos de oxigeno

```
struct factor_oxigenos
  {
   char factor_oxigeno[6];
   char oxigeno_minimo[6];
  };
```

tabla, 2.1.1.3.3.5.a Estructura para datos de oxigeno

b) Estructura de datos de la ventana de captura del programa de mltoxdwn.for

```
typedef struct
  {      /* example record structure */
  long  id_smco2a_struct;  /* campo llave de busqueda 0 */
  char  numero_parametros[3];
  char  parametro_inicio[15];    /* campo de los datos */
  char  guarda_archivo[5];
  char  tipo_edicion[27];
  char  digitos_signif[3];
  char  numero_iterac[5];
  char  valores_default[5];
```

```
                char  sesgo[10];
                char  pendiente[10];
                char  pcor[10];
                char  tcor[10];
                char  wl[5];
                char  retrazo[3];
               }SMCO2A_STRUCT;
```

Tabla 2.1.1.3.3.5.b.. Estructura de datos de la ventana de captura del progeama de mltoxdwn.for

b) Estructura de datos general para la etapa de calibración de oxigeno en su segunda etapa

```
struct esbuctura_smco2a
 {
  struct factor_oxigenos  factores_oxigeno;
  SMCO2A_STRUCT          *rec_smco2a_struct;
 };

struct estructura_smco2a estructura_smco2a_nodo;
```

tabla 2.1.1.3.3.5.c . Estructura de datos general para la segunda etapa de la calibración de oxigeno.

2.1.1.3.3.4.6 CALIBRACIÓN DE OXIGENO (Tercera Etapa)

a) Estructura de datos para la ventana principal de calibración de oxigeno (Tercero Etapa)

```
struct parametros_ajustes
  {
   char modo[12];
   char sesgo[12];
   char pendiente[12];
   char pcor[12];
   char tcor[12];
   char wl[12];
   char retrazo[12];
   char estacion_min[4];
   char estacion_max[4];
  };
```

tabla 2.1.1.3.3.4.6.a. Estructura de datos para la ventana principal de calibración de oxigeno (Tercera Etapa)

b) Estructura de las estaciones disponibles para SMCO3A

```
typedef struct
  {       /* example record structure */
  long  id_smco3a_estacion; /*campo llave de busqueda 0 */
  char  estacion[5];
 }SMCO3A_ESTACION;
```

tabla 2.1.1.3.3.4.6.b.. Estructura de las estaciones disponibles para SMCO3A

c) Estructura de datos general para la etapa de calibración de conductividad en su primera etapa

```
struct estructura_smco3a
 {
  struct parametros_ajustes parametro_ajuste;
  SMCO3A_ESTACION          *rec_smco3a_estacion;
 };
struct estructura_smco3a estructura_smco3a_nodo;
```

tabla . 2.1.1.3.3.4.6.c. Estructura de datos general para la etapa de calibración de conductividad en su primera etapa

2.1.1.3.4 ARCHIVO DE CONTROL

El formato del archivo de control vendrá definido de la siguiente manera:

```
[Cabecera del procedimiento de calibación en turno]
        [Identificador de la RAMA del programa n RAMA]
        [numero de nodos en la lista particular de la rama]
        [datos particulares de la rama]
[Identificador de la RAMA del programa n+1 RAMA]
        [numero de nodos en la lista particular de la rama]
        [datos particulares de la rama]
[Identificador de rama de salida]
```

tabla 2.1.1.3.4. Formato del archivo de control que será utilizado para los programas de calidad.

Un ejemplo de un archivo de control correspondiente a los procedimientos de calidad controlada se encuentra en el apéndice *A*.

2.1.1.3.5 ARCHIVOS DE COMANDOS

Los formatos de los correspondientes archivos de comandos que deberá formar cada procedimiento de calibración son los mismos que el anterior sistema utilizaba.

2.1.2 IDENTIFICACIÓN DE PROGRAMAS.

El sistema MPD se compone de los siguientes programas:

2.1.2.1 PROGRAMA PARA LA CAPTURA DEL ARCHIVO DEL CATÁLOGO DEL CRUCERO.

Este programa genera y opera al archivo de Catálogo de crucero, tomando los datos de entrada de los documentos de bitácora de abordo, y de los datos del laboratorio químico (figura 2.1.2.1).

2.1.2.2 PROGRAMA PARA LA CAPTURA DE NUTRIENTES.

Se encarga de capturar y mantener al archivo de nutrientes. Tratando a cada tipo de dato (silicato, fosfato y nitrato) con un procedimiento en particular.

Toma como datos de entrada la información de las planillas de nutrientes, calculando las correspondientes concentraciones (figura 2.1.2.1).

2.1.2.3 PROGRAMA PARA LA CAPTURA DEL ARCHIVO DE TRANSECTOS.

Obtiene sus datos de entrada de los documentos de transectos que se generan al concluir un crucero oceanográfico(figura 2.1.2.1).

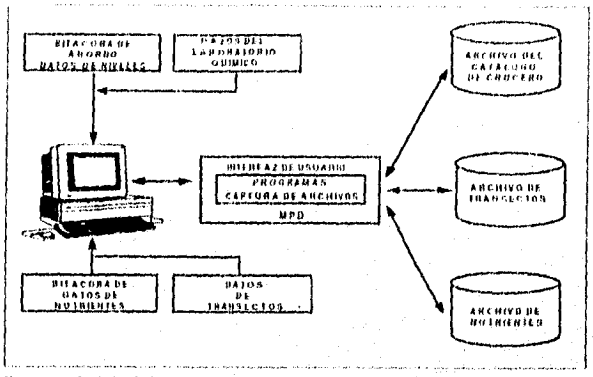


figura 2.1.2.1 diagrama de flujo de información para los programas de capturas de datos (Catálogo, nutrientes y transectos)

2.1.2.4 PROGRAMA QUE GENERA ARCHIVOS INDIVIDUALES DE NUTRIENTES:

Este programa se encarga de obtener los archivos individuales de nutrientes, toma sus datos de entrada de los archivos de índices, y del archivo de nutrientes.

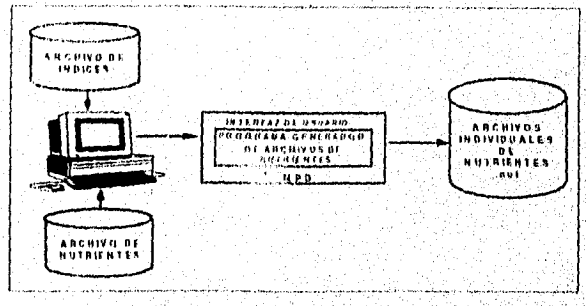


figura 2.1.2.1 diagrama de flujo de información del programa que genera archivos individuales de nutrientes.

2.1.2.5 PROGRAMAS PARA LA OBTENCIÓN DE DATOS DE CALIDAD CONTROLADA.

Este programa obtiene los archivos de datos de calidad controlada a partir de los archivos en formato NBIS y del archivo del catálogo del crucero.

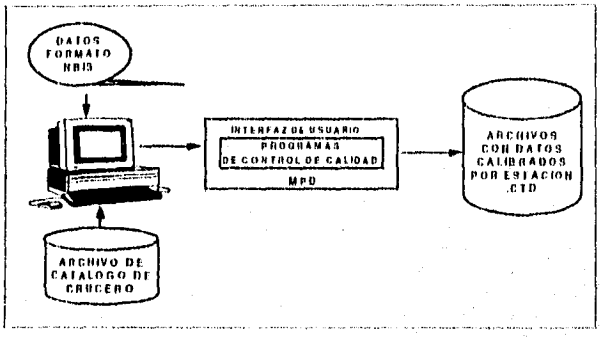


figura 2.1.2.5 diagrama de flujo de información para los programas que generan los datos de calidad controlada

2.1.3 COMPORTAMIENTO DINÁMICO DEL SISTEMA.

La interacción del sistema MPD con el usuario será, como ya se mencionó a través de una interfaz gráfica de usuario y se activará según 1.3.4.1.

La interfaz gráfica de usuario será realizada bajo un sistema de ventanas de X-Window[3], lo anterior quiere decir que (de acuerdo a la filosofía de este tipo de sistemas), nuestra interfaz quedará expuesta a un cierto tipo de eventos[4] de entrada que será necesario controlar y manipular en todo momento, por lo anterior, se debe establecer el comportamiento exacto del sistema al presentarse cualquiera de los posibles eventos de entrada sobre los objetos generales de la interfaz.

A continuación se establece el comportamiento de cada programa del MPD dependiendo de cual haya sido el objeto afectado por el evento de entrada.

Programa	Captura del Catálogo del Crucero
Objeto	Acción
"Acepta"	Se realizará una copia dela lista principal hacia un archivo físico, seguida de la destrucción de las listas en memoria y desapareciendo la interfaz.

[3] Ver apéndice B
[4] Ver apéndice B

"Cancela"	Se destruyen la listas ligadas y se desaparece la interfaz.
"Elige estación"	Se busca dentro de la lista principal los datos de la estación elegida, si se encuentra se presentan sus datos en pantalla, en caso contrario se manda un mensaje de aviso.
"Movimientos"	Se da la oportunidad al usuario de elegir de entre borrar, mover o insertar niveles de datos.
"Comentarios"	Aparece un objeto que facilite la captura de un texto de comentarios.

Tabla 2.1.3.a. Tabla de eventos del programa de captura del catálogo del crucero.

Programa	Captura del archivo de Nutrientes
Objeto	Acción
"Acepta"	Se realizara una copia dela lista principal hacia un archivo fisico, seguida de la destrucción de las listas en memoria y desapareciendo la interfaz.
"Cancela"	Se destruyen la listas ligadas y se desaparece la interfaz.
"Elige estación"	Se busca dentro de la lista principal los datos de la estación elegida, si se encuentra se presentan sus datos en pantalla, en caso contrario se manda un mensaje de aviso.
"Edita Nutrientes"	Aparecerá un objeto tipo ventana donde el usuario podrá registrar los valores de absorbancias registradas en as plantillas de nutrientes.
"Acepta Nutrientes"	Se actualizan la información de la lista de datos de estación y desaparece el objeto tipo ventana.
"Cancela Nutrientes"	Se destruye el objeto tipo ventana utilizado para capturar datos de absorbancias.
"Comentarios"	Aparece un objeto que facilitará al usuario la captura de un texto de tipo comentarios.

Tabla 2.1.3.b. Tabla de eventos del progcama para la captura del archivo de nutrientes

Programa	Captura del Archivo de Transectos
Objeto	Acción
"Acepta"	Se realizara una copia dela lista principal hacia un archivo fisico, seguida de la destrucción de las listas en memoria y desapareciendo la interfaz.
"Cancela"	Se destruyen la listas ligadas y se desaparece la interfaz.
"Elige transecto"	Se elige mediante un objeto de interfaz, un transecto determinado, listando los números de estaciones que lo conforman
"Edita estaciones"	Aparecerá un objeto tipo ventana que permitirá al usuario editar y modificar los números de estaciones para cada transecto.
"Borra Transecto"	Eliminara del objeto tipo ventana el transecto elegido.
"Nuevo Transecto"	Añadirá un nuevo transecto a la lista ligada principal.

tabla 2.1.3.c.. Tabla de eventos de las programas de captura del archivo de transectos

Programa	Generación de archivos individuales de nutrientes
Objeto	Acción
"Genera"	Aparecerá un objeto tipo caja indicando que el proceso se esta llevando a cabo.
"Cancela"	Se aborta la generación de los archivos y de manda el control a la pantalla principal CTDSIS

tabla 2.1.3.d. Tabla de eventos del programa de generación de archivos de nutrientes

Programa	Obtención de datos de calidad controlada
Objeto	Acción
"Guarda"	Toma todos los datos que se encuentran en ese momento en la interfaz, y arma las estructuras que conformaran parte del archivo de control.
"Cancela"	No actualiza los datos de las estructuras y desaparece la interfaz.

tabla 2.1.3.e Tabla de eventos de los programas de control de calidad

2.1.3.1 INTERFAZ GRÁFICA

Para el diseño de la interfaz del sistema MPD y cumpliendo con el requerimiento 1.3.4.11 acerca del nombramiento de los objetos de la interfaz.

Los nombres identificadores de estos objetos se forman tomando en cuanta la siguiente nomenclatura:

A. Prefijo ***k_***
B. Identificador del programa al que pertenece la interfaz
   a) Catalogo del Crucero : ***cac_***
   b) Transectos: ***tra***
   c) Nutrientes ***nut_***
   d) Calidad
      (1) C. Conductividad (1a etapa) ***smcc1a_***
      (2) C. Conductividad (2a etapa) ***smcc2a_***
      (3) C. Conductividad (3a etapa) ***smcc3a_***
      (4) Series de presión ***smp_***
      (5) C.Oxigeno (Primera etapa) ***smco1a_***
      (6) C. Oxigeno (Segunda etapa) ***smco2a_***
      (7) C. Oxigeno (Tercera etapa) ***smco3a_***
   e) Nutrientes Individuales ***checa_***
C. tipo del objeto (boton_, texto_, win_, toggle_, etc)
D. funcion especifica del objeto

p.ej El identificador ***k_smcc1a_toggle_switch_3.*** Se refiere a un objeto tipo toggle button perteneciente a la interfaz de Calibración de conductividad en su primera etapa, que se denomina switch número 3.

figura 2.1.3.1. Nomenclatura para nombrar cada una de los objetos que conforman la interfaz

2.1.4 LAS INTERACCIONES ORGANIZACIONALES.

El sistema MPD para su ejecución poseerá una serie de restricciones, que vendrán determinadas por las condiciones del capitulo 1. Además las relaciones con MGD y con MRG también estarán regidas por este documento.

2.2 DOCUMENTO DE DISEÑO PARA EL SISTEMA MPD.

2.2.1 PANORAMA GENERAL.

El sistema MPD representa un módulo del sistema CTDSIS, como se ha documentado ya en la introducción. Su objetivo es realizar las capturas de archivos de datos (catálogo del crucero, transectos y nutrientes), así como de obtener los datos de calidad controlada a partir de información recabada por una sonda hidrográfica. Todo lo anterior, a través de una interfaz gráfica construida bajo el sistema X-Window.

Los submódulos principales del MPD son 8 y se citan en 1.3.3.2.

2.2.2 DIAGRAMAS DE ESTRUCTURAS.

Debido al gran número de módulos que presenta cada uno de los programas componentes del MPD, se mostrara primero un diagrama de estructura general, para posteriormente descomponer cada uno de los subsecuentes submódulos en diagramas de estructuras mas detallados.

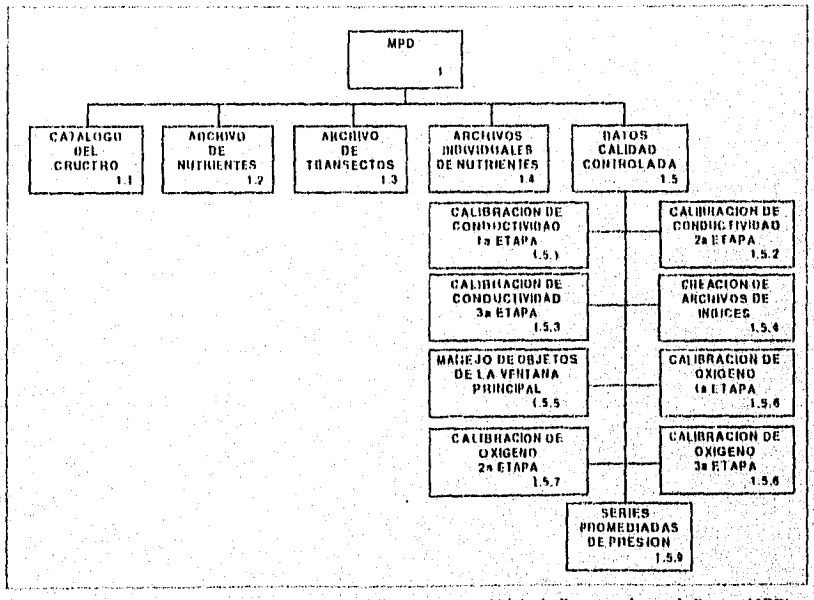


Figura 2.2.2 Diagrama de estructura general para el sistema Módulo de Procesamiento de Datos (MPD)

2.2.2.a Tabla de abreviaciones de procedimientos

A continuación se presentan los diagramas de estructuras individuales de cada uno de los módulos pertenecientes a MPD

La tabla 2.2.2.b presenta las abreviaciones para cada procedimiento contenido dentro de los programas de Catálogo del Crucero, de Nutrientes y de transectos; la tabla 2.2.2.a muestra las funciones para el manejo de las listas ligadas y de ventanas de interfaz, y la tabla 2.2.2.c. contiene las abreviaciones para los procedimientos de los programas de Calidad (Conductividad, Oxigeno, Archivos de Índices, Presión y para el manejos de los objetos de la interfaz principal)

Funciones para el manejo de la listas ligadas	Abreviación	Funciones para el manejo de ventanas en la interfaz	Abreviación
LastRec	LR	VUIT_Unm,anege	VU
MoveRec	MR	VUIT_Manage	VM
DeleteRec	DR		
DeleteList	DL		
InsertRec	IR		
GotRecByKey	GR		
NextRec	NR		
PrevRec	PR		
FirstRec	FR		
CreateList	CL		

Tabla 2.2.2.a. Funciones para el manejo de las listas ligadas y para el manejo de ventanas de la interfaz

Módulos del programa del archivo del Catálogo del Crucero	Ab	Módulos del programa del archivo de Nutrientes	Ab	Módulos del programa del archivo de Transectos	Ab
sale_proc	csp	nut_sale_proc	nsp	tra_crea_proc	tcp
proc_llama	cpl	nut_vaciado_de_objetos_de_datos	nvd	tra_abre_archivo_transectos	tat
cac_proc_crea	cpc	nut_busca_estacion_en_cat	nbc	tra_abre_archivo_catalogo	tac
scroleo_header	csh	nut_busca_estacion_en_nut	nbn	tra_arrows_proc	tap
scroleo_win	csw	nut_proc_llama	npl	tra_set_something	tss
scroleo_win_var	csv	nut_selecciona_nivel	nsn	tra_salidas_proc	tsp
cac_abre_catalogo	cac	nut_graba_en_lista_prin	ngl	tra_vacia_datos_os_objeto_texto	tvo
cac_busca_estacion	cbe	nut_acepta_abs	nas	tra_operaciones_transectos	tot
cac_borra_registros	cbr	nut_maneja_comen	nmc	tra_down_transecto	tdt
cac_iguala_valores	cgv	nut_arregla_lista	nal	tra_incluye_transectos	tit
cac_obten_id	coi	nut_acepta_archivo	naa	tra_salva_en_lista	tsl
cac_operaciones	cop	nut_arma_enteros	naa	tra_acepta_archivo	taa
cac_arregla_lista	cal	nut_arma_decimas	nad	tra_selecciona_transecto	tst
cac_pon_a_ceros	cp0	nut_arma_prin_buffer	nab	tra_checa_lin_transecto	tlt
cac_captura_fecha	ccf	nut_pon_etiquetas_a_niveles	npe	tra_proc_caja_crea_archivo	tca
cac_vaciado_de_objetos_de_datos	cvo	nut_abre_catalogo	nac	tra_proc_caja_caracter_invalido	tci
cac_set_something	css	nut_abre_nutrientes	nan	tra_proc_nuevo_item	tni
cac_pon_etiquetas_a_niveles	cpe	nut_iguala_valores	niv	tra_busca_estacion_en_catalogo	tec
cac_captura_longitud	ccl	nut_scroleo_header	nsh	tra_maneja_lellongs	tml
cac_captura_hora	cch	nut_scroleo_win	nsw	tra_obten_datos	tod
cac_maneja_coment	cmc	nut_scroleo_win_ver	nsv	tra_pon_datos_caja	tpd
cac_guarda_archivo	cga	nut_proc_crea	npc	tra_repite_transecto	trt
cac_captura_datos	ccd	CapturaValorFloatDeWidgetText	ncf		
cac_arma_ser_decimas	cad	AsignaValorFloatAWidgetText	naf		
cac_guarda_estacion_en_lista	cge	NUT_VALIDA_CARACTERES	nvc		
cac_cancela_estacion_en_lista	cce	NUT_VALIDA_VALOR	nvv		
cac_proc_anade_estacion	cae	nut_set_something	nss		
cac_guarda_registro_datos	cgr	NUT_DESTRUYE_OBJETOS	ndo		

cac_ultima_estacion	cue						
cac_repite_estacion	cre						
cac_borra_estacion	cbe						
cac_arregla_argos	caa						
cac_proc_maneja_catalogo	cpm						

Tabla 2.2.2.b. Abreviaciones de los procedimientos que comprnen a los programas de capturas del Catálogo del Crucero, de Nutrientes y de Transectos

Calibración de Conductividad Primera etapa	**Ab**	**Calibración de Conductividad Segunda etapa**	**Ab**	**Calibración de Conductividad Tercera etapa**	**Ab**
smcc1a_sel_something	c1ss	smcc2a_sel_something	c2ss	smcc3a_sel_something	c3ss
smcc1a_get_something	c1gs	smcc2a_get_something	c2gs	smcc3a_get_something	c3gs
smcc1a_obten_texto_de_opcion	c1ot	smcc2a_obten_texto_de_opcion	c2ot	smcc3a_obten_texto_de_opcion	c3ot
smcc1a_determina_texto_opcion	c1dt	*smcc2a_determina_texto_opcion*	c2dt	*smcc3a_determina_texto*	c3dt
smcc1a_vacia_a_interfaz	c1vi	smcc2a_vacia_a_interfaz	c2vi	smcc3a_vacia_a_interfaz	c3vi
smcc1a_determina_id	c1di	smcc2a_determina_id	c2di	smcc3a_determina_id	c3di
smcc1a_vacia_structura_interfaz	c1vs	smcc2a_vacia_structura_interfaz	c2vs	smcc3a_vacia_structura_interfaz	c3vs
smcc1a_inicia	c1ii	smcc2a_inicia	c2ii	smcc3a_inicia	c3ii
smcc1a_crea_proc	c1cp	*smcc2a_crea_proc*	c2cp	smcc3a_crea_proc	c3cp
smcc1a_determina_variable	c1dv	smcc2a_incluye_archivo_control	c2ia	*smcc3a_selecciona_lista_inter*	c3sl
smcc1a_selecciona_lista_inter	c1si	smcc2a_determina_variable	c2dv	smcc1a_selecciona_lista_funcio	c3sf
smcc1a_incluye_archivo_control	c1ia	smcc2a_selecciona_lista_edita	c2se	smcc3a_inserta_listas	c3il
smcc1a_acepta_proc	c1ap	smcc2a_selecciona_lista_termi	c2st	smcc3a_elimina_listas	c3el
smcc1a_arma_comandos	c1ac	smcc2a_selecciona_lista_histo	c2sh	smco3a_llama_pantalla_prin	c3ls
smcc1a_selecciona_lista_regre	c1sr	*smcc2a_selecciona_lista_promo*	c2sp	smcc3a_incluye_archivo_control	c3ia
smcc1a_saca_pantalla_prin	c1sp	*smcc2a_inserta_lista_edita*	c2ie	*smcc3a_acepta_proc*	c3ap
smcc1a_elimina_listas	c1el	smcc2a_inserta_lista_termi	c2il	smcc3a_arma_comandos	c3ac
smcc1a_up_down_proc	c1ud	smcc2a_inserta_lista_histo	c2ih	smcc3a_modifica_lista	c3ml
smcc1a_nuevos_procs	c1np	smcc2a_inserta_lista_promo	c2ip	smcc3a_determina_variable	c3dv
smcc1a_modifica_listas	c1ml	smcc2a_elimina_lista_edita	c2ee	smcc3a_up_down_proc	c3ud
smcc1a_inserta_listas	c1il	*smcc2a_elimina_lista_termi*	c2et	*smcc3a_nueva_proc*	c3np
smcc1a_inserta_en_estructuras	c1is	smcc2a_elimina_lista_histo	c2eh	smcc3a_inserta_en_estructuras	c3is
smcc1a_lee_archivo_control	c1la	smcc2a_elimina_lista_promo	c2ep	smcc3a_lee_archivo_control	c3ln
		smcc2a_saca_pantalla_prin	c2sp		
Creacion de Archivos de Indices	**Ab**	smcc2a_arma_comandos	c2ac	**Manejo de objetos De la pantalla Principal**	**Ab**
		smcc2a_acepta_proc	c2ap		
		smcc2a_mod_lista_edita	c2me		
crdsu_ejecuta_crsumary	ecr	smcc2a_mod_lista_termi	c2ml	ctd_mpd_ejecuta_calmuos	ctct
mpd_indices_arranque	iarr	smcc2a_mod_lista_histo	c2mh	ctd_mpd_ejecuta_smcc1a	ctc1
		smcc2a_mod_lista_promo	c2mp	ctd_mpd_ejecuta_smcc2a	ctc2
Calibración de Oxigeno Primera Etapa	**Ab**	smcc2a_up_down_proc	c2ud	ctd_mpd_ejecuta_smcc2a	ctc3
		smcc2a_nuevos_procs	c2np	ctd_mpd_ejecuta_smp	ctsp
		smcc2a_inserta_en_estructuras	c2is	ctd_mpd_ejecuta_crsumarym	ctsm
smco1a_inserta_en_estructuras	o1is	smcc2a_lee_archivo_control	c2la	ctd_mpd_ejecuta_smco1a	cto1
smco1a_lee_archivo_control	o1la			ctd_mpd_ejecuta_smco2a	cto2
smco1a_vacia_a_interfaz	o1vi			ctd_mpd_ejecuta_smco2a	cto3
smco1a_lee_mdirfil	o1lm	**Calibración de Oxigeno Segunda Etapa**	**Ab**	ctd_mpd_ejecuta_crsumaryx	ctsm
smco1a_sel_something	o1ss			ctd_mpd_ejecuta_nutrientes	ctcn
smco1a_get_something	o1gs			ctd_work_proc	ctwp
smco1a_vacia_lista_disponibles	o1vd	smco2a_vacia_datos_caratula	o2vc	ctd_mpd_llena_nombres_structs	ctns
smco1a_vacia_structura_interfaz	o1vs	smco2a_inicia	o2ii	ctd_mpd_llama	ctll
smco1a_inicia	o1ii	*smco2a_crea_proc*	o2cp		
smco1a_crea_proc	o1cp	*smco2a_up_down_proc*	o2ud	**Calibración de Oxigeno Tercera Etapa**	**Ab**
smco1a_incluye_listas	o1il	smco2a_aparece_win_captura	o2aw		
smco1a_excluye_listas	o1le	smco2a_esconde_win_proc	o2ew		
smco1a_incluye_listas_elegida	o1li	smco2a_obten_datos_de_texto	o2ot	smco3a_sel_something	o3ss
smco1a_excluye_listas_elegida	o1el	*smco2a_acepta_ajuste_proc*	o2aa	smco3a_get_something	o3gs
smco1a_cancela_pantalla	o1cp	smco2a_incluye_archivo_control	o2ia	smco3a_lee_mdirfil	o3lm
smco1a_incluye_archivo_control	o1ia	smco2a_acepta_todo_proc	o2at	smco3a_vacia_lista_disponibles	o3vl
smco1a_acepta_proc	o1ap	smco2a_borra_ajuste_proc	o2ba	smco3a_lee_archivo_fil	o3lf
smco1a_arma_comandos	o1ac	smco2a_maneja_toggles	o2mt	smco3a_vacia_estructura_interfa	o3vs
smco1a_acepta_pantalla_rango	o1ar	smco2a_inserta_en_estructuras	o2is	smco3a_inicia	o3ii
smco1a_acepta_pantalla_incluye	o1ai	*smco2a_lee_archivo_control*	o2lc	smco3a_crea_proc	o3cp

smco1a_edita_estaciones	o1es	smco2a_arma_comandos	o2ac	smco3a_incluye_listas	o3li
smco1a_maneja_escala	o1me			smco3a_excluye_listas	o3lo
smco1a_esconde_error	o1eo	**Series Uniformes De presion**	Ab	smco3a_cancela_pantalla	o3cp
				smco3a_incluye_archivo_control	o3ia
				smco3a_acepta_todo	o3at
		smp_lee_archivo_parametro	plap	smco3a_acepta_pantalla_rango	o3ar
		smp_vacia_estructura_interfaz	pvsi	smco3a_acepta_pantalla_incluye	o3ai
		smp_inicia	piui	smco3a_edita_estaciones	o3ee
		mpd_smp_crea_proc	pcpr	smco3a_maneja_escala	o3me
		smp_incluye_archivo_control	piac	smco3a_esconde_error	o3er
		mpd_smp_arma_comando	pac	smco3a_selecciona_listas	o3sl
		mpd_smp_acepta_todo	pact	smco3a_incluye_listas_elegida	o3il
		mpd_cancela_wins	pcw	smco3a_excluye_listas_elegida	o3el
		mpd_smp_maneja_toggle	pmt	smco3a_inserta_en_estructuras	o3is
		mpd_smp_trae_ventana	ptv	smco3a_lee_archivo_control	o3ac
		mpd_smp_acepta_cambio	pac		
		mpd_smp_inserta_en_estructura	pies		
		mpd_smp_lee_archivo_control	plac		

Tabla 2.2.2.c, Abreviaciones de los procedimientos de calibración de conductividad (primera, segunda y tercera etapa), oxigeno (primera, segunda y tercera etapa), series de presión, manejo de objetos de la lista principal y de archivos de Índices.

2.2.2.1 DIAGRAMA DE ESTRUCTURA DEL PROGRAMA DE CAPTURA DEL CATÁLOGO DEL CRUCERO

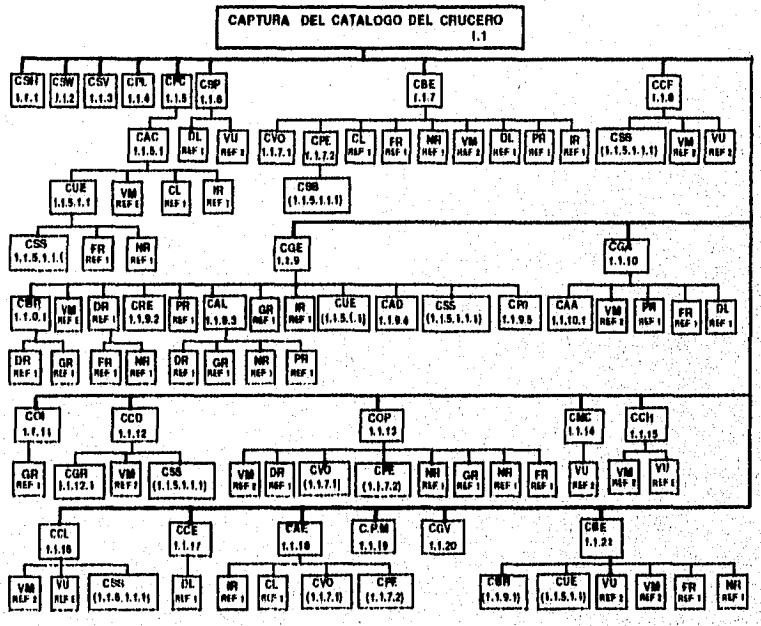


Figura 2.2.2.1. Diagrama de estructura del programa de Captura del Catálogo del Crucero

2.2.2.2 DIAGRAMA DE ESTRUCTURA DEL PROGRAMA DE CAPTURA DE DATOS DE NUTRIENTES

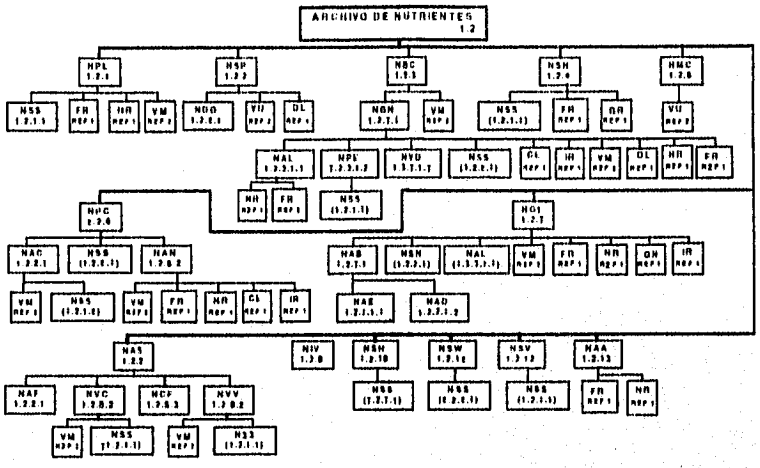


Figura 2.2.2.2. Diagramas de estructura del programa de captura de datos de nutrientes

2.2.2.3 DIAGRAMA DE ESTRUCTURA DEL PROGRAMA DE CAPTURA DE DATOS DE TRANSECTOS.

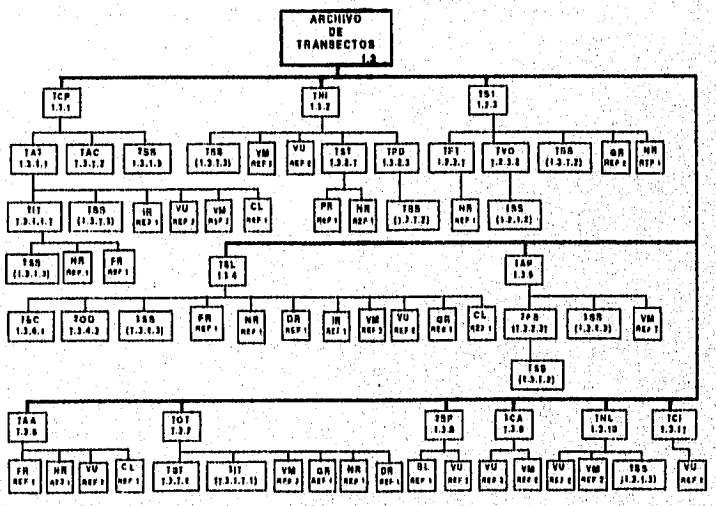


Figura 2.2.2.3. Diagrama de estructura del programa de captura de datos de transectos

2.2.2.4 DIAGRAMA DE ESTRUCTURA DEL PROGRAMA DE CALIBRACIÓN DE CONDUCTIVIDAD EN SU PRIMERA ETAPA

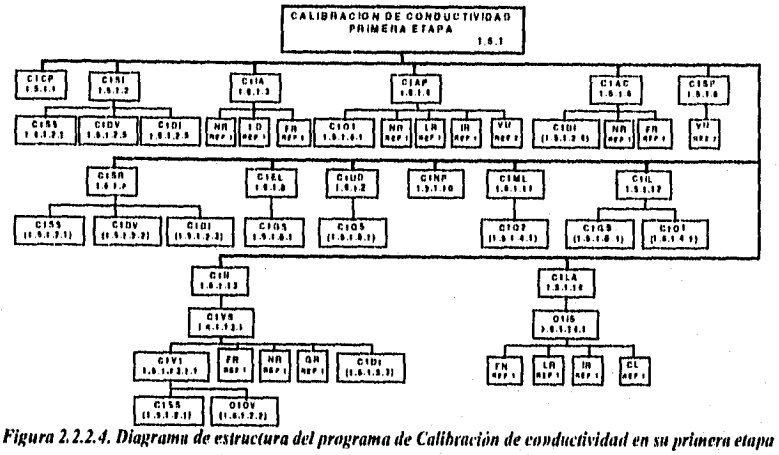


Figura 2.2.2.4. Diagrama de estructura del programa de Calibración de conductividad en su primera etapa

2.2.2.5 DIAGRAMA DE ESTRUCTURA DEL PROGRAMA DE CALIBRACIÓN DE CONDUCTIVIDAD EN SU SEGUNDA ETAPA.

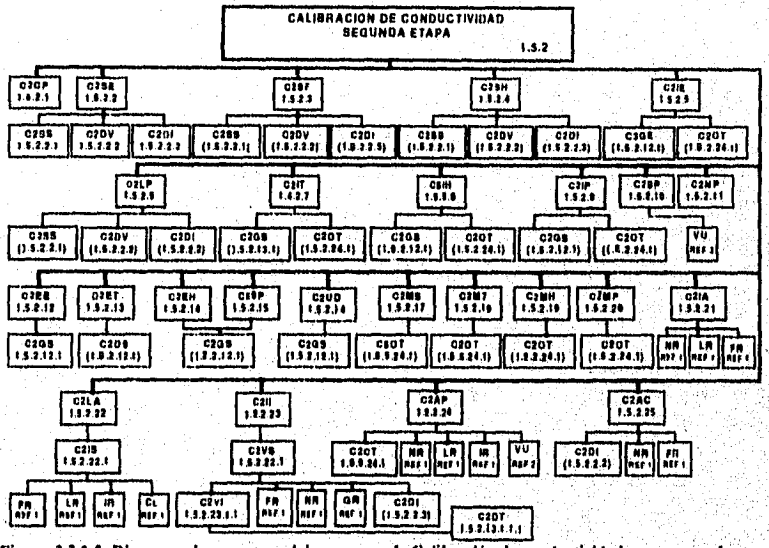


Figura 2.2.2.5. Diagrama de estructura del programa de Calibración de conductividad en su segunda etapa

3.2.3.6 DIAGRAMA DE ESTRUCTURA DEL PROGRAMA DE CALIBRACIÓN DE CONDUCTIVIDAD EN SU TERCERA ETAPA

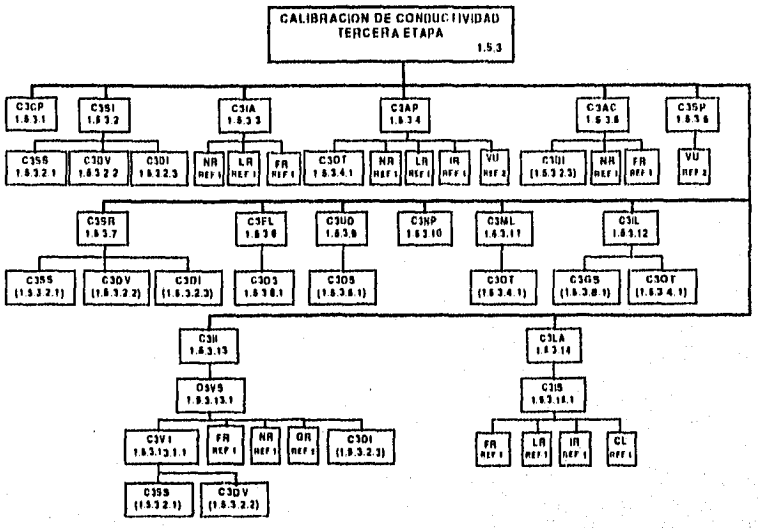


Figura 2.2.2.6. Diagrama de estructura del programa de Calibración de conductividad en su tercera etapa

2.2.2.7 DIAGRAMA DE ESTRUCTURA DEL PROGRAMA DE CALIBRACIÓN DE OXÍGENO EN SU PRIMERA ETAPA

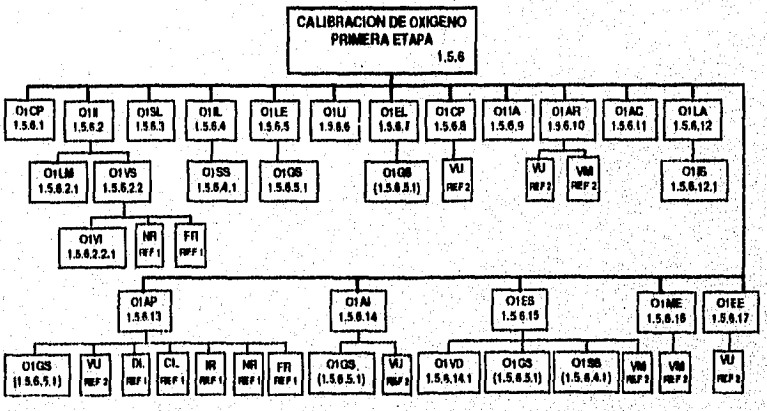


Figura 2.2.2.7. Diagrama de estructura del programa de Calibración de oxigeno en su primera etapa

2.2.2.8 DIAGRAMA DE ESTRUCTURA DEL PROGRAMA DE CALIBRACIÓN DE OXÍGENO EN SU SEGUNDA ETAPA

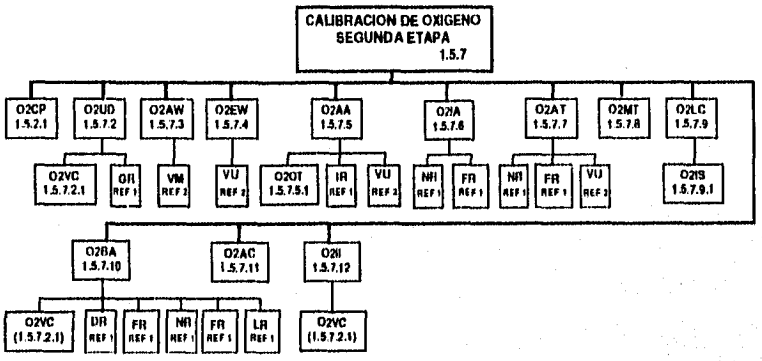


Figura 2.2.2.8. Diagrama de estructura del programa de Calibración de oxígeno en su primera etapa

2.2.2.9 DIAGRAMA DE ESTRUCTURA DEL PROGRAMA DE CALIBRACIÓN DE OXÍGENO EN SU TERCERA ETAPA

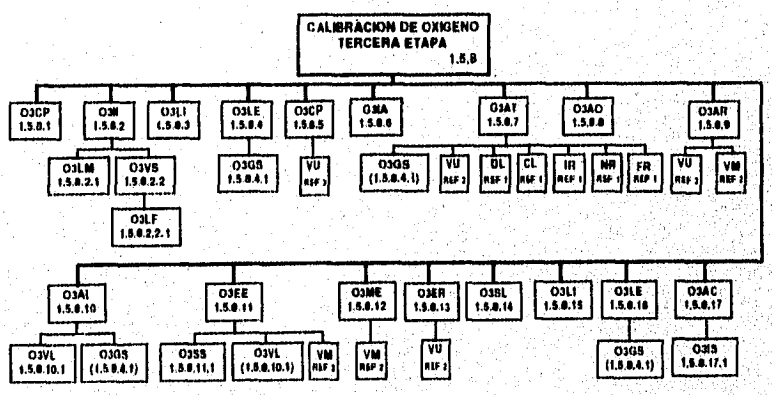


Figura 2.2.2.9. Diagrama de estructura del programa de Calibración de oxígeno en su tercera etapa

2.2.2.10 DIAGRAMA DE ESTRUCTURA DEL PROGRAMA DE SERIES PROMEDIADAS DE PRESIÓN

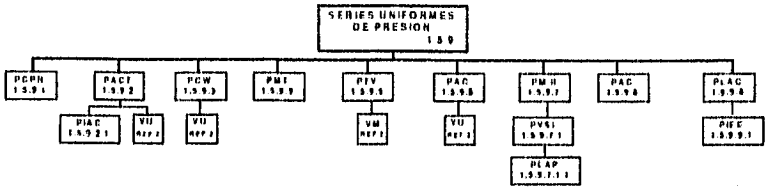


Figura 2.2.2.10. Diagrama de estructura del programa de Series Promediadas de Presión

2.2.2.11 DIAGRAMA DE ESTRUCTURA DEL PROGRAMA DE CREACIÓN DE ARCHIVOS DE ÍNDICES

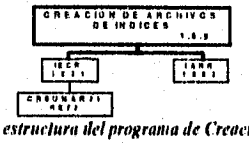


Figura 2.2.2.11. Diagrama de estructura del programa de Creación de archivos de índices

2.2.2.12 DIAGRAMA DE ESTRUCTURA DE LOS PROCEDIMIENTOS QUE MANEJAN LA VENTANA PRINCIPAL

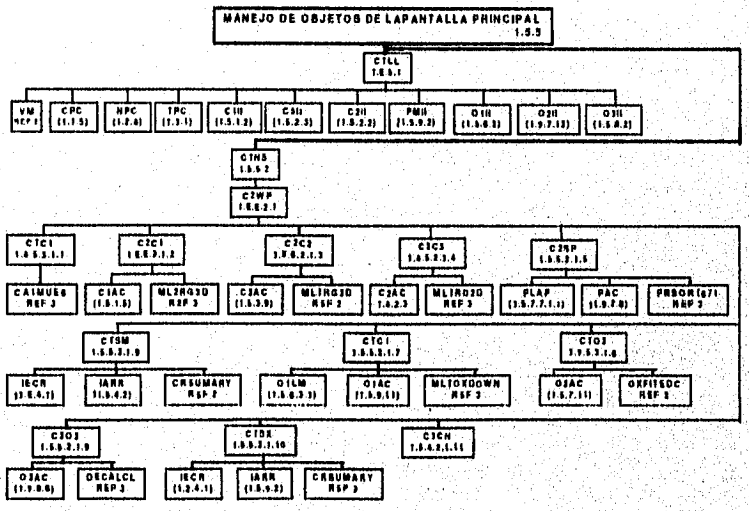


Figura 2.2.2.12. Diagrama de estructura de los procedimientos que manejas los objetos de la ventana principal

2.2.3 DESCRIPCIÓN DE LOS PROCEDIMIENTOS DEL PROGRAMA MPD.

A continuación se incorpora la descripción de cada uno de los módulos que aparecen en los diagramas de estructuras de la sección 2.2.2. Esta sección se subdivide en subsecciones numeradas de acuerdo al diagrama de estructura en la siguiente forma:

1. Habrá una subsección por cada bloque del diagrama de estructuras principal (2.2.2)
2. El número de cada subsección será el del bloque precedido de un 2.2.3. Por ejemplo, la sección 2.2.3.1.1 corresponderá al módulo 1.1; la subsección 2.2.3.1.1.5 corresponderá al módulo 1.1.5.

Cada subsección contiene la descripción del módulo dividida de la siguiente forma:

1. Descripción del Proceso
2. Variables Externas.
3. Variables Globales.
4. Variables locales.
5. Pseudocódigos.

Debido al gran número de procedimientos[1] que conforman todas y cada uno de los programas de MPD, solo se presentaran algunos de estos, ya que de otro modo el número de páginas correspondientes a 2.2.3 serían demasiadas (cercanas a las 100) lo cual crearía una abrumación de información. Si usted quiere consultar las cameterlsticas y pseudocódigos de todos los procedimientos, estos se encuentran disponibles en el documento de Diseño Detallado de MPD.

2.2.3.1. MÓDULO DE PROCESAMIENTO DE DATOS.

2.2.3.1 1 PROGRAMA PARA LA CAPTURA DEL ARCHIVO DEL CATÁLOGO DEL CRUCERO.

La tabla que se muestra a continuación, presenta todas y cada una de la variables globales que se utilizan en el programa para la captura del catálogo del crucero.

VARIABLES GLOBALES	DESCRIPCIÓN
File *p1, p2	Apuntadores a archivos fisicos a usar en la aplicacion
LIST lista_prin	Apuntador a la lista principal (guarda todo el archivo de calalogo de crucero)
LIST lista_esta	Apuntador a la lista de estacion (almacena los datos para una sola estacion)
Widget widget_array [100]	Arreglo de widgets que contienen los id de los objetos de la interfaz
Widget widget_array_datos[230]	Arreglo de widget que contienen los id's de los objetos texto para los datos de estacion
Widget scrollbar_header	widget tipo scrollbar de la ventana de datos de cabecera
Widget scrollbar_win	Widget tipo scrollbar horizontal de la ventana de scroll de datos
Widget scrollbar_win_vertical	Widget tipo scrollbar vertical de la ventana de scroll de datos

[1] Vea 2.2.2

char *a_cac_comet	Apuntador que apunta a la cadena de comentarios que se manda al objeto texto
char cac_coment[400]	Cadena que guarda el contenido de los comentarios que se capturan en el objeto texto
char i_cac_fecha_estacion[10]	Cadena que guarda la fecha de la estacion elegida
char i_cac_long_2 [8]	Cadena que guarda la longitud linea dos de la estacion
char i_cac_hora_local [6]	Cadena que contiene la hora local de la estacion elegida
char i_cac_long_3 [8]	Cadena conteniendo la longitus linea tres de la estacion
char i_cac_lat_2 [8]	Cadena que contiene la latitud linea dos de la estacion elegida
char i_cac_drt [6]	Cadena que contiene la hora DRT de la estacion elegida
char nombre_archivo[70]	Ruta y Nombre del archivo de catalogo de crucero que sera abierto
int nivel_max	Variable que guarda el numero maximo de estaciones para un crucero
int lineas_coment	Numero de lineas de comentarios en los que se grabara o leera del archivo del cac
int total_lineas	Numero con el total de lineas,incluyendo de coment y de datos para la estacion elegida
int num_linea	Registra el numero de linea donde empieza la estacion elegida dentro de la lista principal
int i_numero_niveles	Numero de niveles que componen a cada estacion
char apuntador_de_vaciado_de_datos	Apuntador que sirve para vaciar y leer desde o hacia objetos tipo texto
char direccion [60]	Cadena conteniendo la direccion en disco del archivo cac a abrir

Tabla 2.2.3.1.1. Variables globales que pertenecen al programa para la captura del catalogo del crucero

2.2.3.1.1.5 *cac_proc_crea*

PROCEDIMIENTO:	*cac_proc_crea (w, tag, reason)*	
Descripcion:	Procedimiento que crea un identificado de cada objeto para poder ser referenciados en el programa en "C" Obtiene el elemento scrollbar de las scrollwindow usadas y se les asignan sus correspondientes callbacks	
Variables Locales	**Descripcion**	**Dominio**
int widget_num	Almacena el identificadel del Widget que invoco este procedimiento	Todos los identificadores de widgets (1-100)
int ac	Numero de argumentos que se modifican de un objeto	1<=ac<=10
Arg al[]	parametro que permita sustraer o introducir argumentos de objetos tipo scrollbar	Valoras de 1-10
Variables Globales	widget scrollbar_header, widget scrollbar_win, widget scrollbar_vertical	

```
Empieza cac_proc_crea (w, tag, reason)
        widget_array[widget_num]= tag
        caso (widget_num)
                caso k_cac_scrollwin_header
                Obten  XmNhorizontalScrollBar de objeto k_cac_scrollwin_header
                Añade CallBacks  a scrollbar_header
        caso k_cac_scroll_win
                Obten  XmNhorizontalScrollBar y XmNVerticalscrollBar de objeto k_cac_scroll_win
                Añade CallBacks a scrollbar_win y a scrollbar_win_vertical
        fin caso
```

2.2.3.1.1.7 *cac_busca_estacion*

PROCEDIMIENTO:	*cac_busca_estacion(w, tag, reason)*	
Descripcion:	Procedimiento que realiza la busqueda secuencial dentro del archivo de catalogo de crucero de la estacion capturada en el objeto texto repartiendo los datos en cada uno de los objetos correspondientes, (datos de cabecera, de niveles y comentarios) ademas coloca todos los datos de la estacion elegida en la lista de datos para su posterios uso	
Variables Locales	**Descripcion**	**Dominio**
char i_cac_num_esta[5]	cadena numerica que contiene la estacion elegida	Cualquier caracter numerico 1-max estacion
char i_cac_prof_max[5]	cadena numerica que contiene el valor de profundida maxima	Cualquier caracter numerico
char buff_inter [130]	cadena que se utiliza para leer una linea de datos del archivo fisico del catalogo de cruc	Cualquier cadena de 130 caracteres
char i_cac_num_bot[4]	cadena numerica que contiene el numero de botella o de nivel para la estacion elegida	Cualquier caracter numerico
char *e_estacion	apuntador a caracter que posee el numero de estacion alegida para procesar	Valor de entre 1 hasta max-estacion
int i,j	Variable auxiliarea para incrementar procesos *for ( )*	0<=i,j=>(largo de cadenas a armar)

int fin_arch	bandera que determina si se encontro fin de archivo	cero, uno, dos
Variables Globales	*apuntador de vaciado de datos	
	estructura REC_ESTA nodo rec_esta, rec2_esta	
	estructura REC_HEADER, num_linea, pl (apuntador a archivo), cac_coment	
	i_num_niveles, total_lineas, lineas_coment, a_cac_coment	

```
EMPIEZA cac_busca_estacion
        Obten valor a_estacion de estacion de objeto k_cac_texto_estacion
        Mientras no sea fin de lista principal
                lee primer nodo
                Si el primer caracter es igual a '*'
                        lee numero de estacion en i_cac-num_esta
                        Si i_cac-num_esta es igual a a_estacion
                                fin_arch igual a cero
                        si no
                                fin_arch igual a 1
                        fin (si)
            Fin (si)
      Fin (mientras)
        Si fin_arch es igual a 1
                sal del procedimiento
            Aparece ventana cac_mensaje_no_estacion
        fin (si)
        lee  nodos de cabecera de datos
        identifica cada dato de la cadena leida y vacia e la estructura de datos da cabecera
        vucia datos  de la estructura de datos de cabecera a los objetos de datos de cabecera
        Crea lista de datos de estacion
        lee nodo de la lista principal
        Mientras primer caracter del nodo sea diferente de '*' y de '('
                identifica cada dato de la cadena leida y vacia a la estructura de datos de estacion
                Inserta nodo a la lista de estacion
            llama cac_vaciado_de_objetos_de_estacion()
        fin (mientras)
        llama cac_pon_etiqueta_e_niveles()
        Mientras primer caracter sea iguel a '('
                lee nodo de la lista principal
                concatene nodo leida a cac_coment
        fin (mientras)
    coloca sensitives correspondientes
```

2.2.3.1.1.8 *cac_captura_fecha*

PROCEDIMIENTO:	***cac_captura_fecha(w,tag,reason)***	
Descripcion:	Procedimiento que controla todos los objetos pertenecientes a la caja de dialogo que permite capturar la fecha del crucero . Captura de mes,dia, anio, botones de incremento/decremento ,etc	
Variables Locales	**Descripcion**	**Dominio**
int widget_num	Almacena el identificador del Widget que invoco este procedimiento	Todos los identificadores de widgets (1-100)
Arg al[10]	parametro que permite modificar los argumentos de el objeto DialogBox	Valores de 1-10
int i,j;	Variable auxiliares para incrementar procesos *for ( )*	0<=i,j=>(largo de cadenas a armar)
int jw	Conserva el identificador del objeto al cual se hace referencia en la DialogBox	42<=jw<=49 (objetos de comentario)
char a_fecha_dia[4]	Guarda el valor cadena del el dia de fecha	Caracter numerico
char a_fecha_anio[4]	Conserva el valor cadena del anio de fecha	Caracter numerico
char a_fecha_mes[4]	Guarda el valor cadena de el mes de fecha	Caracteres para mes ENE,FEB,MAR etc
int i_fecha_dia	Valor numerico da dia de fecha	entero de 0 a 31
int i_fecha_anio	Valor numerico de anio de fecha	entero de 80 a 96
Variables Globales	*apuntador_de_vaciado_de_datos	
	i_cac_fecha_esta	

```
EMPIEZA cac_captura_fecha
        caso (widget_num)
      * casa k_cac_row1_col13
                Aparece ventana  cac_caja_captura_fecha
                extrae dia, mes y ano de i_cac_fecha_esta
                Coloca a_fecha_dia , a_fecha-mes y a-fecha_ano en objeto k_cac_fecha_dia_label
        caso k_cac-boton_fecha_dia_up
```

```
                    incrementa i_fecha_dia
                    coloca i_fecha_dia a atributo XmNlabel en objeto k_cac_fecha_dia_label
          caso k_cac_boton_fecha_dia_down
                    decrementa i_fecha_dia
                    coloca i_fecha_dia a atributo XmNlabel en objeto k_cac_fecha_dia_label
          caso k_cac_boton_fecha_ano_up
                    incrementa i_fecha_amo
                    coloca i_fecha_anio a atributo XmNlabel en objeto k_cac_fecha_anio_label
          caso k_cac_boton_fecha_ano_down
                    decrementa i_fecha_anio
                    coloca i_fecha_anio a atributo XmNlabel en objeto k_cac_fecha_anio_label
          caso k_cac_captura_fecha_boton_aceptar
                    concatena a_fecha_dia, a_fecha_mes, a_fecha_anio en fecha_esta
                    coloca fecha_esta en objeto k_cac_row1_col13
                    Desaparece ventana cac_caja_captura_fecha
```

2.2.3.1.1.9 *cac_guarda_estacion_en_lista*

PROCEDIMIENTO:	***cac_guarda_estacion_en_lista(w, tag, reason)***	
Descripcion:	Procedimiento que agrupa todos los datos de todos los objetos de texto y los manda a la lista principal para que, mediante otro procedimiento se genere un nuevo archivo de datos de crucero	
Variables Locales	**Descripcion**	**Dominio**
int i,j	Variable auxiliares para incrementar procesos *for ( )*	0<=i,j=>(largo de cadenas a armar)
char buffer_1[130];	Cadena que concatenara los datos recolectados de los widgets y los asignara a la lista	Cadena numerica cuya strlen<130
	principal	
Variables Globales	i_numero_niveles, lineas_coment, cac_coment	
	estructura REC_PRIN nodo rec3_prin,rec2_prin, indica_arreglo	
	lista_esta,lista_prin (apuntadores a las listas principal y de datos)	

```
EMPIEZA cac_guarda_estacion_en_lista
          Coloca sensitivos de objetos
          (*inserta en  la lista principal el numero de nodos para la estacion elegida*)
          coloca el apuntador de la lista principal donde se insertaran los nodos
          Repite i=0, mientras i sea menor al total_lineas, incrementa i
                    reserva memoria para nodo
                    inserta nodo en blanco en lista principal
          fin (repite)
          Actualiza flaves de la lista principal
          Concatena los valores de los objetos texto a cabecera en buffer_1
          inserta buffer_1 en el primero nodo reservado en la lista principal
          Repite rec_esta=primer registro de la lista de estancias, mientras rec-esta sea diferente de nulo
                    concatena  texto de widget_array_datos[i] en buffer_1
                    concatena  texto de widget_array_datos[i+1] en buffer_1
                    concatena  texto de widget_array_datos[i+2] en buffer_1
                    concatena  texto de widget_array_datos[i+..] en buffer_1
                    concatena  texto de widget_array_ainten[i+18] en buffer_1
                  Inserta buffer_1 como nodo en la lista principal
          fin (repite)
        Borra Lista de estacion
```

2.2.3.1.1.10 *cac_guarda_archivo.*

PROCEDIMIENTO:	***cac_guarda_archivo(w,tag,reason)***	
Descripcion:	Procedimiento que abre un archivo fisico y vacia en el toda la lista principal, creando una nueva version del catalogo de Crucero	
Variables Locales	**Descripcion**	**Dominio**
Variables Globales	i_codbarco, _codcrucero, i_codproyecto, estructura externa *archivos argos*	
	estructura REC_PRIN nodo rec_prin, *p2 (apuntador al archivo)	
	lista_prin (apuntador a la lista principal)	

```
EMPIEZA cac_guarda_archivo
          (*abre archivo fisico para escritura del archivo de catalogo*)
          Abre archivo cac_argos.cac_entrada (p2 ) para escritura
          Repite rec_prin igual a primer nodo, mientras rec-prin sea diferente de nulo
                    escribe el nodo de la lista principal en el archivo p2
        fin (repite)
```

2.2.3.1.1.13 *cac_operaciones*

PROCEDIMIENTO:	*cac_operaciones(w,tag,reason)*	
Descripcion:	Procedimiento que se encarga de controlar las operaciones de insercion, movimiento y borrado de niveles en el area de captura de datos asi como en la lista de estacion.	
Variables Locales	**Descripcion**	**Dominio**
int widget_num	Almacena el identificador del Widget que invoco este procedimiento	Todos los identificadores de widgets (1-100)
int llava	llave almacena el identificador del widget_llave de renglon que sera borrado,insert, etc.	Todos los identificadores de widgets (1-100)
int i,j,	Variable auxiliares para incrementar procesos *for ( )*	0<=i,j=>(numero de objetos para un renglon)
Variables Globales	*apuntador_de_vaciado_de_datos, i_numero_niveles	
	estructura REC_ESTA, nodo rec3_esta,rec2_esta	

```
EMPIEZA cac_operaciones
        caso (widget_num)
        caso k_cac_estacion_inserta_nivel
                reserva memoria para nodo rec3_esta
                iguala rec3_esta a nulo
                llama cac_arregla_lista
                inserta nodo rec3_esta despues de rec2_esta
                incrementa i_numero_niveles
                llama pon_etiquetas_n_niveles
        caso k_cac_estacion_mueve_arriba
                Mueve registro rec2_esta arriba de la lista
        casa k_cac_estacion_mueve-despues
                Aparece ventana cac_caja_mueve_estacion
        caso k_cac_estacion_mueve_despues_acepta
                mueve nodo rec3_esta despues de rec2_esta
        caso k_cac_estacion_mueve_abajo
                mueve nodo rec2_esta hacia abajo de la lista de estacion
        caso k_cac_estacion_borre_nivel
                Borra node rec2_esta
                Actualiza llaves de listn principal
                llama cac_pon-etiqueta a niveles
        fin (caso)
        Rapite rec_esta = primer nodo, mientras rec_esta sea diferente de nulo
                llama cac_vaciado_do_objetos_de_datos()
        fin (repitn)
```

2.2.3.1.2 PROGRAMA PARA LA CAPTURA DE DATOS DE NUTRIENTES

La tabla que se muestra a continuación, presenta todas y cada una de la variables globales que se utilizan en el programa para la captura de datos de nutrientes.

VARIABLES GLOBALES	DESCRIPCION
LIST lista_nut_prin	Lista principal para la aplicacion pues guarda una copia exacta del archivo fisico de
	nutrientes. Cada linea del archivo es un registro en la lista mprincipal
LIST lista_nut_esta	Lista de datos de la estacion. Esta lista guarda todos los datos de nutrientes para
	una sola estacion, teniendo un registro por cada nivel de muestreo de la estacion.
	Cada campo de registro en la lista representa un dato relacionado con nutrientes; ejem
	campo concentrasion, campo absorbancia promedio, campo factor de columna, etc.
rec_lista_prin,rec2_nut_lista_prin	Apuntadores a los registros que se utilizaran para la lista principal.
rec_lista_esta, rec2_nut_lista_esta	Apuntadores los registro que seran usados para la lista de datos de estacion
Widget nut_widget_array [100]	Arreglo de widgets que contienen los id de los objetos de la interfaz
Widget nut_widget_array_datos	Arreglo de widget que contienen los id's de los objetos texto para los datos de estacion
Widget nut_scrollbar_header	widget tipo scrollbar de la ventana de datos de cabecera
Widget nut_scrollbar_win	Widget tipo scrollbar horizontal de la ventana de scroll de datos
Widget nut_scrollbar_win_vertic	Widget tipo scrollbar vertical de la ventana de scroll de datos

File *nut_p1, *nut_p2	Apuntadores a los archivos fisicos de cac y de nutrientes
char apuntador_de_vaciado_de_datos	Apuntador que sirve para vaciar y leer desde o hacia objetos tipo texto
char nut_anio[3]	Cadena que contiene el anio del crucero a procesar
char nut_sigla [3]	Cadena que contiene la siglas del crucero cuyo archivo cac queremos abrir
char nut_version [2]	Cadena que contiene la version del crucero a trabajar
char nut_nombre_archivo_cac	Cadena que guarda el nombre del archivo de catalogo que abrira para la aplicacion
char nut_archivo_nutrientes	Cadena que guarda el nombre del archivo de nutrientes a abrir para la aplicacion
int nut_nivel_max	Variable que guarda el numero maximo de estacionee para el crucero a procesar
int nut_posicion_de_encuentro	Registra el numero de linea donde empieze la estacion elegida dentro de la lista principal
char nut_coment[550]	Cadena que almacena los comentarios correspondientes para cada estacion
int nut_ultime_rec	Registra el numero de registro de la lista principal
int i_nut_numero_niveles	Registra el numero de niveles que posee cada estacion
int nut_lineas_coment	Registre el numero de lineas de comentarios que se registran para la estacion
int nut_lineas_coment_before	Numero total de lineas de comentarios antes de capturar nuevos comentarios
int nut_total_lineas	Numero total de lineas pertenecientes e una estacion, incluyendo titulo y comentarios
int nut_indice_datos	Indice de arreglo de widgets de datos
int nut_num_linea_coment	Valor que almacena el ki del registro donde se encontro el inicio de los comentarios

Tabla 2.2.3.1.2. Variables globales que pertenecen al programa para la captura de datos de nutrientes

2.2.3.1.2.1 *nut_proc_llama*

PROCEDIMIENTO:	*nut_proc_llama (w,tag,reason)*	
Descripcion:	Procedimiento que realiza las llamades a la ventanas de capturas de nutrientes, sean silicatos, fosfatos o nitratos y a la ventana de captura de comentarios. Los numeros de niveles para cada estcion se introduces en el objeto lista de niveles	
Variables Locales	**Descripcion**	**Dominio**
int widget_num	Almacena el identificedor del Widget que invoco este procedimiento	Todos los identificadores de widgets (1-100)
XmString item_nutriente_entr	Tipo compound que guarda los item que se incorporeran al objeto lista de niveles	Cadena con numero de nivels valido (1-12)
char nut_nivel [5]	Tipo char que alamcena en valor del item elegido del objeto lista de niveles	Cadeba con numero de niveles validos (1-12)
Variables Globales	*apuntador_de_vaciado_de_datos, nut_coment, Lista lista_nut_esta	
	Estructura REC_NUT_LISTA_ESTA noda rec_nut_lista_esta	

```
EMPIEZA nut_proc_llama
        caso (widget_num)
        caso k_nut_boton_anade_estacion
                Repite /oe_nut_lista_esta igual a primer nodo, mientras rec_nut_lista_esta sea diferente de ultimo nodo
                        Inserta rec_nut_lista_esta->id_nut_nivel_esta al objeto lista k_nut_caja_lista_esta
                fin(repite)
        caso k_nut_caja_boton_edita_coment
                Aparece ventana nut_coment
                Pon nut_coment en objeto k_nut_coment_text
        fin(caso)
```

2.2.3.1.2.3.1 *nut_busca_estacion_en_nut*

PROCEDIMIENTO:	*nut_busca_estacion_en_nut()*	
Descripcion:	Busca en la lista principal el numero de estacion elegida, una vez encontrade se extraen sus datos y se forma la liste de eetacion. Tambien se leeran la lineas de comentarios en caso de que existan	
Variables Locales	**Descripcion**	**Dominio**
int i2,,j,i;	Variable auxiliares para incrementar procesos *for ( )*	i2,,j,i>=0
char i_nit_num_esta[10]	Cadena que alamcena el numero de estacion que se lee en la lista principal	Cualquier cadena numerica con num de estacion
char i_nit_num_bot[10]	Caden que almacena el numere de botella o nivel para cada estacion de la lista prin	Cualquier cadena numerica con num de botella
Variables Globales	Estructura REC_NUT_LISTA_ESTA node rec_nut_lista_esta	
	Lista lista_nut_prin, *apuntador_de_vaciado_de_datos, nut_indice_datos	
	num_total_lineas, nut_lineas_coment_before, nut_num_linea_coment	
	nut_coment, *nut_p2 (apuntador al archivo de nutrientes)	

```
EMPIEZA nut_busca_estacion_en_nut
        Obten valor apuntador_de_vaciado_de_objetos de estacion de objeto k_nut_texto_estacion
        Mientras no sea fin de lista_nu_prin
                lee primer nodo
                Si el primer caracter es igual a '*'
                        lee numero de estacion en i_nut_num_esta
                        Si i_cac_num_esta es igual a apuntador_de_vaciado_de_objetos
                                fin_arch igual a cero
                        si no
                                fie_arch igual a 1
                        fin (si)
            Fin (si)
    Fin (mientras)
        Si fin_arch es igual a 1
                sal del procedimiento
            Aparece ventana nut_mensaje_no_estacion
        fin (si)
        Crea lista de datos de estacion (lista_nut_prin)
        lee nodo de la lista principal
        Mientras primer caracter del nodo sea diferente de '*' y de '('
                Identifica cada dato de la cadena leida y vacia a la estructura rec_nut_lista_esta
                Inserta nodo a la lista lista_nut_prin
                llama nut_vaciado_de_objetos_de_datos(8)
        fin (mientras)
        llama nut_pon_etiqueta_a_niveles()
        Mientras primer caracter sea igual a '('
                lee nodo de la lista principal
                concatena nodo leido a nut_coment
        fin (mientras)
    coloca sensitives correspondientes
```

2.2.3.1.2.6.2 *nut_abre_nutricates.*

PROCEDIMIENTO:	*nut_abre_nutrientes()*	
Descripcion:	Procedimiento que abre el archivo de nutrientes, si es que este existe, incluye todo el archivo en lo que llamamos lista principal, identifica el numero de estaciones, y el numero de estacion maxima, pasando este datos al objeto SCALE.	
Variables Locales	**Descripcion**	**Dominio**
char archivo [50]	cadena que contiene el nombre del archivo de catalogo de cruicero a ser abierto	cualquier cadena con nombre de archivo valido
int f2,,j,l;	Variable auxiliares para incrementar procesos *for ( )*	f2,,j,l>=0
char buffer [130]	cadena que se utiliza pa leer un renglon del archivo fisico	cualquier cadena que se lea del archivo de cac
Variables Globales	*nut_p2 (apuntador al archive fisico de nutrientes), Lista lista_nut_prin	
	nut_ultimo_rec, Estructura REC_NUT_LISTA_ESTA node rec_nut_lista_esta	

```
EMPIEZA nut_abre_nutrientes
        (*incluye archivo a lista principal*)
            Abre Archivo fisico (archivo)
            Crea lista lista_nut_prin
            Mientras no sea fin de archivo (nut_p2)
                    reserva memoria para nodo
                    lee linea de archivo
                    inserta linea de archivo como un nodo de la lista
            Fin mientras
            Coloca Sensitives de Widgets
```

2.2.3.1.2.7 *nut_graba_en_lista_prin.*

PROCEDIMIENTO:	*nut_graba_en_lista_prin( )*	
Descripcion:	Procedimiento que acepta toda la lista de estacion ya modificada, incorporandola a la lista principal, exactamente en el mismo lugar de donde fue extraida.	
Variables Locales	**Descripcion**	**Dominio**
int h1,h,l;	Variable auxiliares para incrementar procesos *for ( )*	h1,h,l>=0
char buffer_1[130]	Se utiliza para "partir" la cadena de comentarios en cadenas mas pequenas	Cualquier cadena que contenga comentarios
Variables Globales	cursor, nut_indice_datos, rec2_nut_lista_esta, nut_coment	
	nut_lineas_coment_before, nut_num_lines_coment, Lista lista_nut_esta	
	Estructura REC_NUT_LISTA_ESTA node rec_nut_lista_esta,	

```
EMPIEZA nut_grabo_en_lista_prin
        Posiciona rec_nut_lista_prin en lista_nut_prin segun llave igual a nut_posicion_de_encuentro
        Repite rec_nut_lista_esta = primer nodo, mientras rec_nut_lista_esta no sea diferente de ultimo nodo
                Llama nut_arma_prin_buffer()
                Llama nut_vaciado_de_objeto_de_datos()
        fin(repite)
        Posiciona en el primitivo nodo de lista_nut_prin a rec_nut_lista_prin
        Actualiza llaves de lista_nut_prin
```

2.2.3.1.2.8 *nut_acepta_abs*

PROCEDIMIENTO:	*nut_acepta_abs()*	
Descripcion:	Procedimiento que contiene los algoritmos para el calculo de los correspondientes nutrientes, asi como el manejo de todos los objetos involucrados para este calculo. Aqui se manda validar los datos, si alguno es invalido el procedimiento se cancela	
Variables Locales	**Descripcion**	**Dominio**
int *j,i,*	Variable auxiliares para incrementar procesos *for ( )*	j,i>=0
float nut_prom	Conserva el absorbancia promedio	0<=nut_prom <=0.5
float float_abs	Variable para leer las absorbancias de los objetos texto que las contienen	0<=float_abs<=0.5
char char_float_abs[15]_	Cadena que guarda los valores de las operaciones que se van realizando	Cadena numerica con valores validos
Variables Globales	*apuntador_de_vaciado_de_datos*	
	Estructura REC_NUM_LISTA_ESTA node rec_num_lista_esta	
	nut_widget_array	

```
EMPIEZA nut_acepta_abs(w, cosmico, tag)
        Llama NUT_VALIDA_CARACTER (k_nut_caja_edita_texto_abs1)
        Llama NUT_VALIDA_CARACTER (k_nut_caja_edita_texto_abs2)
        Llama NUT_VALIDA_CARACTER (k_nut_caja_edita_texto_abs3)
        Llama NUT_VALIDA_CARACTER (k_nut_caja_edita_texto_error)
        Repite i=35, mientras i sea menor o igual a 37, incrementa i
                Obten float_abs en CapturaValorFloatDeWidgetTxt(nut_widget_array(i))
                NUTVALIDAVALOR(nut_widget_array(i))
                nut_prom es igual a nut_prom + float_abs
                incrementa j en 1
        fin(repite)
      obten rec_nut_list_esta->i_nut_abs_prom AsignaValorFloatAWidget_Txt (nut_prom/j,k_nut_caja_edita_texto_absprom)
        NUTVALIDAVALOR (k_nut_caja_edita_texto_error) en rec_nut_list_esta->i_nut_abs_error
        obten (rec_nut_list_esta->i_nut_abs_corr de (rec_nut_list_esta->i_nut_abs_prom - rec_nut_list_esta->i_nut_abs_error)

        (**Pseudocodigo para concentracion de silicatos segun**)
                (**determina absorbancia real**)
        rec_nut_lista_esta->i_nut_abs_real = ((rec_nut_lista_esta->i_nut_abs_corr)*(1.0+(0.0028*rec_nut-lista_esta
                                                                                ->i_nut_salinidad)))

        Si Toggle k_nut_caja_toggle_colda10 esta encendido
                        rec_nut_lista_esta->i_nut_abs_real = (rec_nut_lista_esta->i_nut_abs_real/10)
        fin(si)
                (*determina concentracion de silicatos*)
        rec_nut_lista_esta->i_nut_concentra =  (94.727*(rec_nut_lista_esta->i_nut_abs_real))- 0.4833
        Pon rec_nut_lista_esta->i_nut_concentra en objeto k_nut_caja_edita_texto_conmcentra
        (**Pseudocodigos para concentracion de fosfatos segun**)
        rec_nut_lista_esta->i_nut_concentra = (4.386 * rec_nut_lista_esta->i_nut_abs_real)
        Pon rec_nut_lista_esta->i_nut_concentra en objeto k_nut_caja_edita_texto_conmcentra
        (**Pseudocodigos para concentracion de nitratos segun**)
                (*checar tipo de celda*)
        Si Toggle k_nut_caja_toggle_celda10 esta encendido
                rec_nut_lista_esta->i_nut_abs_corr = (rec_nut_lista_esta->i_nut_abs_corr/10)
        fin(si)
                (*checar disolucion*)
        Si Toggle k_nut_caja_toggle_dilucion esta encendido
                rec_nut_lista_esta->i_nut_abs_corr = (rec_nut_lista_esta->i_nut_abs_corr*2)
        fin(si)
        rec_nut_lista_esta->i_nut_concentra = rec_nut_lista_esta->i_nut_factor* rec_nut_lista_esta->i_nut_abs_corr
        Pon rec_nut_lista_esta->i_nut_concentra en objeto k_nut_caja_edita_texto_conmcentra
```

2.2.3.1.2.13 *nut_acepta_archivo*

PROCEDIMIENTO:	*nut_acepta_archivo (valor, widget)*	
Descripcion:	lee toda la lista principal y la pasa a un archivo fisico en disco, creando una nueva version del archivo de nutrientes	
Variables Locales	**Descripcion**	*Dominio*
Variables Globales	nut_archivo_nutrientes, lista nut_lista_prin	
	estructura REC_NUT_LISTA_PRIN node rec_nut_lista prin	
	*nut_p2 (apuntador a archivo fisico de nutrientes)	

```
EMPIEZA nut_acepta_archivo
        Abre archivo fisico nut_p2
        Repite rec_nut_lista_prin = Primer nodo, mientras rec_nut_lista_prin sea al ultimo nodo
                copia nodo de lista_nut_prin a nut_p2
        fin (Repite)
    Cierra nut_p2
```

2.2.3.1.3 PROGRAMA PARA LA CAPTURA DE DATOS DE TRANSECTOS.

La tabla que se muestra a continuación, presenta todas y cada una de la variables globales que se utilizan en el programa para la captura de datos de transectos

VARIABLES GLOBALES	DESCRIPCION
Widget tra_widget_array[90]	Arreglo de widgets que contienen los id de los objetos de la interfaz
File *pl	Apuntador a archivo fisico de transectos a usar en la aplicacion
LIST tra_lista_prin	Lista donde se almacena todo el archivo de transectos a utilizar, teniendo como campo de la lista un renglon de este archivo fisico
*tra_rec2_prin, *tra_rec_prin	Apuntadores a los registros que se utilizaran en la lista TRA_LISTA_PRIN
char apuntador_de_vaciado_de_datos	Apuntador que sirve para vaciar y leer desde o hacia objetos tipo texto
char anio[3]	Cadena que contiene el anio del crucero a procesar
char sigla [3]	Cadena que contiene la siglas del crucero cuye archivo cac queremos abrir
char version [2]	Cadena que contiene la version del crucero a trabajar
char archivo_transecto[70]	Nombre del archivo de transectos con el cual quremos trabajar
char tra_datos_para_edicion[500]	Se utiliza para almacenar la cadena del objeto texto para captura de estaciones
int tra_pos_item	Guarda la posicion del item elegido en el objeto lista de transectos
int num_estaciones	Numero de estaciones capturadas para un transecto espesifico
int i_tran_arriba	almacena el indice del identificador del transecto
int tra_anade_transecto	Bandera que indica si se van a anidir nuevos transectos al archivo.
XmString valor_item;	Almacena el valor del item elegido dentro del objeto lista de transectos
int no_existe_estacion	Bandera que indica si la esacion capturada existe o no.
int tra_archivo_nuevo	Bandera que indica si se creara un nuevo archivo de transectos o no

Tabla 2.2.3.1.3. Variables externas del programa de captura de datos de transectos

2.2.3.1.3.1 *tra_crea_proc*

PROCEDIMIENTO:	*tra_crea_proc (w,tag,reason)*	
Descripcion:	Procedimiento que crea un identificador para cada objeto de la interfaz utilizado en la aplicacion de captura de transectos	
Variables Locales	**Descripcion**	**Dominio**
int widget_num	Almacena el identificador del Widget que invoco este procedimiento	Todos los identificadores de widgets (1-100)
char dir_jstra[70]	Almacena la direccion completa de los archivos a abrir	Cadena tipo direccion
char archivo_catalogo[100]	Almacena el nombre del archivo del catalogo del crucero que bedera ser abierto	Cadena con cualquier nombre de archivo
int tra_operacion_a_realizar	Valor que identifica el modo de trabajo para el programa de captura de transectos (altas, bajas, cambios, consultas)	mayor o igual a 1 menor o igual a 4

```
EMPIEZA tra_crea_proc ( w,tag,reason)
        Inserta w en tra_widget_array[widget_num]
        Concatena en dir_jdka (codbarco,"0",codcrucero,"d00",codcrucero," cac]")
        Concatena en archivo_catalogo (codbarco,"0",codcrucero,"d00",codcrucero," cac")
        Llama tra_abre_archivo_catalogo(archivo_catalogo)
        case (tra_oporacion_a_realizar)
        caso 1
                Prende objeto k_tra_boton_anade_transecto
        ...
        fin(caso)
        Lama tra_abre_archivo_transectos(archivo_transectos)
```

2.2.3.1.3.1.1 *tra_abre_archivo_transectos*

PROCEDIMIENTO:	*tra_abre_archivo_transectos(arch)*	
Descripcion:	Procedimiico que apartir de la variables globales, forma el nombre del archivo de transectos a abrir, e introduce todo el archivo a la lista principal	
Variables Locales	**Descripcion**	**Dominio**
int i,j;	Variable auxiliares para incrementar procesos *for ( )*	i,j >=0
Variables Globales		
	*pl ¡apuntador a archivo de transectos), estructura TRA_REC_PRIN nodo tra_rec_prin	

```
EMPIEZA tra_abre_archivo_transectos(arch)
        Abre archivo fisico para lectura pl(arch)
        Si no se puede abrir
                Abre archivo fisico para lectura pl(nuevo)
        fin(si)
        Creo lista principal tra_lista_prin
        Mientras no sea fin de archivo
                lee linea de archivo (pl) en buffer
                reserva memoria para nodo tra_rec_prin
                Inserta buffer como nodo a tra_lista_prin
        fin(mientras)
        Llama tra_incluye_transectos()
        cierra pl
```

2.2.3.1.3.3 *tra_selecciona_transecto*

PROCEDIMIENTO:	*tra_selecciona_transecto (w, client_data, call_data)*	
Descripcion:	Procedimiento que coloca el apuntador de la lista principal segun el item elegido en el objeto lista de transectos	
Variables Locales	**Descripcion**	**Dominio**
int i,j;	Variabla auxiliares para incrementar procesos *for ( )*	i,j >=0
char pos[5]	cadena que almacena la pesicion dentro del objeto lista del item elegido	cualquier cadena numerica con posicion valida
Variables Globales	*apuntador_de_vaciado_de_datos, tra_pos_item, arreglo_transectos[]	
	estructura TRA_REC_PRIN node tra_rec2_prin, lista tra_lista_prin	

```
EMPIEZA tra_selecciona_transecto ( w, client_data, call_data)
        Obten en call_data->item el item seleccionado de la lista de traosectos
        (*tran_datos.tran_tarnsectos es parte de la estructura qua recibe maricela y jou para sus aplicacionos*)
        Iguala el id del transecto elegido a tra_datos.tran_transecto
        Obten en call_data->item_position la posicion di item alegido
        Posiciona tra_rec_prin en tra_lista_prin segun llave call_data->itero_positioo
        Llama tra_checa_fin_transecto()
        Llama tra_vacia_datos_es_objeto_texto()
```

2.2.3.1.3.3.1 *tra_checa_fin_transecto.*

PROCEDIMIENTO:	*tra_checa_fin_transecto ( )*	
Descripcion:	Procedimiento que checa las lineas en la lista principal que pertenecen a determinado transecto, una vez que estas se determinan se conforman en un solo buffer y se mandan a la caja de captura de estaciones	
Variables Locales	**Descripcion**	**Dominio**
int i,j;	Variable auxiliares para incramentar procesos *for ( )*	i,j >=0
Variables Globales	estructura TRA_REC_PRIN node tra_rec2_prin	
	lista tra_lista_prin	

```
EMPIEZA tra_checa_fin_transecto ( )
        Repite i=0, mientras j sea diferente de cero, incrementa i
                tra_rec2_prin es igual al siguiente nodo de tra_lista_prin
                Si tra_rec2_prin->tra_buffer es diferente de "fin de transecto"
                        concatena tra_rec2_prin->tra_buffer a tra_datos_para_edicion
                fin(si)
        fin(repite)
```

2.2.3.1.3.4 *tra_salva_en_lista*

PROCEDIMIENTO:	***tra_salva_en_lista (w,tag,reason)***	
Descripcion:	Procedimiento que guarada en la lista de datos principal las modificaciones que se hallan registrado en la captura de numera de estaciones para un transecto espesifico, o si se anadio o borro un transecto. Aqui se validan tambien los numeros de estaciones.	
Variables Locales	**Descripcion**	**Dominio**
int widget_num	Almacena el identificador del Widget que invoco este procedimiente	Todos los identificadores de widgets (1-100)
char tra_num_transecto	Cadena que guarda el identificador del item da transecto qua es elegido	cadena numerica respresentando id de transecto
char buff_datos	Cadena auxiliar para almacenar el valor del objeto texto de captura de estaciones	Cualquier cadena con caracteres validos
char cadena_correcta	Cadena donde se almacena los datos ya corregidos de los numeros de estaciones	Cualquier cadena numerica con estaciones valida
int objete_extrano	Bandera que indica si existieron caracteres invalido o no en la cadena capturada	0(caracter invalido), 1 (caracter valido)
int i,j;	Variable auxiliares para incrementar procesos *for ( )*	i,j >=0
int num_lineas	valor entero con el numero de lineas en que se almacenara la cadena en la lista prin.	1<= num_lineas <=10
char dato[5]	Cadena que va almacenado el numero de estacion leido de la cadena de datos	cadena numerica con estacion valida
char mensaje_no_estac.	Cadena qua se forma cuando no existe una estacion capturada , esta cadena contiene el numero de estacion que no se encontro y se manda a la caja de dialogo	cadena numerica con numere de estacion
Variables Globales	*apuntador_de_vaciado_da_datos, Lista tra_lista_prin, no_existe_estacion	
	Estrcutura TRA_REC_PRIN nodo tra_rec_prin, num_estaciones,	
	tra_pos_item, num_transecta, tra_anade_transectos	

```
EMPIEZA tra_salva_en_lista ( w,tag,reason)
        Obten buff_datos de objete texto k_tra_caja_captura_datos_texto
        Repite i=0, mientras i sea menor a longitud de cadena de buff_datos (*valida caracteres invalidos*)
                Si caracter i de buff_datos es diferente de digito,"," o " "
                        bandera objete_extrano = 1
                fin(si)
        fin(repita)
        Repita i =1, mientras i sea igual al numero de estaciones en buff_datos
                Extrae estacion de buff_datos
                tra_busca_estacion_en_catalogo(esta) (*coteja que cada estacion exista en el archivo de crucero*)
                Si bandera no_existe_estacion = 1
                        aparece ventana mensaje_no_estacion
                        sale del procedimiento
                fin(si)
        Anade estacion a objeto k_tra_lista_datos
        fin(repita)
        (*se va a modificar los datos de un transecto ya existente*)
        Si banderas tra_anade_transecto, objeto_extrano y tra_archivo_nuevo son diferentes de 1
                Borra nodos del transectos anterior
                Inserta nodos del transecto corregido
                Actualiza id de la lista (vincipal
                Reemplaza nodos con datos actualizados
        fin(si)
        (*se va a anadir un nuevo transecto a la liste de tranaectos*)
        Si tra_anade_transecto es diferente de cero
                Inserta al final de la liste principal el 3 nodos
                Substituye nodos con datos actualizados
                Borra items de k_tra_lista_transectos
                Llama tra_incluye_transectos()
        fin(si)
```

```
                (*se creara un nuevo archivo de transectos*)
                Si tra_nuevo_archivo es diferente de 0
                        Crea lista tra_lista_prin
                        Inserta al final de la lista principal el 3 nodos
                        Subsituye nodos con datos actualizados
                        Borra items de k_tra_lista_transectos
                        Llama tra_incluye_transectos()
                fin(si)
                desaparece ventana tra_caja_captura_datos_tran
```

2.2.3.1.5.1 CALIBRACIÓN DE CONDUCTIVIDAD (Primera Etapa)

La tabla que se muestra a continuación, presenta todas y cada una de la variables globales que se utilizan en el programa para la calibración de la conductividad en su primera etapa.

VARIABLES GLOBALES	DESCRIPCION
File *p_smcc1a	Apuntador a un archivo fisico para el maneja de ficheros de entrada/salida
int pos_item_inter	Valor que guarda la posición de un item perteneciente al objeto tipo lista de intercambios
int pos_item_regre	Valor que guarda la posicionde un item perteneciente al objeto tipo lista de regresiones
int num_cambios	Entero que almacena el numero de items del objeto tipo lista de intercambios
int num_regresion	Entero que almacena el numero total de items del objeto tipo lista de regresiones
char ch_regresion[60]	Cadena que guarda un item que sera introducido al objeto tipo lista de regresion
char ch_inter[60]	Cadena que guarda un item que será incluido al objeto lista de intercambios
char ch_indice_de[5]	Almacena el id de la variable en el objeto option_menu_de
char ch_indice_pro[5]	Almacena el id de la variable en el objeto option_menu_por

tabla 2.2.3.1.5.1. Variables globales para los procedimientos que realizan la calibración de conductividad en su primera etapa.

2.2.3.1.5.1.1 *smcc1a_crea_prac*

PROCEDIMIENTO:	*smcc1a_crea_proc (w, tag, reason)*	
Descripcion:	Crea un identificador para cada widget de la interfaz en la aplicacion, guardado estos al arreglo smcc1a_widget_num.	
Variables locales	Descripcion	Dominio
int widget_num	Almacena el identificador del Widget que invoca este procedimiento	Todos los identificadores de widgets (1-100)
Variables globales	smco1a_widget_num	

```
Empieza smcc1a_crea_proc (w, tag, reason)
        iguala smcc1a_widget_num[widget_num] a w
fin procedimiento
```

2.2.3.1.5.1.2 *smcc1a_selecciona_lista_inter(w, client_data,call_data)*

PROCEDIMIENTO:	*smcc1a_selecciona_lista_inter(w, client_data,call_data)*	
Descripcion:	Procedimiento que maneja los datos del objeto k_smcc1a_liste_intercambia, se toma la cadena que se lee del item, se subdivide en sus componentes, y se manda cada uno a su correspondiente objeto.	
Variables locales	Descripcion	Dominio
regre [5][60]	Arreglo de cadenas que guarda cada dato que compone a un item	arreglo de cadenas
Variables globales	apuntader_de_vaciado_de_datos	
	pos_item_inter, ch_inter	

```
Empieza smcc1a_selecciona_lista_inter(w, client_data,call_data)
        copia call_data->item a apuntador_de_vaciado_de_datos
        reparte apuntador_de_vaciado_de_datos a arreglo regre[ ][ ]
        reparte cada datos de regre[ ][ ] a sus correspondientes objetos
fin procedimiento
```

fin procedimiento

2.2.3.1.5.1.2.2 *smcc1a_determina_variable (opcion, var)*

PROCEDIMIENTO:	***smcc1a_determina_variable (opcion, var)***	
Descripcion:	Funcion que recibe el id de la variable a minipular y el id del optio menu correspondiente y regrasa el id del objeto pushbutton que deberá ser desplegado en el option_menu	
Parámetros	**Descripcion**	**Dominio**
int option	Identificador del option_menu que se maneja	1<= option<=4
int var	Identificador de la variable a manipular	1<= var <=8
Variables locales		
int regreso	Id del pushbutton elegido que sera devuelto	

```
Empieza smcc1a_determina_variable (opcion, var)
        caso (var)
                caso PRESION
                        caso (option)
                                caso 1
                                        copia id de buttoo a regreso
                                ...
                                caso 6
                                        capia id de button a regreso
                        fin (caso)
                ....
                caso ESTACION
                ...
        fin caso
fin procedimiento
```

2.2.3.1.5.1.3 smcc1a_incluye_archivo_control(control)

PROCEDIMIENTO:	***smcc1a_incluye_archivo_control(control)***	
Descripcion:	Procedimiento qus incluye al archivo de control generalla parte correspondiente a la calibracion de conductividad en su primera etapa	
Parametros		
*FILE control	Apuntador el archivo de control general	
Variables Locales	**Descripcion**	**Dominio**
char buffer[100]	Cadena que serlvira para guardar en *control una linea de datos para el archivo de control general	Cualquier cadena
Variables Globales	(todas las estructuras)	

```
Empieza smcc1a_incluye_archivo_control(control)
        copia "[CALIBRACION_CONDUCTIVIDAD_1a_ETAPA]" a buffer
        incluye buffer a control
        ...
        (formato "")
        ...
        copia "    [SALIDA_RAMA_5]   (2) \n"
        incluye buffer a control
        cierra control
fin (procedimiento)
```

2.2.3.1.5.1.4.1 *smcc1a_obten_texto_de_opcion (id)

PROCEDIMIENTO:	***smcc1a_ obten_texto_de_opcion (id)***	
Descripcion:	Procedimiento que obtiene la etiqueta del pushbutton que se encuentre en ese momento visualizada en el option_menu.	
Parámetros	**Descripcion**	**Dominio**
int id	identificador del objeto option_menu el cual queremos determinar su etiqueta	
Variables locales		
XmString cad_string	Cadena tipo xmstring donde reside la "label " del option menu elegido	
Variables Globales		
	widget smcc1a_menu_history	
	apuntador_de_vaciado_de_datos	

```
Empieza smcc1a_obten_texto_de_opcion (id)
        Obten smcc1a_menu_history de smcc1a_widget_array[id]
        Obten XmNlabelString de smcc1a_menu_history en cad_string
        regresa cad_string
fin procedimiento
```

2.2.3.1.5.1.12 *smcc1a_inserta_listas(w, tag, reason)*

PROCEDIMIENTO:	*smcc1a_inserta_listas(w, tag, reason)*	
Descripcion:	Procedimiento que inserta un nuevo item a un objeto lista en particular	
Variables locales	**Descripcion**	**Dominio**
int widget_num	Identificador del widget que invoco al procedimiento	
XmString arma_string	Cadena tipo string que se insertara en el objeto lista como item modificado	Caden tipo string
Variables Globales	apuntador_de_vaciado_de_datos	
	num_regresion, num_cambios, pos_item_regre, pos_item_inter	

```
Empieza smcc1a_inserta_procs(w, tag, reason)
        caso (widget_num)
                caso k-_smcc1a_boton_intercambia
                        arma item a insertar en arma_string
                        incrementa num_cambios en 1
                        anade a k_smcc1a_lista_intercambia item arma_string
                caso k-_smcc1a_boton_regresion_acepta
                        arma item a insertar en arma_string
                        incrementa num_regresion en 1
                        anade a k_smcc1a_lista_regresion item arma_string
        fin (caso)
fin (procedimiento)
```

2.2.3.1.5.1.14.1 *smcc1a_inserta_en_estructuras(branch, nodos, calfile)*

PROCEDIMIENTO:	*smcc1a_inserta_en_estructuras(branch, nodos, calfile)*	
Descripcion:		
Parametros		
*FILE calfile	Apuntador al archivo de control general	
int nodos	numero de nodos que se reciben del archivo de control general	
char branch[5]	numero de rama que se recibe del archivo de control general	
Variables Locales	**Descripcion**	**Dominio**
char datos[10][15]	arreglo de cadenas que ayuda a leer las lineas del archivo de control	

```
Empieza smcc1a_inserta_en_estructuras(branch, nodos, calfile)
        caso (branch)
                lee cadena  buffer de calfile
                divide cadena buffer en datos
                reparte datos en estructura_smcc1a_nodo
        fin (caso)
fin (procedimiento)
```

2.2.3.1.5.2 CALIBRACIÓNDE CONDUCTIVIDAD (Segunda Etapa)

La tabla que se muestra a continuación, presenta todas y cada una de la variables globales que se utilizan en el programa para la calibración de la conductividad en su segunda etapa.

VARIABLES GLOBALES	DESCRIPCION
File *p_smcca	Apuntador a un archivo fisico para el manejo de ficheros de entrada/salida
int pos_item_edita	Valor que guarda la posición de un item perteneciente al objeto tipo lista de edicion
int pos_item_termi	Valor que guarda la posicionde un item perteneciente al objeto tipo lista de terminos
int pos_item_histo	Valor que guarda la posición de un item perteneciente al objeto tipo lista de histogramas

int pos_item_prome	Valor que guarda la posicionde un item perteneciente al objeto tipo lista de promedios
int num_ediciones	Entero que almacena el numero de items del objeto tipo lista de ediciones
int num_terminos	Entero que almacena el numero total do items del objeto tipo lista de terminos
int num_histogramas	Entero que almacena el numero de items del objeto tipo lista de histogramas
int num_parametros	Entero que almacena el numero total de items del objeto tipo lista de parametros
char ch_indice_de[5]	Almacena el id de la variable en el objeto option_menu_de
char ch_indice_pro[5]	Almacana el id de la variable en el objeto option_menu_por

tabla 2.2.3.1.5.2. Variables globales para los procedimientos que realizan la calibración de conductividad en su segunda etapa.

2.2.3.1.5.2.2 *smcc2a_selecciona_lista_edita(w, client_data,call_data)*

PROCEDIMIENTO:	***smcc2a_selecciona_lista_edita(w, client_data,call_data)***	
Descripcion:	Procedimiento que maneja los datos de los objeto tipo lista k_smcc2a_lista_edita, k_smcc2a_lista_termi, k_smcc2a_lista_histo y k_smcc2a_lista_prome; se toma la cadena que se lee del item, se subdivide en sus componentes, y se manda cada uno a su correspondiente objeto.	
Variables locales	**Descripcion**	**Dominio**
regre [5][60]	Arreglo de cadenas que guarda cada dato que compone a un item	arreglo de cadenas
Variables globales	apuntador_de_vaciado_de_datos	
	pos_item_edita, pos_item_histo, pos_item_termi, pos_item_prome, ch_inter	

```
Empieza smcc2a_selecciona_lista_edita(w, client_data,call_data)
        copia call_data->dsm a apuntador_de_vaciado_de_datos
        reparte apuntador_de_vaciado_de_datos a arreglo regre[ ][ ]
        reparte cada datos de regre[ ][ ] a sus correspondientes objetos
fin procedimiento
```

2.2.3.1.5.2.10 *smcc2a_saca_pantalla_prin()*

PROCEDIMIENTO:	***smcc1a_saca_pantalla_prin()***
Descripcion:	Procedimiento que se activa el cancelar la ventana principal

```
Empieza smcc2a_saca_pantalla_prin()
        Llama Vuit_unmenage("smcc2a_main_win")
fin (procedimiento)
```

2.2.3.1.5.2.11 *smcc2a_nuevos_proc(w, tag, reason)*

PROCEDIMIENTO:	***smcc2a_nuevos_proc(w, tag, reason)***	
Descripcion:	Procedimiento que anade un nuevi item al objeto lista seleccionado	
Variables locales	**Descripcion**	**Dominio**
int widget_num	Identificsdor del widget que invoco al procedimiento	

```
Empiaza smcc2a_nuevos_proca(w, tag, reason)
        caso (widget_num)
                caso k-_smcc1a_boton_edita_nueva
                        coloca "" a todos los objetos texto
                caso k-_smcc1a_botcn_termi_nueva
                        coloca "" a todos los objetos texin
                caso k-_smcc1a_botonhisto__nueva
                        cnloca "" a todos los objetos texto
                caso k-_smcc1a_botnn_promoi_nusva
                        coloca "" a todos los objetos texto
        fin (caso)
fin (procedimiento)
```

2.2.3.1.5.2.12 *smcc2a_elimina_lista_edita(w, tag, reason)*

PROCEDIMIENTO:	***smcc2a_elimina_lista_edita(w, tag, reason)***	
Descripcion:	Elimina un item (previamente elegido) de cualquiera de los dos objetos listas que se encuentran en la pantalla de captura.	
Variables locales	**Descripcion**	**Dominio**

XmStringTable tabla_string	Arraglo tipo XmString que contendrá todos los items de un objeto tipo lista	arreglo de cadenas tipo Xmstring
int num_items	Numero de elementos del arreglo tipo Xmstring	1<=num_items<=25
XmString new-item	Cadena tipo string que almacenara los nuevos valores de items despues de borrar uno	
Variables Globales	Apuntador_de_vaciado-de_vaciado	
	num_histogramas, num_ediciones, num_parametros, num_terminos	
	pos_item_edita, pos_item_histo, pos_item_termi, pos_item_prome, ch_inter	

```
Empieza smcc2a_elimina_lista_edita(w, tag, reason)
        elimina item de lista k_smcc2a_lista_edita de posicion pos_item_edita
        obten numero de items en k_smcc2a_lista_edita
        obten tabla_string de lista k_smcc2a_lista_edita
        actualiza indices de items restantes
fin (procedimientos)
```

2.2.3.1.5.3 CALIBRACIÓNDE CONDUCTIVIDAD (Tercera Etapa)

La tabla que se muestra a continuación, presenta todas y cada una de la variables globales que se utilizan en el programa para la calibración de la conductividad en su tercera etapa.

VARIABLES GLOBALES	DESCRIPCION
File *p_smcca	Apuntador a un archivo fisico para el manejo de ficheros de entrada/salida
int pos_item_inter	Valor que guarda la posición de un item perteneciente al objeto tipo lista de intercambios
char ch_inter[80]	Cadena que almacena el contenido de un item para el objeto lista de intercambios
char smcc_3a_formula[4]	Cadena que guarda el id de la formula de conversion a utilizar
char funcion_elegida[60]	Nombre de la función de conversión a utilizar
int num_cambios	Entero que almacena el numero de items del objeto tipo lista de intercambios

tabla 2.2.3.1.5.3. Variables globales para los procedimientos que realizan la calibración de conductividad en su tercera etapa.

2.2.3.1.5.3.5 *smcc3a_arma_comandos( )*

PROCEDIMIENTO:	***smcc1a_arma_comandos()***	
Descripcion:	procedimiento que genera el archivo de comandos que será ejecutado por el procedimiento encargado de realizar la primete etapa de calibración de oxigeno	
Variables Locales	**Descripcion**	**Dominio**
char cadena_entrante[50]	Cadena que sera armada para su inclusion como renglon de p_smcc3a	Cualquier cadena
Variables Globales	FILE *p_smcc1a	
	codcrucero,codproyecto, codbarco	
	extructura estructura_smcc3a_nodo.set_switche	
	estrcutura estructura_smcc3a_nodo.elige_funcion	
	estrcutura estructura_smcc1a_nodo.rec_smcc3a_inter	

```
Empieza smcc1a_arma_comandos ()
        Abre archivo "lg13:[cld.ctdsoft[cld_mpd_comandos.com" para escritura
        copia "5,,,,,,50/b\n" a cadena_entrante
        incluye cadena_entrante a p_smcc1a
        ...
        (formato "")
        ...
        copia "18/b\n"
        incluye cadena_entrante a p_smcc3a
        cierra p_smcc1a
fin (procedimiento)
```

2.2.3.1.5.3.7 *smcc3a_selecciona_lista_funcion(w, client_data,call_data)*

PROCEDIMIENTO:	*smcc3a_selecciona_lista_funcion(w, client_data,call_data)*	
Descripcion:	*Procedimiento que controla la elección de algún item del objeto lista de funciones*	
Variables locales	Descripcion	Dominio
int pos_item	Posicion del item elegido del onjeto lista de finciones	1<=pos_item<=4
Variables globales	*smcc3a_formula, funcion elegida*	

```
Empieza smcc3a_selecciona_lista_funcion(w, client_data,call_data)
        iguala pos_item a call_data->item_position
        caso (pos_item)
                caso 1:
                        iguala funcion elegida a "Cond->Sal (antigua formula salinidad)"
                ...
                caso 4:
                        iguala funcion elegida a "Sal->Cond(UNESCO-78 'SAL78')"
        fin caso
fin procedimiento
```

2.2.3.1.5.4 PROGRAMA PARA LA CREACIÓN DE ARCHIVOS DE INDICES

2.2.3.1.5.4.1 *crsu_ejecuta_crsumary( )*

PROCEDIMIENTO:	*crsu_ejecuta_crsumary( )*	
Descripcion:	Procedimiento que general el archivo de comandos necesario para ejecutar el programa CRSUMARY	
Variables Locales	Descripcion	Dominio
char instruccion[60]	cadena que guardara la instruccion que se ejecutara para el sistema	cualquier cadena de instruccion
char js_dir[30]	cadena que almacena la direcion del archivo de comandos a crear	cadena formato dir
FILE *p_crsum	Apunatador a un archivo fisico (del archivo de comandos	
char buffer[40]	cadena auxiliar para la creacion del archivo de comandos	

```
Empieza crsu_ejecuta_crsumary( )
        arma direccion de archivo de comandos en js_dir
        abre archivo cld_mpd_comandos.com en p_crsu
        copia "mdirfil.dat\n" en buffer
        graba linea buffer en p_crsum
        ...
        copia "$An" en buffer
        graba linea buffer en p_crsum
        cierra p_crsum
        ejecuta mediante el sistema al programa "[cld.cldsoft.mpd.smle]crsumary)"
fin procedimiento
```

2.2.3.1.5.4.2 *mpd_indices_arranque( )*

PROCEDIMIENTO:	*mpd_indices_arranque( )*	
Descripcion:	Procedimiento que crea el fichero de directorio de archivo *.cld"	
Variables Locales	Descripcion	Dominio
char instruccion[60]	cadena que guardara la instruccion que se ejecutara para el sistema	cualquier cadena de instruccion
char js_dir[30]	cadena que almacena la direcion del archivo de de ficheros .cld	cadena formato dir
FILE *p_firfil	Apunatador a un archivo fisico (de ficheros .cld)	
char buffer[40]	cadena auxiliar para la creacion del archivo de comandos	

```
Empieza mpd_indices_arranque( )
        arma direccion de archivo en js_dir
        manda a actuar desde el sistema al archivo m_x_difil.dat
        copia archivo m_x_difil.dat al directorio raiz (ctd ctdsoft)
fin (procedimiento)
```

2.2.3.1.5.5 PROGRAMA QUE MANEJA LOS OBJETOS DE LA PANTALLA PRINCIPAL

2.2.3.1.5.5.1 *ctd_mpd_llama(w, tag, reason)*

PROCEDIMIENTO:	*ctd_mpd_llama*	
Descripcion:	Procedimiento que se invoca cada vez que se llama a una ventana de MPD, dependiendo de esto, y del tipo de operacion que se quiera realizar se llamana a ciertas rutinas de inicio	
Variables Locales	Descripcion	Dominio
int widget_num	Guarda el identificador del objeto que invoco al procedimiento	307<=widget_num<=334

```
Empieza ctd_mpd_llama(w, tag, reason)
        caso (widget_num)
                caso k_ctd_alias_boton_catalogo
                        iguala cac_operacion a realizar a 1
                        llama VUIT_Manage("cac_main_win")
                        llama cac_inicia()
                fin caso
                ... (asi para bajas, cambios y consultas)
                caso k_ctd_alias_boton_transectos
                        iguala tra_operacion_a_realizar a 1
                        llama VUIT_Manage("tra_main_win")
                        llama tra_inicia()
                fin caso
                ... (asi para bajas, cambios y consultas)
                caso k_ctd_alias_boton_nitrato
                        iguala tro_operacion_a_realizar a 1
                        llama VUIT_Manage("nit_main_win")
                        llama nit_inicia()
                fin caso
                ... (asi para alias, cambios de silicatos, fosfatos y nitratos)
                caso k_ctd_calidad_boton_smce1a
                        llama VUIT_Manage ("smce1a_main_win")
                        llama smce1a_inicia()
                fin caso
                ... (asi para la segunda y tercera etapa de calibracion de conductividad)
                caso k_ctd_calidad_boton_smce1a
                        llama VUIT_Manage ("smce1a_main_win")
                        llama smce1e_inicia()
                fin caso
                ... (asi para la segunda y tercera etapa de calibracion de oxigeno)
                caso k_ctd_calidad_boton_presion
                        llama VUIT_Manage ("mpd_smp_main_win")
                        llame smp_inicia()
                fin caso
        fin caso
fin procedimiento
```

2.2.3.1.5.5.1 *ctd_mpd_llena_nombres_struct (w, tag, reason)*

PROCEDIMIENTO:	*ctd_mpd_llena_nombres_struct (w, tag, reason)*	
Descripcion:	Procedimiento que activa a la funcion tipo work_proc (ctd_work_proc)	
Variables Locales	Descripcion	Dominio
XtWorkProcId id	id es un tipo work_proc que controla el flujo hacia la cola de eventos	

```
Empieza ctd_rupd_llena_nombres_struct (w, tag, reason)
        llama funcion XtAppAddWorkProc(app_context,ctd_work_proc,Null)
fin (procedimiento)
```

2.2.3.1.5.5.2.1 *ctd_work_proc (client_data)*

PROCEDIMIENTO:	*ctd_mpd_llena_nombres_struct (w, tag, reason)*
Descripcion:	Procedimiento que ejecuta cada uno de los procesos que se involucran en la calibración, realizando un vaciado de la cola de eventos despues de cada proceso.

```
Empieza ctd_rupd_llena_nombres_struct (w, tag, reason)
        caso (eje)
                caso 1:
                        llama ctd_mpd_ejecuta_calmues()
                        incrementa eje en 1
                        regresa falso
                caso 2:
                        llama ctd_mpd_ejecuta_smco1a()
                        incrementa eje en 1
                        regresa falso
                caso 3:
                        llama ctd_mpd_ejecuta_smco2a()
                        incrementa eje en 1
                        regresa falso
                caso 4:
                        llama ctd_mpd_ejecuta_smco3a()
                        incrementa eje en 1
                        regresa falso
                caso 5:
                        llama ctd_mpd_ejecuta_crsumaryx()
                        incrementa eje en 1
                        regresa falso
                caso 6:
                        llama ctd_mpd_ejecuta_smp()
                        incrementa eje en 1
                        regresa falso
                caso 7:
                        llama ctd_mpd_ejecuta_smco1a()
                        incrementa eje en 1
                        regresa falso
                caso 8:
                        llama ctd_mpd_ejecuta_smco2a()
                        incrementa eje en 1
                        regresa falso
                caso 9:
                        llama ctd_mpd_ejecuta_smco3a()
                        incrementa eje en 1
                        regresa falsa
                caso 10:
                        llama ctd_mpd_ejecuta_crsumaryx()
                        regresa verdadero
        fin caso
fin (procedimiento)
```

2.2.3.1.5.6 CALIBRACIÓNDE OXÍGENO (Primera Etapa)

La tabla que se muestra a continuación, presenta todas y cada una de la variables globales que se utilizan en el programa para la calibración de oxígeno en su primera etapa.

VARIABLES GLOBALES	DESCRIPCION
File *p_smco1a	Apuntador a un archivo fisico para el manejo de ficheros de entrada/salida
int smco_array_estaciones[150]	Arreglo que conserva los numeros de estaciones que estan disponibles para un crucero
int indice_mayor	Contiene el indice mayor del arreglo de estaciones para el crucero
int smco1a_tipo	bandera que indica si se trabajara con todas las estaciones, o con solo algunas

char smco1a_inclusion[5]	cadena que conserva un item que sera incluido en un objeto lista
char smco1a_exclusion[5]	cadena que conserva un item que sera excluido de un objeto lista

tabla 2.2.3.1.5.6. Variables globales para los procedimientos que realizaa la calibración de oxigeno en su primera etapa.

2.2.3.1.5.6.1 *smco1a_crea_proc*

PROCEDIMIENTO:	*smco1a_croa_proc (w, tag, reason)*	
Descripcion:	Crea un identificador para cade widget de la interfaz en la aplicacion, guardado estos al arreglo smco1a_widget_num.	
Variables locales	Descripcion	Dominio
int widget_num	Almacena el identificador del Widget que invoco este procedimiento	Todos los identificedores de widgets (1-100)
Variables globales	smco1a_widget_num	

```
Empieza smco1a_crea_proc (w, tag, reason)
        iguala smco1a_widget_num[widget_num] a w
fin procedimiento
```

2.2.3.1.5.6.2.1 *smco1a_lee_mdirfil*

PROCEDIMIENTO:	*smco1a_leo_mdirfil (archivo_lectura)*	
Descripcion:	Abre el archivo mdirfil e inserta cada una de las estacienes al arreglo de estaciones sorteadas	
Parametros		
char archivo_lectura	cadena con el nombre del archivo m_dirfil que sera abierto	
Variables locales	Descripcion	Dominio
FILE p_mdirfil	Apuntador al archivo fisico mdirfil que sera abierto	apuntador tipo FILE
char buffer[50]	Cadena que sirve para lee una linea completa de archivo de p_mdirfil	Cualquier cadena
char ch_estacion[6]	Cedena que guarda el numero de estacion que se lee de buffer	cadena numerica
int indice	Indice del arreglo smce_array_estaciones	
Variables globales	smco_array_estacion	
	indice_mayor	

```
Empieza smco1a_lon_mdirfil (archivo_lectura)
        abre en  p_mdirfil el archivo archivo_lectura
        mientras no sea fin de archivo
                lee renglon de archivo
                si el renglon contiene una estacinn
                        incementa indice en 1
                        incluye ch_estacion a smco_array_estacion
                fin si
        fin_mientras
        iguala indice_mayor a indice
fin procedimiento
```

2.2.3.1.5.6.2.2.1 *smco1a_vacia_a_interfaz*

PROCEDIMIENTO:	*smco1a_vacia_a_interfaz(i_num_datos,a_datos)*	
Descripcion:	Procedimienio que recibe el arreglo de datos *datos[]* y los rafleja en los objetos de la interfaz	
Parámetros	Descripcion	Dominio
char a_datos[5][38]	Arreglo que recibido desde proc. smco1a_vacia_estructura_interfaz conteniendo los datos de las estrucuras	Arreglo tipo cadena con datos de estructura
int i_num_datos	Representa el numero de datos qua contiene el arreglo a_datos	1<=i_num_datos<=5
Variable globales	smco_array_estacion[]	
	indice_mayor	
	estructura estructura_smco1a_nodo.muestras_de_agua	
	estructura estructura_smco1a_nodo.rec_smco1a_estacion	

```
Empieza smco1a_vacia_a_interfaz(i_num_datos,a_datos)
        ciclo(a_datos[0])
                caso(2)
                        Apaga boton k_smco1u_toggle_algunas
                        Anade como item a k_smco1a_lista_elegida el datos a_datos[3]
                caso (6)
                        Si smco_array_estacion[indice_mayor-1] es igual e a_datos[2]
```

```
                                        oncionda bolon k_smco1a_toggle_todas
                                fin (si)
                                Repite de i=i, mientras smco_array_estacion[i] sea diferente de a_datos[2],    incrementa i
                                        introduce como item a k_smco1a_lista_elegidas el valor de smco_array_estacion[i]
                                fin (repite)
        fin caso
```

2.2.3.1.5.6.10 *smco1a_acepta_pantalla_rango(w, tag, reason)*

PROCEDIMIENTO:	*smco1a_acepta_pantalla_rango(w, tag, reason)*	
Descripcion:	Procedimiento que acepta todoa los datos de la ventana "smco1a_caja_rango_max_min" y la desaparace	
Variables Locales	**Descripcion**	**Dominio**
XmString arma_string	Cadena tipo xmstring que es usada para insertar un item a un objeto lista	Cualquier cadena tipo xmstring
int scala_valor_min	Entero que recibe el valor minimo del objeto scale k_smco1a_scale_rango_minimo	1<=scala_valor_min<= indice mayor
int scala_valor_max	Entero que recibe el valor maximo del objeto scale k_smco1a_scale_rango_maximo	1<=scala_valor_min<= indice mayor
Variables Globales	indice_mayor	
	smco1a_array_estaciones	

```
Empieza smco1a_acepta_pantalla_rango(w, tag, reason )
        Obten scale_valor_min del objeto k_smco1a_scale_rango_minimo
        Obten scale_valor_max del objeto k_smco1a_scale_rango_maximo
        Repite de i=0, mientras i sea menor a indice-mayor, incrementa i
                Si smco_array_estaciones es igual a scala_valor_min
                        existe_min = 1
                fin (si)
        fin (repite)
        Si existe_min es igual a 1
                Si scala_valor_min es menor a scala_valor_max
                        Llama vuil_unmanage ("smco1a_caja_rango_max_min")
                        Elimina items de objeto k_smco1a_lista_elegidas
                        Repite i = 0, mientras smco_array_estaciones[i] sea diferentn de scala_valor_max
                                inserta arma_string en listn k_smco1a_lista_elegidas
                        fin (repite)
                si (no)
                        vuil_manage("smco1a_error_ranga")
                fin (si)
        si (no)
                vuil_manage("smco1a_estacion_inexiste")
        fin (si)
fin (procedimiento)
```

2.2.3.1.5.7 CALIBRACIÓN DE OXÍGENO (Segunda Etapa)

La tabla que se muestra a continuación, presenta todas y cada una de la variables globales que se utilizan en el programa para la calibración de oxígeno en su segundaetapa.

VARIABLES GLOBALES	DESCRIPCION
File *p_smcoa	Apuntador a un archivo fisico para el manejo de ficheros de entrada/salida
int tipo_de_captura_flag	bandera que indoca el tipo de captura que se realizara sobre los ajustes
int num_max_nodos	numero maximo de nodos para la lista legada
int pos_actual_de_la_lista	Posicion del apuntador en la lista ligada

tabla 2.2.3.1.5.17. Variables globales para los procedimientos que realizan la calibración de oxigeno en su segunda etapa.

2.2.3.1.5.7.1 *smco2a_crea_proc*

PROCEDIMIENTO:	*smco2a_crea_proc (w, tag, reason)*	
Descripcion:	Crea un identificador para cada widget de la interfaz en la aplicacion, guardado estos al arreglo stnco3a_widget_num.	
Variables locales	**Descripcion**	**Dominio**
int widget_num	Almacena el identificador del Widget que invoco este procedimiento	Todos los identificadores de widgets (1-100)

2.2.3.1.5.7.2 *smco2a_up_down_proc (w, tag, reason)*

PROCEDIMIENTO:	***smco2a_up_down_proc (w, tag, reason)***	
Descripcion:	Crea un identificador para cada widget de la interfaz en la aplicacion, guardado estos al arreglo smco2a_widget_num.	
Variables locales	**Descripcion**	**Dominio**
int widget_num	Almacena el identificador del Widget que invoco este precedimiento	Todos los identificadores de widgets (1-100)
char max_nodos	Cadena que Almacena en forma alfanumerica el id del ultimo nodo de la lista	Char numerica formato id
Variables globales	estructura estructura_smco2a_nodo.rec_smco2a_struct	
	pos_actual_de_la_lista, lista lista_smco2a, apuntador_de_vaciado_de_datos	

```
EMPIEZA smco2a_up_down_proc (w, tag, reason)
        casn (widget_num)
        caso k_smco2a_boton_up_ajuste
                incrementa  pos_actual_de_la_lista en 1
                Obten estructura estructura_smco2a_nodo.rec_smco2a_struct de la lista segun llave pos_actual_de_la_lista
                Llama smco2a_vacia_datos_caratula (pos_actual_de_la_lista)
        caso k_smco2a_boton_down_ajuste
                decrementa pos_actual_de_la_lista en 1
                Obten estructura estructuu_smco2a_nodo.rec_smco2a_struct de la lista segun llave pos_actual_de_la_lista
                Llama smco2a_vacia_datos_caratula (pos_actual_de_la_lista)
        fin(caso)
```

2.2.3.1.5.7.2.1 *smco2a_vacia_datos_caratula (nodo)*

PROCEDIMIENTO:	***smco2a_vacia_datos_caratula (nodo)***	
Descripcion:	Vacia un nodo de la lista lista_smco2a a sus correspondientes widgets de la caratula, es decir, de la pantalla de inicio, el parametro que se pasa es el numero de nodo a pasar	
Parametro	**Descripcion**	**Dominio**
int nodo	El es identificador del nodo que sera vaciado a la pantalla inicial de smco_2a	cualquier id que exista en lista_smco2
Varisbles_globales	estructura estructura_smco2a_nodo.rec_smco2a_struct lista lista_smco2a	

```
EMPIEZA smco2a_vacia_datos_caratula ( nodo)
        Obten nodo de lista_smco2a en estructura_smco2a_nodo.rec_smco2a_nodo segun llave nodo
        Coloca estructura_smco2a_nodo.rec_smco2a_nodo->numero_parametros en objeto k_smco2a_label_on_parametros
        ...
        Coloca estructura_smco2a_nodo.rec_smco2a_nodo->guarda_archivo en k_smco2a_label_registro_archivo
```

2.2.3.1.5.7.8 *smco2a_maneja_toggles(w, tag, reason)*

PROCEDIMIENTO:	***smco2a_maneja_toggles (w, tag, reason)***	
Descripcion:	Procedimiento que realiza el manejo del toggle_botton "datos de defaul" encendiendo o apagando el sensitive de los objetos "iteraciones" y de "digitos significativos"	
Variables globales	arreglo smco2a_widget_array[50]	

```
Empieza smco2s_maneja_toggles (w, tag, reason)
        Si toggle k_smco2a_toggle_valor_default esta encendido
                apaga sensitivo de k_smco2a_texto_nn_iteracion
        si (nn)
                prende sensitivo de k_smco2a_texto_nn_iteracion
        fin (si)
fin (procedure)
```

2.2.3.1.5.8 CALIBRACIÓNDE OXÍGENO (Tercera Etapa)

La tabla que se muestra a continuación, presenta todas y cada una de la variables globales que se utilizan en el programa para la calibración de oxígeno en su tercera etapa.

VARIABLES GLOBALES	DESCRIPCION
File *p_smco1a	Apuntador a un archivo fisico para el manejo de ficheros de entrada/salida
int smco_array_estaciones[150]	Arreglo que conserva los numeros de estaciones que estan disponibles para un crucero
int smco_array_estaciones_rango[150]	Arreglo de estaciones que conserva solamente un rango de ellas
intsmco_array_estaciones_algunas[150]	Arreglo de estaciones que contiene solamente algunas de todas las disponobles
int indice_msyor	Contiene el indica mayor del arreglo de estaciones para el crucero
int indice_mayor_rango	Contiene al indice mayor del arreglo de estaciones de rango para el crucero
int indice_mayor_algunas	Contiene el indice mayor del arreglo de estaciones de algunas para el crucero
char smco3a_inclusion[5]	cadena que conserva un item que sera incluido en un objeto lista
char smco3a_exclusion[5]	cadena que conserva un item que sera excluido de un objeto lista

tabla 2.2.3.1.5.8. Variables globales para los procedimientos que realizan la calibración de oxígeno en su tercera etapa.

2.2.3.1.5.8.1 *smco3a_crea_proc*

PROCEDIMIENTO:	*smco3a_crea_proc (w, tag, reason)*	
Descripcion:	Crea un identificador para cada widget de la interfaz en la aplicacion, guardado estos al arreglo smco3a_widget_num.	
Variables locales	Descripcion	Dominio
int widget_num	Almacena al identificador del Widget que invoco este procedimiento	Todos los identificadores de widgets (1-100)
Variables globales	smco3a_widget_num	

```
Empieza smco3a_crea_proc (w, tag, reason)
        iguala smco3a_widget_num[widget_num] a w
fin procedimiento
```

2.2.3.1.5.8.2 *smco3a_inicia*

PROCEDIMIENTO:	*smco3a_inicia ()*	
Descripcion:	Establece las condiciones iniciales de la ventana smco3a, que se necesitan cada vez que se llama a la a esta interfaz	
Variables locales	Descripcion	Dominio
char dir_jsmco[70]	Cadena que guarda la direccion en disco de los archivos que se utilizaran paca esta ventana	cadena con formato de direccion
char_smco3a_archivo_mdirfil [100]	Cadena que contendra la direccion y nombra del archivo mdirfil a abrir	cadena con formato de direccion de archivo
Variables globales	codbarco, codproyecto, codcrucero	

```
Empieza smco3a_inicia ()
        Concatena datos para formar dir_jsmco
        concatena datos para formar smco3a_archivo_msubindmx
        concatena datos para formar smco3a_archivo_msamples
        llama smco3a_lee_adirfil(smco3a_archivo_mdirfil)
        llame smco3a_vacia_estructura_interfaz
fin procedimiento
```

2.2.3.1.5.8.2.2.1 *smco3a_lee_archivo_fit()*

PROCEDIMIENTO:	*smco3a_lee_archivo_fit()*	
Descripcion:	Procedimiento que recibe los datos de las estructuras, los reparte a los correspondientes objetos en la interfaz	
Variables locales	Descripcion	Dominio
char datos[5][38]	Arreglo que sirve pare leer linea de parametros de archivos fit.	Arreglo de cadenas

char buffer[100]	Cadena que sirve para leer una linea de archivo fit	cadena de archivo
Variable globales	lista_smco3a	
	estructura estructura_smco3a_nodo parametro_ajuste	

```
Empieza smco3a_lee_archivo_fit()
        Abre archivo de parametros de ajuste .fit
                Haz mientras no sea fin de archivo
                        Si la cadena leida en buffer existe la subcadena "PARMS="
                                lee parametros de arreglo datos[]
                                reparte datos[] en estructura_smco3a_nodo parametro_ajuste
                        fin(si)
                        lee siguiente linea de archivo
                fin (haz)
fin (procedimiento)
```

2.2.3.1.5.8.3 *smco3a_incluye_listas (w, client_data, call_data)*

PROCEDIMIENTO:	*smco3a_incluye_lista(w, client_data, call_data)*	
Descripcion:	Procedimiento que realiza la inclusion de un item en la lista de estaciones incluidas,elegido mediante un XmNdefaultActionCallback	
Variables Locales	Descripcion	Dominio

```
Empieza smco3a_incluye_listas ( w, client_data, call_data)
        Si arma_string ya existe como item en k_smco3a_lista_incluida
        si (no)
                añade item arma_string a objeto k_smco3a_lista_inserta
        fin (si)
fin (procedimiento)
```

2.2.3.1.5.8.4 *smco3a_excluye_listas (w, client_data, call_data)*

PROCEDIMIENTO:	*smco3a_excluye_lista(w, client_data, call_data)*	
Descripcion:	Procedimiento que realiza la exclusion de un item en la lista de estaciones incluidas,elegido mediante un XmNdefaultActionCallback	
Variables Locales	Descripcion	Dominio
int num_items	Guarda el numero de items de un objeto tipo lista	1<=num_items<=25
Variable globales	apuntador_de_vaciado_de_datos	

```
Empieza smco3a_excluye_listas ( w, client_data, call_data)
        Obtiene numero de items de k_smco3a_lista_incluidas
        Si num_items es diferente de cero
                Posiciona en item 0 del objeto lista k_smco3a_lista_incluidas
        si (no)
                Apaga sensitive de k_smco3a_boton_excluye
        fin (si)
fin (procedimiento)
```

2.2.3.1.5.8.11 *smco3a_edita_estaciones(w, tag, reason)*

PROCEDIMIENTO:	*smco3a_edita_estaciones(w, tag, reason)*	
Descripcion:	Procedimiento que dependiendo de los toggle buttons toda_algunas_rango, aparece la caja de dialogo adecuada	
Variables Locales	Descripcion	Dominio
int widget_num	Identificadir del objeto tipo widget que convoca al procedimiento	1<=widget_num<=100
Variables Globales	apuntador_de_vaciado_de_datos	
	smco_array_estaciones_rango	

```
Empieza smco3a_edita_estaciones(w, tag, reason)
        case (widget_num)
                caso k_smco3a_toggle_todas
                        Si objeto toggle k_smco3a_toggle_todas esta encendido
                                Elimina todos los items de objeto lista k_smco3a_lista_elegidas
                                Llama smco3a_vacia-lista_disponibles(k_smco3a_lista_elegidas)
                        fin (si)
                caso k_smco3a_toggle_algunas
                        Si objeto toggle k_smco3a_toggle_algunas esta encendido
                                Elimina todos los items de objeto lista k_smco3a_lista_elegidas
                                Obtiene tabla_strings de k_smco3a_lista_incluida
                                Obtiene num_items de k_smco3a_lista_incluida
```

```
                                        Repita de i = 0; mientras i sea menor a num_items
                                                anado items tabla_string[i] n k_smco3a_lista_elegidas
                                        fin (repita)
                                fin (si)
                        caso k_smco3a_toggle_rango
                                si el objeto k_smco1_scale-rango_minimo es diferente de NULL
                                        Llama smco3a_acepta_pantalla_rango();
                                fin (si)
                        caso k_smco3a_boten_edito
                                Si toggle button  k_smco3a_toggle_algunas esta encendido
                                        Llama ventana ("smco3a_caja_incluye_boten")
                                        Llama smco3a_vacia_lista_disponibles (k_smco3a_liste_disponible)
                                Obten table_string de k_smco3a_lista_elegidas
                                        Obten num_items de k_smco3a_lista_elegidas
                                Si (no)
                                        Si toggle button  k_smco3a_toggle_rango esta encendido
                                                Llama ventana ("smco3a_ceja_rango_mux_min")
                                                Coloca valores max y min a objetos tipo scala
                                        Si(no)
                                fin (si)
                fin (caso)
        fin (procedimiento)
```

2.2.3.1.5.9 SERIES PROMEDIADAS DE PRESIÓN

2.2.3.1.5.9.1 *mpd_smp_crea_proc (w, tag, reason)*

PROCEDIMIENTO:	*mpd_smp_crea_proc (w, tag, reason)*	
Descripcion:	Crea un identificador para cada widget de la interfaz en la aplicacion, guardado estos al arreglo smp_widget_num.	
Variables locales	**Descripcion**	**Dominio**
int widget_num	Almacena el identificador del Widget que invoco este procedimiento	Todos los identificadores de widgets (1-100)
Variables globales	smp_widget_num	

```
Empieza mpd_smp_crea_proc (w, tag, reason)
        Iguala smp_widget_nom[widget_num] a w
fin procedimiento
```

2.2.3.1.5.9.2 *mpd_smp_acepta_todo (w, tag, reason)*

PROCEDIMIENTO:	*mpd_smp_acepte_todo (w, tag, reason)*
Descripcion:	Procedimiento que acepta todos los cambios de la ventana, y los coloca en las estructuras correspondien

```
        empieza mpd_smp_acepta_todo (w, tag, reason)
                llama smp_incluye_archivo_control()
                llama VUIT_Unmanage ("mpd_smp_main_win")
        fin procedimiento
```

2.2.3.1.5.9.2.1 *mpd_smp_incluye_archivo_control (control)*

PROCEDIMIENTO:	mpd_smp_incluiye_archivo_control ()	
Descripcion:	Procedimiento que incluye al archivo de control generalla parte correspondiente al sorteo de presion	
Parametros		
*FILE control	Apuntador al archivo de control general	
Variables Locales	**Descripcion**	**Dominio**
char buffer[100]	Cadena que servira para guardar en *control una linea de datos para el archivo de control genetal	Cualquier cadena
Variables Globales	extructura estructura_smp_nodo_datos_presion	

```
        Empieza  smp_incluye_archivo_control()
                copia "|CALIBRACION_PRESION_PROMEDIA|" a buffer
                incluye buffer a control
                ...
                (formato "")
                ...
                copia "    |SALIDA_RAMA_12|  (2) \n"
```

```
                incluye buffer a control
                cierra control
        fin (procedimiento)
```

2.2.3.1.5.9.7.1 *smp_vacia_estructura_interfaz ()*

PROCEDIMIENTO:	*smcofa_vacia_estructura_interfaz()*	
Descripcion:	Procedimiento que recibe los datos de las estructuras, los reparte a los correspondientes objetos en la interfaz	
Variables locales	Descripcion	Dominio
char texto_show_state[40]	cadena que guarda la leyenda que sera incluida al objeto k_mpd_smp_texto_state_intervalo	

```
        Empieza smp_vacia_estructura_interfaz ()
                Si estructura_smp_nodo_datos_presion.entrar es igual a "YES"
                        copia a texto_show_state la leyenda "Extraer datos de subida del rid 2 0db
                si (no)
                        arma leyenda con val_actual para k_mpd_smp_texto_state_intervalo en texto_show_state
                fin (si)
                incluye cadena texto_show_state a objeto  k_mpd_smp_texto_state_intervalo
                llama smp_lee_archivo_parametro()
                vacia a los objetos texto cada una de los elementos de la estructura estructura_smp_nodo.datos-presion
```

2.2.3.1.5.9.8 *mpd_smp_arma_comandos( )*

PROCEDIMIENTO:	*mpd_smp_arma_comandos( )*	
Descripcion:	procedimiento que genera el archivo de comandos que será ejecutada por el procedimiento encargado de realizar el sorteo de presion	
Variables Locales	Descripcion	Dominio
char cadena entrante[50]	Cadena que sera armada para su inclusion como renglon de p_smcofa	Cualquier cadena
Variables Globales	FILE *p_smcofa	
	codcrucero,codproyecto, codbarco	
	estructura estreuctura_smp_nodo.datos_presion	

```
        Empieza smp_arma_comandos ()
                Abre archivo "/g13/cid.cids01/cid_mpd_comandas.com" para escritura
                copia "13,1/'bin'" a cadena_entrante
                incluye cadena_entrante a p_smp
                ...
                (formato "")
                ...
                copia "12,1/'bin'"
                incluye cadena_entrante a p_smp
                cierra p_smp
        fin (procedimiento)
```

Capítulo 3

Fase de Desarrollo

Los objetivos perseguidos durante la fase de desarrollo del sistema MPD son los siguientes:

1. Codificar cada uno de los procedimientos que conforman a cada programa de MPD.[1]
2. Verificar el correcto funcionamiento de cada procedimiento
3. Integrar los procedimientos para formar cada uno de los programas.[2]
4. Integrar los programas para conformar lo que finalmente será el sistema MPD.
5. Elaborar el diseño e integración de la interfaz con la aplicación.

Para llevar a cabo esta etapa de nuestro sistema se subdividió la fase de desarrollo en tres etapas, las cuales son[3]:

a) Codificación:

i) De la Aplicación.

Los estándares o normas de codificación de programación son un conjunto de reglas de estilo que permiten que los programas sean fáciles de leer, revisar, modificar y por lo consiguiente mantener.

Este conjunto de normas generales de programación también tienen como objetivo unificar el estilo de programación, sin limitar la creatividad del programador. Se entiende por estilo, la selección adecuada de hábitos de programación que favorecen la claridad y eficiencia del programa.

El empleo unificado de estos hábitos coadyuva a la producción de buenos programas, cuya principal característica desde luego es que deben trabajar. La velocidad, la utilización del tiempo de máquina, el número de líneas de código y otras variables cuantificables no tienen significado si el programa trabaja. Con relación a esas características puede afirmarse que:

1. Debe tener bajos costos de desarrollo
2. Debe tener bajos costos de prueba
3. Debe tener costos mínimos de mantenimiento.
4. Debe ser fácilmente modificable
5. Debe tener un diseño poco complicado

ii) De la Interfaz.

Como se menciono en el requerimiento *1.3.4.11.e* se codificará la interfaz empleando funciones de alto nivel (XUI)[4]. Se ejemplificará con el código de una ventana en particular y se explicarán cada una de las secciones de este y lo que significa.

---

[1] Todos los procedimientos se describen en 2.2.2
[2] Ver 2.2.3
[3] Según [6]

b) Integración

El objetivo principal de esta etapa es la integración funcional de cada uno de los procedimientos codificados para formar a cada uno de los programas que componen a MPD.

En esta etapa de integración se tomó en cuenta también otro aspecto: el diseño de las pruebas que permitieron identificar errores de codificación, para lo cual se empleó una estrategia de integración incremental[5], la cual consiste en validar los nuevos módulos o procedimientos (no aprobados) agregándolos a módulos o procedimientos ya probados e integrados.

c) Desarrollo de la Interfaz.

En esta etapa nos abocaremos al desarrollo e implementación de todas y cada una de las cajas de diálogo que compondrán la interfaz de MPD. Se detallarán las secciones principales de las ventanas, así como se mencionarán los identificadores de cada unos de los objetos (botones, textos, cajas de diálogo, etc.) que las componen. También se describirán los callbacks o procedimientos asociados a cada objeto tipo botón.

3.1 CODIFICACIÓN

A continuación se listan cada una de las normas que se tomaron de base para la codificación de cada uno de los procedimientos que conforman a MPD. Se ilustrará con un procedimiento real la implementación de cada uno de estos puntos.

3.1.1 UNA SOLA FUNCIÓN.-

Todo procedimiento que componen a MPD posee una y solo una función específica. Esto gracias a las características del lenguaje de programación utilizado que permite modularidad. La función de cada procedimiento esta perfectamente especificada en 2.2.3.

3.1.2 CONVENCIÓN DE NOMBRAMIENTO.-

Los nombres asignados a cada uno de los procedimientos que conforman a MPD son representativos del contenido del mismo.

3.1.3 LONGITUD DE LOS PROCEDIMIENTOS.-

El cuerpo de cada uno de los procedimientos no excede de mas de 100 instrucciones ejecutables.

3.1.4 ASIGNACIÓN DE LÍNEAS DE RASTREO Y DE DEPURACIÓN

Cada uno de los procedimientos que realiza alguna labor en especial lleva incluidas dentro de su código, instrucciones para mandar mensajes e información a pantalla. Lo anterior con la finalidad de saber exactamente el estado que los datos del procedimiento presenta, para detectar

[4] Ver OSF/MOTIF PROGRAMMER'S REFERENCE
[5] Vea Gerez,Victor; Mier, Mauricio, "Desarrollo y administración de Programas de computadora"

problemas al momento de ejecutarse. La instrucción en lenguaje "C" que se utiliza para imprimir esta información es PRINTF[6].

A continuación presentamos el código en lenguaje "C" de un procedimiento perteneciente al programa de captura del catálogo del crucero.

```
/* PROCEDIMIENTO QUE BUSCA EL NUMERO DE ESTACION DESEADO DENTRO DEL
   ARCHIVO DEL CATALOGO DEL crucero*/

void cac_busca_estacion (w,tag,reason)
Widget        w;
int           *tag;
unsigned long  *reason;
{
        /*Variables Tipo Char De Los Datos De La Cebecera De La Estacion*/
 char i_cac_num_esta[5]='\0';
 char i_cac_prof_max[8]='\0';

        /* Modifica la apariencia del cursor */
cursor= DXmCreateCursor(toplevel_widget, decw$c_wait_cursor);
XIDefineCursor(XtDisplay(toplevel_widget),  XtWindow(widget_array[k_cac_main_win]), cursor);

/****  obtiene la estacion capturada del widget texto_estacion */
 a_estacion = XmTextGetString(widget_array[k_cac_texto_estacion]);

/******inicia busqueda secuencial de la estacion****/
 rec_prin=FirstRec(lista_prin);
  while(atoi(i_cac_num_esta)!=atoi(a_estacion))
        {
                /* Bucle para leer las cabeceras de estaciones
                   es decir las cadenas que empiezan con '*'
                   si no se localiza ninguna, la bandera de fin de
                   archivo se pone a cero */
                while (rec_prin->buffer[0]!='*')
                        {
                                rec_prin=NextRec(lista_prin,rec_prin);
                                if (rec_prin->id_prin == cac_ultimo_rec)
                                        { /* pone la bandera de fin de archivo a 1*/
                                        fin_arch=1;
                                                break;
                                        }
                        }

                /*  Checar si el numero de estacion es el deseado*/
                                for (i=1;i<=3;i++)
                                        {
                                                if ((rec_prin->buffer[i]=='0')|| (rec_prin->buffer[i]==' '))
                                                {
                                if (f2=1) {i_cac_num_esta[j]=rec_prin->buffer[i];j++; }
                                                }
                                  else
                                                {
                                                        i_cac_num_esta[j]=rec_prin->buffer[i];j++; f2= 1;
                                                }
                                                }
                                i_cac_num_esta[j]='\0';
        /* Lee el siguiente registro de  la lista principal*/
                        rec_prin=NextRec(lista_prin,rec_prin);
```

[6] Vea [10]

```
                        }

                num_linea=rec_prin->id_prin;
                cac_flag=2;
                /* La bandera indica que se tranajara con una estacion que ya estaba
                        en el archivo, es decir no se anadio estacion */
                /*  captar del buffer el dato de profundidad maxima*/
                 for(i=5;i<=10;i++) {
                        i_cac_prof_max[j]=rec_prin->buffer[i]; j++;
                } i_cac_prof_max[j]='\0';j=0;

        /*  captar del buffer el dato de la hora local */
                for (i=12;i<=20;i++) {
                        i_cac_fecha_esta[j]=rec_prin->buffer[i]; j++;
                } i_cac_fecha_esta[j]='\0';j=0;
        /* Vaciar datos en los objetos texto*/
apuntador_de_vaciado_de_datos=i_cac_num_esta;
XmTextSetString(widget_array[k_cac_row1_col1],apuntador_de_vaciado_de_datos);
XtFree(apuntador_de_vaciado_de_datos);
apuntador_de_vaciado_de_datos=NULL;
apuntador_de_vaciado_de_datos=i_cac_prof_max;
XmTextSetString(widget_array[k_cac_row1_col2],apuntador_de_vaciado_de_datos);
XtFree(apuntador_de_vaciado_de_datos);
apuntador_de_vaciado_de_datos=NULL;
apuntador_de_vaciado_de_datos=i_cac_fecha_esta;

/* lectura de la primera linea de datos del catalogo del crucero*/
rec_prin=NextRec(lista_prin,rec_prin);

 /* creacion de la lista de datos de la estacion*/
if ((lista_esta = CreateList()) == NULL) {
printf("Error not enough memory ");
exit(-1);
        }
printf("acabo de crear la lista de estacion\n");

while((rec_prin->buffer[0]!='*')&&(rec_prin->buffer[0]!='(')&&(bot!=1))
        {
                rec_esta = (REC_ESTA *) calloc(1, sizeof(REC_ESTA));
                rec_esta->id_esta=atoi(i_cac_num_esta);
                j=0;        /*** lectura del buffer del numero de botella*/
                for (i=0;i<=2;i++) {
                        i_cac_num_bot[j]=rec_prin->buffer[i];j++;
                }
                i_cac_num_bot[j]='\0';
                rec_esta->botella=atoi(i_cac_num_bot);
                /* vaceado de los datos a sus correspondientes widget de la cabnecera de
                estacion */
                bot=rec_esta->botella;

                    cac_vaciado_de_objetos_de_datos();
                /*restauramos el cursor ya que la accion ha finilizado
                        */
                 XUndefineCursor(XtDisplay(toplevel_widget),
                XtWindow(widget_array[k_cac_main_win]));
}
```

Figura 3.1.a. Ejemplo ilustrativo de los puntos 3.1.1, 3.1.2, 3.1.3 y 3.1.4 utilizando un procedimiento del programas de captura del archivo de Catálogo del Crucero.

Segmento de código de interfaz. En la siguiente sección mostraremos una porción de código UIL[7] de la interfaz.

[7] Ver apéndice B sobre programación en UIL.

```
!** FONDO PRINCIPAL DE LA PANTALLA DE CAPTURA DEL CATALOGO DEL CRUCERO
  cac_main_win: XmFormDialog  !** Caja tipo formdialog con nombre cac_main_win
    {
    arguments
      {
      XmNx = 489;
      XmNy = 27;
      XmNwidth = 681;
      XmNheight = 588;
      XmNborderWidth = 1;
      XmNnoResize = true;
      XmNresizable = false;
      XmNresizePolicy = XmRESIZE_NONE;
      XmNdialogTitle = compound_string("Lectura de Catalogo del Crucero");
      };
    controls
      {
      XmScrolledWindow cac_scrollwin_hoador;
      XmForm cac_row_labels;
      XmForm cac_form_base_datos;
      XmFrame cac_frame_estacion;
      XmFrame cac_frame_archivos;
      };
    callbacks
      {
      MrmNcreateCallback = procedures
        {
        cac_proc_crea(k_cac_main_win);
        };
      };
    };

cac_coment_borrar: XmPushButton   /** Widget tipo Botón para realizar oporación do "Borrar" un valor
    {
    arguments
      {
      XmNlabelString = compound_string("BORRAR");
      XmNx = 764;
      XmNy = 416;
      XmNleftOffset = 218;
      XmNtopOffset = 8;
      XmNleftAttachment = XmATTACH_FORM;
      XmNtopAttachment = XmATTACH_FORM;
      };
    callbacks
      {
      MrmNcreateCallback = procedures
        {
        cac_proc_crea(k_cac_coment_borrar);
        };
      XmNactivateCallback = procedures
        {
        cac_maneja_coment(k_cac_coment_borrar);
        };
      };
    };
```

Sección de argumentos del objeto o widget cac_main_win Se establecen caracteristicas como: tamaños, posiciones, titulos, anchos de borde, etc.

Sección de controles. Objetos hijos directos de cac_main_win.

Area de callbacks o funciones asaciadas a cac_main_win

Argumentos a caracteristicas de cac_coment_borrar. Se asignan etiquetas, tamaño, posición dentro del objeto padre.

Sección de callbacks o procedimientos asaciados a cac_coment_borrar. Se determinan dos tipos: createcallback y activatecallback.

Figura 3.1.b Sección de código UIL. (de Interfaz). Para la ventana del programa de capturar del Catálogo del crucero.

3.1.5 Autodocumentación

Todo Programa, contiene una descripción breve de su función, uso, variables que maneja, limitaciones, etc. Estos comentarios son:

1. Programa fuente. Referencia del área en disco que contiene al programa en cuestión
2. Lanzador. Referencia del área en disco que contiene el conjunto de instrucciones o programa que ejecuta dicho módulo
3. Propósito. Descripción en prosa de la función que realiza el procedimiento.
4. Referencias. Menciona las referencias en donde el requerimiento que se esta resolviendo por medio de este programa queda definido
5. Bibliografía. Menciona las fuentes bibliográficas utilizadas como base en el desarrollo de este programa
6. Limitaciones. Sirve para indicar en forma breve las restricciones en el volumen de información.
7. Aclaraciones. Sirve para indicar las condiciones de terminación de la ejecución.
8. Nombre y fecha de la implementación.
9. Nombre y fecha de revisiones

```
/*****************************************************************/
/*                  INSTITUTO DE INVESTIGACIONES ELECTRICAS      */
/*                  GRUPO DE ESTUDIOS OCEANOGRAFICOS             */
/*                  MODULO DE PROCESAMIENTO DE DATOS             */
/*****************************************************************/
/*                                                               */
/* PROGRAMA:                                                     */
/*                  Captura de Datos de Transectos               */
/*          lg13 |ctd.ctdsoft mpd.smc|transactos.c               */
/* LANZADOR:                                                     */
/*                  lg13 |ctd.ctdsoft|ctdsis.c                   */
/* PROPOSITO:                                                    */
/*          Conjunto de procedimientos que permiten la captura de datos de transectos */
/*          a travez de una ventana de interfaz                  */
/* REFERENCIAS:                                                  */
/*                  Seccion ****                                 */
/* BIBLIOGRAFIA:                                                 */
/*                  No existe                                    */
/* LIMITACIONES:                                                 */
/*                  Lenguaje "C", librerias de MOTIF             */
/* ACLARACIONES:                                                 */
/*                  No se tiene un limite definido sobre el numero maximo de transectos */
/*                  que se puedan soportar                       */
/* NOMBRE Y FECHA DE IMPLEMENTACION:                             */
/*                  P.I.C Carlos Adrián Delgado Diaz  15/agosto/1995 */
/* NOMBRE Y FECHA DE REVICION                                    */
/*                  M.C. Eustorgio Meza Conde  15/agosto/1995    */
/*****************************************************************/
```

figura 3.1.5 Ejemplo de autodocumentacion en un procedimiento de MPD

3.2 INTEGRACIÓN

En esta sección se presenta el tipo de prueba que se utilizó para identificar los errores de cómputo que se originaron en la etapa de integración. Como ya se mencionó, se utilizará un tipo de integración incremental [6] con pruebas de abajo hacia arriba.

La estrategia de pruebas de abajo hacia arriba comienza con probar el primer modulo del nivel jerárquico mas alto dentro de los diagramas de estructura mostrados en *2.2.3* para después ir descendiendo de nivel.

Por ejemplo; en la figura 3.2 se muestra el diagrama de estructura del programa de captura de datos de transecto (la tabla de abreviaciones se muestra en 2.2.3). Se arranca con las pruebas de los módulos de los niveles de mayor jerarquía, como lo son *1.3.1, 1.3.2, 1.3.3, 1.3.4, 1.3.5, 1.3.6, 1.3.7, 1.3.8, 1.3.9, 1.3.10, 1.3.11*; posteriormente el análisis sigue a los procedimientos del nivel jerárquico siguiente hacia abajo.

Así, de esta manera se sigue la implemetación de los módulos restantes en forma descendente y de uno en uno.

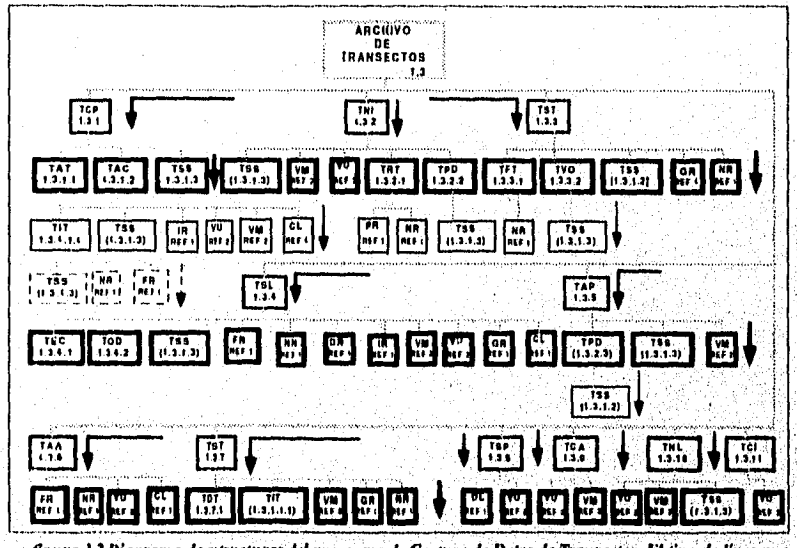


figura 3.2 Diagrama de estructuras del programa de Captura de Datos de Transectos. El tipo de línea que envuelve al módulo indica el nivel jerárquico al que pertenece. ——— *Primer nivel jerárquico.* ———— *Segundo nivel.* ———— *Tercer nivel.* ..... *Cuarto nivel jerárquico.*

De esta forma se integraron todos y cada uno de los procedimientos y módulos codificados de todos los programas que conforman a MPD.

3.3 DESARROLLO DE LA INTERFAZ

3.3.1 VENTANA PRINCIPAL DE CTDSIS

Como se mencionó en la introducción el sistema CTDSIS esta formado por tres módulos construidos de manera independiente : MPD (producto de este trabajo de tesis), MGD[4] y MRG[5]. Existe una ventana principal común a estos tres módulos. Esta ventana fue diseñada y construida para servir de frente a CTDSIS y de arranque a las tres aplicaciones ya mencionadas.

Esta ventana posee un menú principal que contiene todas las posibilidades que entre MPD, MGD y MRG pueden ofrecer.

Para nuestro caso todas la opciones de MPD están repartidas dentro de este menú y solamente a través de él se podrán accesar a todas sus opciones. La estructura del menú de la ventana principal se muestra en la figura 3.3.1.a.

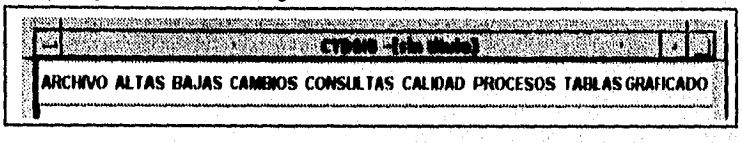


figura 3.3.1.a . Estructura del menú principal de CTDSIS

Dentro de esta ventana se encuentran una serie de objetos tipo texto (Ver apéndice ***B***) que nos permiten visualizar información del crucero oceanográfico que se encuentra en uso. Estos datos son:

1. ***Disco de datos de proyecto.-*** Muestra el disco de la VAX-Station donde residirá el archivo de control relacionado a este crucero.
2. ***Código del barco.-*** Siglas del buque oceanográfico empleado.
3. *Código del crucero.-* Año en el que se llevó a cabo el crucero.
4. ***Código de Datos.-*** Indica que los datos de este crucero fueron procesados bajo el formato CTD78 y representados bajo series precedidas de presión a cada dos decibarios.

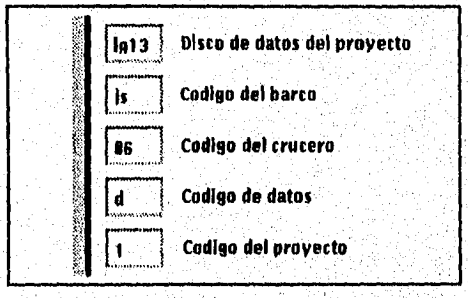


Figura 3.3.1.b, Objetos tipo texto para visualizar información acerca del crucero oceanográfico utilizado.

3.3.1.1 OPCIÓN ARCHIVO

Botón tipo Option Menu (Apéndice B) que despliega un PullDownMenu conteniendo 5 opciones para el manejo del archivo de control. Estas opciones son:

1. *NUEVO* Crea una estructura de datos de default para el manejo de datos de un nuevo crucero oceanográfico.

2. *ABRIR...* A partir de esta opción se despliega una caja de diálogo que permite navegar dentro del disco y elegir un archivo de control previamente creado .
3. *SALVAR* Salva todas las estructuras de datos para formar el archivo de control
4. *SALVAR COMO* Salva las estructuras de datos dentro de un archivo de control con diferente nombre al que fue abierto.
5. *SALIR* Termina la ejecución de CTDSIS.

Figura 3.3.1.1 Contenido de la opción ARCHIVO del menú principal de CTDSIS.

3.3.1.2 OPCIÓN ALTAS

Botón tipo Option Menu (Apéndice B) que despliega un PullDownMenu conteniendo 3 opciones para dar de alta datos del catálogo de crucero, de transectos y de nutrientes. Estas opciones son:

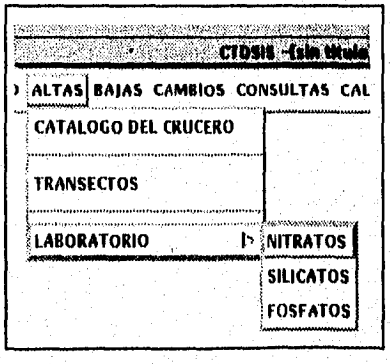


Figura 3.3.1.2. Contenido de la opción ALTAS del Menu principal de CTDSIS.

1. ***CATALOGO DEL CRUCERO.-*** Aparece la ventana para la captura del archivo del catálogo de crucero en su modo de altas (Permite incluir nuevas estaciones).
2. ***TRANSECTOS.-*** Se llama a la ventana de captura de datos de transectos en su modo de altas (Añade nuevas estaciones al archivo de transectos).
3. ***LABORATORIO.-*** Este es un CacadeButton que posee tres opciones;
   a) ***NITRATOS.-*** Permite capturar nuevos datos de nitratos e incluirlos al archivo de nutrientes.
   b) ***SILICATOS.-*** Permite capturar nuevos datos de silicatos e incluirlos al archivo de nutrientes.

c) ***FOSFATOS.***- Permite capturar nuevos datos de fosfatos e incluirlos al archivo de nutrientes.

3.3.1.3 OPCIÓN BAJAS

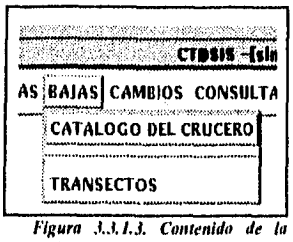


Figura 3.3.1.3. Contenido de la opción BAJAS del Menu principal de CTDSIS.

PullDownMenu conteniendo 2 opciones para dar de baja datos del catálogo de crucero y de transectos . Estas opciones son:.

1. ***CATALOGO DEL CRUCERO.***- Da de baja una estación del Catálogo de Crucero.
2. ***TRANSECTOS.***- Da de baja algún transecto del archivo de transectos

3.3.1.4 OPCIÓN CAMBIOS.

PullDownMenu conteniendo 3 opciones para dar de baja datos del catálogo de crucero, de transectos y de nutrientes. Estas opciones son:

1. ***CATALOGO DEL CRUCERO.***- Aparece la ventana para la captura del archivo del catálogo de crucero en su modo de cambios (Permite realizar cambios en los datos de las estaciones).
2. ***TRANSECTOS.***- Se llama a la ventana de captura de datos de transectos en su modo de cambios (Modifica los datos de transectos en el archivo de transectos).
3. ***LABORATORIO.***- Este es un CacadeButton que posee tres opciones:
   a) ***NITRATOS.***- Permite modificar datos de nitratos en el archivo de nutrientes.
   b) ***SILICATOS.***- Permite modificar datos de silicatos en el archivo de nutrientes.
   c) ***FOSFATOS.***- Permite modificar datos de fosfatos en el archivo de nutrientes

3.3.1.5 OPCIÓN CONSULTAS.

PullDownMenu conteniendo 3 opciones para consultar datos del catálogo de crucero, de transectos y de nutrientes . Estas opciones son:

1. ***CATALOGO DEL CRUCERO.***- Aparece la ventana para la captura del archivo del catálogo de crucero en su modo de consultas (Permite realizar consultas de los datos de las estaciones).
2. ***TRANSECTOS.***- Se llama a la ventana de captura de datos de transectos en su modo de consultas (Consulta los datos de transectos en el archivo de transectos).
3. ***LABORATORIO.***- Este es un CacadeButton que posee tres opciones:
   a) ***NITRATOS.***- Permite consultar datos de nitratos en el archivo de nutrientes.
   b) ***SILICATOS.***- Permite consultar datos de silicatos en el archivo de nutrientes.
   c) ***FOSFATOS.***- Permite consultar datos de fosfatos en el archivo de nutrientes.
4. ***CTD.***- Opción que pertenece al Módulo Generador de Datos, ajeno a este trabajo de tesis [4]

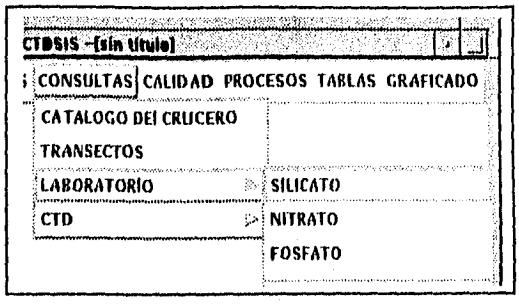


Figura 3.3.1.5. Contenido de la opción CONSULTAS del menú principal de CTDSIS.

3.3.1.6 OPCIÓN CALIDAD

Opción del menú principal que ofrece todas las opciones para la obtención de datos de calidad controlada (calibración de conductividad, de oxígeno y series uniformes de presión). Los botones disponibles para esta opción son:

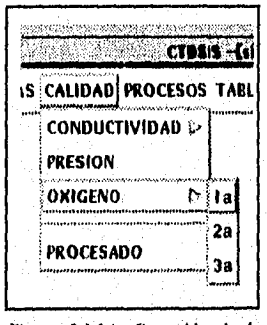


Figura 3.3.1.6. Contenido de la opción CALIDAD del menú principal de CTDSIS.

1. ***CONDUCTIVIDAD.-*** Despliega un Cascade-Button conteniendo tres opciones:
   a) ***1a.-*** Despliega la ventana para la calibración de conductividad en su primera etapa.
   b) ***2a.-*** Despliega la ventana para la calibración de conductividad en su segunda etapa.
   c) ***3a.-*** Realiza la ventana de calibración de conductividad en su tercera etapa.
2. ***PRESIÓN.-*** Trae al dispositivo de despliegue la ventana para ordenar por series de presión.
3. ***OXIGENO.-*** Despliega un Cascade-Button conteniendo tres opciones:
   a) ***1a.-*** Despliega la ventana para la calibración de oxígeno en su primera etapa.
   b) ***2a.-*** Despliega la ventana para la calibración de oxígeno en su segunda etapa.
   c) ***3a.-*** Realiza la ventana de calibración de oxígeno en su tercera etapa.
4. ***PROCESADO.-*** Activa al procedimiento "ctd_work_proc" [1] el cual ejecuta todos los pasos para que la calibración de datos se lleve a cabo.

[1] Ver Apéndice B.

3.3.2 PANTALLA PRINCIPAL DE LA VENTANA DE CATÁLOGO DE CRUCERO Y DE NUTRIENTES

Tanto la pantalla para la captura de datos del archivo del catálogo del crucero, como para la captura del archivo de nutrientes, poseen tres secciones principales:

a) ***Área de control.*** Posee los objetos (widgets) que controlan las características de las ventanas. Botones de salvar, cancelar, mover, objetos tipo escala para la elección de estaciones, etc. Estas secciones serán analizadas mas adelante para cada tipo de pantalla.

b) ***Área de Datos de Cabecera.-*** Área donde se despliega la información de cabecera para cada estación hidrográfica. Se despliegan datos como fechas, horas, localizaciones, profundidades, etc.(2.1.1.3). Esta área contiene los mismos datos para la pantalla de Catálogo de Crucero que para la pantalla de captura de nutrientes, aunque cabe señalar que esta sección solo es editable para el caso de altas o modificaciones de Catálogo, y solo sirve de área de consulta para la pantalla de nutrientes.

La figura 3.3.2.b muestra parte de la estructura de esta sección.

NUMERO ESTACION	PROFUNDIDAD MAXIMA	FECHA ESTACION	HORA LOCAL	HORA DRT	HORA GMT 2	LATITUD LINEA 2	LONGITUD LINEA 2
026	3700.0	20/JUL/85	04:26	00:34	10:26	23,00.1	93,59.9

Figura 3.3.2.b. Estructura de la sección de datos de cabecera para las pantallas del Catálogo de Crucero y de Nutrientes.

c) ***Área de Datos de Estación.*** En esta sección se encuentran todos los datos agrupados por niveles que pertenecen a una determinada estación hidrográfica. Se tiene un total de doce niveles de despliegue, es decir doce renglones de datos; y un rango de entre 17 o 19 columnas , dependiendo del tipo de ventana que se trate. Esta sección es editable solo para el caso de altas o modificaciones del Catálogo del Crucero; para el caso de la ventanas de nutrientes (silicatos, fosfatos, o nitratos) es solo una sección informativa. Los datos desplegados para el caso del Catálogo del crucero en esta sección se muestran en 2.1.1.3.3.1, mientras que para el caso de nutrientes el formato aparece en 2.1.1.3.3.2.

Esta sección esta compuesta por una barra de títulos, indicando el contenido de la columna, una barra de niveles, donde se despliega el número de nivel del renglón, y un conjunto de objetos texto agrupados en forma de hoja tabular.

La figura 3.3.2.c muestra parte de la estructura de esta sección.

	NUMERO BOTELLA	CABLE METROS	PRESION DECIBARES	TEMPERATURA AGUA	CONDUCTIVIDAD AGUA	SALINIDAD
11		2.0	0.4	28.658	58.826	36.223	23.1
10		25.0	25.4	28.601	59.058	36.523	23.3
9		50.0	50.1	25.200	54.710	36.115	23.1

Figura 3.3.2.c. Estructura de la sección de datos de estación para las pantallas principales de Catálogo y de nutrientes.

3.3.3 VENTANA DEL CATÁLOGO DE CRUCERO

A continuación se presenta el área de objetos de control para la ventana de catálogo de crucero. Se especificará el identificador de cada objeto, así como el callback asociado a ese objeto en particular.

1. ***Salvar.*** Botón que acepta todas la modificaciones realizadas durante la sesión de trabajo con esta ventana, vacía la estructura principal (fig 2.1.1.3.3) creando un nuevo archivo físico de catálogo de crucero. El identificador del botón es *k_cac_file_save_boton* y su callback es *cac_guarda_archivo* .
2. ***Salir.-*** PushButton que termina la ejecución de esta ventana, cerrando las listas correspondientes sin salvar al archivo. Si ID es *k_cac_file_exit_boton* y su callback *cac_sale_proc.*
3. ***Añade Estación.-*** Botón que tiene asociado un callback que permite la adición de una nueva estación de datos a la lista principal del catálogo. Identificador : *k_cac_añade_estacion.* Callback: *cac_proc_onade_estacion*
4. ***Borrar Estación.-*** Botón que permite eliminar de la lista principal del catálogo una determinada estación hidrográfica. Su id es *k_cac_boton_file_erase* y su callback es *cac_borra_estación* .
5. ***Movimientos.-*** CascadeButton que controla los movimientos de los niveles de datos para las estaciones hidrográficas. Contiene tres opciones, como se muestra en la figura (3.3.3b).
   a) ***Inserta.-*** Habilita un nuevo nivel de datos para la inserción de información. Si ya existen un total de 12 niveles ocupados, esta opción aparecerá desabilitada. El id de esta de esta opción es *k_cac_estacion_inserta_nivel,* su callback es *cac_operaciones.*
   b) ***Mueve.-*** Mueve un nivel de datos ya existente a otra posición dentro del área de datos. El id es *k_cac_estacion_mueve_nivel,* de esta opción se desprenden tres nuevas opciones:
      i) ***Arriba.-*** Mueve un nivel elegido hasta arriba de la pantalla. *k_cac_estacion_mueve_arriba.*

ii) ***Despues de.-*** Mueve un nivel elegido después de algún otro nivel, también elegido. *k_cac_estacion_mueve_despues.*

iii) ***Abajo.-*** Mueve un nivel de datos previamente elegido hasta abajo de la pantalla. *k_cac_estacion_mueve_abajo.*

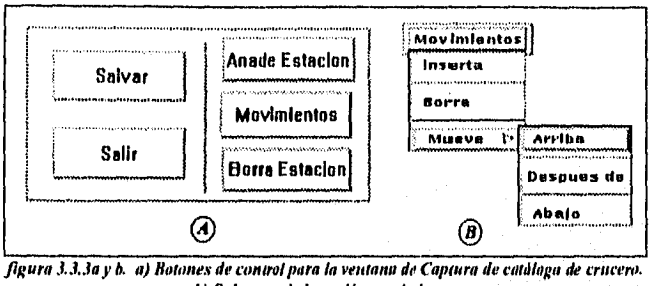


figura 3.3.3a y b. a) Botones de control para la ventana de Captura de catálogo de crucero. b) Submenu de la opción movimientos.

6.- ***Objeto ESCALA.-*** Permite al usuario elegir un número específico de algunas de estación para trabajar con ella. Id: *k_cac_scacale_estacion.* Callback: *cac_iguala_valores.*

7.- ***Aceptar.-*** Acepta el número de estación elegido con el objeto SCALE. id: *k_cac_acep_estacion,* callback: *cac_busca_estación.*

8.- ***Comentarios.-*** Aparece una nueva ventana llamada *cac_coment,* que tiene como fin el capturar comentarios a los datos capturados. El id del botón comentarios es *k_cac_comen_boton* y su callback es *cac_proc_llama.*

9.- ***Acepta datos de estación.-*** Acepta todos las modificaciones que se hayan realizado en la ventana, guardando estos datos en la lista principal de datos del catálogo. El id de este botón es *k_cac_guarda_datos_estacion_boton* y su callback es *cac_guarda_estacion_en_lista.*

10.- ***Cancela.-*** Cancela las modificaciones que se hayan realizado a la pantalla de datos. id: *k_cac_cancela_datos_estacion_boton,* callback: *cac_cancela_estacion_en_lista.*

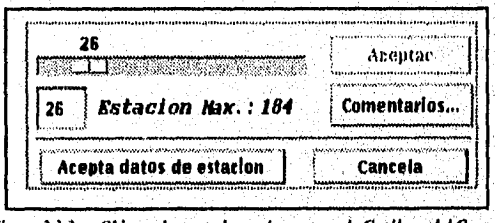


Figura 3.3.3.c. Objetos de control para la ventana de Catálogo del Crucero.

3.3.3.1 ELEMENTOS PARA LA CAPTURA DE DATOS TIPO HORA Y FECHAS

1. ***Botones flecha hacia arriba*** (▲) .- Objetos tipo ArrowButton que nos permiten incrementar una cantidad numérica. Incrementar datos como horas, minutos, dias o años. Existen este tipo de objetos en cajas de diálogo para la captura de horas y fechas. Los identificadores para este tipo de objetos son : *k_boton_fecha_anio_up, k_boton_fecha_dia_up, k_boton_hora_hora_up, k_boton_hora_minuto_up.* Los callbacks asociados son *cac_captura_fecha* y *cac_captura_hora.*

2. ***Batones flecha hacia abajo*** (▼) .- Objetos tipo ArrowButton que nos permiten decrementar una cantidad numérica. Decrementar datos como horas, minutos, dias o años. Existen este tipo de objetos en cajas de diálogo para la captura de horas y fechas. Los identificadores para este tipo de objetos son : *k_boton_fecha_anio_down, k_boton_fecha_dia_down, k_boton_hora_hora_down, k_boton_hora_minuto_down.* Los callbacks asociados son *cac_captura_fecha* y *cac_captura_hora.*

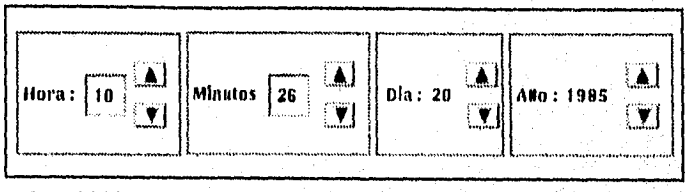


figura 3.3.3.1. Objeto para la captura de datos tipo fecha y hora para el catálogo del Crucero

3.3.3.2 OBJETOS PARA LA CAPTURA DE DATOS TIPO LATITUD O LONGITUD

1. ***Objetos tipo ESCALA.*** se utilizan objetos tipo escala para la captura de datos tipo latitud o longitud. Se utiliza un widget scale para capturar independientemente datos de grados, minutos y segundos. Los id's de estos tres objetos son: *k_cac_captura_longitud_grados, k_cac_captura_longitud_minutos* y *k_cac_captura_longitud_segundos.* El callback asociado a estos tres objetos son: *cac_captura_longitud.*

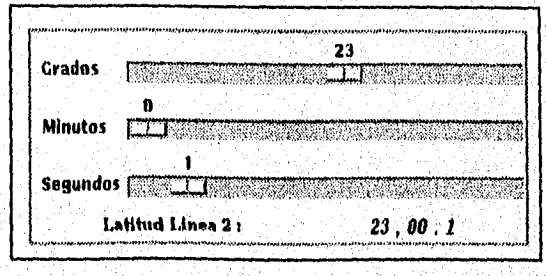


figura 3.3.3.2. Objetos tipo scale para la captura de datos tipo latitud y longitud.

3.3.4 VENTANA PARA LA CAPTURA DEL ARCHIVO DE TRANSECTOS.

3.3.4.1 OBJETOS PARA EL MANEJO DE TRANSECTOS.

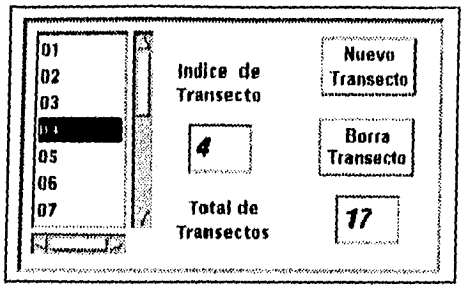


Figura 3.3.4.1. Objetos para el manejo de Transectos

1. ***Nuevo Transecto.*** Botón que aparece una caja de diálogo cuya finalidad es capturar el identificador del nuevo transecto que se quiere incluir al archivo de transectos.
El identificador de este widget botón es *k_tra_boton_anade_transectos.*
El callback asociado a este objeto es *tra_operaciones_transectos.*

2.- ***Borra Transecto.-*** Widget que elimina todo un transecto de la lista principal de transectos (2.1.1.3.3.3). Id : *k_tra_boton_borra_transectos* , callbacks: *tra_operaciones*

3. ***Índice de Transecto.*** Objeto tipo texto (*k_tra_label_numero_transecto*) que muestra el id del transecto elegido dentro de la lista de transectos.
4. ***Total de Transectos.-*** Widget tipo texto (*k_tra_texto_total_transectos*) que muestra el número total de transectos para el crucero oceanográfico seleccionado.
5. ***Objeto tipo Lista de Transectos.*** Widget tipo ScrolledList que despliega cada una de los transectos disponibles, permitiendo la elección de alguno de ellos. (*k_tra_lista_transectos*).

3.3.4.2 OBJETOS PARA EL MANEJO DE LA EDICIÓN DE LA ESTACIONES DE CADA TRANSECTO.

1. ***Edita Estación.-*** Botón que llama a la caja de diálogo *tra_caja_captura_datos_tran* que es útil para capturar loas estaciones del transecto, así como para añadir atributos a este.

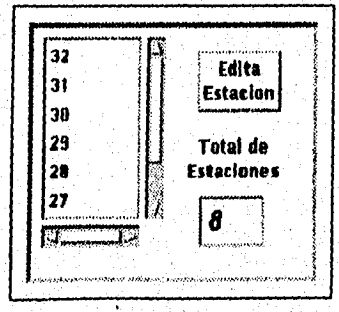


figura 3.3.4.2 Objetos para el manejo de estaciones de los transectos

El id de este botón es *k_tra_boton_edita_estacion*
El callback asociado es *tra_arrow_proc.*

2.- ***Total de Estaciones.-*** Objeto tipo texto que muestra el número total de estaciones asociadas al transectos elegido. El identificador de este widget es k_tra_texto_total_estaciones.

3.- ***Objeto Lista de Estaciones.-*** Widget tipo ScrolledList que muestra el conjunto de estaciones contenidas para el transecto seleccionado, permitiendo la elección de uno de ellos.
El identificador del objeto es *k_tra_lista_datos*

3.3.4.3 OBJETOS PARA EL MANEJO DE CARACTERÍSTICAS DE TRANSECTOS.

1. ***Texto para inserción de estaciones.-*** Objeto tipo ScrolledText que tiene como función el capturar los números de estaciones que conforman a un determinado transecto. Los números de estaciones se tendrán que capturar separados por coma o espacio. Si el número total de estaciones sobrepasa el área visual del texto, la barra de scroll entrará en acción visualizando las estaciona que se vayan perdiendo. El identificador de este widget es *k_tra_texto_total_estaciones.*

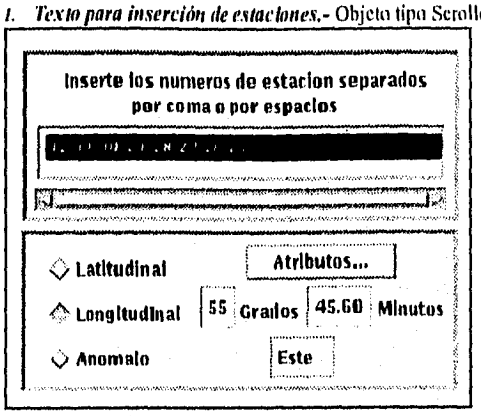


figura 3.3.4.3. Objetos para la captura de estaciones y manejo de características de transectos

2. ***Atributos.*** Botón que despliega una caja de diálogo llamada *cac_caja_base_log_win* que permite modificar los atributos del transecto. (*k_tra_boton_atributos*), callback : *tra_maneja_lat_long.*
3. ***ToggleButtons (Latitudinal, Longitudinal, Anómalo).*** Widgets tipo Toggle que permiten elegir el tipo de transecto con el que se trabaja. Si se elige el Toggle *Anómalo,* el botón de *Atributos* de desabilitará. Los id's de estos objetos son *k_tra_toggle_lati, k_tra_toggle_longi* y *k_tra_toggle_anomalo.*

3.3.4.4 OBJETOS PARA ATRIBUTOS DEL TRANSECTO.

ToggleButrons Norte_Sur_Este_Oeste. Widgets tipo toggle que permiten elegir la dirección de transecto, pudiendo ser, y dependiendo del tipo de transecto, norte o sur, si este es latitudinal; o este u oeste, si el transecto es longitudinal. Los id de esto objetos son: *k_tra_toggle_lat_norte, k_tra-toggle_lat_sur, k_tra_toggle_long_este* y *k_tra_toggle_long_oeste.*

1. ***Objetos para capturar latitudes o longitudes y dirección.*** Objetos tipo texto que permiten

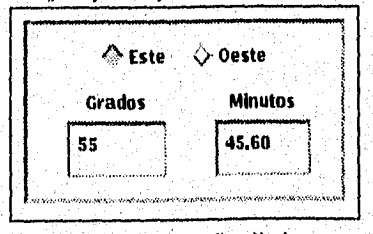


Figura 3.3.4.4. Objetos para dirección de transecto.

Objetos tipo texto que permiten capturar datos de grados y minutos para formar latitudes o longitudes. El texto de minutos acepta cifras con punto decimal para capturar segundos.
Los identificadores de estos objetos textos son: *k_tra_texto_lat_grados, k_tra_lat_minutos, k_tra_long_grados.*

3.3.5 VENTANA PARA LA CAPTURA DEL ARCHIVO DE NUTRIENTES

3.3.5.1 VENTANA INICIAL DE NUTRIENTES

La interfaz de inicio para la captura del archivo de nutrientes posee la siguiente estructura:

1. Área de objetos de control
2. Área de objetos para mostrar datos de cabecera.
3. Área de objetos para mostrar datos característicos de cada nivel de muestreo.

Tanto el área de objetos de datos de cabecera como la de datos para cada nivel de muestreo están estructuradas de la forma que se muestra en 3.3.2.

3.3.5.1.1 ÁREA DE OBJETOS DE CONTROL

1. ***Botón Aceptar.-*** Este PushButton acepta el número de estación que se haya elegido en el objeto scale. Tiene un callback asociado llamado *nut_busca_estacion_en_cat*[1] y su identificador es *k_nut_acep_estacion*.
2. ***Botón Edita Nutriente...*** Botón que llama a la ventana de edición de nutrientes[2] por niveles según sea el caso. El id de este objeto es *k_nut_edita_nutriente* y su callback es *nut_proc_llama*.
3. ***Objeto Scale (Estación).*** Objeto tipo scale que nos permite elegir el número de estación hidrográfica con la cual queremos trabajar. El id de este objeto es *k_nut_scale_estacion*.
4. ***Etiqueta Estación Max.*** Etiqueta (*k_nut_estacion_max_label*) que muestra el número total de estaciones disponibles en el archivo de catálogo de crucero.

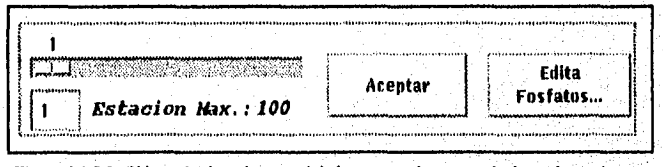


Figura 3.3.5.1. Objetos del área de control de la ventana de captura de datos de nutrientes.

3.3.5.2 VENTANA PARA LA EDICIÓN DE NUTRIENTES.

1. ***Objeto Lista de niveles.-*** Objeto XmScrolledList que almacena el total de niveles de muestreo que para la estación existen. Se da la oportunidad de elegir un nivel de datos y vaciar la información relacionada con ese nivel a los objetos de texto de edición. El id de ese objeto es *k_nut_lista_niveles*.
2. ***Botón Comentarios.-*** PushButton (*k_nut_coment*) que aparece una caja de diálogo para la captura de comentarios de la estación hidrográfica. El callback asociado es *nut_proc_llama*.

---

[1] La búsqueda de la estación elegida se hace sobre el archivo de catálogo de crucero antes de hacerse sobre el archivo de nutrientes.

[2] Existen tres tipos de ventanas *nut_caja_edita_nutrientes* para edición de nutrientes, una para cada clase (silicatos, fosfatos y nitratos). La etiqueta del objeto Pushbutton cambiará según el caso.

3. ***Objetos tipo texto.-*** Dependiendo del tipo de nutrientes que se vayan a capturar, existe una serie de objetos que sirven para capturar los datos[1] para cada nivel. En la siguiente tabla se enumeran cada uno de ellos y se mostrará en que tipo de ventana de nutriente aparece..

Nombre	Identificador	Descripción
Absorbancia 1	k_nut_caja_edita_texto_abs1	Dato de primera absorbacia. Para obtener silicatos, fosfatos y nitratos. Objeto de captura.
Absorbancia 2	k_nut_caja_edita_texto_abs2	Dato de segunda absorbancia. Para obtener silicatos, fosfatos y nitratos. Objeto de captura.
Absorbancia 3	k_nut_caja_edita_texto_abs3	Dato de tercera absorbancia. Para obtener silicatos, fosfatos y nitratos. Objeto de captura.
Error	k_nut_caja_edita_texto_error	Dato para capturar error. Para obtener silicatos, fosfatos y nitratos. Objeto de captura.
Factor de columna	k_nut_caja_edita_texto_factor	Dato para capturar factor de columna. Solo para obtener nitratos. Objeto de captura.
Absorbancia Promedio	k_nut_caja_edita_texto_absprom	Objeto de no_edicion. Despliega el valor promedio de los tres datos de absorbancias.
Concentración nutriente	k_nut_caja_edita_texto_concentra	Objeto de no_edición. Muestra el valor calculado de concentración del nutriente correspondiente.

Tabla 3.3.5.2. Relación de objetos tipo texto para la captura y despliegue de datos para el calculo de concentración de nutrientes.

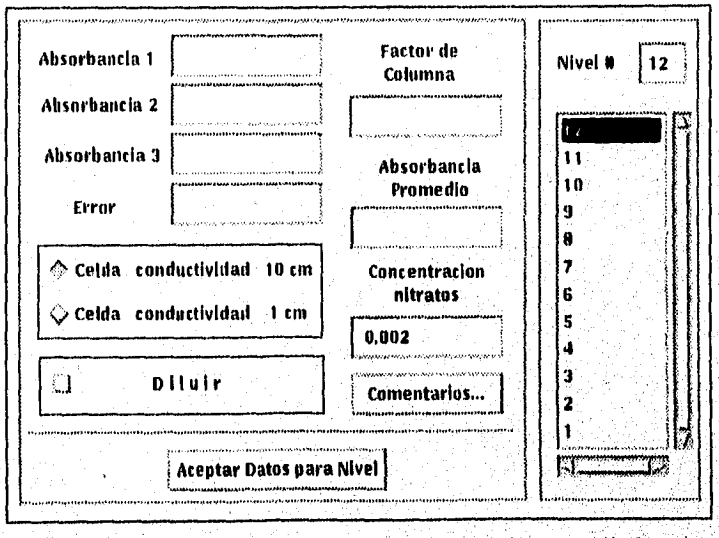


figura 3.3.5.2. Objetos para la captura y despliegue de datos para el calculo de concentración de nutrientes por nivel hidrográfico

[1] Se capturan datos de absorbancias, errores y factores de columna.

4. **Botón *Aceptar Datos Para Nivel.-*** Botón (*k_nut_caja_boton_acepta_datos*) que acepta los valores que se hayan capturado en los objetos texto de edición. Se dispara un callback (*nut_acepta_abs*) que calcula la absorbancia promedio y la concentración del nutriente.

5. **Toggle_Buttons *Celda de Conductividad.-*** Dos toggle_buttons que dentro de un Radio_Box que determinan el tipo de celda de conductividad a usar para determinar concentraciones de nitratos o de silicatos. Los id's de los objetos son: Para el toggle de 10 cm : *k_nut_caja_toggle_celda10*, y para el toggle de 1 cm : *k_nut_caja_toggle_celda01*.

6. ***Toggle_Button Diluir.-*** Toggle Button (*k_nut_caja_toggle_diluir*) que permite elegir al calcular nitratos si se diluye concentración o no.

3.3.6 VENTANA PARA LA CALIBRACIÓN DE CONDUCTIVIDAD EN SU PRIMERA ETAPA.

Para la ventana de calibración de conductividad[4] en su primera etapa solo se presentarán dos de sus siete secciones, ya que el presentar todas llevaría a un desbordamiento de información, debido a que solo para esta ventana existen mas de 100 objetos (widgets). El diseño de esta ventana cumple con el requerimiento mencionado en 1.3.4.8.2.

Cada uno de los objetos de esta ventana tiene un equivalente en las estructuras de datos[5]. Al leerse las estructuras al inicio de CTDSIS, estas se vacían a sus correspondientes widgets de la ventana..

3.3.6.1 Sección de Switches del simulador de HP2100.

Existen 8 switches representados por un toggle_button cada uno, el estado del toggle button representa el estado del switch. Los identificadores de estos widgets son:
k_smcc1a_toggle_switch3, k_smcc1a_toggle_switch4, k_smcc1a_toggle_switch6, k_smcc1a_toggle_switch10, k_smcc1a_toggle_switch11, k_smcc1a_toggle_switch12, k_smcc1a_toggle_switch15, k_smcc1a_toggle_switch16.

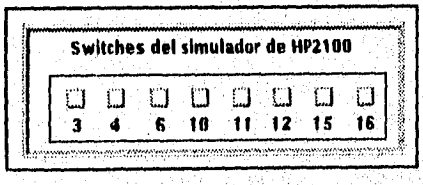


Figura 3.3.6.1. Objetos que forman la sección de la rama 5 en la calibración de conductividad en su primera etapa

[4] El formato completo de la ventana de calibración de conduct. en su primera etapa se presenta en el apéndice E.
[5] Ven las estructuras para la calibración de conductividad en 2.1.1.3.3.4.1

3.3.6.2 Sección de Intercambio de columnas (Rama 12)

Esta sección de la ventana de calibración de conductividad (primera etapa) contiene 4 tipos de widgets: (PushButton, OptionMenu, ArrowButton y Text). Este estilo de secciones es común para el diseño de las ventanas de calibración de conductividad, es decir la mayoría de las secciones poseen esta clase de objetos; y todas utilizan listas ligadas[1] para almacenar la información que en estos se capturen. La función de cada widget de esta sección tiene un comportamiento similar en cada una de las demás secciones en las que aparezcan..

1. ***Botón Nuevo:*** Este Pushbutton (*k_smcc1a_boton_nuevo*) tiene asociado un callback (*smcc1a_nuevos_procs*) el cual permite la inserción a la lista de un nuevo nodo.
2. ***Botón Elimina.-*** Botón que permite eliminar un nodo de la lista de intercambio mediante el callback *smcc1a_elimina_listas*. El id es *k_smcc1a_boton_elimina*.
3. ***Botón Intercambia.-*** Incluye un nodo a la lista de intercambio, eligiendo previamente los datos de los option menu correctos. El callback asociado es *smcc1a_inserta_listas* y su id es *k_smcc1a_boton_intercambia*.
4. ***Botón Modifica.-*** Al elegir previamente un determinado nodo de la lista de intercambio, este widget nos permite modificar los datos de este. El id es *k_smcc1a_boton_modifica* y su callback es *smcc1a_modifica_listas*.
5. ***Botones Option Menus.-*** Botones tipo menu que nos permiten elegir una de entre ocho opciones de variables. El valor que presente cada uno de estos option menus será el que se acepte para ingresar a la lista ligada. Los id's de cada uno de estos option menus son:
   Para el option_menu DE: *k_smcc1a_option_menu_inter_de*.
   Para el option_menu POR: *k_smcc1a_option_menu_inter_por*.

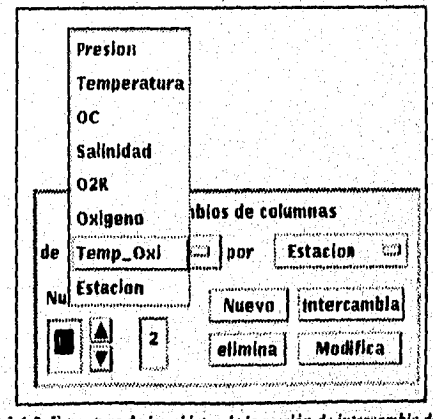


Figura 3.3.6.2. Estructura de los objetos de la sección de intercambio de columnas de la ventana de calibración de conductividad en su primera etapa.

[1] La estructura de las listas ligadas de cada sección de las interfaces de calibración de datos se encuentran en 2.1.1.3.3.4.1. Los procedimientos que pertenecen a cada ventana de calibración se pueden consultar en 2.2.2.

3.3.7 VENTANA DE CALIBRACIÓN DE CONDUCTIVIDAD EN SU SEGUNDA ETAPA.

Esta ventana tiene solamente tres secciones:

a) Una sección de switches similar a 3.3.6.1.
b) Una sección de intercambio de columnas como en 3.3.6.2.
c) Una sección de elección de función de conversión.

3.3.7.1 Sección para la elección de la función de conversión.

Esta sección consta de solo dos objetos, uno tipo XmScrolledList y un Toggle_button.

Objeto Lista.- Desde el widget tipo "lista" (*k_smcc2a_lista_función*) podemos elegir una de entre 4 diferentes fórmulas para la conversión de datos de salinidad a conductividad, o viceversa.
Las cuatro fórmulas[2] disponibles son:

a) De Conductividad a Salinidad (antigua formula de salinidad)
b) De Salinidad a Conductividad (antigua formula de salinidad)
c) De Conductividad a Salinidad (UNESCO-78 'SAL78')
d) De Salinidad a Conductividad (UNESCO-78 'SAL78')

2. ***Objeto Toggle_button.-*** Toggle_button (*k_smcc2a_toggle_aplica_funcion*) nos permite elegir si deseamos que la fórmula elegida en el widget "lista" se aplique a los procedimientos de calibración.

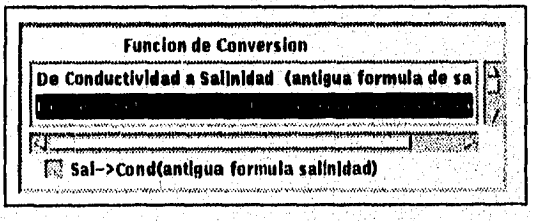


Figura 3.3.7.1. Sección De elección de función de conversión para la calibración de conductividad en su segunda etapa.

3.3.8 VENTANA PARA LA CALIBRACIÓN DE CONDUCTIVIDAD EN SU TERCERA ETAPA.

La ventana para la calibración de conductividad en su tercera etapa esta compuesta por seis secciones, las cuales son:

[2] La descripción completa de cada fórmula de conversión se pueden consultar en Millard, R.C. CTD Calibrationand Data Pnecssing Techniques at WHOI using the 1978 Practical Salinity Scale, Proceddings International STD Conference and Workshop, 1982.

1. Edición de tabla de datos.
2. Sección de switches del simulador de HP2100.
3. Sección para promediar columna y determinar desviación.
4. Sección de comentarios
5. Edición de términos de regresión
6. Sección de Histogramas.

Se presentará el diseño de la sección tres (rama 8 y rama 29) y de la sección seis (rama 4).

3.3.8.1 Sección de edición de histogramas de la ventana de calibración de conductividad en su tercera etapa (rama 4).

1. ***Botón Nuevo:*** Este Pushbutton (*k_smcc3a_boton_nuevo*) tiene asociado un callback (*smcc3a_nuevos_procs*) el cual permite la inserción a la lista[1] de un nuevo nodo.

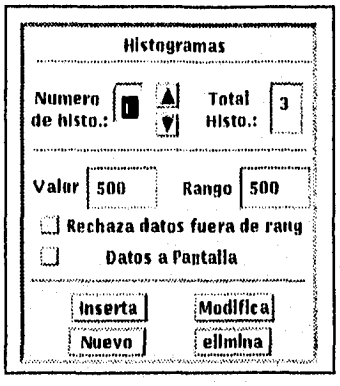


Figura 3.3.8.1. Objetos de la sección de edición de histogramas de smcc3a etapa.

2. ***Botón Elimina.-*** Botón que permite eliminar un nodo de la lista de histogramas mediante el callback *smcc1a_elimina_lista_histo.* El id es *k_smcc2a_boton_elimina_histograma.*

3. ***Botón Inserta.-*** Incluye un nodo a la lista de histogramas, capturando previamente los datos correctos de los objetos de edición . El callback asociado es *smcc1a_inserta_lista_histo* y su id es *k_smcc3a_boton_inserta_histograma.*

4. ***Botón Modifica.-*** Al elegir previamente un determinado nodo de la lista de histograma, este widget nos permite modificar los datos de este. El id es *k_smcc3a_boton_modifica_histo* y su callback es *smcc3a_mod_lista_histo.*

5. ***Toggle Rechaza datos.-*** El id de este objeto es: *k_smcc3a_toggle_histograma_rechaza* y permite elegir si se desean rechazar datos fuera de rango.

6. ***Toggle Datos a pantalla.-*** *k_smcc3a_toggle_histograma_pantall* al ser verdadero nos permite ver en pantalla el resultado de los histogramas.

7. ***Objetos Texto.-*** *k_smcc3a_texto_histograma_rango y k_smcc3a_texto_histograma_valor* son widget tipo texto para capturar datos de rango y valor respectivamente de las variables a utilizar.

[1] La estructura de la lista de histogramas (Rama 4) se encuentra disponible en 2.1.1.3.3.4.2.f.

8. ***Widget List.-*** Objeto que almacena cada uno de los nodos de la lista de histogramas. Nos permite realizar un recorrido de la lista mediante los objetos ArrowButton. El id de la lista es *k_smcc3a_lista_histograma*

3.3.8.2 Sección para promediar columna y determinar desviación (rama 8 y 29).

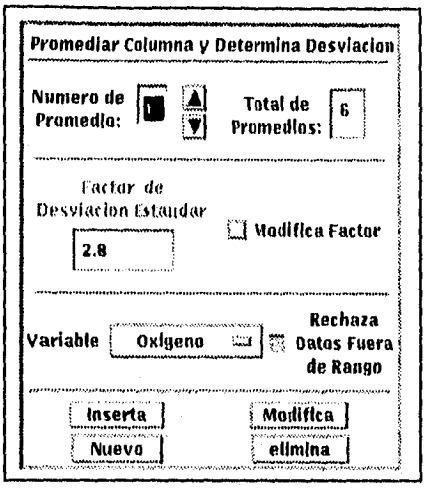


Figura 3.3.8.2. Sección para promediar columna y determinar desviación (rama 8 y 29).

1. ***Botón Nuevo:*** Este Pushbutton (*k_smcc3a_boton_nuevo*) tiene asociado un callback (*smcc3a_nuevos_procs*) el cual permite la inserción a la lista[4] de un nuevo nodo.
2. ***Botón Elimina.-*** Botón que permite eliminar un nodo de la lista de histogramas mediante el callback *smcc3a_elimina_lista_prome.* El id es *k_smcc3a_boton_elimina_promedio.*
3. ***Botón Inserta.-*** Incluye un nodo a la lista de histogramas, capturando previamente los datos correctos de los objetos de edición . El callback asociado es *smcc3a_inserta_listaprome* y su id es *k_smcc3a_boton_inserta_promedia.*
4. ***Botón Modifica.-*** Al elegir previamente un determinado nodo de la lista de histograma, este widget nos permite modificar los datos de este. El id es *k_smcc3a_boton_modifica_prome* y su callback es *smcc3a_mod_lista_prome.*
5. ***Toggle Rechaza datos.-*** El id de este objeto es: *k_smcc3a_toggle_promedia_rechaza* y permite elegir si se desean rechazar datos fuera de rango.
6. ***Toggle Modifica Factor.-*** Toggle Button que habilita o desabilita al objeto texto *k_smcc3a_texto_promedia_factor* para modificar su valor. El id del Toggle es *k_smcc2a_toggle_promedia_modifica.*
7. ***Option Menu Variable.-*** Widget tipo OptionMenu el cual despliega una serie de ocho tipos de variables de las cuales tenemos la oportunidad de elegir solo una. El Id es *k_smcc3a_option_menu_promedia*
8. ***Widget List.-*** Objeto que almacena cada uno de los nodos de la lista de promedios. Nos permite realizar un recorrido de la lista mediante los objetos ArrowButton. El id de la lista es *k_smcc3a_lista_promedia.*
9. ***Objeto texto Factor Desviación Estándar.-*** *(k_smcc3a_texto_promedia_factor)* widget tipo texto en el cual capturamos el nuevo factor de para la desviación estándar. Este objeto aparece inhabilitado cuando el objeto k_smcc2a_toggle_promedia_modifica se encuentra en la posición de falso.

[4] El formato de esta lista se encuentra en 2.1.1.3.3.4.2.d.e

3.3.9 VENTANA PARA ORDENAR DATOS BAJO SERIES ORDENADAS DE PRESIÓN.

1. ***Objetos Texto.-*** Se presentan un total de cinco widgets tipo texto donde se depositan datos extraídos de un archivo[1] generado en la etapa de calibración anterior (calibración de conductividad en su tercera etapa), es decir, estos objetos no son para edición; sino solo para desplegar los datos ya mencionados.

   Los identificadores de los objetos tipo texto son: *k_mpd_smp_texto_sesgo, k_mpd_smp_texto_pendiente, k_mpd_smp_texto_retrazo, k_mpd_smp_texto_atr1, k_mpd_smp_texto_atr2.*

2. ***Botón Modificar Intervalo de Presión.-*** Este widget tipo Pushbutton (*k_mpd_smp_madifica_intervalo*) tiene asociado un callback (*mpd_smp_trae_ventana*) el cual trae una caja de diálogo (*mpd_smp_caja_cambia_intervalo*) que nos permite modificar el intervalo de presión mediante el cual queremos ordenar las series de datos.

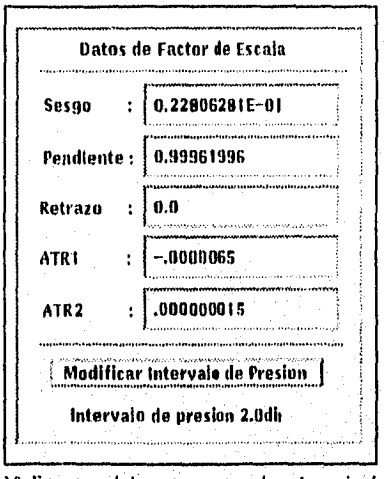


Figura 3.3.9. Estructura de la ventana para ordenar las series de presión.

3.3.9.1 CAJA DE DIÁLOGO PARA MODIFICAR INTERVALO DE PRESIÓN.

1. ***Widget Scale Intervalo de Presión.-*** Objeto que nos permite elegir el nuevo valor de intervalo de presión. El id es *k_mpd_smp_scale_intervalo*

[1] El archivo de donde se extraen los datos mostrados en estos widgets tipo texto es p.j. JS861bp.con

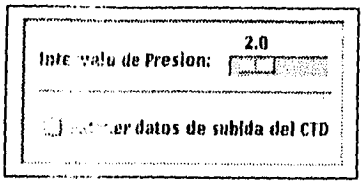


Figura 3.3.9.1. Caja de Diálogo para modificar el intervalo de presión.

2. ***Widget Toggle Extraer Datos de Subida.-*** Elemento tipo ToggleButton que permite elegir si se desean contemplar los datos que se sensaron a la subida del CTD. Si id es *k_mpd_smp_toggle_extraer.*

3.3.10 VENTANAS PARA LA CALIBRACIÓN DE DATOS DE OXÍGENO EN SU PRIMERA Y TERCERA ETAPA.

Debido a que el diseño de las ventanas para la calibración de oxigeno en su primera y en su tercera etapa son muy similares, bastará con mostrar la estructura de los objetos de la ventana de calibración en su primera etapa, y solo se hará la referencia a los identificadores de los widgets correspondientes de la tercera etapa.

Las estructuras de objetos que explicaremos son :

1. Un sección de las cajas de diálogo principales para la calibración de oxígeno en su primera y tercera etapa.
2. Una parte de la caja de diálogo para elegir ciertas estaciones.
3. Una sección de la caja de diálogo para elegir un rango de estaciones.

3.3.10.1 ELEGIR EL TIPO DE CONJUNTO DE ESTACIONES.

Esta sección nos permite elegir el tipo de conjunto de estaciones con el cual queremos trabajar. Esta sección la componen tres tipo de objetos : ToggleButtons, PushButton y ScrolledList.

1. ***Widgets Toggle Buttons.-*** Existen tres toggle buttons para esta sección, cada uno de ellos elige un tipo de conjunto de estaciones.

a) ***Toggle Todas.-*** Este toggle Button (*k_smco1a_toggle_todas*[2]) elige como datos de entrada a todas las estaciones que para ese crucero se encuentren disponibles[3]. Coloca todos los identificadores de estaciones en el objeto "lista" y desabilita al botón *Selecciona Estaciones*

[2] *k_smco3a_toggle_todas* para la ventana de la tercera etapa.

[3] La aplicación *smco1a_lee_mdirfil y smco3a_lee_mdirfil* (Ver 2.2.3.1.5.8.2.1) son las encargadas de determinar cuantas estaciones se encuentran disponibles para algún crucero oceanográfico.

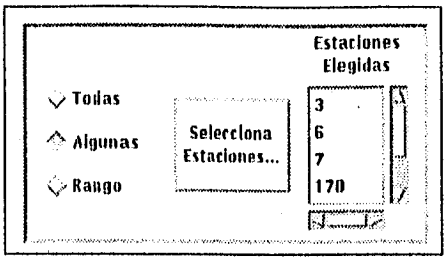


Figura 3.3.10.1. Estructura de los objetos para la ventana de calibración de axigeno en su primera y tercera etapa. Permite elegir el tipo de conjunto de estaciones con la que se quiera trabajar.

b) ***Toggle Algunas.-*** *Toggle (k_smcola_toggle_algunas*[4]*)* que habilita al botón *Selecciona Estaciones..* y coloca en el objeto lista solo las estaciones elegidas por la caja de diálogo *smcola_caja_incluye_estaciones*[5].

c) ***Toggle Rango.-*** *Toggle (k_smcola_toggle_rango*[6]*)* que habilita al botón *Selecciona Estaciones..* y coloca en el objeto lista solo el rango de estaciones señaladas por la caja de diálogo *smcola_caja_rango_max_min*[7]

2. ***Botón Selecciana Estaciones.-*** Objeto tipo PushButton (*k_smcola_boton_edita*[8]) que aparece una caja de diálogo para capturar los identificadores de las estaciones deseadas. Dependiendo del estado del los toggle buttons este botón aparecerá habilitado o desabilitado; y desplegará uno de dos diferentes cajas de diálogo según el caso:

   a) Si el toggle button Algunas esta encendido, el botón k_smcola_boton_edita aparecerá habilitado, y traerá la ventana smcola_caja_incluye_estaciones.

   b) Si el toggle button Rango esta encendido, el botón k_smcola_boton_edita aparecerá habilitado, y traerá la ventana smcola_caja_rango_max_min.

   c) Si el toggle Todas esta encendido, el botón k_smcola_boton_edita aparecerá desabilitado.

3. ***Objeto Scrolled List Estaciones Elegidas.-*** Los identificadores de las estaciones elegidas se mostrarán en este widgel (*[9]k_smcola_listas_elegidas*).

---

[4] *k_smco3a_toggle_algunas* para la ventana de la tercera etapa.
[5] *k_smco3a_caja_incluye_estaciones* para la tercera etapa.
[6] *k_smco3a_toggle_rango* para la ventana de la tercera etapa.
[7] *k_smco3a_caja_estacion_max_min* para la tercera etapa.
[8] *k_smco3a_boton_edita* para el caso de la ventana de tercera etapa
[9] *k_smco3a_listas_elegidas* para la ventana de la tercera etapa.

3.3.10.2 CAJA DE DIÁLOGO PARA ELEGIR ALGUNAS ESTACIONES
(*k_smcola_caja_incluye_estaciones*[10]).

1. ***Objeto Lista Estaciones Disponibles.-*** Objeto tipo *Scrolled* List que posee a todas las estaciones disponibles del crucero oceanográfico en cuestión. (*smcola_lista_disponibles*[11]).

2. ***Objeto Lista Estaciones Elegidas.-*** Widget tipo "lista" que guarda solo los identificadores de las estaciones elegidas. (*k_smcola_lista_incluidas*[12]).

3. ***Botón Incluir.-*** Botón que dada una estación elegida es la lista de estaciones disponibles, la coloca en la lista de estaciones elegidas. El callback es *smcola_incluye_lista_elegida*. El id es *k_smcola_botón_incluye*[13].

4. ***Botón Excluir.-*** PushButton (*k_smcola_botón_excluye*) que elimina una estación (previamente elegida) de las lista de estaciones elegidas. El callback asociado a este botón es *k_smcola_boton_excluye*[14].

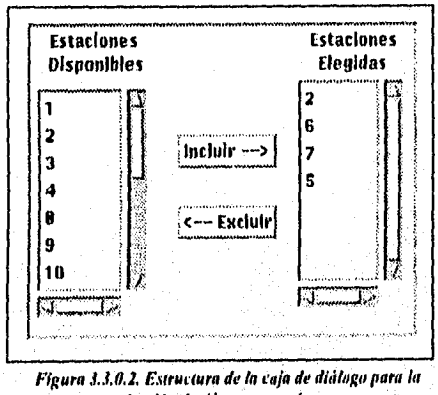


Figura 3.3.0.2. Estructura de la caja de diálogo para la elección de Algunas estaciones.

[10] *k_smco3a_incluye_estaciones* para la calibración de conductividad en su tercera etapa.
[11] *k_smco3a_lista_disponibles* para la tercera etapa.
[12] *k_smco3a_lista_incluidas* para la tercera etapa.
[13] *k_smco3a_botón_incluye* para la tercera etapa.
[14] *k_smco3a_boton_excluye* para la tercera etapa.

3.3.10.3 CAJA DE DIÁLOGO PARA ELEGIR UN RANGO DE ESTACIONES (*k_smcola_caja_rango_max_min*[15]).

1. ***Objeto scale Rango Mínimo:*** Widget tipo scale (*k_smcola_scale_rango_minimo*) que nos permite capturar el rango mínimo de estaciones.
2. ***Objeto sacle Rango Máximo:*** Widget tipo scale (*k_smcola_scale_rango_maximo*) para capturar el rango máximo de las estaciones.
3. ***Texto Rango Mínima:*** Texto que nos indica cual es el mínimo rango que se puede elegir. (*k_smcol_texto_rango_mínimo*).
4. ***Texto Rango Máximo.-*** Texto que nos indica cual es el mínimo rango que se puede elegir. (*k_smcol_texto_rango_máximo*).

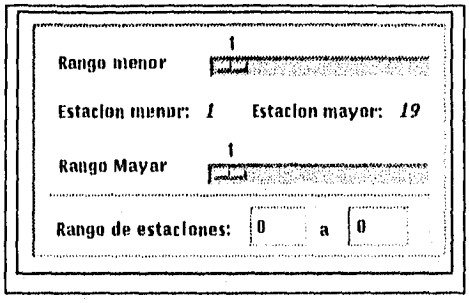


Figura 3.3.10.3 . Caja de diálogo para capturar un rango de estaciones.

3.3.11 VENTANA PARA LA CALIBRACIÓN DE OXÍGENO EN SU SEGUNDA ETAPA.

La interfaz para la calibración de datos de oxígeno en su segunda etapa esta compuesta por dos cajas de diálogo. La primera caja presenta datos de todos los ajustes capturados hasta ese momento, en la segunda caja se pueden editar cada uno de los datos de estos ajustes.

3.3.11.1 VENTANA DE PRESENTACIÓN DE AJUSTES.

Esta ventana presenta cuatro secciones de objetos:

1. Área para capturar datos de Factor de Oxígeno y de Oxígeno Mínimo.
2. Área de control para recorrer la lista de ajustes[16].
3. Área de despliegue de datos para cada ajuste.
4. Área de objetos tipo botón para añadir, eliminar o modificar un ajuste determinado.

[15] *k_smcola_caja_rango_max_min* para la calibración de conductividad en su tercera etapa.
[16] La estructura de la lista ligada para la ventana de calibración de oxígeno se encuentra en 2.1.1.3.3.4.4.

ZNAM - GEO - JJE

3.3.11.1.1 ÁREA DE OBJETOS TIPO BOTÓN PARA AÑADIR, ELIMINAR O MODIFICAR UN AJUSTE DETERMINADO.

Figura 3.3.11.1.1. Botones para añadir, borrar o modificar un ajuste.

1. ***Botón Nuevo.-*** PushButton que permite insertar un nuevo nodo de datos de ajuste a la lista ligada de smco2a. Se tiene asociado un callback (*smco2a_aparece_win_captura*) el cual aparece la caja de diálogo smco2a_captura_ajuste_win. El id de este widget es *k_smco2a_boton_nuevo_ajuste*.
2. ***Botón Borra.-*** Widget PushButton que elimina de la lista de ajustes un nodo determinado. Este nodo es elegido mediante los objetos ArrowButton. El id es *k_smco2a_boton_borra_ajuste* y el callback asociado es *smco2a_borra_ajuste_proc*.
3. ***Wieget Modifica.-*** Objeto (*k_smco2a_boton_modifica_ajuste*) que previamente habiendo elegido un ajuste determinado, nos permite modificar los datos relacionados con este. Se trae a la ventana *smco2a_aparece_win_captura* para este propósito.

3.3.11.1.2 ÁREAS PARA CAPTURAR DATOS DE OXÍGENO Y RECORRER LA LISTA DE AJUSTES.

1. ***Texto Factor de Oxígeno.-*** Objeto tipo XmText para capturar el valor del factor de oxígeno. *k_smco2a_texto_factor_o2*.
2. ***Texto Oxígeno Mínimo.-*** Objeto tipo XmText para capturar el valor de oxígeno mínimo. *k_smco2a_texto_o2_minimo*.
3. ***Objetos ArrowButton (Flechas).-*** Widgets tipo ArrowButton que nos permite recorrer la lista ligada de ajustes (hacia abajo o hacia arriba). El callback asociado a estos objetos es *smco2a_up_down_proc*. Los id's de cada widget son *k_smco2a_boton_up_ajuste* y *k_smco2a_boton_down_ajuste*.
4. ***Texto Número de Ajuste.-*** Muestra el número de ajuste o nodo de la lista de ajuste en el que nos encontramos. *k_smco2a_texto_nn_ajuste*.
5. ***Texto Total de Ajustes.-*** Muestra el número total de ajustes o nodos disponibles en la lista ligada de ajustes. *k_smco2a_texto_total_ajuste*.

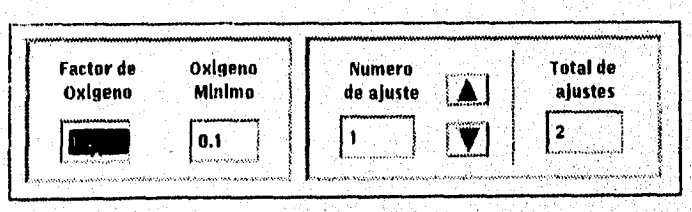


Figura 3.3.11.1.2 Áreas para capturar datos de oxígeno y recorrer la lista de ajustes.

3.3.11.2 VENTANAS PARA EDICIÓN DE DATOS DE AJUSTE.

La ventana para edición de datos de ajuste posee cinco áreas específicas las cuales son:

1. Datos de edición.
2. Valores de Parámetros.
3. Empezar desde parámetro.
4. Valores de Proceso.
5. Número de parámetros a ajustar.

3.3.11.2.1 ÁREA DE VALORES DE PROCESO.

1. ***Texto Dígitos Significativos.-*** Se edita el número de dígitos significativos de la cantidades usadas para este proceso. *k_smco2a_texto_nn_digitos.*
2. ***Texto Número de Iteraciones.-*** Objeto que capta el número de iteraciones necesarias para este proceso. *k_smco2a_texto_nn_iteracion.*

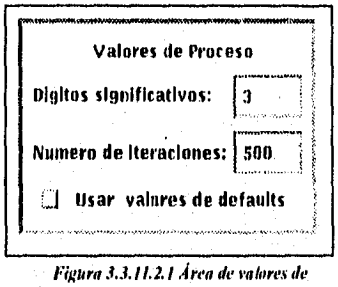


Figura 3.3.11.2.1 Área de valores de proceso de smco2a.

3. ***Toggle Usa Valores de Default.-*** Widget tipo toggle button, que determina si se utilizan valores de default para los datos *k_smco2a_texto_nn_digitos* y *k_smco2a_texto_nn_iteracion* . En caso de que el toggle se encuentre encendido, los objetos texto aparecerán desabilitados. El id es *k_smco2a_toggle_valor_default.*

3.3.11.2.2 ÁREA PARA DETERMINAR EL NÚMERO DE PARÁMETROS A UTILIZAR.

Esta sección solo contiene un objeto tipo Scale que permite elegir el número de parámetros a ajustar en este proceso. El id es *k_smco2a_scale_nn_parametros.*

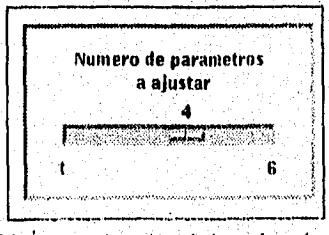


Figura 3.3.11.2.2 Área para determinar el número de parámetros a utilizar.

Capítulo 4

Garantía y Verificación de la Interfaz Gráfica.

INTRODUCCIÓN

La importancia de la prueba del software y sus implicaciones en la calidad del software no se pueden pasar por alto. *"El desarrollo de sistemas de software implica una serie de actividades de producción en las que las posibilidades de que aparezca la falibilidad humana son enormes. Los errores pueden empezar a darse desde el primer momento del proceso, en el que los objetivos.... puede estar especificados en forma errónea o imperfecta, así como [los errores que aparecen] en los posteriores pasos de diseño y desarrollo... Debido a la imposibilidad humana de trabajar y comunicarse en forma perfecta, el desarrollo de software ha de ir acompañado de una actividad que garantice la calidad".*

Las pruebas del software es un elemento crítico para la garantía de calidad del software y representa una revisión final de las especificaciones, del diseño y de la codificación.

OBJETIVOS DE LAS PRUEBAS.

Los objetivos que se persiguen al desarrollar un número determinado de casos de prueba son los siguientes:

1. La Prueba es un proceso de ejecución de un programa con la intención de descubrir un error.
2. Un buen caso de prueba es aquel que tiene una alta probabilidad de mostrar un error no descubierto hasta entonces.
3. Una prueba tiene éxito si descubre un error no detectado hasta entonces.

Conociendo la función especifica para el que fue diseñado el sistema MPD, se pueden llevar a cabo pruebas que demuestren que cada función es completamente operativa. Este enfoque de prueba se denomina: *prueba de la caja negra.*

PRUEBA DE LA CAJA NEGRA.

Los casos de prueba de la caja negra que implementamos para MPD nos permitió obtener conjuntos de condiciones de entrada que ejercitaron completamente todos y cada uno de los requisitos que se listaron en el capítulo 1.

Este tipo de pruebas intenta encontrar errores de las siguientes categorías:

- Funciones Incorrectas o Ausentes
- Errores de Interfaz
- Errores en estructuras de datos o en accesos a bases de datos externas.
- Errores de rendimiento.
- Errores de Inicialización o de terminación.

Todas y cada una de las pruebas realizadas se efectuaron sobre datos recopilados durante el crucero oceanográfica de 1985 en su segunda versión. Es decir se usar datos del crucero JS085D002.

4.1 CASO NÚMERO 1

Nombre: *Altas, Bajas, Cambios y Consultas del subsistema para la captura del archivo del Catálogo del Crucero.*

Objetivo: Verificar el correcto funcionamiento de los procedimientos correspondientes (así como de la interfaz) de la captura de datos para el archivo de Catálogo de Crucero. Se checará la correcta visualización de todos los datos de estación en la estructura de la interfaz gráfica, así como cada una de las funciones diseñadas para este programa, es decir, altas, bajas, cambios y consultas de estaciones hidrográficas.

Descripción de la Prueba:

a) *Altas:* Partiendo de la ventana principal CTDSIS, se activa el menú *ALTAS* opción *Catálogo de Crucero.*

b) *Bajas:* Activa menú *BAJAS* de CTDSIS, eligiendo opción *Catálogo de Crucero.*

c) *Cambios:* Activar en CTDSIS el menú *CAMBIOS* para escoger *Catálogo del Crucero.*

d) *Consultas:* Elige de menú *CONSULTAS* la opción *Catálogo del Crucero.*

Criterios de Aceptación:

Los datos correspondientes a cada estación hidrográfica del Catálogo de Crucero se deberán mostrar en forma correcta dentro de cada una de las secciones de la ventana para dicho fin. De igual forma al realizar cambios de datos de alguna estación (inserción, borrado y movimiento de algún nivel), estos deberán reflejarse en forma correcta.

Se deberá comprobar cualquier tipo de acción de los datos de catálogo dentro del archivo físico JS085d002.CAT

Procedimientos de la Prueba:

a) ***Altas:*** Se oprime el botón [*Anade Estación*][1], al encenderse el sensitive de las secciones de cabecera y de datos[2] se procede a insertar datos en cada una de estas secciones (por lo menos un nivel de datos). En seguida se oprime [*Acepta Datos de Estación*][3] para aceptar los datos recién capturados e incrustarlos en la lista ligada principal[4]. Se presiona [*Salvar*][5] para general el archivo físico.

---

[1] Observar en 3.3.3 (3)
[2] Ver 3.3.2.b y 3.3.2.c respectivamente
[3] Ver 3.3.3 (9)
[4] Ver 2.1.1.3.3

b)***Bajas:*** Elige el número de estación '3' en el objeto *Scale*[1]. Oprime [*Aceptar*]. Presiona [*Borra Estación*][2]. Oprime [*Salvar*].

c)***Modificaciones:*** Elige estación '2' en objeto *Scale*. Oprime [*Aceptar*]. Elige el nivel '3' de datos. Oprime el menú [*Movimientos*][3] y elige opción [*Arriba*]. Selecciona [*Salvar*]

d)***Consultas***: Elige número de estación '8' en objeto *Scale*. Oprime [*Aceptar*].

Resultados de la Prueba:

Para todos los casos en el que se requirió abrir el archivo JS085D002.DAT, el resultado fue satisfactorio, es decir, se mostraron cada uno de los datos sin error alguno en cada uno de los objetos correspondientes de la interfaz gráfica[1]. Los procedimientos que habilitan y desabilitan algunos objetos de la ventana según el caso también funcionó satisfactoriamente.

Por último, se mostró que el archivo físico JS085D002.CAT[5] si sufrió las modificaciones echas por esta prueba.

Desempeño de la Prueba:

La prueba demostró que los requerimientos 4.1 y 4.2 respecto al Catálogo de Crucero fueron cubiertos satisfactoriamente.

4.2 CASO NÚMERO 2

Nombre: *Altas, Cambios y Consultas del subsistema para la captura del archivo de Nutrientes.*

Objetivo: Verificar el correcto funcionamiento de los procedimientos correspondientes (así como de cada una de las ventanas para dicho fin, Silicatos, Nitratos y Fosfatos) a la captura de datos para el archivo de Nutrientes. Se checará la correcta visualización de todos los datos de nutrientes en la estructura de la interfaz gráfica, así como cada una de las funciones diseñadas para este programa, es decir, altas, cambios y consultas de datos de nutrientes para una determinada estación hidrográfica.

Se verificará de igual modo los cálculos de concentración de Nutriente que se calcule y se muestre dentro de la interfaz.

---

[1] Ver 3.3.3 (1)
[2] Ver 3.3.3 (6)
[3] Ver 3.3.3 (4)
[4] Ver 3.3.3 (5)
[5] Ver figura D-1 en Apéndice D
[6] Ver formato de archivo físico JS085D002.CAT en la figura E-1 del apéndice E. Su estructura se puede consultar en 2.1.1.3.3.1

Descripción de la Prueba:

a) *Altas:* Partiendo de la ventana principal CTDSIS, se activa el menú *ALTAS*[1] opción *LABORATORIO*[2]. Elegir botón *NITRATOS*[3].

b) *Cambios:* Activar en CTDSIS el menú *CAMBIOS* para escoger *LABORATORIO. Presionar SILICATO*[4].

c) *Consultas:* Elige de menú *CONSULTAS* la opción *LABORATORIO.* Elige *FOSFATO*[5]

Criterios de Aceptación:

El archivo de datos de Nutrientes se deberá mostrar en forma correcta y sin error alguno dentro de la sección de datos de cada una de las ventanas de nutrientes (Silicatos, Fosfatos y Nitratos).

El área de datos de cabecera deberá mostrar correctamente los datos leídos del archivo de Catálogo.

Los datos de concentración de Nutriente deberán ser los correctos.

Se deberá comprobar cualquier tipo de acción de los datos de Nutrientes dentro del archivo físico JS085d002.NUT

Procedimientos de la Prueba:

a) ***Altas y Modificaciones:*** Elegir estación '9' con objeto tipo *Scale*[6]. Presiona *[Aceptar]*[7] *y [Edita Nitratos]*[8]. Inserta datos de Absorbancias, error y factor de scala [9]. Oprime [*Acepta Datos para Nivel*]. Selecciona [*Salvar*].

Se repitieron esta tarea para tres niveles mas de estaciones.

d)***Consultas***: Elige número de estación '8' en objeto *Scale*. Oprime [*Aceptar*]. Consultar cinco estaciones mas.

Resultados de la Prueba:

Se accesó correctamente al archivo del Catálogo para leer datos de cabecera.

Para todos los casos en el que se requirió abrir el archivo JS085D002.NUT, el resultado fue satisfactorio, es decir, se mostraron cada uno de los datos sin error alguno en cada uno de los objetos correspondientes de la interfaz gráfica[10]. Los

---

[1] Ver 3.3.1.2
[2] Ver 3.3.1.2 (3)
[3] Observar 3.3.1.2. (3.a)
[4] Consulta 3.3.1.4. (3.b)
[5] Consultar en 3.3.1.5 (3.c)
[6] Ver 3.3.5.1.1 (3)
[7] Ver 3.3.5.1.1.(1)
[8] Ver 3.3.5.1.1 (2)
[9] En objetos que se muestran en 3.3.5.2
[10] Ver figura D-2 en Apéndice D

procedimientos que habilitan y desabilitan algunos objetos de la ventana según el caso también funcionó adecuadamente.

Por última, se mostró que el archivo físico JS085D002.NUT[1] si sufrió las modificaciones echas por esta prueba.

Desempeño de la Prueba:

La prueba demostró que los requerimientos 5.1 y 5.2 respecto a la captura del archivo de datos de Nutrientes fueron cubiertos satisfactoriamente.

4.3 CASO NÚMERO 3

Nombre: *Altas, Bajas, Cambios y Consultas del subsistema para la captura del archivo de Transectos.*

Objetivo: Verificar el correcto funcionamiento de los procedimientos correspondientes (así como de la interfaz) de la captura de datos para el archivo de Transectos. Se checará la correcta visualización de todos los datos de transectos en la estructura de la interfaz gráfica, así como cada una de las funciones diseñadas para este programa, es decir, altas, bajas, cambios y consultas de transectos.

Descripción de la Prueba:

a) *Altas:* Partiendo de la ventana principal CTDSIS, se activa el menú *ALTAS* opción *Transectos.*
b) *Bajas:* Activa menú *BAJAS* de CTDSIS, eligiendo opción *Transectos..*
c) *Cambios:* Activar en CTDSIS el menú *CAMBIOS* para escoger *Transectos.*
d) *Consultas:* Elige de menú *CONSULTAS* la opción *Transectos.*

Criterios de Aceptación:

Dentro de la estructura visual de la ventana de transectos, se deberá presentar correctamente la información correspondiente a los datos del archivo del mismo tipo. Se efectuarán operaciones de añadir, eliminar, modificar y consultar diferentes transectos correspondientes al crucero oceanográfico de 1985 en su segunda versión. Se deberán comprobar los procedimientos de validación de datos de transectos.
Se comprobará cualquier tipo de acción de los datos de transectos dentro del archivo físico JS085d002.TRA

Procedimientos de la Prueba:

a) ***Altas:*** Se oprime el botón [*Nuevo Transecto*][2]. Se captura del identificador del nuevo transecto. Acto seguido se procedió a capturar separados por una coma o

[1] Observar formato de archivo JS085D002.NUT en Apéndice E, figura E-2. La estructura de este archivo se localiza en 2.1.1.3.3.2

espacio los identificadores de las estaciones que pertenecerán a este transecto. Se aceptan estos datos.

b)*Bajas:* Elige el número de transectos '9' en el objeto *tipo lista de transectos*[1]. Presiona [Borra Transecto][2]. Oprime [*Aceptar*].

c)*Modificaciones:* Elige el transecto con identificador '5' del objeto tipo lista de transectos.. Oprime [*Edita Estación*][3]. Se modificaron los identificadores de estaciones que ahí se encontraron. Se seleccionó [*Atributos...*][4]. Se modifican los datos de minutos y segundos[5].Selecciona [*Aceptar*]

d)*Consultas*: Elige número de transectos '00'. Verificar los datos de transectos.
Elige número de transectos '01'. Verificar los datos de transectos.
Elige número de transectos '08'. Verificar los datos de transectos.

Resultados de la Prueba:

La ventana para la captura del archivo de Transectos[6] mostró correctamente todos y cada uno de los datos que se leyeron de este. Los procedimientos para la validación de los identificadores de transectos repetidos funcionaron satisfactoriamente, así como los procesos que validan los números de estaciones en la archivo de Catálogo de Crucero.

Por último, se mostró que el archivo físico JS085D002.TRA[7] si sufrió las modificaciones echas por esta prueba.

Desempeño de la Prueba:

La prueba demostró que los requerimientos 6.1 y 6.2 respecto al archivo de transectos fueron cubiertos satisfactoriamente.

4.4 CASO NUMERO 4

Nombre:

Procedimientos para abrir y salvar el Archivo de Control.

Objetivo:

El Verificar la correcta lectura y escritura del Archivo de control desde y hacia las estructuras de datos (listas ligadas).

---

[2] Ver 3.3.4.1(1)
[1] Ver 3.3.4.1 (5)
[3] Ver 3.3.4.1 (2)
[4] Observar 3.3.4.2 (1)
[5] Ver 3.3.4.3 (2)
[6] En los objetos tipo teto 3.3.4.3 (2)
[7] Se puede observar la estructura de la ventana de transectos en el apéndice D figura D-3
Ver formato de archivo físico JS085D002.TRA en la figura E-3 del apéndice E. Su estructura se puede consultar en 2.1.1.3.3.1

Así como checar que todas las estructuras se datos se incorporen correctamente a cada uno de los objetos de las ventanas de calibración.

Descripción de la Prueba:

a) ***Abrir:*** De la pantalla principal CTDSIS. Del menú *[ARCHIVO]*[1] .Elegir *[ABRIR...]*[2].

b) ***Salvar:*** De la pantalla principal de CTDSIS. Del menú *[ARCHIVO]* elegir opción *[GUARDAR]*[3].

Criterios de Aceptación:

Al momento de abrir el archivo de control JS085D002.MAD se revisarán cada una de las ventanas para la calibración de datos, y la información ahí localizada deberá coincidir con la encontrada en el archivo ya mencionado.

Para la opción de guardar, los datos que se graben en el archivo de control deberán ser los correctos de acuerdo a las modificaciones que hayan sufrido los contenidos de las ventanas de calibración.(Calibrarción de conductividad en su primera, segunda y tercera etapa, Ordenamiento Mediante series de presión, y Calibrarción de Oxígeno en su primera segunda y tercera etapa).

Procedimientos de la Prueba:

Elegir del objeto *FileSelectionBox* el archivo JS085d002.MAD. Activar del menú *[CALIDAD]*[4]. Elegir cada una de las opciones que ahí se encuentran:

a)[Conductividad][5].- Seleccionar cada una de las tres opciones que posee esta opción.

b)[Presion][6]

c)[Oxigeno][7].- Seleccionar cada una de las tres opciones que posee esta opción.

Para todos los ventanas modificar de forma aleatoria todos y cada una de las secciones de cada ventana, observando que cambios fueron los que se realizaron. Al elegir [GUARDAR] del menú principal de CTDSIS se deberá verificar que el archivo de control (JS085D002.MAD) contenga en forma correcta los datos modificados

Resultados de la Prueba:

Las ventanas diseñadas para la calibración[8] si mostraron (en primera instancia) en forma correcta cada uno de los datos existentes en el archivo de control.

---

[1] Ver 3.3.1.1

[2] 3.3.1.1 (2)

[3] Observar en 3.3.1.1 (3)

[4] Checar 3.3.1.6

[5] Ver 3.3.1.6 (1)

[6] Observar 3.3.1.6 (2)

[7] Ver 3.3.1.6 (3)

[8] Las ventanas para la calibración de datos pueden consultarse en el Apéndice D.2

También se verificó el correcto funcionamiento de cada uno de los objetos que forman a este tipo de ventanas. Las secciones para mostrar el contenido de las listas ligadas también funcionaron de la forma deseada.
Al final cuando se ejecuto *[GUARDAR]*. Se verificó que el archivo de control (JS085D001.MAD[1])sí contiene la información modificada para las acciones realizadas.

Desempeño de la Prueba:

La anterior prueba demostró que los requerimientos 8.1, 8.2, 9.0 y 10.0 descritos en el capítulo 1 fueron satisfechos completamente.

4.5 CASO NÚMERO 5

Nombre: Ventanas y procedimientos para la obtención de los datos de calidad controlada..

Objetivo: Verificar cada una de las ventanas y procedimientos involucrados en la calibración de los datos hidrográficos.
Verificar ventana y procedimientos para la calibración de conductividad.
Verificar ventana y procedimientos para obtener series promediadas de presión.
Verificar ventana y procedimientos para la calibración de Oxígeno.
Realizar un procedimiento de calibración para los datos del crucero oceanográfico del año de 1985 en su segunda versión.

Descripción de la Prueba:

Del menú principal, activar [*CALIDAD*].
a) Presionar [*Conductividad*]
- **i)** [*Primera Etapa*]
- **ii)** [*Segunda Etapa*]
- **iii)** [*Tercera Etapa*]

b) Seleccionar [*Presion*]
c) Presionar [*Oxigeno*]
- i) [*Primera Etapa*]
- ii) [*Segunda Etapa*]
- iii) [*Tercera Etapa*]

Criterios de Aceptación:

Se deberán comprobar que cada ventana de calibración adquiera, al ser invocada, de la correcta serie de datos leídos de las estructuras de datos.
Se checará el correcto funcionamiento de cada una los objetos que componen a cada ventana de calibración.
Se verificará la correcta grabación de los archivos de comandos individuales para cada procedimiento de calibración.

[1] El formato del archivo de control se epcuentra en el Apéndice A.

Se cotejarán los datos calibrados obtenidos por el sistema MPD contra datos calibrados con la metodología anterior. No debe existir diferencia alguna.

Procedimientos de la Prueba:

Se capturará para cada ventana de calibración, la información que se muestra en los archivos de comandos que se encuentran señalados en [3]. Se acepta la información de cada una de estas ventanas.

Se selecciona del menú [*CALIDAD*] la opción [*Procesado*][1]. Se ejecutarán cada uno de los procedimientos. Al terminar los procedimientos seleccionar [*SALVAR*] del menú de datos principal para después seleccionar [*SALVAR*].

Resultados de las Pruebas:

Las estructuras de las ventanas de calibración mostraron un desempeño aceptable al manejar cada uno de los datos leídos del archivo de control.

Las ventanas para la calibración de datos se muestran en el apéndice D.

Los archivos de comandos individuales fueron grabados correctamente y en los directorios indicados.

Los resultados arrojados por los procedimientos de calibración de datos fueron cotejados contra los datos obtenidos por el sistema anterior, y estos no tuvieron ningún tipo de diferencia.

Desempeño de la prueba:

Los requerimientos satisfechos por esta prueba fueron el 7.0, 8.1, 8.2, 9.0 y 10.0

[1] Ver 3.3.1.6 (4)

Recomendaciones y Conclusiones

A pesar de que el sistema desarrollado y documentado dentro de este trabajo de tesis cumplió satisfactoriamente los requerimientos establecidos para su desarrollo, sabemos que durante las etapas de diseño, arquitectura e implementación (y debido a la complejidad del sistema) siempre aparecen modificaciones a estos requerimientos iniciales, e incluso se presentan nuevos requerimientos que se hubieran deseado incluir desde el principio del diseño del sistema. Algunas veces estas modificaciones no representan ningún problema y se pueden ir implementado durante el desarrollo del sistema; pero existen algunas otras que, debido al tiempo asignado para la implementación del sistema, no se pueden llevar a cabo.

Es por lo anterior que formularemos una serie de recomendaciones, que de llevarse a cabo, harán que el sistema MPD sea un programa mas completo y con mayores capacidades.

RECOMENDACIONES

1.- Chequeo de los archivos NBIS.

Se implementará un procedimiento que deberá realizar un chequeo de los archivo de datos individuales código NBIS, que en ese momento se encuentren en el disco de la computadora, reconocerá la estación hidrográfica a la que hace referencia y cotejara su existencia dentro del archivo de Catálogo de Crucero. En caso de existir alguna anomalía, se desplegará una caja de aviso indicando que alguno de los archivos código NBIS no esta referenciado en el Catálogo, impidiendo continuar con el proceso de calibración.

2.- Actualización automática de los archivos de Nutrientes y de Transectos.

Si se realiza alguna modificación al archivo del Catálogo del Crucero, es decir, si se realizan altas o bajas de estaciones hidrográficas, los archivos de nutrientes y de transectos se deberán actualizar automáticamente de acuerdo y sólo conforme a las estaciones que existan en el Catálogo.

3.- Verificación de la información dentro de las ventanas para la calibración de datos.

Debido a que los datos que componen a cada una de las ventanas para la calibración son bastante especializados, no se pudo llevar a cabo una validación de estos. Con el apropiado estudio y análisis de estos procedimientos se podrían implementar rutinas de validación para estas ventanas de datos.[1]

4.- Transcripción de datos.

Permitirá que, desde de la interfaz, se ejecuten los procedimientos para la transcripción de datos, es decir, traducirlos de código NBIS a código VAX.

[1] Todas las ventanas para la captura y manejo de archivos de catálogo, de nutrientes y de transectos SÍ cuentan con un número suficiente de rutinas de validación.

ZNAM - GEO - JJE

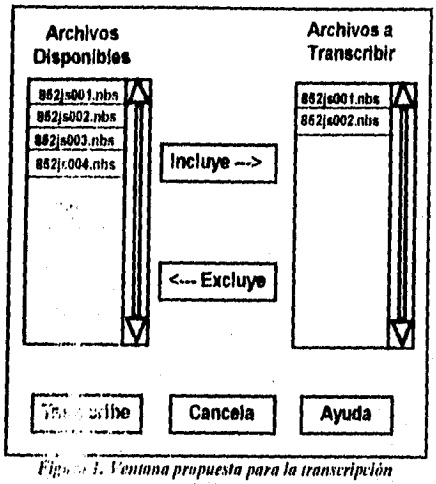


Figura 3. Ventana propuesta para la transcripción de datos

Se propone la siguiente estructura de interfaz para realizar la transcripción de datos NBS a VAX.

Archivos Disponibles.- Objeto tipo scrolledlist que muestra cada uno de los archivos NBS disponibles en el disco de la VaxStation.
Incluye.- Botón que incluye el identificador de algunas estación disponible en la lista de archivos a transcribir.
Excluye.- Botón que borra de la lista de archivos a transcribir al fichero elegido.
Archivos a Transcribir.- Objeto tipo lista que almacena los archivos seleccionados para transcripción.
Transcribe.- Botón que dispara los procedimientos para transcribir archivos elegidos.

5.- Sistema de Ayudas.

Este es un requerimiento que no se consideró indispensable al principio del desarrollo del sistema, por lo que se incluyo dentro de los requerimientos iniciales, pero el elaborar un sistema de ayudas incrementaría la productividad de MPD.

CONCLUSIONES

Uno de los objetivos mas importantes logrados al final del desarrollo de este trabajo de tesis fué, el lleva a cabo el análisis, diseño e implementación de un sistema de información útil, funcional, sólido y confiable.

Algunas de las características que MPD posee al término de su desarrollo son:

a) Portabilidad Se refiere a la facilidad del sistema MPD para correr en un ambiente diferente a aquel en que fue creado. Un software que es portable corre en hardware diferentes y bajo sistemas operativos diversos.

b) Seguimiento de la tendencia Tecnológica Se trató de seguir una tendencia tecnológica para incrementar la longevidad y la calidad de MPD.

Se procuró que MPD fuera un:

- *Sistema abierto.* Un sistema que cumple con ciertos estándares para permitir su fácil adaptación de software o hardware de otras marcas. Se construyó el sistema MPD basado en estándares de sistemas abiertos para lograr proporcionar al usuario todas las ventajas que poseen este tipo de sistemas.
- *Sistema conectado en red.* Una red es un conjunto de computadoras conectadas entre si o con otros dispositivos (tales como impresoras, unidades de almacenamiento etc.) a través de un medio de comunicación de tal forma que es posible transferir, recibir y compartir información. Se creó un sistema que trabaja en red, para extraer las ventajas que esta proporciona. Como por ejemplo, el compartir atributos, objetos y hasta procedimientos que se encuentren en otra computadora.

c) Versatilidad Se refiere a que nuestro puede ser modificado y mantenido de una manera fácil. Se buscó esta versatilidad para que MPD se adapte rápidamente a diferentes usuarios.

d) Transparencia MPD tiene la capacidad de manejar sus procedimientos de tal forma que el usuario inexperto no se entere de los complicados procesos que en realidad se estén llevando a cabo.

e) Consideración del factor humano para la realización de la Interfaz :Los factores humanos fueron estudiados para colaborar a que la interfaz fuera amable al usuario, fácil de usar y de aprender. Se construyó tomando en cuenta este factor y se logró sistema estándar que cumple con todos los requisitos establecidos de una buena interfaz.

Otro objetivo alcanzado con la finalización del sistema fue el haber proporcionado al Grupo de Estudios Oceanográficos una herramienta útil y confiable que se podrá emplear para sustituir a los tediosos métodos que hasta ahora se venían utilizando para la captura de archivos y calibración de datos. Además el MPD permite a los usuarios conceptualizar de una forma mas compacta al sistema de calibración, ya que, en una sola herramienta, se tienen disponibles todas y cada una de operaciones que la calibración involucra, desde la captura del Archivo del Catálogo del Crucero hasta la obtención del archivo de Índices de datos ya calibrados.

Con respecto a la captura y manejo de archivos de datos(de catálogo, de nutrientes y de transectos), este sistema será de gran utilidad, ya que nunca se necesitará otra vez de un editor de texto para modificar o actualizar estos archivos, además, las funciones incluidas de validación permiten capturar solo el tipo de datos adecuados para cada archivo.

En general, el manejo de la interfaz gráfica no representa demasiadas complicaciones para el manejo e interpretación de cada función que posea, pero para el caso de las ventanas de calibración si es necesario de un conocimiento especializado de cada dato que estas contienen, por lo que es indispensable que una persona con experiencia en este tipo de procesos sea el encargado de manejar a este tipo de ventanas.

Los elementos gráficos y de texto presentes en el sistema, dan al usuario (GEO) una idea clara y precisa de la información que se desea representar. Los usuarios contemplan la información que será utilizada para realizar el manejo de archivos de datos y llevar a cabo los procedimientos de calibración de una forma sólida y completa. Un objetivo importante (como ya se mencionó) es el haber integrado de un solo golpe toda la información que antes el usuario contemplaba en forma dispersa.

En trabajos anteriores dicha información se presentaba en diferentes escenarios. Este trabajo la presenta en uno solo y con mayor significado.

Con el desarrollo de la interfaz gráfica (IGU) se permitió tener con los usuarios una interacción de manera mas expedita y confiable. La construcción de la interfaz, que se realizó con la herramienta de apoyo en el desarrollo de IGU X-Window, permitió que el usuario pudiera realizar de una forma mas sencilla el manejo de la WorkStation, aprender a utilizar el programa en menos tiempo y realizar sus tareas de una forma mas rápida y con mayor confiabilidad.

El desarrollo del sistema de información usando X-Window, permite además, que el usuario pueda trabajar en diferentes tipos de terminales gráficas. Lo anterior es posible, ya que la mayoría de las instrucciones dadas al sistema de información son a traves del "ratón" y usando únicamente las teclas alfanuméricas. El uso extendido de X-Window ha hecho que diferentes compañías desarrollen computadoras que permiten el uso de este software, logrando con ello que nuestro sistema (MPD) pueda ser transportado a otras plataformas. Por otra parte, el software de emulación de terminales tipo X para pc's, permite que este tipo de trabajos pueda ser activado en dichos equipos.

Dentro de las experiencias adquiridas en la realización de este trabajo de tesis se encuentra el haber desarrollado un sistema cuyo uso será real, es decir, un sistema que estará funcionando como parte de las herramientas cotidianas del Grupo de Estudios Oceanográficos.

Otra grata experiencia fue el haber implementado un sistema apoyado en una tecnología relativamente nueva, (una en la que al menos su servidor no tenía ni idea de que existía), por lo que la satisfacción, así como la experiencia adquirida fueron enormes.

Lo anterior nos deja la enseñanza de que toda persona que desarrolle aplicaciones de software debe mantenerse informada de todos los avances que diariamente se presentan en esta área, ya que de lo contrario se corre el riesgo de quedar obsoleto y de no ser competitivo en un país en donde lo mas importante en este momento es precisamente la búsqueda de nueva tecnología para desarrollar mas fácilmente los actividades cotidianas.

Glosario

Abreviaciones

CFE	Comisión Federal de Electricidad
CNSNS	Comisión Nacional de Seguridad Nacional y Salvaguardas
CONACYT	Consejo Nacional de Ciencia y Tecnología
CTDSIS	Conductivity Temperature and Depht System
GEO	Grupo de Estudios Oceanográficos
GUI	Graphical User Interface
IIE	Instituto de Investigaciones Eléctricas
MGD	Módulo Generador de Datos
MIT	Massachusetts Institute of Technology
MPD	Módulo de Procesamiento de Datos
MRG	Módulo de representación Gráfica
NBIS	Neil Brown Instruments System
UIL	User Interface Languaje
WHOI	Woods Hole Oceanographics Institute

Términos Oceanográficos

Alcalinidad. Exceso de iones oxidrilo (OH-) en una solución acuosa; es lo opuesto a la acidez.

Archivo de Comandos. Archivo que almacena los datos necesarios para la ejecución de los programas escritos en lenguaje FORTRAN, los cuales son los encargados de realizar la calibración de los datos hidrograficos

Archivo de Control. Archivos donde residirán todos los datos correspondientes a entradas, salidas, cambios, consultas y procesamientos de una campaña hidrografica especifica. Este archivo es leído y cargado a la interfaz cada vez que se abra un archivo con datos de alguna campaña en particular. A su vez, se guardaran cada uno de los cambios que haya sufrido la interfaz en este archivo

Archivo de Nutrientes. Archivo dividido por estaciones hidrografica que contiene los datos necesarios para el calculo de la concentración de nutriente. En este archivo de encuentran los datos de Silicatos. Fosfatos y Nitratos

Archivo de Transectos. Archivo que contiene cada uno de los identificadores de transectos que posee alguna campaña oceanográfica. A su vez, se muestran también los identificadores de cada estación hidrografica para cada transecto

Atlas Oceanográfico. Documento dirigido a la comunidad científica abocada a estudios marinos y al sector productivo, donde se incluye un compendio de las Investigaciones del GEO en el Golfo de México. Este documento incluye información de la naturaleza física, química, biológica y geológica de los cuerpos de agua contiguos a las estaciones hidrograficas sondeadas.

Abreviaciones

CFE	Comisión Federal de Electricidad
CNSNS	Comisión Nacional de Seguridad Nacional y Salvaguardas
CONACYT	Consejo Nacional de Ciencia y Tecnología
CTDSIS	Conductivity Temperature and Depht System
GEO	Grupo de Estudios Oceanográficos
GUI	Graphical User Interface
IIE	Instituto de Investigaciones Eléctricas
MGD	Módulo Generador de Datos
MIT	Massachusetts Institute of Technology
MPD	Módulo de Procesamiento de Datos
MRG	Módulo de representación Gráfica
NBIS	Neil Brown Instruments System
UIL	User Interface Languaje
WHOI	Woods Hole Oceanographics Institute

Términos Oceanográficos

Alcalinidad. Exceso de iones oxidrilo (OH-) en una solución acuosa; es lo opuesto a la acidez.

Archivo de Comandos. Archivo que almacena los datos necesarios para la ejecución de los programas escritos en lenguaje FORTRAN, los cuales son los encargados de realizar la calibración de los datos hidrograficos

Archivo de Control. Archivos donde residirán todos los datos correspondientes a entradas, salidas, cambios, consultas y procesamientos de una campaña hidrografica especifica. Este archivo es leído y cargado a la interfaz cada vez que se abra un archivo con datos de alguna campaña en particular. A su vez, se guardaran cada uno de los cambios que haya sufrido la interfaz en este archivo

Archivo de Nutrientes. Archivo dividido por estaciones hidrografica que contiene los datos necesarios para el calculo de la concentración de nutriente. En este archivo de encuentran los datos de Silicatos. Fosfatos y Nitratos

Archivo de Transectos. Archivo que contiene cada uno de los identificadores de transectos que posee alguna campaña oceanográfica. A su vez, se muestran también los identificadores de cada estación hidrografica para cada transecto

Atlas Oceanográfico. Documento dirigido a la comunidad científica abocada a estudios marinos y al sector productivo, donde se incluye un compendio de las Investigaciones del GEO en el Golfo de México. Este documento incluye información de la naturaleza física, química, biológica y geológica de los cuerpos de agua contiguos a las estaciones hidrograficas sondeadas.

Bitácora de abordo. Planillas con Datos de absorbancias y otros parámetros necesarios para determinar concentraciones de nutrientes.

Campaña Oceanográfica. Crucero que tiene la finalidad de llevar a cabo mediciones oceanográficas regionales de sitios costeros. Cada campana oceanográfica se compone por un numero determinado de transectos y estaciones hidrograficas.

Catalogo del Crucero. Archivo que alberga datos caracteristicas de las estaciones hidrograficas, estos pueden ser: posición geográfica de la estación, profundidades, presiones, temperaturas, corriente del sensor de oxigeno, alcalinidades, etc.

Código NBIS. Representa los datos que fueron recolectados por el CTD. Código de grabación de la información de la compañía Neil Brown Instruments System.

Conductividad. Magnitud especifica y característica de cada sustancia, que se denomina también conductancia especifica. conductor es aquel cuerpo que permite el paso de la corriente eléctrica, oponiendo una resistencia relativamente pequeña.

Estación Hidrografica. Posición geográfica dentro del mar territorial del Golfo de México donde se suelta la sonda hidrografica para la recolección de datos.

Fosfato. Nombre de las sales de varios ácidos fosfóricos. Entre los fosfatos naturales el mas importante es el tricalcico.

Hidrografía. Ciencia que estudia el conjunto de las aguas marinas, terrestres y atmosféricas, en todos sus aspectos.

Nitrato. Sal, derivada del ácido nitrico, que contiene el ion monovalente NO3. Los nitratos metálicos son, casi todos, solubles en agua y se producen a partir del ácido nitrico.

Oxigeno. Elemento quimico de simbolo O. Es el primer termino del sexto grupo del sistema periódico y el elemento mas abundante en la corteza terrestre.

Presión. Relación entre la intensidad de la fuerza elemental dF que actúa normalmente a una temperatura elemental dS y el área de dicha superficie.

Salinidad. Proporción de sales contenidas en el agua. La salinidad se expresa en tanto por mil, en peso.

Silicato. Ácidos o esteres del ácido silicico.

Sonda Oceanográfica. Instrumento de detección y medición de datos Fisicos, quimicos y biológicos dentro del agua de mar.

Transecto. Grupo de estaciones hidrograficas unidas por una linea imaginaria y con un parámetro geografico en común.

Términos Computacionales

Aplicación Los componentes de procesamiento que se encargan de realizar operaciones y cálculos sobre la información que realiza el sistema. El usuario no interactua directamente con ésta.

Base de Datos La base de datos concentra toda la información que es producida, mantenida y utilizada por un sistema. Su diseño adecuado es uno de los factores mas importantes en el desarrollo de un sistema de cómputo.

Codificación Es la traducción del diseño detallado de los módulos que componen un programa a un lenguaje de programación especifico.

Contexto Gráfico Los contextos gráficos son recursos de X-Window que permite agrupar atributos útiles para gráficas y textos.

CPU Unidad Central de Procesamiento

Cursor. El cursor es la representación visible del apuntador en la pantalla.

Diagrama de estructuras Representa la subdivisión jerárquica y funcional de un programa. A cada elemento de este diagrama le llamaremos "módulo".

Evento Es toda acción que es llevada a cabo sobre la interfaz gráfica (servidor)

Font Este tipo de recursos tienen la información necesaria para desplegar texto en una ventana.

Interfaz Lo que el usuario ve o siente, debe entender y puede (o debe) hacer para obtener los resultados apropiados de un sistema. Las acciones que el usuario puede tomar, apoyado por su memoria y control sobre la serie de pasos requeridos para obtener los resultados, son influidas por los objetos que se le presenten.

MOTIF Motif es una colección de objetos que un programador o un usuario tiene disponible al desarrollar o utilizar una ventana de interfaz

Petición Son solicitudes de información o eventos que el cliente envía al servidor de interfaz.

Pixmap Un pixmap, como una ventana, es una área rectangular de trabajo donde la aplicación puede dibujar.

Protocolo Un protocolo es una descripción de formatos de mensajes y reglas sobre cuando cada una de las dos partes puede transferir esos mensajes.

Prueba de la Caja Negra (De entrada-salida) Son los que permiten diseñar casos de prueba a partir de la especificación del programa, sin requerir del conocimiento de la forma en que el programa ha sido realizado.

Pseudo código Descripción en lenguaje natural del flujo y tareas de un determinado procedimiento.

Recursos. Caracteristicas de los objetos que componen la interfaz y con las cules se controla su apariencia.

UIL UIL es un lenguaje de especificación para describir el estado inicial de una interfaz de usuario para una aplicación en MOTIF.

Ventana Estos recursos son áreas rectangulares dentro de la pantalla de video de la workstation.

Widget Un widget es un objeto de interfaz que tiene dos perspectivas, la del usuario, que lo visualiza solo como un objeto en la pantalla; y la del programador, que lo representa como un objeto con un conjunto de recursos y de callbacks.

X-Toolkit Los toolkits de X ofrecen bibliotecas de funciones para manipular los widgets que los componen.

X-Window El sistema X-Window es un software de sistema estándar que permite a los programadores desarrollar interfaces gráficas de usuario portables.

XLIB Xlib es una capa de software sobre el protocolo de X. Está implementada como una biblioteca que se emplea programando en lenguaje "C".

Apéndice A

Archivo de Control

Como se mencionó en el requerimiento 9.0 en el capítulo de definición de requerimientos, se utilizó, como parte de la estructura de la interfaz gráfica, lo que llamamos "*Archivo de Control*"; que no es mas que un archivo de texto en donde se almacenan todos los datos que sean leídos o escritos por parte de la interfaz gráfica.

Al principio se encuentra el identificador del crucero oceanográfico representado por este archivo. Los números entre paréntesis indican el total de datos que pertenecen a esta rama del archivo. Los renglones "NODOS" indican el número de nodos que posee la estructura lista de esa rama.

A continuación mostramos el archivo de control para el crucero oceanográfico de 1985 en su segunda versión.

```
[JS085D002]
archivo= js085d002.MAD,1
fecha= Mon Aug 14 13:35:08 1995
[CALIDAD] (7)
        [CALIBRACION_CONDUCTIVIDAD_1A_ETAPA] (10)
                [SET_ISSW_ARRAY_RAMA_5]  (8)
                        NODOS    = 0
                        SWITCH_3 = APAGADO
                        SWITCH_4 = APAGADO
                        SWITCH_6 = APAGADO
                        SWITCH_10 = APAGADO
                        SWITCH_11 = APAGADO
                        SWITCH_12 = APAGADO
                        SWITCH_15 = APAGADO
                        SWITCH_16 = APAGADO
                [DATOS_LIMITE_RAMA_7]  (3)
                        NODOS    = 0
                        VARIABLE = Presion
                        VALOR    = 4000
                        RANGO    = 4000
                [ARCHIVO_ENTRADA_RAMA_22]  (4)
                        NODOS    = 0
                        DATO     = 4000
                        ESTACION = 27
                        VARIABLE = Presion
                        ARCHIVO  = [CTD.js085d002.CAC]js852.mlt
                [INTERCAMBIO_COLUMNA_RAMA_12]  (2)
                        NODOS    = 2
                        VARIABLES1 = Temp_Oxl , Oxigeno ,
                        VARIABLES2 = Estacion , Estacion ,
                [TABLA_DATOS_RAMA_21]  (4)
                        NODOS    = 0
                        VARIABLE = Salinidad
                        VALOR    = 26
                        RANGO    = 12
                        RECHAZA  = YES
                [REORE_POLI_RAMA_20]  (5)
                        NODOS    = 2
                        VARIABLE = Presion , Temperatura ,
                        PRIMER   = 1 , 1 ,
                        ULTIMO   = 20 , 2000 ,
```

```
            ABS2    = 0.0 , 0.0 ,
            ABS3    = 0.0 , 0.0 ,
        [REESCALA_RAMA_11]  (5)
          NODOS   = 2
            VARIABLE  = Presion , Temperatura ,
            A0      = 0.0 , 0.0 ,
            A1      = 0.05 , 0.0005 ,
            A3      = 0.0 , 0.0 ,
            A4      = 0.0 , 0.0 ,
        [ESCRIBE_MLT_RAMA_25] (1)
          NODOS   = 0
            ARCHIVO   = [CTD.js085d002.smcc]js852s.mlt
        [LISTA_TABLA_RAMA_10] (4)
            NODOS   = 0
            UNIDAD  = 6
            PRIMER  = 1
            ULTIMO  = 30
        [SALIDA_RAMA_18]

    [CALIBRACION_CONDUCTIVIDAD_2A_ETAPA] (7)
      NODOS   = 0
        [EDITA_TABLA_RAMA_21] (4)
            NODOS   = 2
            VARIABLES = Estacion , Presion ,
            VALOR   = 53.5 , 600 ,
            RANGO   = 52.5 , 600 ,
            RECHAZA  = YES , YES ,
        [SET_ISSW_ARRAY_RAMA_5] (6)
            NODOS   = 0
            SWITCH_3 = APAGADO
            SWITCH_4 = APAGADO
            SWITCH_8 = APAGADO
            SWITCH_10 = APAGADO
            SWITCH_11 = APAGADO
            SWITCH_12 = ENCENDIDO
            SWITCH_15 = APAGADO
            SWITCH_16 = ENCENDIDO
        [TERMINOS_REGRESION_RAMA_3] (4)
            NODOS   = 2
            COEFF_1  = 0 , 0 ,
            COEFF_2  = 0 , 0 ,
            COEFF_3  = 0 , 1 ,
            COEFF_4  = 0 , 0 ,
        [PROMEDIAR_COLUMNA_RAMA_8] (4)
            NODOS   = 6
            VARIABLE          = Oxigeno , Oxigeno , Oxigeno , Oxigeno , Oxigeno , Oxigeno
            CAMBIA_FACTOR_STNDV  = NO , NO , NO , NO , NO , NO
            FACTOR            = 2.8 , 2.8 , 2.8 , 2.8 , 2.8 , 2.8
        [EDITA_DATA_TABLE_RAMA_29] (1)
            NODOS   = 6
            RECHAZA_FUERA_RANGO  = YES , YES , YES , YES , YES , YES
        [EDITA_COMENTARIOS_RAMA_23] (1)
            NODOS   = 0
            COMENTARIOS = " CTD 1 julio de 1985 ARGOS85-2"
        [HISTOGRAMAS_RAMA_4] (4)
            NODOS   = 3
            VALOR             = 500 , 500 , 1500
            RANGO             = 500 , 500 , 500
```

```
                RECHAZA_FUERA_RANGO    = NO , NO , NO
                DATOS_A_PANTALLA       = NO , YES , YES
        [SALIDA_RAMA_18]

[CALIBRACION_CONDUCTIVIDAD_3A_ETAPA] (3)
  NODOS   = 0
        [SET_ISSW_ARRAY_RAMA_5] (8)
                NODOS   = 0
                SWITCH_3  = APAGADO
                SWITCH_4  = APAGADO
                SWITCH_6  = APAGADO
                SWITCH_10 = APAGADO
                SWITCH_11 = APAGADO
                SWITCH_12 = APAGADO
                SWITCH_15 = APAGADO
                SWITCH_16 = APAGADO
        [INTERCAMBIO_COLUMNA_RAMA_12] (2)
                NODOS   = 3
                VARIABLES1 = OC , OC , O2R
                VARIABLES2 = Salinidad , Salinidad , Estacion
        [FUNCION_DE_CONVERSION_RAMA_00] (2)
                NODOS           = 0
                FUNCION_DE_CONVERSION= ""
                APLICAR         = YES
        [SALIDA_RAMA_18]

[CALIBRACION_PRESION_PROMEDIA] (2)
        [CAMBIA_INTERVALO_PRESION_RAMA_20] (2)
                NODOS           = 0
                EXTRAE_DATOS_DE_SUBIDA  = NO
                INTERVALO_DE_PRESION    = 2.0
        [VALORES_DE_FACTOR_DE_ESCALA_RAMA_18] (2)
                NODOS           = 0
                SESGO           = .011011335
                PENDIENTE       = 0.00009956426
                RETRAZO         = 0.0
                ATR1            = -.0000065
                ATR2            = .000000015
        [SALIDA_RAMA_12]

[CALIBRACION_OXIGENO_1A_ETAPA]  (3)
  NODOS   = 0
        [ARCHIVOS_MUESTRAS_AGUA_RAMA_3]  (1)
                NODOS   = 0
                ARCHIVO  = js852s.dyn
        [ESTACION_RANGO_ELEGIDA_RAMA_6]  (3)
                NODOS   = 0
                MODO    = TODAS
                RANGO_MIN  = 1
                RANGO_MAX  = 19
        [ALGUNAS_ESTACIONES_RAMA_2]  (2)
                NODOS   = 6
                ESTACION = 3 , 6 , 7 , 170 , 171 , 172 ,
        [SALIDA_RAMA_5]  (2)

[CALIBRACION_OXIGENO_2A_ETAPA] (6)
  NODOS   = 2
```

```
            [CAPTURA_FACTOR_OXIGENO_PROC_1] (2)
                    FACTOR_OXIGENO      = 1.2
                    OXIGENO_MINIMO      = 0.1
            [CUANTOS_PARAMETROS_PROC_2] (1)
                    NUMERO_PARAMETROS       = 4 , 4 ,
            [EMPIEZA_DESDE_PARAMETRO_PROC_3] (1)
                    INICIO      = sesgo , sesgo ,
            [DATOS_DE_EDICION_PROC_4] (2)
                    GUARDA_ARCHIVO      = YES , NO ,
                    TIPO_EDICION        = don't_abs_edt_2.8_stndev , do_abs_edt_+/-0.3 ,
            [VALORES_DE_PROCESO_PROC_5] (3)
                    DIGITOS_SIGNIFICA   = 3 , 2 ,
                    NUMERO_ITERACIONES  = 500 , 500 ,
                    VALORES_DE_DEFAULT  = NO , YES ,
            [VALORES_DE_PARAMETRO_PROC_6] (6)
                    SESGO           = 0.0 , 0.0 ,
                    PENDIENTE       = 1.0 , 1.0 ,
                    PCOR            = .00015 , .00015 ,
                    TCOR            = -.036 , -.036 ,
                    WT              = 1 , 1 ,
                    RETRAZO         = 4 , 4 ,
            [SALIDA_PROC_7]

        [CALIBRACION_OXIGENO_3A_ETAPA] (3)
          MODO          = TODAS
            [PARMETROS_DE_AJUSTE_RAMA_0] (6)
                    NODOS           = 0
                    SESGO           = 0.085
                    PENDIENTE       = 2.044
                    PCOR            = 0.2949E-03
                    TCOR            = -0.1677E-01
                    WT          = 0.1000E+01
                    RETRAZO         = 0.4000E+01
            [PROCESAR_RANGO_DE_ESTACIONES_RAMA_6] (2)
                    NODOS           = 0
                    ESTACION_MIN    = 1
                    ESTACION_MAX    = 19
            [PROCESAR_ESTACIONES_SUELTAS_RAMA_2] (1)
                    #_ESTACIONES    = 4
                    ESTACIONES      = 2 , 6 , 7 , 5 ,
            [SALIDA_RAMA_5]
```

Apéndice B

X-Window . Motif

INTRODUCCIÓN

El sistema X-Window es un software de sistema estándar que permite a los programadores desarrollar interfaces gráficas de usuario portables. Una de las mas importantes características del sistema X es su arquitectura independiente de dispositivo. X permite a los programadores desplegar ventanas conteniendo textos y gráficos en cualquier tipo de hardware que soporte al protocolo X sin ninguna modificación, recompilación o recligado de la aplicación . Esta independencia de dispositivo permite a las aplicaciones basadas en X trabajar en ambientes heterogéneos como por ejemplo en mainframes, workstations o en computadoras personales.

1. OBJETIVOS DE X-WINDOW

Durante la realización del proyecto Athena en el MIT (Massachusetts Institute of Technology), se detectó la necesidad de contar con un sistema de desarrollos de interfaces para estaciones de trabajo distribuidas en una red de computadoras que explotara las capacidades gráficas de dichos equipos. Esta necesidad dio origen a un proyecto, en que participaron Robert Scheifler y Jim Gettys (1986). Jim Gettys había creado en la universidad de Stanford el sistema W (por Windows), que proponía el desarrollo de un estándar de interfaz gráfica para estaciones de trabajo en un ambiente distribuido. Tomando este proyecto como base se emprendió el proyecto X en el MIT. Sus principales Objetivos son los siguientes (Jones, 1989):

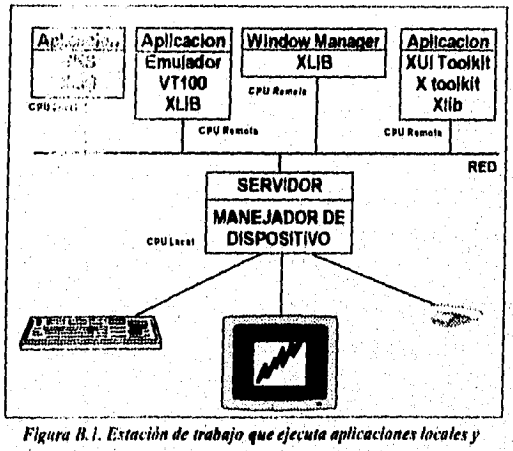


Figura B.1. Estación de trabajo que ejecuta aplicaciones locales y remotas.

- **Dar transparencia a la red.** Ello permite el acceso a varios nodos distribuidos en una red y tener múltiples procesos desplegándose simultáneamente. Además, un proceso puede exhibiese en múltiples ventanas, ya sea en la pantalla conectada al CPU local o a cualquier otra pantalla conectada a otro CPU. Estas facilidades se deben a que X-Window esta integrado en un ambiente de red local.

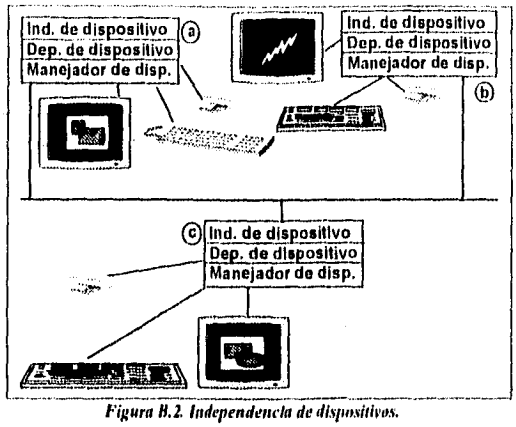


Figura B.2. Independencia de dispositivos.

- **Proveer independencia de dispositivos.** Esto permite que cualquier aplicación basada en X sea portable. Así, solo es necesario compilar y ligar la aplicación para un CPU y en el sistema operativo deseado. La aplicación puede ejecutarse desde cualquier terminal, sin importar las características específicas de marca, el sistema operativo, ni las capacidades gráficas.

- **Proveer capacidades de salidas gráficas.** X-Window organiza las ventanas en una jerarquía que permite traslaparlas. Permite además que una ventana anide otras ventanas y da facilidad para controlar su tamaño, color y visibilidad. Con X-Window pueden obtenerse dibujos y textos de alta calidad y manejar estaciones monocromáticas y de color.

- **Ofrece mecanismos de distribución de entradas.** Las entradas del usuario son discriminadas de manera que se dirigen a la aplicación adecuada, aun cuando se despliegan varias a aplicaciones en la pantalla.

- **Proveer facilidades para que mas de una aplicación este activa en una estación de trabajo.**

Las empresas que apoyaban el proyecto Athena (DEC e IBM) apreciaron el potencial de X-Window y tuvieron la gran idea de donar recursos substanciosos para su desarrollo.

En enero de 1987, un adocena de compañías anunciaron un esfuerzo cooperativo para estandarizar y promover X-Window. En la actualidad, la mayoría de los productores de estaciones de trabajo aceptan al sistema X-Window como una interfaz estándar de sus equipos. Estas compañías componen lo que ahora se conoce como el Consorcio X.(Southerton, 1989).

2. LA ARQUITECTURA DE X-WINDOW

En esta sección se describe la arquitectura del sistema X-Window mediante la definición de los conceptos que la conforman.

2.1 EL MODELO CLIENTE-SERVIDOR

X-Window es un sistema que opera a través de un red, por lo cual se emplea el modelo cliente-servidor. El concepto tradicional parece invertirse: el servidor es ahora un servidor de interfaz y, en ves de que este fuera de nuestra vista, se encuentra justo frente a nosotros, mientras que el cliente (aplicación) de la interfaz esta ejecutándose en alguna parte de la red, probablemente lejos de nuestro alcance. Al servidor de interfaz se le denomina comúnmente servidor de la X. Al conjunto formado por el servidor y el hardware que lo ejecuta se le designa como estación de trabajo. Existen terminales X que pueden realizar esta función; asimismo, existe software disponible para que un a PC actúe como terminal X. (Martínez, 1990a)

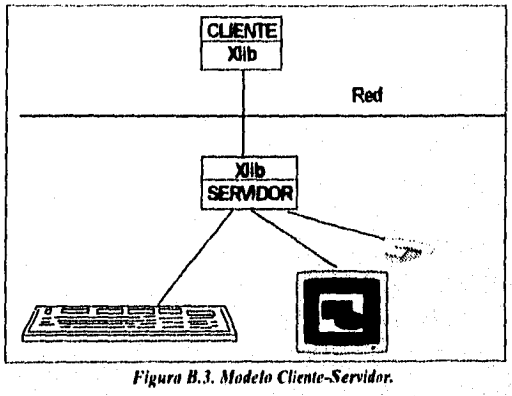


Figura B.3. Modelo Cliente-Servidor.

El hardware de la estación de trabajo consta de un video de despliegue de bitmaps, un ratón, un teclado y una computadora que corre el software del servidor. El servidor sigue las ordenes del cliente, entre ellas la de iniciar una sesión. Los clientes son programas de aplicación que se comunican con el servidor a través de un protocolo de X. Esta comunicación puede llevarse a cabo a través de una red o dentro del mismo equipo. Si desea iniciar una conexión, el cliente debe nombrar al servidor, su dirección en la red, su nombre de nodo, e identificar su terminal o pantalla. Y se sigue este procedimiento aun si ambos, cliente y servidor se encuentran en la misma computadora.

2.2 PROTOCOLOS, CONEXIONES, PETICIONES Y EVENTOS.

La característica distintiva del sistema X-Window es que se basa en un protocolo de red y no en llamadas a procedimientos (lo que sucede en otros ambientes de programación). Un protocolo es una descripción de formatos de mensajes y reglas sobre cuando cada una de las dos partes puede transferir esos mensajes (Nye, 1989).

Los mensajes fundamentales del protocolo del sistema X son las peticiones y los eventos. A continuación describiremos su función y se definen además otros dos tipos de mensajes : la respuesta y el error.

Todos los programas de aplicación interactuan con la estación de trabajo por medio de una conexión en pantalla (display conciten), que es una conexión lógica a través de los circuitos de la red entre la aplicación y la terminal. Todo acceso de la aplicación a la pantalla, ratón, y teclado se hace por medio de la conexión a pantalla (Jones, 1989).

El servidor recibe y responde peticiones (requests) originadas en la aplicación (cliente) a través de la conexión de pantalla. Algunas peticiones requieren que el servidor responda inmediatamente; la respuesta (reply) es entonces otro tipo de mensaje que viaja por la red.

La estación de trabajo también emplea la conexión de pantalla para enviar eventos (e.g.: notificación de que el usuario ha presionado el botón del ratón) a las aplicaciones. Un tipo particular de evento es el error, el cual requiere un manejo distinto por parte de los clientes.

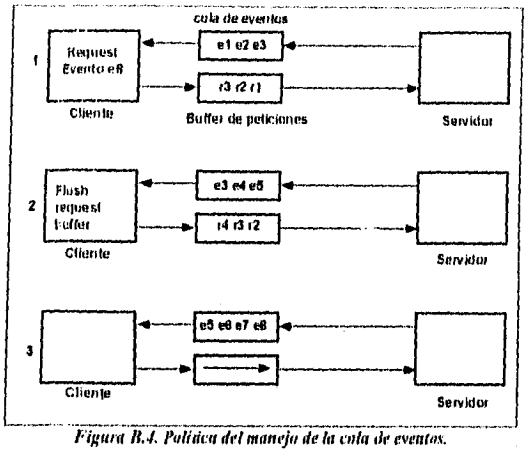
Una vez comenzada la aplicación en el cliente, este entra en un ciclo, en espera de que el servidor le notifique la existencia de algún evento. Los ciclos (loops) son comunes en los programas en C, pero el concepto es llevado hasta sus ultimas consecuencias en X-Window : el ciclo siempre esta esperando a que en el servidor suceda algo que indique al cliente lo que ha de realizarse, hasta que el programa de aplicación termina (al recibir el evento que se lo indique). Este tipo de programación se denomina programación dirigida por eventos (event-programming) (Jones, 1989).

Cuando el cliente recibe la notificación de un evento, ejecuta un bloque de código que el programador designo especificamente para dicho evento. Después de que este código se ejecuta, X-Window devuelve el control al ciclo principal, que nuevamente entra en vigilia, en espera de otra notificación.

2.3 COMUNICACIÓN ASINCRONA

El proceso de comunicación en su conjunto se maneja mediante una comunicación de torrente (stream) entre procesos, lo que permite que el cliente entregue solicitudes al servidor aun sin haber obtenido respuesta a solicitudes anteriores. A esta forma de comunicación se le denomina comunicación asincrona. Para lograrla, los eventos enviados al cliente son colocados en una cola de eventos (event queue), y el cliente normalmente los procesa en orden. Por otro lado, el cliente también cuenta con un buffer de peticiones, en el que se almacenan las peticiones a enviar al servidor siguiendo la politica que se describe enseguida (Nye, 1989).

Conforme el cliente (aplicación) se ejecute, recibe los eventos y los coloca en un cola hasta que ocurra una de las siguientes situaciones :



a) Una aplicación (cliente) llama a una rutina para obtener un evento, pero no has un evento del tipo requerido en la cola de eventos. Puesto que entre las peticiones que se encuentran en el buffer puede estar una que solicite al usuario el tipo de evento que la aplicación espera, tiene sentido vaciar el buffer de peticiones (flush request buffer).

Figura B.4. Política del manejo de la cola de eventos.

b) Una de las peticiones del cliente solicita información al servidor. En este caso, todas la peticiones del buffer se envían antes de que el cliente se dedique a esperar la respuesta del servidor.

c) El cliente también puede desear vaciar el buffer de peticiones manualmente cuando no se esperan eventos del usuario ni se esta solicitando información.

d) También se vacía el buffer de peticiones cuando se llena.

2.4 RECURSOS DE X-WINDOW

X-Window pone a disposición de los usuarios y programadores varios recursos, con los cuales pueden armar las interfaces que deseen y controlar su apariencia. Estos recursos pueden manejarse de diferentes formas, ya sea por programa, con un manejador especial llamado resource manager; hay aplicaciones que permiten especificar algunos recursos como parámetros en el momento de ejecución.

Las aplicaciones X usan y destruyen estos recursos en el curso de su operación. Estos recursos son esenciales en la forma en que la aplicación interactua con la workstation. Estos recursos permiten transmitir informaciones de estado, tales como puede ser la posición y colores de una ventana, solo cuando se creen la ventana para usar estos atributos posteriormente.

Los recursos que ofrece el sistema X-Window son (Sheifler y Gettys, 1989):

- Ventanas
- Contextos Gráficos
- Fonts
- Tablas de colores
- Pixmaps
- Cursores

2.4.1 VENTANAS.

Estos recursos son áreas rectangulares dentro de la pantalla de video de la workstation. Estas presentan al usuario información de salida, y le dan a las aplicaciones una forma racional de manejar el espacio de la pantalla. Cualquier aplicación que genere salida gráfica, debe especificar una ventana es particular para recibir esta salida.

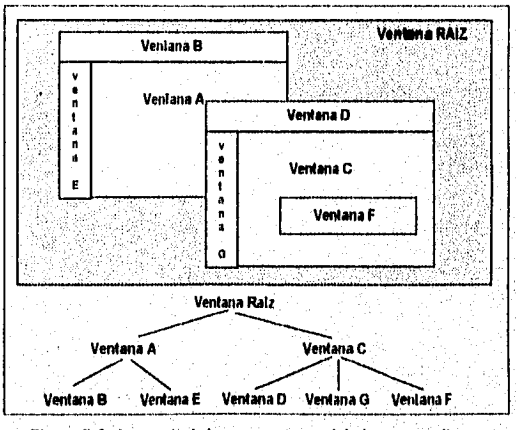


Figura B.5. Jerarquía de las ventanas y su árbol correspondiente.

Las ventanas se pueden traslapar, cambiar de tamaño y contenido, y pueden anidarse dentro de otras ventanas. X provee a la aplicación de muchas formas de controlar la visibilidad, anidamiento , traslapado, tamaño y color de las ventanas. Cada pantalla de una workstation contiene una ventana raíz (root window), la cual completamente la cubre. Este tipo de ventanas son especiales ya que existe solo un por pantalla estas no pueden ser creadas o destruidas, todas las demás ventanas (hijas) se anidan jerárquicamente dentro de esta ventana raíz.

X-Window no restringe el numero de ventanas que puede crear un aplicación. Todas las ventanas tienen una ventana padre (excepto la ventana raiz), y el tamaño de la ventana padre limita el tamaño de las ventanas hijas.

Todas las ventanas tienen atributos que controlan principalmente:

- **Apariencia:** El ancho del borde, el color del fondo, los colores y la forma en que el cursos a aparece en ella. Cada ventana puede tener su propio recurso de cursor asociado, el cual le permite a la aplicación controlar el tamaño, forma y color de su propio cursor.

- **Manejo de eventos:** Cada ventana tiene tres atributos con los cuales puede solicitar y controlar eventos así como su propagación dentro de la jerarquía de ventanas, estos tres atributos son:

◊*event_mask*. Es una mascara de bits, donde cada bit solicita ciertos eventos dentro de ciertas circunstancias. Todos estos bits tiene nombres simbólicos, por ejemplo , existe un bit llamado ButtonPressMask el cual solicita eventos de presionado de botón.

◊*do_not_propagate_mask*. Este es también una mascara de bits, este tipo de eventos evita que se propaguen hacia arriba, sobre la jerarquía, los eventos que no son solicitados por una ventana en particular.

◊*override_redirect*. Este es un valor boleano cuyo valor por default es False. Cuando se coloca su valor a True, se esta previniendo al manejador de ventanas de que se quiere configurar o desplegar una ventana en particular.

- **Manejo de la Imagen:** lo que sucede al contenido de una ventana cuando cambia de tamaño.

- **Refrescado.-** Algunas, pero no todas, workstations con software X son capaces de mantener el contenido de las ventanas cuando estas están total o parcialmente cubiertas por otra ventana. Cuando se tiene esta característica el contenido de la ventana se almacena en un pixmap dentro de una porción de memoria conocida como almacenamiento de respaldo (backing store).

Las ventanas también tienen una serie de características definidas al momento de su creación y que permanecen constantes durante su existencia:

- **Clase:** La clase de una ventana puede ser ya sea de EntradaSalida (InputOutput) o de SoloEntrada (InputOnly). La mayoría de las ventanas son de tipo EntradaSalida, pudiendo tener este tipo de ventanas todos las capacidades de una intrínsecas de una ventana, mientras que las ventanas de SoloEntrada, por otro lado, son invisibles, y no poseen ni colores ni bordes.

- **Profundidad.** La profundidad de una ventana esta definida por el numero de bits por pixel en su parámetro de valor_de_pixeles (Este parámetro es dependiente del tipo de estación de trabajo). Las ventanas del tipo SoloEntrada tienen una profundidad de cero .

Como las ventanas son recursos, residen en el servidor y pueden ser compartidas por otras aplicaciones de la estación de trabajo que conozcan su identificador. El Manejador de ventanas (Window Manager) es una aplicación que usa esta facilidad y se encarga de organizar y reacomodar las ventanas de las aplicaciones que estén en la pantalla; ofreciendo facilidades para moverlas, cambiarles el tamaño y sobreponer unas a otras.

2.4.2 CONTEXTOS GRÁFICOS:

Los contextos gráficos son recursos de X-Window que permite agrupar atributos útiles para gráficas y textos. Los atributos pueden ser de tipo y ancho de línea, patrones de asurado y tipos de texto. los contextos se crean asociados a una ventana y posteriormente pueden ser compartidos (por medio de su identificador) con cualquier ventana. El usuario crea los contextos gráficos que necesite y los utiliza al hacer dibujos o textos; les puede cambiar los valores que le asigno originalmente y puede destruirlos.

2.4.3 FONTS:

Este tipo de recursos tienen la información necesaria para desplegar texto en una ventana. Un font describe el tamaño y forma de una colección de caracteres. Las workstation con software X generalmente poseen varios tipos de font almacenados en librerías las cuales no pueden ser accesadas directamente desde la aplicación ; la workstation carga en memoria los archivos de fonts y los hace disponible a las aplicaciones es respuesta a peticiones de X.

2.4.4 MAPAS DE COLORES.

Un mapa de colores es la definición de los identificadores de colores y sus valores correspondientes en intensidades de rojo, verde y azul (su definición RGB). Existen varias políticas para definir los mapas de colores, según tipo de aplicación y la estación de trabajo esto a su vez depende del numero de planos de bits que tiene la estación). Con este recurso se define que política se va a seguir y si, por ejemplo, se va a utilizar una tabla de colores por definición, si se va a generar una específica de la aplicación o si se va a compartir esta aplicación con otras.

2.4.5 PIXMAPS

Un pixmap, como una ventana, es una área rectangular de trabajo donde la aplicación puede dibujar. Pero a diferencia de las ventanas, un pixmap nunca es visible en la pantalla; los pixeles de un pixmap son almacenados en memoria dentro de la workstation. La mayoría de la aplicaciones hacen visibles a los pixmap copiándolos dentro de una ventana. Los pixmaps tienen tres características:

Ancho y Alto .Este valor siempre es positivo y una vez que el pixmap es creado estos datos ya no pueden ser modificados.
Profundidad. Define el numero de bits por píxel en el pixmap.
Pantalla. En workstation con múltiples pantallas físicas, se debe decidir que pantalla estará asociada con un determinado pixmap.

2.4.6 CURSORES.

El cursor es la representación visible del apuntador en la pantalla. Cada ventana tiene un recurso de cursor particular asociado con esta. Es posible especificar explícitamente un cursor para cada una de las aplicaciones o permitir a cada una de las ventanas tomar el cursor de su ventana padre.

3. PROGRAMACIÓN CON X-WINDOW

Existen diferentes niveles a los que se puede programar la interfaz para una aplicación. Los dos niveles de programación de interfaces son:

- Xlib
- Tollkit

Es posible programar directamente con llamadas al protocolo de X (Nye, 1989); sin embargo, esto sería tan complicado que resultaría imprácticо. Por tanto, se creó sobre el protocolo X una capa de software llamada Xlib, que pone a disposición del programador un conjunto de rutinas para accesar los recursos disponibles de X-Window.

El siguiente nivel de programación son los toolkits, que poseen un conjunto de componentes de interfaz (cajas de diálogo, botones, menús, barras de scroll, etc) y sus respectivas funciones. Se combinan estos componentes para construir la interfaz deseada. La figura B.6 muestra la arquitectura estratificada de X-Window

En las siguientes secciones se verá cada uno de estos niveles y la estructura básica del programa que los emplea.

3.1 XLIB

Xlib es una capa de software sobre el protocolo de X. Está implementada como una biblioteca que se emplea programando en lenguaje "C". El protocolo de X se accesa utilizando las funciones que la componen (Nye, 1987, y Wood, 1989).

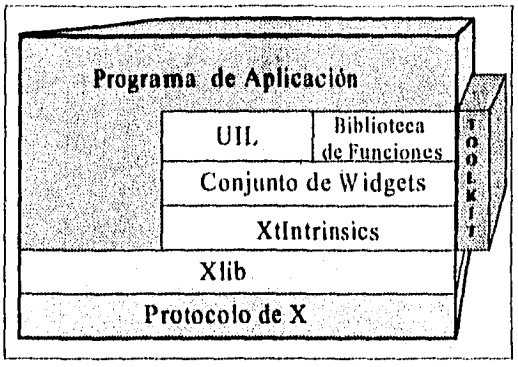


Figura B.6 Arquitectura estratificadora de X Window

Un programa que emplea Xlib sigue el concepto denominado *event-driven-programming* : programación dirigida por eventos. La aplicación se encuentra constantemente esperando eventos del usuario. Cada evento del usuario tiene una semántica definida. Cuando se recibe un evento se ejecutan las acciones que implementan su semántica. La aplicación suele estar construida por un ciclo en que se esperan eventos del usuario. Se encuentra además con un árbol de decisiones que clasifica el evento y decide las acciones a realizar. Los pasos necesarios para programar en Xlib son los siguientes:

1. Conectarse al display. Con esto se crea la conexión a pantalla
2. Crear la ventanas. En esta etapa se definen los atributos de las ventanas que componen la interfaz
3. Hacer visibles las ventanas. Procesar eventos. Se reciben los eventos y, según la ventana de origen y el evento recibido, se decide cuales acciones se realizan.
4. Terminar. Uno de los eventos recibidos implementa la terminación de la aplicación

3.2 INTRINSECAS Y TOOLKITS

Aunque los programadores pueden servirse de Xlib y lo hacen para construir sus aplicaciones, esta biblioteca puede resultar tediosa y difícil de usar (Rosenthal, 1987). Para ello de ha desarrollado una capa de software sobre Xlib denominadas *Intrinsecas* de X (Xt instrinsics). Esta capa se conoce como el primer componente de un toolkit de X; sirve de un ambiente que permite la creación de componentes de interfaz. A dichos componentes se les denomina widgets, y constituyen el segundo componente de un toolkit de X. Un widget es un objeto de interfaz que tiene dos perspectivas, la del usuario, que lo visualiza solo como un objeto en la pantalla; y la del programador, que lo representa como un objeto con un conjunto de

recursos y de callbacks. Los widgets incluyen objetos como menús, barras de scroll, cajas de diálogo y otras partes visibles de la interfaz (Martínez, 1990b, Young, 1990b y c, y León et al.,1991).

Las Intrínsecas sirven de base a la definición de varios conjuntos de widgets. El conjunto de widgets Athena (de dominio público) se entrega con la distribución estándar de X. otros conjuntos de widgets son OFS/Motif de la "Open Software Foundation", XUI de Digital y Open Look de AT&T.

Los toolkits de X ofrecen bibliotecas de funciones para manipular los widgets que los componen. Estas funciones permiten crear widgets, modificar su atributos, hacerlos visibles, asociarles acciones cuando reciben eventos y destruirlos.

Los recursos de un widget le dan al programador el control para dar la apariencia y conducta requerida para cada uno de los objetos de la interfaz. Por ejemplo, todo widget maneja recursos de ancho (width) y largo (height) que se pueden manipular para hacer aparecer al objeto del tamaño deseado.

Los widgets de una interfaz tiene asociadas acciones que ejecutan cuando reciben eventos del usuario. Esta es la manera en que se programa la semántica de la interfaz. Las acciones que se asocian a un widget para ejecutarse al recibir eventos se denominan callbacks. Todos los widgets de Motif poseen callbacks, estos sirven para disparar determinadas acciones en respuesta a eventos del usuario, es decir, si un usuario manipula a un objeto de la interfaz, son los callbacks los encargados de avisar a la aplicación que hubo un cambio en la interfaz. Por ejemplo, un usuario oprime, con la ayuda del ratón, un objeto tipo botón (pushbutton) dentro de una ventana de interfaz, la persona esperará que se realice alguna acción, pero nada sucederá si este widget no tiene asociado un callback que le indique a la aplicación que debe hacer algo, es decir, el programa necesita saber acerca del "click" a nuestro botón para realizar una acción determinada.

Toda aplicación que emplea un toolkit de X se encuentra ejecutando un ciclo (XtMainLoop). Este ciclo puede verse como el administrador de los widgets de la interfaz, y se encarga de despachar los eventos a los widgets que los corresponde; los widgets reaccionan a los eventos activando las funciones de callback correspondientes al evento que reciben. Uno de ellos es normalmente la terminación del programa y corresponde al callback de un widget realizar esta función y así terminar la ejecución del programa y del ciclo de despacho. Este estilo de programación recibe el nombre de programación dirigida por despacho (dispatch-driven-programming).

Las funciones típicas de un programa que emplea un toolkit son las siguientes:

1. Inicializar la Xt intrinsics. Este paso permite realizar la conexión a pantalla.
2. Crear los widgets. Esto se logra mediante las funciones de manejo de widgets propias de cada toolkit.

3. Asociar funciones de callback a los widgets. En este paso se define la manera en que los widgets reaccionan a eventos de usuario
4. Administrar los widgets. En esta etapa se definen las jerarquías de widgets y se decide cuales será visibles de cada jerarquía.
5. Hacer los widgets visibles. En esta etapa se hacen visibles las jerarquías de widgets que inician la comunicación del usuario.
6. Entrar en el ciclo de eventos. Esta es la parte principal del programa de aplicación, dentro de este ciclo se despachan los eventos a los widgets de la interfaz.

Algunos toolkits proporciona interfaces de programación de mas alto nivel; por ejemplo, XUI y Motif, los cuales cuentan con un lenguaje denominado UIL (User Interface Language) que permiten definir las características de los widgets independientemente del código de la interfaz. Esto ofrece la ventaja de que la apariencia de la interfaz puede ser modificada sin necesidad de recompilar el código que la maneja. Los programas que emplean UIL deben realizar las mismas funciones que arriba se describen (DEC, 1989).

4. MOTIF

Motif es una colección de objetos que un programador o un usuario tiene disponible al desarrollar o utilizar una ventana de interfaz ; menús tipo pull_down, menús, cajas de diálogo, barras de scroll, botones, etc. A cada uno de este conjunto de objetos se les denomina para su mejor identificación como *widgets*. Para construir una aplicación en Motif, el programador selecciona un conjunto de widgets para crear una interfaz de usuario, escribiendo el código que hace que estos widgets aparezcan en el momento justo y que además tengan el comportamiento requerido.

4.1 CLASES DE WIDGETS

Las xtintrinsics definen una arquitectura orientada a objetos que organiza los widgets en clases.

En general una clase es un conjunto de objetos que tiene características similares. Una intrinseca de una clase es un objeto individual.

La herencia es otro concepto útil que soporta el xtintrinsics, en la mayoría de los sistemas orientados a objetos, una clase puede heredar alguna o todas las carácteristicas d otra clase.

Las caracteristicas que cada clase hereda de su sepercluse incluye todos los recursos y todas las listas de callbacks.

El Xtintrinsic definen varias clases de widgets básicos que sirven como superclases para otros widgets. Esta clase de widgets definen la arquitectura usada por otros widgets y también proporcionan caracteristicas fundamentales para las demas clases de widgets. La suigiente lista muestra las clases de widgets del Xtintrinsics.

- Core Widget class
- Composite widget calss

- Constraint widget class
- Shell widget class
- Top_level shell
- Transient shell
- Override shell

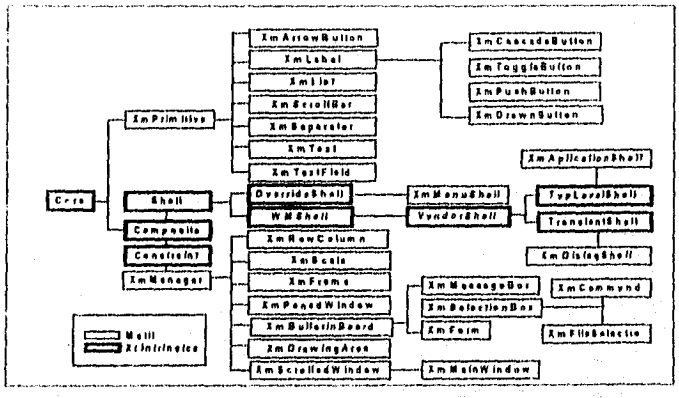


Figura B.7 Jerarquía de los widgets de MOTIF.

El conjunto de widgets de motif posee muchos componentes que pueden combinarse para crear la interfaz d usuario. Se pueden subdividir las clases de widgets en varias categorías de a cuerdo a la funcionalidad que ofrecen:

Widgets de despliege. Los los widgets qu tiene como funcion el despliegue de información. Esto incluye:

XmArrowButton	XmDrawButton	XmLabel	XmList
XmPushButton	XmScrollBar	Xmseparator	XmTextEdit
XmToggleButton			

Widgets Contenedores. Son widgets que pueden ser usados para combinar otros widgets. Permite agrupar botones, etiquetas scrollbars, etc., en una aplicación.

XmDrawArea	XmFrame	XmMainWindiw	XmRowColum
XmScale	XmScrollWindow	XmPanedWindow	XmBulletinBoard
XmForm			

Widgets de diálogo. Gracias a las facilidades del "popup" proporcionadas por las intrinsics, Motif construye un conjunto de widgets versatiles como son:

XmCommand	XmSelectionBox	XmMessageBox	XmFileSelectionBox

4.2 ESTILO DE PROGRAMACIÓN DE MOTIF

4.2.1 CONTROLES

Se presentan tres tipos: Botones, cajas y Valuadores

Tipos de botones.

Las aplicaciones de motif usan tres tipos de botones. pushbutons, radio buttons y toggle buttons. dependiendo del tipo de control que se decea alanzar, es el tipo de boton que debe usarse

Tipos de cajas:

Existen dos tipos de cajas: cajas de listas y cajas de entrada.

Cajas de entrada. Permiten la introduccion de texto. Puede contener un scroll horizontal o vertical, para ver el contenido del texto en caso de que la caja sea pequeña

Valuadores

Prover un estilode control análoga. Los mas comunes son los tipo escala y los tipo scrollbar

Scrollbar. Se usan para buscar rápidamente atraves de los contenidos de una ventana o para elegir un conjunto continuamente variable de valores.

4.2 MENUS

Generalmente constan de un rtítulo y una lista de selecciones de las cuales el usuario puede elegir la acción apropiada. Motif provee de los siguientes ambientes de menús:

Pulldwns. Son aquellos que caen de una posición fija en el menu-bar

Pop-Ups. Menus que aparecen en la posición actual del apuntador cuando algún botón del mouse es presionado

Cascade. Son los menus que caen a la derecha de algun otro menu, proporcionando selecciones mas detalladas relacionadas con la selección original.

Option. Son menus que se despliegas desde un bolon tipo "option" en una ventana.

4.3 CAJAS DE DIALOGO.

Son ventanas que contiene cotroles gráficos que sirven para dialogar con la aplicación. Desde un punto de vista técnico, una caja de diálogo es una ventana que sirve para solicitar y/o desplegar información o instrucciones. Puede incluir combinaciones de texto y gráficas, así como una variedad de controles: botones, etiquetas, etc.

Algunas cajas de diálogo proporcionan un mensaje solicitan la confirmacuón d alguna acción. Este tipo de cajas son llamados cajas de mensajes, y son desplegadas por la aplicación

como resultado de algun evento. Se dividen en cinco tipos dependiendo del tipo de mensaje que despliegan:

Información, progeso, interrogación, advertencia y acción.

Este tipo d cajas de diálogo normalmente incluye un icono relacionado con el tipo de mensaje que genera y tres botones de default: Aceptar, Cancelar y Ayuda.

5. PROGRAMACIÓN EN UIL (User Interface Language)

UIL es un lenguaje de especificación para describir el estado inicial de una interfaz de usuario para una aplicación en MOTIF. La especificación describe los objetos (por ejemplo, menúes, cajas de diálogo, etiquetas, pushbuttons, etc.) usados en una interfaz y especifica las funciones a ser llamadas cuando algún miembro de la interfaz cambia de estado como resultado de alguna interacción con el usuario.

Para crear una interfaz de usuario en UIL, implica desarrollar los siguientes puntos:

1. Generar el Código en UIL, el cual es almacenado en un archivo con extensión "*.UIL*".
2. Compilar el archivo .UIL para general un archivo con extensión UID (User Interface Definition)
3. En el programa de aplicación, se usarán funciones en MOTIF para abrir el archivo extensión UID y accesar a la estructura de la interfaz.

Usando UIL, se puede especificar lo siguiente:

- Objetos (widgets) que comprenden la interfaz.
- Argumentos (atributos) de los widgets que se especifican.
- Funciones "Callback" para cada objeto.
- El árbol de widgets para la aplicación.
- Valores de literales que representan a cada objeto dentro de la aplicación.

5.1 BENEFICIOS DE LA PROGRAMACIÓN EN UIL.

Crear una interfaz para una aplicación en MOTIF utilizando UIL representa las siguientes ventajas:

Codificación mas sencilla.

Se puede especificar rápidamente una interfaz en UIL, ya que no se tiene que conocer a fondo las funciones especificas para la creación de objetos (widgets), o el formato de su lista de argumentos. Solo basta con incluir aquellos argumentos que se desean cambiar. En general, se pueden especificar estos argumentos en cualquier orden.

Detección de errores con anticipación.

Se detectan errores en la interfaz rápidamente ya que se cuenta con un compilador de código UIL, el cual detecta todos los errores que pueda contener el código de interfaz.

Separación de la aplicación y de la interfaz.

El código de la interfaz reside en un archivo con extensión ".UIL". Mientras que la aplicación se encuentra en un archivo en código "C".

Un desarrollo rápido de prototipos.

UIL ayuda a desarrollar prototipos de la interfaz. Se pueden crear una variedad de interfaces en un tiempo relativamente corto, y darse un a idean de la apariencia final de la interfaz antes de codificar la aplicación.

Apéndice C

Referencias a Procedimientos Externos

REFERENCIA 1

Funciones para el manejo de las lista ligadas

Nombre: ***CreateList (Crear Lista)***
Abreviación : CL
Propósito: Reservar memoria para la lista LIST
Parámetros: Ninguno
Regresa: LIST

Nombre: ***InsertRec (Insertar Registro)***
Abreviación : IR
Propósito: Insertar el registro "rec" despues del registro "after_rec" en la lista "list". El parametro "where" espesifica la posición en la lista.
Parámetros:
list - La lista que contiene a "rec"
rec - Dirección del registro a ser insertado
where- Posición en la lista donde "rec" será insertado
FIRST = al inicio de la lista
LAST = al final de la lista
AFTER = despues de "after_rec"
after_rec- dirección del registro despues del cual "rec" será insertado
where = AFTER, de otra manera es igual a NULL
Regresa: void

Nombre: ***DeleteRec (borrar registro)***
Abreviación : DR
Propósito: Remover el dato apuntado por "rec" en la lista "list" y liberar la memoria asociada
Parámetros:
list- La lista que contiene a "rec"
rec - dirección delregistro a ser borrado de la lista
Regresa nada

Nombre: ***ClearList (Borrar Lista)***
Abreviación : CL
Propósito: Borra todos los registros de la lista, pero no elimina la lista
Parámetros: list - La lista a ser limpiada
Regresa nada

Nombre: ***DeleteList (Elimina lista)***
Abreviación : DL
Propósito: Libera la memoria asociada con los registros en "list".
Parámetros: list - La lista a ser eliminada
Regresa nada

Nombre: ***MoveRec (Mover registro)***
Abreviación: MR
Propósito: Mueve el registro "rec" a la locación espesificada en "where"
Parámetros: list - la lista que contiene a "rec"
rec - direccion del registro a ser movida
where- donde se colocara "rec" en la lista
FIRST = al principio de la lista
LAST = al final de la lista
AFTER = despues de "after_rec"
after_rec - dirección del registro despues del cual sera insertado "rec"
"where"=="AFTER" de otra manera es igual a NULL.
Regresa nada

Nombre: ***GetRecByKey (Obten registro mediante llave)***
Abreviación: GR
Propósito: Obtener un registro a travez de su campo llave
Parámetros: list - la lista que contiene a "rec"
key - El campo llave mediante el cual se realiza la busqueda
value- El valor a buscar dentro del campo llave
Regresa La dirección del primer registro donde "value" coinside con el valor de su campo llave.

Nombre: ***NextRec (siguiente registro)***
Abreviación: NR
Propósito: Obtener el siguiente registro de "rec" en "list"
Parámetros: list- la lista que contiene a "rec"
rec- el registro previo al buscado
Regresa la dirección del registro despues de "rec"

Nombre: ***PrevRec (registro anterior)***
Abreviación: PR
Propósito: Obtiene el registro previo a "rec" dentro de "list"
Parámetros: list- la lista que contiene a "rec"
rec - el registro despues al que buscamos
Regresa: el registro previo a "rec"

Nombre: ***FirstRec (Primer registro)***
Abreviación: FR
Propósito: Obtiene la dirección del primer registro en la lista "rec"
Parámetros: list - La lista
Regresa: La direccion del primer registro de la lista "list"

Nombre: ***LastRec (Ultimo registro)***
Abreviación: LR
Propósito: Obtiene la dirección del último registro en "list"
Parámetros: la lista
Regresa: diección del últimno registro en "list"

REFERENCIA 2

Procedimientos que maneja la aparición y desaparición de ventanas.

Nombre : ***VUITManage***
Propósito: Visualiza una ventana de interfaz.
Sintaxis: VUITManage("*NOMBRE*"); donde *NOMBRE* es el nombre de la ventana dentro de la interfaz.

Nombre : ***VUITUnmanage***
Propósito: Esconde una ventana de interfaz.
Sintaxis: VUITUnmanage("*NOMBRE*"); donde *NOMBRE* es el nombre de la ventana dentro de la interfaz.

REFERENCIAS 3

Procedimientos que realizan las tareas de calibración de datos.

Nombre : ***Catmues***
Propósito: Extrae del archivo de Catálogo de Crucero los valores de salinidad y de oxígeno de las muestras que se requieren para la calibración de los datos del CTD.
Programó: M.C. Computacionales José María Pérez Molero (1985)

Nombre: ***MLTRG2D.for***
Propósito: Corrige la conductividad del CTD por la deformación que sufre el sensor con la temperatura y la presión como describe fofonoff y colaboradores en [2].
Obtiene la conductividad de las muestras de agua, debido a que el laboratorio químico solo reporta salinidad.
Ajusta con un polinomio de primer grado la conductividad del CTD a las conductividades de la muestra sde agua usando mínimos cuadrados.

Nombre: ***PRSORT871***
Propósito: Genera una serie de datos uniformes de temperatura, salinidad y oxigeno a intervalos de 2 decibarios de presion. Toma los archivos en formato CTDVAX y genera archvos por estacion oceanográfica en los que la presion ha sido promediada y sorteada.

Nombre: ***CRSUMARY***
Propósito: Genera la documentación de todos los archivos CTD de un crucero oceanográfico. Genera un archivo de indices (AI).

Nombre: ***MLTOXDWN***
Propósito: Extrae del archivo de Catálogo del Crucero y de los archivos CTD código "M" los valores de las observaciones del CTD.

Nombre: ***OXFITEDC***
Propósito: Ajusta los parámetros que genera el anterior programa generando archivos con valores ajustados.

Nombre: ***OXCALCL***
Propósito: Aplica los parámetros obtenidos en oxfited a los perfiles del CTD (código M) generando los archivos CTD código X

Apéndice D

Ventanas de la Interfaz Gráfica

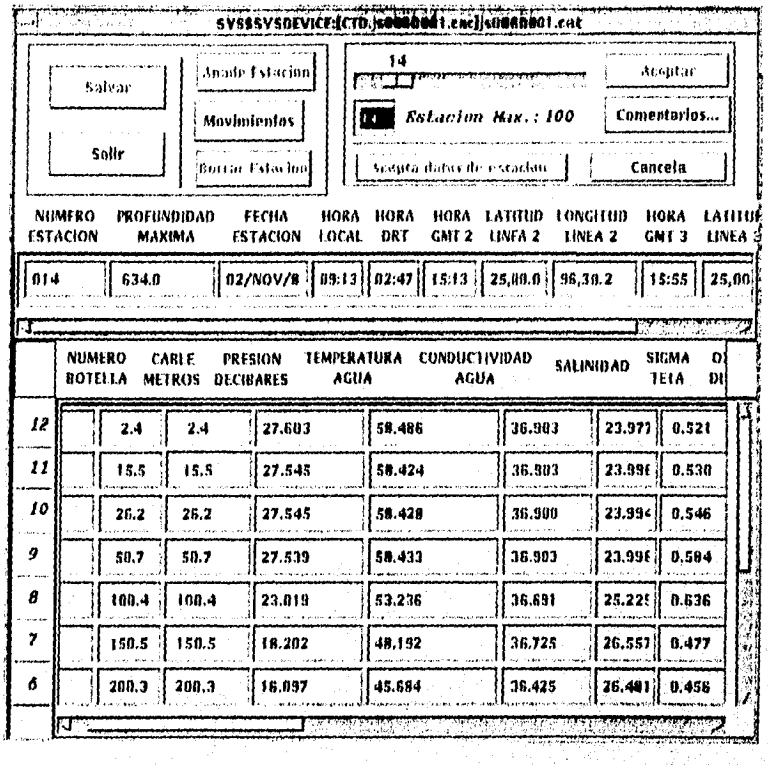


Fig D.1 Ventana del Programa para la captura del archivo de Catálogo del Crucero.

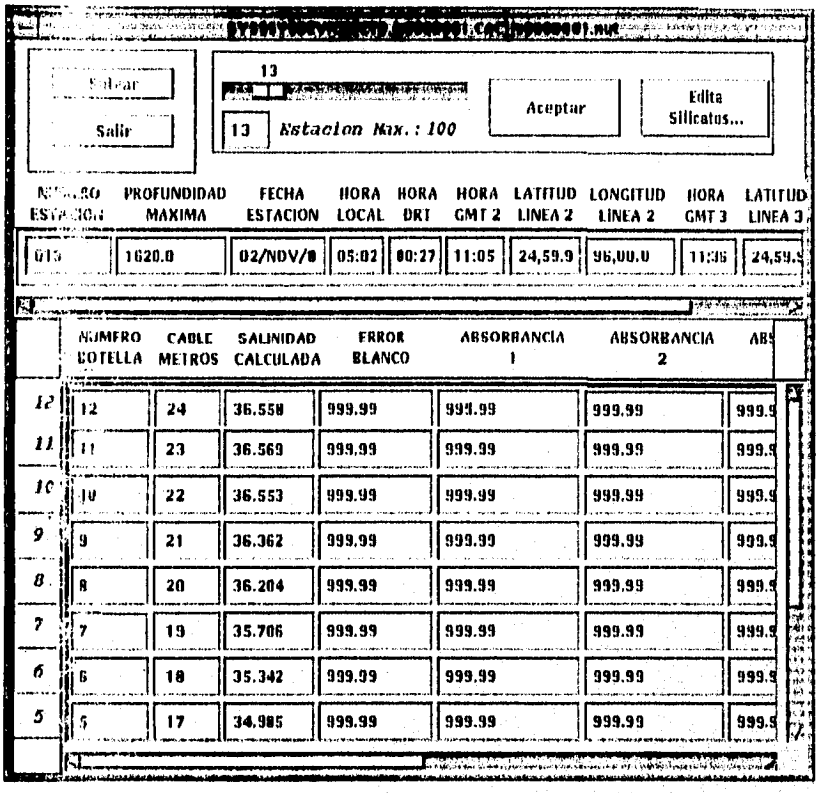


Figura D.2 Ventana para el programa de Captura del Archivo de nutrientes (Ventana Inicial).

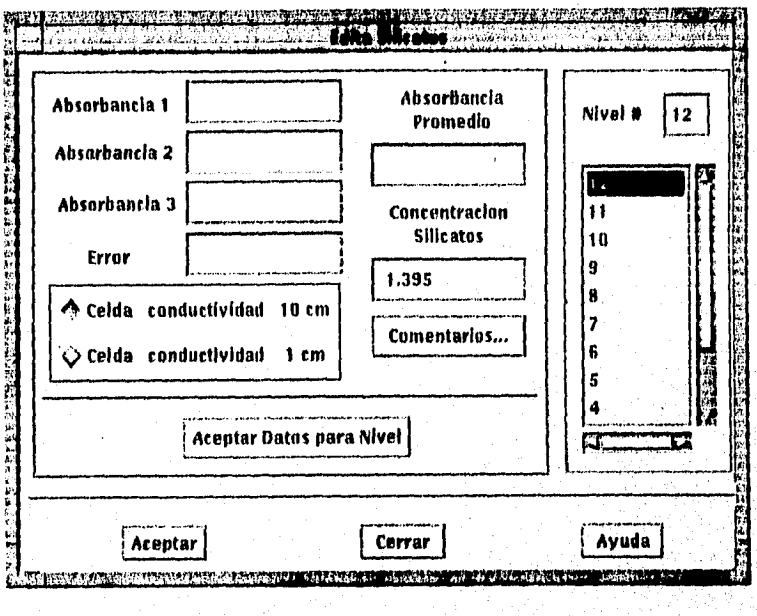


Figura D.3 Ventana Para la captura de datos de Silicatos.

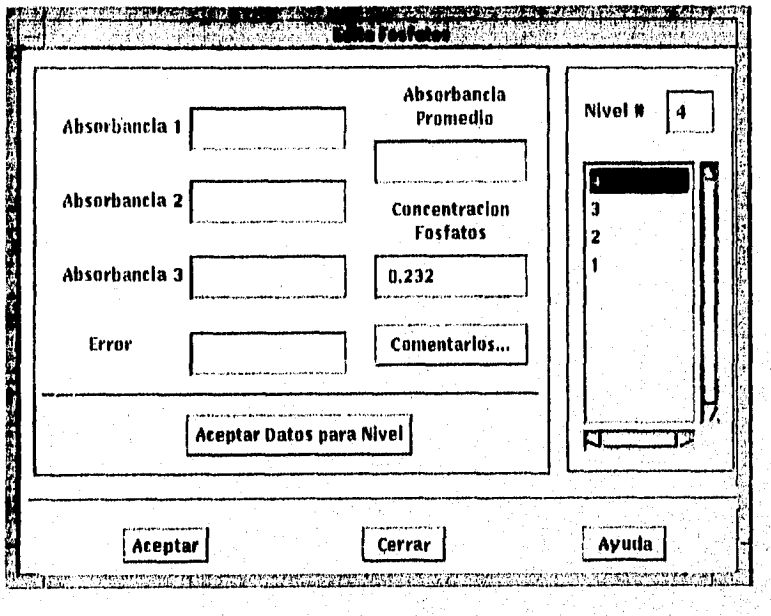


Figura D.4 Ventana para la captura de datos de Fosfatos.

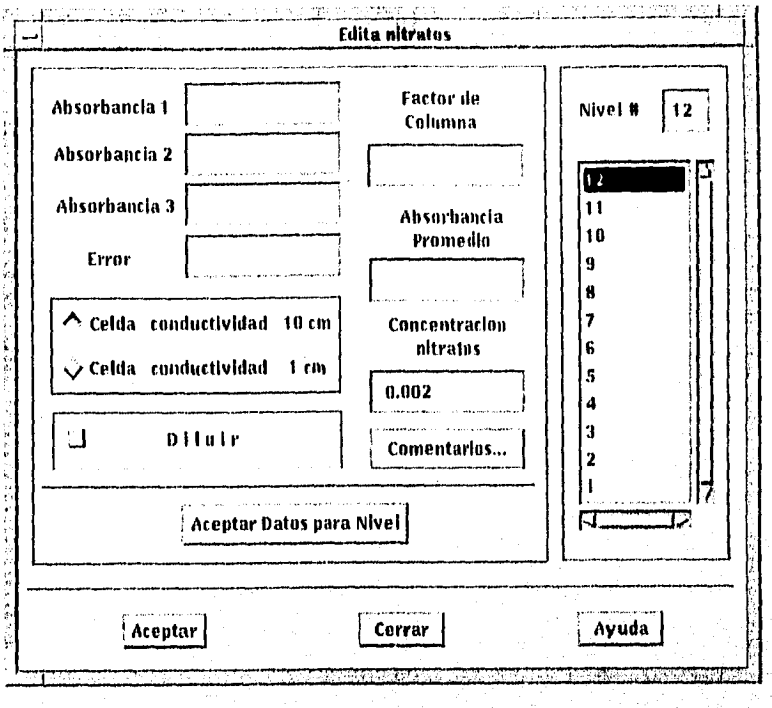
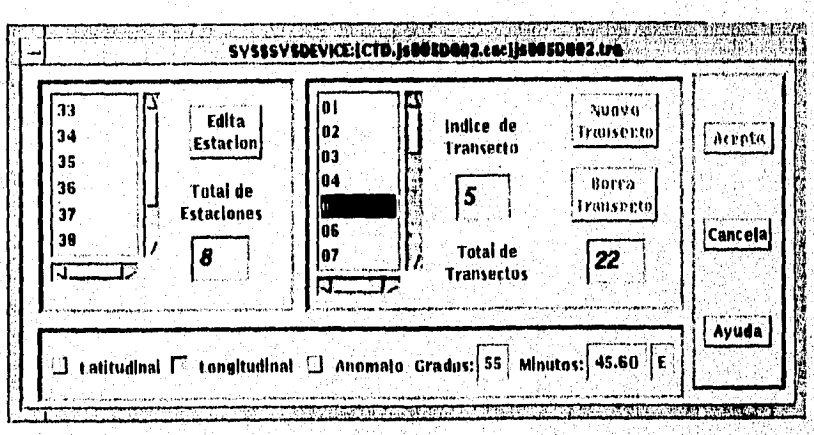


Figura D.5 Ventana para la captura de datos de Nitratos.



Ventana D.6 Ventana para la Captura de Datos de Nutrientes (Ventana Principal)

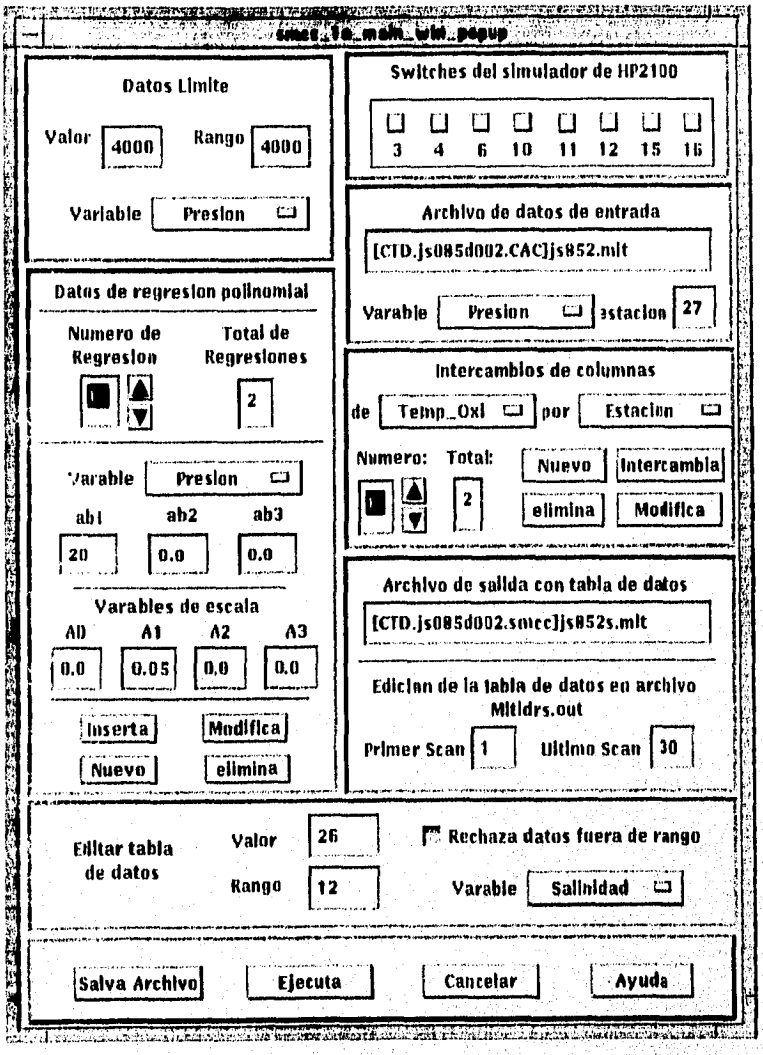


Figura D.7 Ventana para la captura de datos para el procedimiento de calibración de Conductividad en su Primera Etapa.

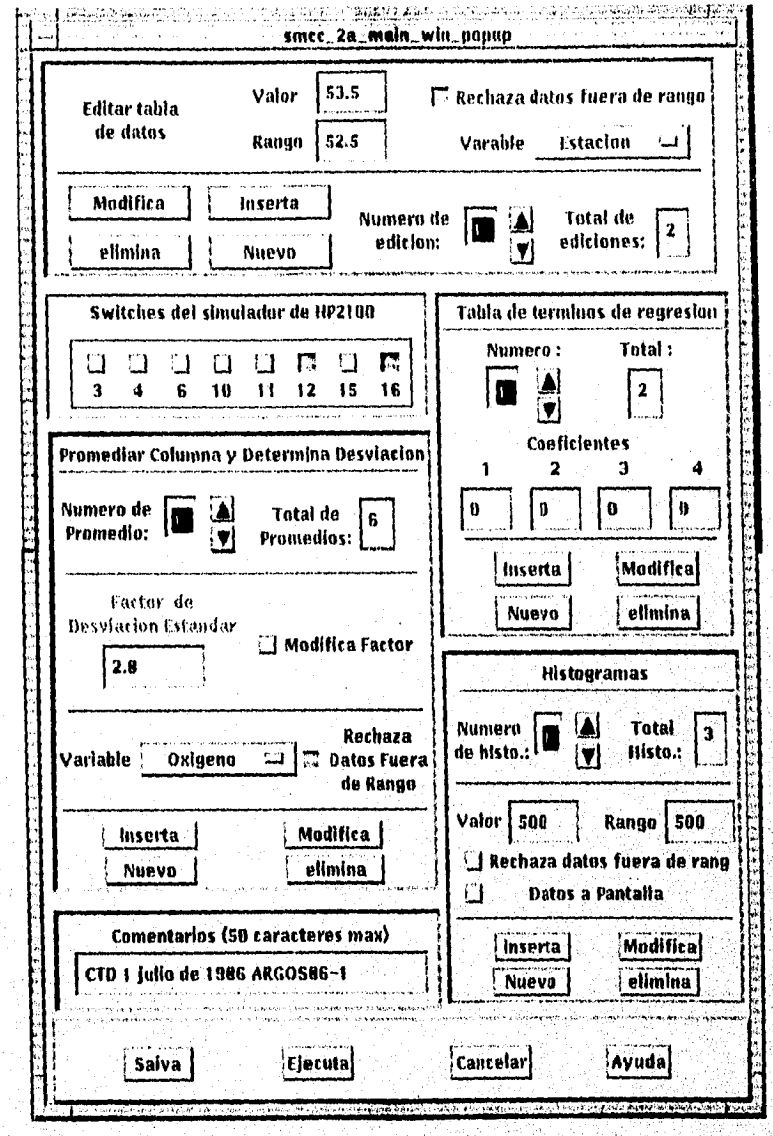


Figura D.8 Ventana para la captura de datos para el procedimiento de la calibración de conductividad en su segunda Etapa

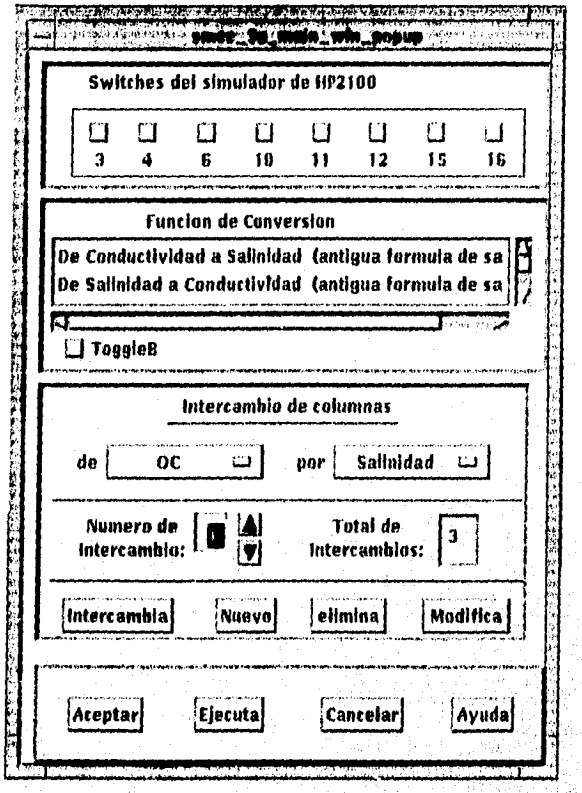


Figura D.9 Ventana para la captura de datos para el procedimiento de la calibración de conductividad en su Tercera Etapa

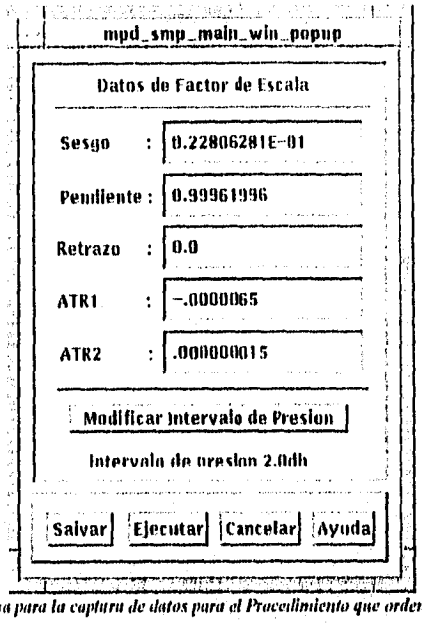


Figura D.10 Ventana para la captura de datos para el Procedimiento que ordena series de Presión.

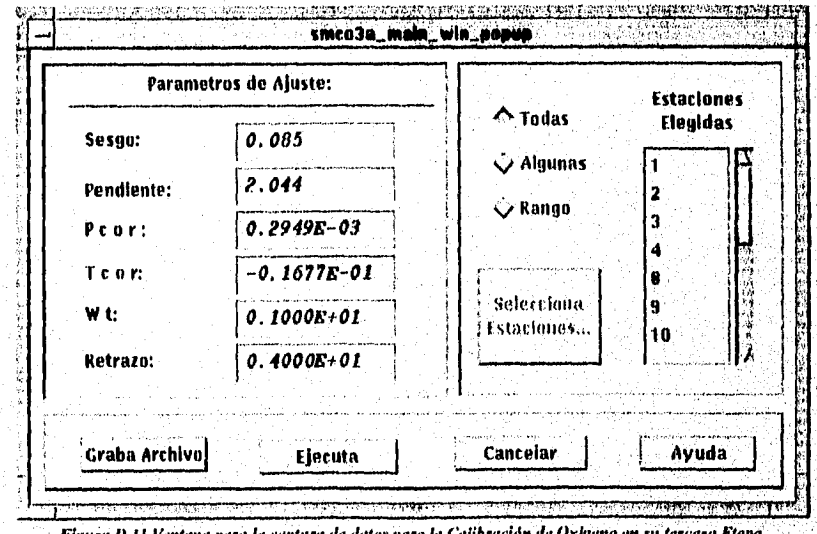


Figura D.11 Ventana para la captura de datos para la Calibración de Oxigeno en su tercera Etapa.

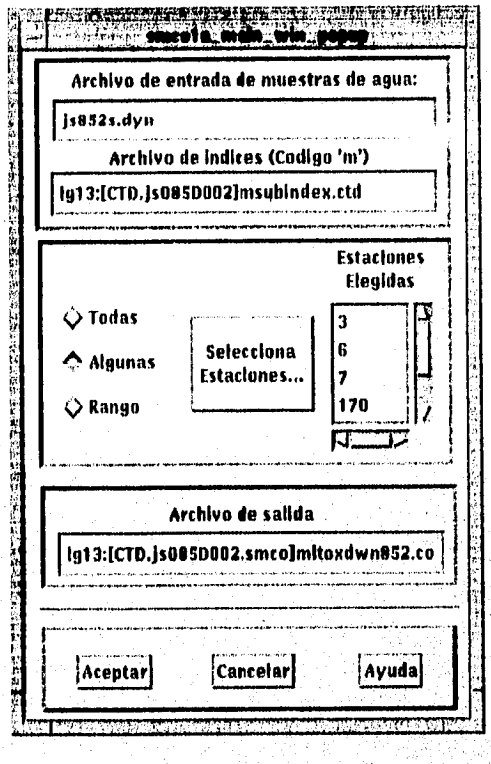


Figura D.12 Ventana para capturar datos para el procedimiento de calibración de Oxígeno en su Primera Etapa.

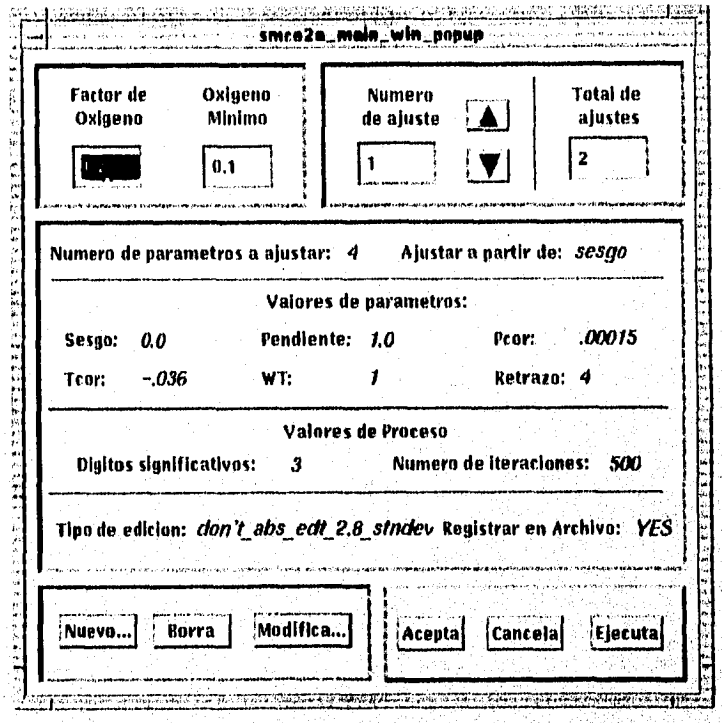


Figura D.13 Ventana Inicial Para Mostrar Datos del procedimiento de calibración de Oxígeno en su segunda etapa.

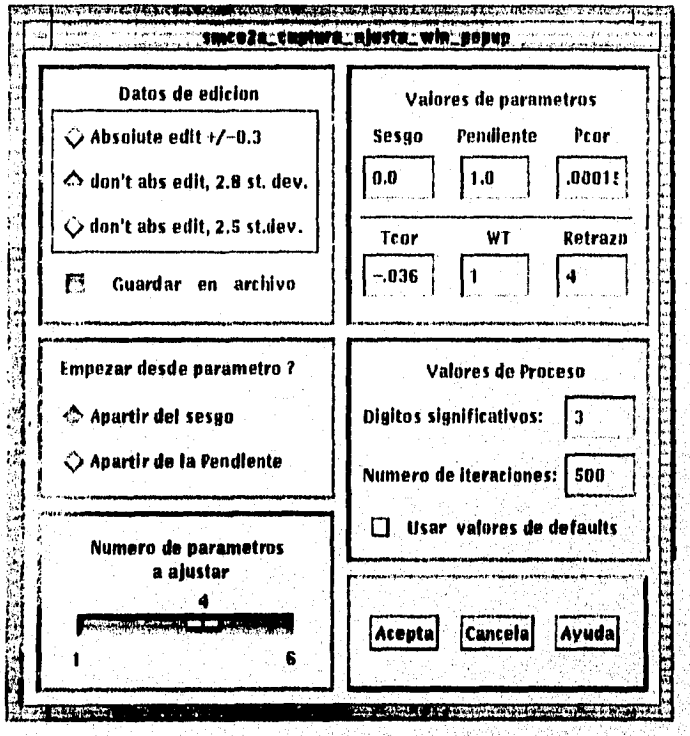


Figura D.14 Ventana para la captura de datos para el procedimiento de calibración de Oxígeno en su Segunda Etapa.

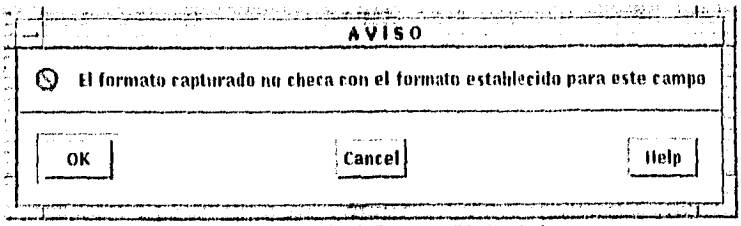


Figura D.15 Ventana de AVISO para validación de datos.

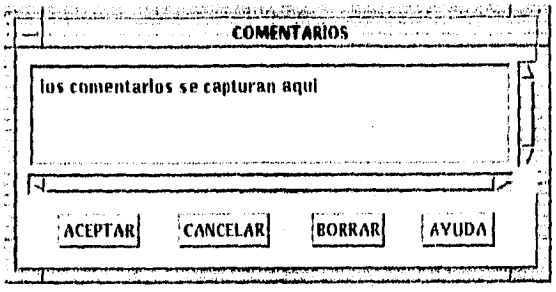


Figura D.16 Ventana para capturar COMENTARIOS.

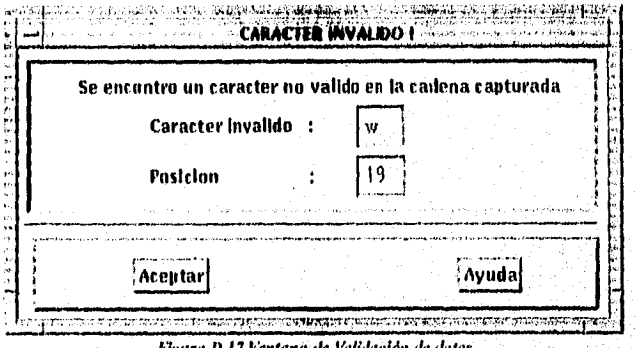


Figura D.17 Ventana de Validación de datos.

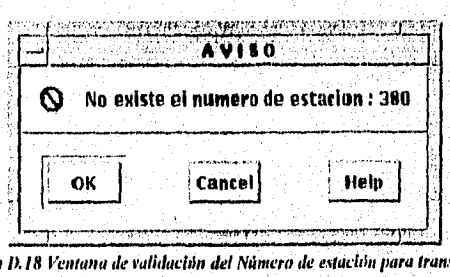


Figura D.18 Ventana de validación del Número de estación para transectos.

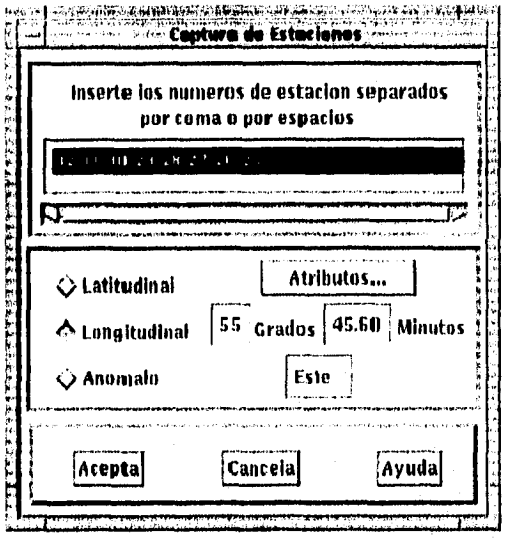
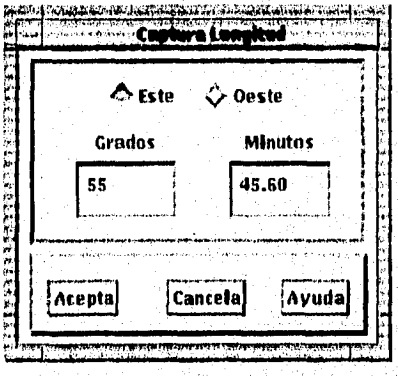
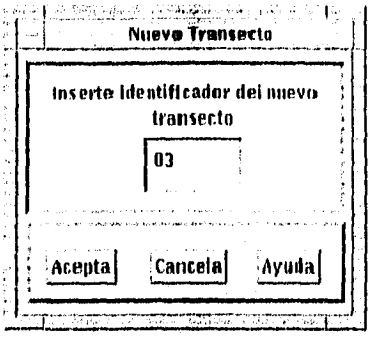


Figura D. 19 Ventana para capturar datos de estaciones y caracteristicas de Transectos.



Ventana D.20 Ventana para capturar tipos de transectos.


Nuevo Transecto
Inserte identificador del nuevo transecto
03
Acepta
Cancela
Ayuda

Apéndice E

Archivos de Datos

```
*001   29.0 30/OCT/85 07:50
02:01 13:50 25,55.0 96,59.9 14:21 25,55.0 96,59.8
 4   1.6   1.6  24.874 47.413 31.068 20.498  1.566 25.600 0.797  26 19  8  8     8   31.428  5.060
 3  10.6  10.6  24.737 47.630 31.173 20.558  1.596 25.856 0.818  25 17  7  7  7  7   31.575  5.010
 2  17.4  17.4  24.927 48.233 31.495 20.743  1.584 26.498 0.818  24 15  6  6     6   31.973  4.820
 1  24.9  24.9  26.732 54.954 35.042 22.856  0.897 25.856 0.499  23 12  5  5     5   35.724  2.810
* 02   50.0 30/OCT/85 09:55
04:06 15:54 25,55.0 96,45.0 16:07 25,55.1 96,45.0
 5   1.7   1.7  25.507 51.827 33.704 22.230  1.366 27.008 0.739  33 30 13 13    13   33.961  4.410
 4  10.7  10.7  26.647 54.777 34.979 22.826  1.236 26.752 0.705  32 28 12 12    12   35.149  3.670
 3  20.7  20.7  26.982 56.519 35.978 23.481  1.184 26.624 0.682  31 26 11 11    11   36.273  3.700
 2  35.7  35.7  26.995 56.619 36.032 23.518  1.101 26.240 0.652  30 24 10 10    10   36.358  3.880
 1  44.3  44.3  26.996 56.623 36.030 23.515  1.207 25.216 0.875  29 21  9  9     9   36.354  3.760
* 03   99.0 30/OCT/85 12:06
01:07 18:05 25,55.0 96,24.9 18:18 25,55.1 96,24.9
 7   1.8   1.8  27.062 56.533 35.929 23.416  1.256 26.624 0.734  45 20 20       20   36.322  4.615
 6  10.2  10.2  26.928 56.378 35.918 23.456  1.369 25.984 0.766  41 43 19 19    19   36.309  4.460
 5  21.0  21.0  26.912 56.387 35.918 23.459  1.370 25.216 0.768  40 41 18 16    18   36.309  4.475
 4  30.3  30.3  26.921 56.389 35.925 23.460  1.392 24.192 0.763  39 39 17 17    17   36.313  4.395
 3  50.2  50.2  26.916 56.404 35.934 23.467  1.324 23.168 0.710  38 37 16 16    16   36.318  4.475
 2  75.8  75.8  22.782 52.023 35.943 24.749  1.240 24.192 0.584  37 35 15 15    15   36.324  4.225
 1  91.2  91.2  21.271 50.512 36.028 25.231  1.035 26.368 0.473  36 33 14 14    14   36.370  3.810
* 4  960.0 30/OCT/85 15:00
00:26 21:00 25,55.0 96,00.0 21:20 25,55.1 96,00.0
12   1.8   1.8  27.070 56.702 36.043 23.501  1.368 25.088 0.763  57 70  8  8     8   36.348  4.615
11  25.1  25.1  26.992 56.660 36.069 23.545  1.265 22.912 0.684  56 68  7  7     7   36.344  4.695
10  50.9  50.9  26.856 56.621 36.131 23.634  1.316 19.456 0.662  55 66  8  6     6   36.358  4.760
 9 100.8 100.8  23.529 53.681 36.604 25.013  1.021 14.976 0.417  54 64  5  5     5   36.762  3.735
 8 200.5 200.5  16.700 46.044 36.196 26.520  0.785 11.848 0.235  53 62  4  4     4   36.210  3.120
 7 300.4 300.4  13.206 42.014 35.880 26.895  0.715  9.728 0.180  52 60  3  3     3   35.687  2.780
 6 401.3 401.3  10.933 39.504 35.344 27.074  0.738  8.192 0.160  51 58  2  2     2   35.354  2.630
 5 501.7 501.7   9.103 37.568 35.103 27.202  0.784  7.168 0.154  50 56  1  1     1   35.108  2.740
 4 602.1 602.1   7.836 36.129 34.783 27.145  0.832  6.272 0.149  49 53 24 24    24   34.978  2.800
 3 701.6 701.8   6.835 35.004 34.513 27.078  0.903  5.760 0.153  48 51 23 23    23   34.912  3.080
 2 801.8 801.8   6.105 34.414 34.546 27.201  0.960  5.376 0.159  47 49 22 22    22   34.900  3.360
 1 931.3 931.3   5.257 33.655 34.727 27.448  1.154  8.144 0.180  46 47 21 21    21   34.919  3.900
* 5  1820.0 30/OCT/85 19:07
04:34 01:08 25,55.0 95,24.7 02:04 25,55.3 95,24.5
12   1.8   1.8  26.686 56.819 36.273 20.538  1.407 23.296 0.736  73 94 20 20    20   36.273  4.730
11  25.6  25.6  26.699 56.625 36.261 20.529  1.353 20.480 0.679  72 92 19 19    19   36.261  4.750
10  50.4  50.4  26.689 56.625 36.261 20.612  1.356 15.616 0.619  71 90 18 16    18   36.261  4.690
 9 100.4 100.4  21.725 51.693 36.568 22.488  0.989 12.288 0.361  70 88 17 17    17   36.568  3.710
 8 200.7 200.7  15.318 44.426 36.006 23.790  0.848  9.728 0.226  69 85 16 16    16   36.006  2.920
 7 400.7 400.7  10.150 38.649 35.244 24.370  0.799  6.784 0.161  88 84 15 15    15   35.244  2.690
 6 600.2 600.2   7.153 35.627 34.927 24.935  0.909  5.376 0.154  87 82 14 14    14   34.927  3.020
 5 800.6 800.6   5.759 34.419 34.904 25.484  1.098  4.608 0.169  86 80 13 13    13   34.904  3.660
 4 1000.5 1000.5 4.914 33.770 34.939 26.499  1.265  4.224 0.185  85 78 12 12    12   34.939  4.310
 3 1200.9 1200.9 4.449 33.456 34.961 27.035  1.374  3.968 0.194  84 76 11 11    11   34.961  4.730
 2 1500.1 1500.1 4.253 33.421 34.978 27.389  1.429  3.968 0.193  83 74 10 10    10   34.978  4.910
 1 1802.0 1802.0 4.217 33.515 34.980 27.883  1.439  4.096 0.186  82 72  9  9     9   34.980  5.140
.....
.....

.....
```

```
* 53 3080.0 15/NOV/85 16:09
03:52 22:12 20,40.1 94,59.9 23:00 20,40.6 94,59.4
 11  10.7  10.7  26.149 56.496 36.620 24.229  1.664 22.656  0.856 125  95  7  7    7   36.309 4.950
 10  25.2  25.2  26.141 56.499 36.622 24.233  1.650 20.608  0.824 123  94  6  6    6   36.307 4.770
  9  50.8  50.8  23.296 53.474 36.859 25.128  1.689 18.432  0.747 121  93  5  5    5   36.350 4.990
  8 101.2 101.2  20.506 50.590 36.733 25.988  1.158 15.360  0.427 119  92  4  4    4   36.423 3.270
  7 200.7 200.7  15.585 45.087 36.354 26.895  1.022 11.776  0.293 117  90  3  3    3   36.032 2.650
  6 300.2 300.2  12.347 41.353 35.859 27.202  0.929  8.832  0.221 115  89  2  2    2   35.555 2.660
  5 400.0 400.0  10.019 38.793 35.525 27.372  0.928  7.040  0.193 113  88  1  1    1   35.228 2.670
  4 600.8 600 8   6.994 35.738 35.208 27.603  1.070  5.504  0.185 111  87 24 24   24   34.919 2.700
  3 600.2 600.2   5.493 34.428 35.196 27.791  1.342  4.608  0.209 109  65 23 23   23   34.916 3.060
  2 1000.5 1000.5 4.858 33.792 35.229 27.917  1.589  4.224  0.233 107  85 22 22   22   34.945 4.490
  1 1499.6 1499.6 4.223 33.634 35.254 27.985  1.767  4.096  0.240 103  83 20 20   20   34.965 4.850
(No disparo la botella 2. Al llegar a la superficie la botella 12)
(ven'la perforla. )
(se inte.[illegible]aron valores en sal y oxi del nivel 2  RAO,Suarez  16-dic-85)
*100  28 [illegible] 08/NOV/85 13:26
00:00 2[illegible] 21 27.2 97,14.1 20:28 21,27.2 97,14.1
  5  1[illegible]  1 [illegible]  26.470 56.531 36.393 23.956  1.672 26.496  0.946  5  7  5  5    5   36.048 4.760
  4  5 3   5.9  26.447 56.482 36.375 23.951  1.625 26.496  0.912  4  6  4  4    4   36.070 4.730
  3  10.6  10 9 26.454 56.521 36.397 23.964  1.650 26.496  0.930  3  5  3  3    3   36.085 4.730
  2  15.1  15.4 26.457 56.524 36.391 23.959  1.654 26.368  0.932  2  3  2  2    2   36.083 4.720
  1  22.[illegible]  [illegible]2.3 26.433 56.477 36.377 23.958  1.601 26.496  0.892  1  1  1  1    1   36.066 4.620
(Estac[illegible]n el londeadero de Isla Lobos. )
```

Figura E-1. Sección del archivo del Catálogo del Crucero.

```
*S001
 4  6 1.75[illegible]0.99 999.99 999.99 999.99 999.99 31.428 999.99 999.99 999.99 999.99 999.99 10
 3  7 1.7[illegible] 999.99 999.99 999.99 999.99 999.99 31.575 999.99 999.99 999.99 999.99 999.99 10
 2  6 2.[illegible]6 999.99 999.99 999.99 999.99 999.99 31.973 999.99 999.99 999.99 999.99 999.99 10
 1  5 [illegible].[illegible]33 999.99 999.99 999.99 999.99 999.99 35.724 999.99 999.99 999.99 999.99 999.99 10
*S002
 5 12 2.[illegible]17 999.99 999.99 999.99 999.99 999.99 33.961 999.99 999.99 999.99 999.99 999.99 10
 4 12 2.[illegible]24 999.99 999.99 999.99 999.99 999.99 35.149 999.99 999.99 999.99 999.99 999.99 10
 3 11 [illegible]37 999.99 999.99 999.99 999.99 999.99 36.273 999.99 999.99 999.99 999.99 999.99 10
 2 10 4.[illegible]17 999.99 999.99 999.99 999.99 999.99 36.358 999.99 999.99 999.99 999.99 999.99 10
 1  9 4.[illegible]08 999.99 999.99 999.99 999.99 999.99 36.354 999.99 999.99 999.99 999.99 999.99 10
*S003
 7 20 1.708 999.99 999.99 999.99 999.99 999.99 36.322 999.99 999.99 999.99 999.99 999.99 10
 6 19 1.813 999.99 999.99 999.99 999.99 999.99 36.309 999.99 999.99 999.99 999.99 999.99 10
 5 18 1.917 999.99 999.99 999.99 999.99 999.99 36.309 999.99 999.99 999.99 999.99 999.99 10
 4 17 2.126 999.99 999.99 999.99 999.99 999.99 36.313 999.99 999.99 999.99 999.99 999.99 10
 3 16 1.708 999.99 999.99 999.99 999.99 999.99 36.318 999.99 999.99 999.99 999.99 999.99 10
 2 15 2.334 999.99 999.99 999.99 999.99 999.99 36.324 999.99 999.99 999.99 999.99 999.99 10
 1 14 3.965 999.99 999.99 999.99 999.99 999.99 36.370 999.99 999.99 999.99 999.99 999.99 10
..... .....
*F022
12  9 0.009 999.99 999.99 999.99 999.99 999.99 36.677 999.99 999.99 999.99 999.99 999.99 10
11  8 0.018 999.99 999.99 999.99 999.99 999.99 36.673 999.99 999.99 999.99 999.99 999.99 10
10  7 0.026 999.99 999.99 999.99 999.99 999.99 36.675 999.99 999.99 999.99 999.99 999.99 10
 9  6 0.044 999.99 999.99 999.99 999.99 999.99 36.431 999.99 999.99 999.99 999.99 999.99 10
```

```
 8  5 0.390 999.99 999.99 999.99 999.99 36.618 999.99 999.99 999.99 999.99 10
 7  4 1.180 999.99 999.99 999.99 999.99 35.533 999.99 999.99 999.99 999.99 10
 6  3 1.482 999.99 999.99 999.99 999.99 35.143 999.99 999.99 999.99 999.99 10
 5  2 1.658 999.99 999.99 999.99 999.99 34.923 999.99 999.99 999.99 999.99 10
 4  1 1.588 999.99 999.99 999.99 999.99 34.909 999.99 999.99 999.99 999.99 10
 3 24 1.276 999.99 999.99 999.99 999.99 34.976 999.99 999.99 999.99 999.99 10
 2 23 1.303 999.99 999.99 999.99 999.99 34.978 999.99 999.99 999.99 999.99 10
 1 22 1.272 999.99 999.99 999.99 999.99 34.974 999.99 999.99 999.99 999.99 10
*F023
12 21 0.030 999.99 999.99 999.99 999.99 36.607 999.99 999.99 999.99 999.99 10
11 20 0.048 999.99 999.99 999.99 999.99 36.613 999.99 999.99 999.99 999.99 10
10 19 0.077 999.99 999.99 999.99 999.99 36.645 999.99 999.99 999.99 999.99 10
 9 18 0.122 999.99 999.99 999.99 999.99 36.358 999.99 999.99 999.99 999.99 10
 8 17 0.638 999.99 999.99 999.99 999.99 36.336 999.99 999.99 999.99 999.99 10
 7 16 1.359 999.99 999.99 999.99 999.99 35.429 999.99 999.99 999.99 999.99 10
 6 15 1.813 999.99 999.99 999.99 999.99 34.950 999.99 999.99 999.99 999.99 10
 5 14 1.623 999.99 999.99 999.99 999.99 34.902 999.99 999.99 999.99 999.99 10
 4 13 1.535 999.99 999.99 999.99 999.99 34.929 999.99 999.99 999.99 999.99 10
 3 12 1.297 999.99 999.99 999.99 999.99 34.972 999.99 999.99 999.99 999.99 10
 2 11 1.279 999.99 999.99 999.99 999.99 34.973 999.99 999.99 999.99 999.99 10
 1 10 1.238 999.99 999.99 999.99 999.99 34.974 999.99 999.99 999.99 999.99 10
.....
*N052
 6 13 28.982 999.99 999.99 999.99 999.99 35.195 999.99 999.99 999.99 999.99 10
 5 12 31.317 999.99 999.99 999.99 999.99 34.917 999.99 999.99 999.99 999.99 10
 4 11 29.627 999.99 999.99 999.99 999.99 34.906 999.99 999.99 999.99 999.99 10
 3 10 26.737 999.99 999.99 999.99 999.99 34.941 999.99 999.99 999.99 999.99 10
 2  9 25.280 999.99 999.99 999.99 999.99 34.959 999.99 999.99 999.99 999.99 10
 1  8 25.243 999.99 999.99 999.99 999.99 34.972 999.99 999.99 999.99 999.99 10
*N053
11  6 0.620 999.99 999.99 999.99 999.99 36.308 999.99 999.99 999.99 999.99 10
10  5 0.871 999.99 999.99 999.99 999.99 36.307 999.99 999.99 999.99 999.99 10
 9  4 4.580 999.99 999.99 999.99 999.99 36.350 999.99 999.99 999.99 999.99 10
 8  3 8.016 999.99 999.99 999.99 999.99 36.423 999.99 999.99 999.99 999.99 10
 7  2 16.139 999.99 999.99 999.99 999.99 36.032 999.99 999.99 999.99 999.99 10
 6  1 24.123 999.99 999.99 999.99 999.99 35.555 999.99 999.99 999.99 999.99 10
 5 24 30.918 999.99 999.99 999.99 999.99 35.226 999.99 999.99 999.99 999.99 10
 4 23 34.255 999.99 999.99 999.99 999.99 34.919 999.99 999.99 999.99 999.99 10
 3 22 28.485 999.99 999.99 999.99 999.99 34.916 999.99 999.99 999.99 999.99 10
 2 21 26.239 999.99 999.99 999.99 999.99 34.945 999.99 999.99 999.99 999.99 10
 1 20 25.856 999.99 999.99 999.99 999.99 34.965 999.99 999.99 999.99 999.99 10
*N100
 5  5 0.131 999.99 999.99 999.99 999.99 36.048 999.99 999.99 999.99 999.99 10
 4  4 0.138 999.99 999.99 999.99 999.99 36.070 999.99 999.99 999.99 999.99 10
 3  3 0.156 999.99 999.99 999.99 999.99 36.085 999.99 999.99 999.99 999.99 10
 2  2 0.130 999.99 999.99 999.99 999.99 36.083 999.99 999.99 999.99 999.99 10
 1  1 0.106 999.99 999.99 999.99 999.99 36.066 999.99 999.99 999.99 999.99 10
```

Figura E-2 Sección del archivo de Nutrientes

```
*01
8,1,2,3,4,5,6,7,8
1 + 55 4560
fin de transecto
*02
8,16,15,14,13,12,11,10,9
1 + 55 4560
fin de transecto
*03
8,17,18,19,20,21,22,23,24
1 + 55 4560
fin de transecto
*04
8,32,31,30,29,28,27,26,25
1 + 55 4560
fin de transecto
*05
8,33,34,35,36,37,38,39,40
1 + 55 4560
fin de transecto
*06
8,100,46,45,47,44,43,42,41
1 + 55 4560
fin de transecto
*07
6,48,49,50,51,52,53
1 + 55 4560
fin de transecto
*93
6,41,40,25,24,9,8
1 + 55 4560
fin de transecto
*94
6,42,39,26,23,10,7
1 + 55 4560
fin de transecto
*95
7,53,43,38,27,22,11,8
1 + 55 4560
fin de transecto
*96
7,52,44,37,28,21,13,4
1 + 55 4560
fin de transecto
*97
2,14,4
1 + 55 4560
fin de transecto
*98
3,20,13,5
1 + 55 4560
fin de transecto
```

```
*0
52,1,2,3,4,7,8
16,15,14,13,12,11,10,9
17,18,19,20,21,22,23,24
32,31,30,29,28,27,26,25
33,34,35,36,37,38,39,40
100,46,45,47,44,43,42,41
46,49,50,51,52,53
1 + 55 4560
fin de transecto
*51
14,1,2,3,5,7,8
16,15,14,13,12,11,10,9
1 + 55 4580
fin de transecto
*52
22,33,34,35,36,37,38,39,40
100,46,45,47,44,43,42,41
48,49,50,51,52,53
1 + 55 4560
fin de transecto
*50
16,17,18,19,20,21,22,23,24
32,31,30,29,28,27,26,25
1 + 55 4560
fin de transecto
```

Figura E-3. Archivo de Transectos.

Bibliografía

[1] Foffonof, N.P. , y Millard, R,C, "Algorithms for Computation of Fundamental Properties of Seawater": Unesco Technical Papers in Marine Science, No 44 Unesco 1983.

[2] Millard, R.C. "CTD Calibration and Data Processing Techniques at WHOI using the 1978 Practical Salinity Scale", Proceedings International STD Conference and WorkShop, 1982

[3] Pérez Molero, José María, "Algoritmos Numéricos y Diseño de una Base de Datos para el Procesamiento de datos oceanográficos en tiempo Real". Instituto de Investigaciones Eléctricas 1986.

[4] Maricela Claudia Bravo. "Diseño y desarrollo de una interfaz gráfica para el procesamiento de datos hidrográficos recolectados por una sonda oceanográfica." Instituto Tecnológico de Zacatepec.

[5] José Luis Hernandez. "Diseño y desarrollo de una interfaz gráfica para la representación de datos hidrográficos recolectados por una sonda oceanográfica." Instituto Tecnológico de Zacatepec.

[6] Jerez, Victor; Mier, Mauricio. "Desarrollo y administración de Programas de Computadora". Instituto de Investigaciones Eléctricas 1985.

[7] OSF/MOTIF "Programmers's Reference." (Open Software Foundation, revisión 1.1, Prentice Hall, Englewood Cliffs, New Jersey).

[8] Schildt, Herbert. "Programación en Lenguaje 'C'". Osborne McGraw-Hill. 1989

[9] E.Nava, C. Martínez, F.Huesca y A.S. Agüera "Conceptos fundamentales de X-Window". Boletín IIE, Septiembre-Octubre de 1992. Vol. 16 No 5. pp 232-239.

[10] Jones, Oliver. "Introduction to the X-Window System". De. Prentice Hall. New Jersey 1989.

[11] Brain, Marshall. "MOTIF Programming, The Essentials and More". Digital Press. 1992

[12] VAX,VMS Command Language Users Guide, Digital Equipment Corporation. Mayo 1981.

[13] Deutsch, M. "Verification and Validation in Software Engineering". Prentice Hall. 1884