**UNIVERSIDAD PANAMERICANA**

ESCUELA DE INGENIERIA

CON ESTUDIOS INCORPORADOS A LA

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

# "PLANTEAMIENTO DE UNA ALTERNATIVA DE SOLUCION PARA EL CONTROL DE TIEMPO Y ASISTENCIA DE PERSONAL"

## T E S I S

QUE PARA OBTENER EL TITULO DE:
**INGENIERO MECANICO ELECTRICISTA**
**AREA: INGENIERIA INDUSTRIAL**
P R E S E N T A :
**CARLOS FRUTOS RUBIO**

DIRECTOR: ING. ALFONSO GERARDO LEAL GUAJARDO

MEXICO, D. F. 1997

*A mis padres,*
*por su apoyo incondicional.*

*A mi esposa,*
*sin cuyo estímulo no hubiera*
*realizado este trabajo.*

*A mis hijos,*
*quienes son mi máxima*
*motivación en la vida.*

# ÍNDICE

## Introducción

El proceso tradicional de calcular el tiempo trabajado por medio de la tarjeta de tiempo o tarjeta checadora, ha representado siempre una carga de trabajo para el área administrativa de las empresas. El trabajo de procesar tarjeta por tarjeta por cada uno de los trabajadores, es un proceso tedioso donde frecuentemente se cometen errores en su cálculo. Aunado a esto, sólo una parte de la información que puede proporcionar la tarjeta es aprovechada, mientras otros datos que afectan la productividad como retardos y permisos no son aprovechados para la toma de decisiones.

### *Antecedentes*

El presente estudio está basado en mi trabajo realizado en la empresa Consultores Asociados de Chihuahua, S.A. de C.V. Esta empresa fue creada por mi socio y yo precisamente para elaborar y ofrecer soluciones a la problemática descrita anteriormente, basándolas en el uso de la tecnología más moderna disponible en ese momento. Antes de iniciar operaciones, se realizó un sondeo (a finales de 1990 y principios de 1991) para percibir la problemática en su operación cotidiana y de esta manera empezar a vislumbrar las características que debería cumplir la alternativa que se iba a ofrecer. En este sondeo se mantuvieron pláticas con gerentes de personal de diversas empresas de manufactura, sobre todo del ramo de las maquiladoras, ya que en un principio, se pensó en atacar a ese mercado en particular.

Más que mencionar los datos específicos de una empresa de las que se analizaron, este trabajo se basa en una síntesis de los aspectos más relevantes y problemáticos de 5 lugares importantes donde se instaló la solución definitiva, éstos son: Bimbo de Chihuahua, Sigma Alimentos Noroeste, Gobierno del Estado de Chihuahua, Warner Electric y Presidencia Municipal de Chihuahua. (En conjunto, estos 5 centros de trabajo suman cerca de 4,000 trabajadores.)

Esto se debe a que las políticas de cada empresa en cuanto a la contabilización del tiempo trabajado, así como de los descansos, permisos, incapacidades, etc., varían significativamente.

***Generalidades***

El sistema clásico de control de tiempo y asistencia consiste en un reloj checador que imprime la hora y la fecha sobre la tarjeta individual del trabajador al momento de su entrada o salida. Generalmente las tarjetas se encuentran colocadas en un tarjetero de pared, de donde la toma el trabajador, marca su entrada o salida y la coloca en otro tarjetero que se encuentra al otro lado del reloj checador. Algunas empresas colocan en estos muebles sólo las tarjetas de los turnos susceptibles de entrada en ese momento, evitando así que se cometan fraudes por parte de los trabajadores. Este método, además de poco eficiente tiene otros problemas implícitos como son: falta de tinta en el impresor del checador, fallas mecánicas del mismo y otros relacionados directamente con él. Por otro

lado se encuentran los relacionados con los trabajadores, los cuales han encontrado los más diversos métodos para falsear o evitar el checado correcto o legible, tales como: doblado de la tarjeta para alterar el día de checado, engrapado de un día para dañar el checador o evitar una impresión legible, etc

Al final de la semana, se recogen las tarjetas y se llevan al departamento de nómina para su cálculo. Anteriormente, una o más personas realizaban los cálculos de toda la semana por trabajador. Cada día implicaba tres cálculos: la conversión a decimal de las horas y el cálculo de la diferencia entre ambas. Esto da un total de 18 cálculos por trabajador por semana. La empresa hipotética cuenta con 320 trabajadores, lo cual arroja un total de 5760 cálculos. Esta cifra permite vislumbrar la cantidad de errores que pueden cometerse por semana, a veces en perjuicio del trabajador y otras en perjuicio de la empresa.

Otros problemas con los que se enfrenta la persona que hace los cálculos radica en la situación particular de cada trabajador, por ejemplo: no todos los trabajadores descansan el mismo día o el mismo número de días, no todos tienen el mismo horario, hay quienes entrando a la misma hora tienen más tiempo para comer que otros, etc. Todo esto debe tomarlo en cuenta la persona que realiza los cálculos para reportar correctamente el número de días y el total de horas trabajadas.

El método actual para procesar estos datos consiste en tener la mayoría de estos datos almacenados en una computadora personal y mediante un programa, realizar la captura de las horas de checado, generando el cálculo de las horas trabajadas por día. En la mayoría de los casos de la ciudad donde se realizó el estudio, este tipo de servicio es contratado con una empresa externa. Estas empresas se especializan en este tipo de labores, cobrando una comisión por persona procesada por cada nómina. Esto libera de este trabajo tan tedioso a la empresa, pero también la limita en su capacidad de analizar la información y tomar decisiones en el momento oportuno. La empresa externa genera la nómina y la regresa a la empresa solicitante con un desfasamiento de siete días.

***Análisis de costos***

Desde el punto de vista económico, se puede desglosar el costo del sistema de checado convencional de la siguiente manera:

**Tabla 1- Costos**

**Costo anual empresa de 320 trabajadores**

| CONCEPTO | COSTO |
|---|---|
| Tarjetas de checado (320 trabs x 52 semanas x $0.10 x 1.01 merma) | $1,680.00 |
| Mantenimiento del reloj checador (incluye cinta de impresión) N$100 mensual x 12 | $1,200.00 |
| Procesamiento de los datos por despacho externo (320 trabs x 52 x $1.00) | $16,640.00 |
| TOTAL COSTO ANUAL | N$19,520.00 |

Como se puede apreciar, el costo aparentemente "bajo" del método tradicional de control de tiempo por tarjetas checadoras, es en realidad bastante significativo, siendo casi de $20,000 pesos al año para el caso supuesto de 320 trabajadores. Nótese que no se está incluyendo el costo en si del aparato checador.

Al comparar alternativas, además de comparar los costos de operación de cada una, se deben considerar los beneficios intangibles que poseen. Es difícil estimar en pesos y centavos cuánto representa el hecho de poder tener información confiable al instante. En un sistema ideal, se desearían conocer los eventos que ha realizado un trabajador en cuanto a entradas y salidas, al instante siguiente en que ocurrieron, y no tener que esperar varios días.

En el capítulo siguiente se analizarán los métodos existentes de captura de datos.

# 1 Métodos de captura de datos

Existen básicamente dos métodos para la captura de datos: manual y automático.

## *1.1 Método manual*

El método tradicional de introducir información a una computadora ha consistido en escribir manualmente en un teclado la información contenida en hojas impresas. Los estudios muestran que la tasa de error para este método de captura es aproximadamente de 1 error por cada 300 caracteres tecleados. Este tipo de captura no proporciona información en tiempo real, ya que los datos que están siendo capturados generalmente reflejan acontecimientos pasados. Debido a que la información es primero transcrita al papel y luego capturada, existen varias oportunidades para cometer errores.

## *1.2 Método automático*

Diversos métodos de captura automática han sido desarrollados para sobrevenir las desventajas de la captura manual. Al hablar de "automático" se quiere decir que un evento de captura individual puede resultar en un flujo de datos (desde un sólo carácter hasta docenas de éstos). Bajo esta definición, un operador humano, puede o no ser parte del proceso de captura.

Al evaluar las diferentes alternativas de identificación o captura automática, deben considerarse dos factores: la tasa de error de sustitución[1] comúnmente conocida como tasa de error. Este término describe la probabilidad de que un caracter dado contenga un error. El número de errores que se pueden esperar de una aplicación determinada es igual a la tasa de error de sustitución multiplicada por el número de caracteres capturados:

$$C_e = SER \times n$$

donde:

$C_e$ es el número de caracteres con error

$SER$ es la tasa de error de sustitución

$n$ es el número total de caracteres capturados

El segundo parámetro a considerar es la Tasa de Primer Lectura [2]. Este término (expresado como porcentaje), se refiere a la probabilidad de que un evento de captura de datos, resulte en datos capturados al primer intento. Por ejemplo, si se fueran a capturar 1000 datos con un sistema con un FRR de 75%, entonces se requerirían aproximadamente 1333 intentos para lograrlo. Para cualquier tecnología dada existe una fuerte correlación entre la tasa de error de sustitución y la tasa de primera lectura.

---

[1] SER-Substitution Error Rate
[2] FRR-First Read Rate

## 1.3 *Tecnologías de recolección automática de datos*

### 1.3.1 Magnético

Es posible codificar una gran cantidad de datos en una banda magnética, del tipo encontrado en las tarjetas de crédito. La información es almacenada en regiones con diferente polaridad, como en las cintas y discos de computadoras. Los datos pueden leerse o escribirse a la banda magnética, agregando flexibilidad al sistema de información. Esta tecnología no ha sido usada en todos los ámbitos que requieren recolección automática de datos debido principalmente a:

- Equipo de rastreo [3] remoto (no contacto) no disponible
- La imposibilidad de los métodos tradicionales de impresión de codificar bandas magnéticas
- Mayores costos comparados con tecnologías ópticas que utilizan imágenes impresas en sustratos de papel.

### 1.3.2 Reconocimiento de voz

Existe equipo que puede "entender" el lenguaje hablado. Esta tecnología es apropiada para ciertas labores manuales (tales como la distribución de equipaje, o el control de calidad) donde el operario necesita tener libres ambas manos. Sin embargo, el reconocimiento de voz no encaja en la definición dada con anterioridad; los números de parte o códigos tendrían que ser deletreados carácter por carácter. Además el equipo disponible

[3] *Scanning equipment*

actualmente requiere que cada operario "entrene" al sistema para que reconozca su voz y cuenta con un vocabulario limitado. En esta tecnología el operario es una parte integral del proceso de captura y es la fuente principal de todos los errores.

### 1.3.3 Radio frecuencia

Los sistemas que utilizan este tipo de tecnología, pueden "leer" etiquetas o tarjetas que no estén directamente a la vista del sistema. La forma de operar de este sistema es la siguiente: una señal de radio es dirigida hacia la tarjeta, y ésta responde con otra señal de radio que se modula con la información contenida en la tarjeta. Inicialmente se utilizó para el rastreo y control de ganado, pero recientemente han sido desarrolladas muchas aplicaciones para el transporte y manufactura.

Las tarjetas pueden ser preprogramadas con datos, o pueden permitir que los datos sean cambiados en respuesta a comandos modulados por la señal de origen. La mayoría de las tarjetas programables requieren de una batería propia, mientras que las no programables derivan su energía de la señal de origen.

Esta tecnología ofrece sustanciales ventajas en muchas aplicaciones donde el costo de la tarjeta puede ser justificado.

### 1.3.4 Código de barras

Un código de barras consiste en una sucesión de barras y espacios paralelos de anchos diferentes. La información se encuentra codificada en

el patrón de la anchura de las barras y los espacios. La altura del código permite que existan múltiples caminos de rastreo posibles, requiriendo que uno solo se encuentre libre de imperfecciones. Esta es una tecnología óptica, ya que la información es rastreada usando luz que se refleja de las partes claras y oscuras del código.

Los símbolos de código de barras pueden ser impresos por una amplia gama de técnicas a bajo costo y las dimensiones del símbolo pueden variarse para ajustarse a los requerimientos.

Los códigos de barras ofrecen muy alta seguridad de los datos ya que la tasa de error de sustitución es menor a 1 en un millón de caracteres. La tasa de primera lectura es generalmente mayor al 80%, y muchos de los sistemas de rastreo automático aumentan esta tasa a cerca del 100%.

Existen otras tecnologías de recolección automática de datos que son muy específicas a su campo de aplicación, entre ellas se encuentran:

- OCR Reconocimiento óptico de caracteres. Consiste de caracteres escritos con una tipografía especial la cual permite a equipo especializado leer los caracteres impresos. Es el precursor del código de barras.
- MICR Reconocimiento de tinta magnética. Generalmente usados en cheques y aplicaciones bancarias.
- Machine vision Visión mecánica o visión de máquina. Consiste en una cámara de video de alta resolución conectada a una computadora, su principal función es inspección y selección.

- Smart cards  Tarjetas "inteligentes". Consisten en una tarjeta similar a las de crédito que contienen memoria no volátil, un controlador y un mecanismo de seguridad. Usadas ampliamente en la industria telefónica.

### *1.4 Elección de la tecnología de captura*

En la siguiente tabla, se presenta un resumen de las técnicas de captura descritas. Los datos mostrados, asumen que un operador humano realiza la captura de una etiqueta que se encuentra adherida a un objeto. El tiempo de captura incluye el tiempo requerido para tomar el *instrumento rastreador*[4] y ponerlo frente a la etiqueta.

[4] *Scanner*

Tabla 2 - Comparación de tecnologías

| | Tiempo para capturar un campo de 20 caracteres. | Tasa de error de sustitución SER | Dimensión de la etiqueta. | Costo de la etiqueta. | Costo del equipo de lectura. | Ventajas | Desventajas |
|---|---|---|---|---|---|---|---|
| Tecleado Manual | 10 segundos | Alto | 10 x 12 cm. | Bajo | Bajo | Bajo costo inicial del equipo | Requiere operador humano. Poca flexibilidad. Baja velocidad y SER alto. |
| OCR | 4 segundos | Medio | 11 x 13 cm | Bajo | Medio | Puede ser leído por humanos | SER alto, inflexibilidad del equipo lector. |
| MICR | Rastreado por máquina | Medio | 11 x 13 cm | Medio | Alto | Puede ser leído por humanos | Equipo caro e inflexible. |
| Banda magnética | 4 segundos | Bajo | 10 x 12 cm. | Medio | Medio | Pueden codificarse gran cantidad de datos. Los datos pueden cambiarse. | Afectado por campos magnéticos. Requiere contacto. Equipo lector inflexible. |
| Reconocimiento de voz | 20 segundos | Alto | 10 x 12 cm | Bajo | Alto | Operación a "manos libres" | Requiere operador humano. Alta tasa de error. El equipo debe ser "entrenado" por cada operador. |
| Visión de máquina | Rastreado por máquina | Depende de la técnica. | Variable | Variable | Muy alto | Puede ser parte del sistema de inspección. | Caro. No aplicable a todos los problemas. |
| Radio frecuencia | 2 segundos | Bajo | 2.5 x 4 x 0.4 cm | Alto | Alto | La etiqueta no requiere estar visible. | Alto costo de las etiquetas. |
| Código de barras | 4 segundos | Bajo | 1.5 x 6.5 cm | Bajo | Bajo | Flexibilidad del equipo impresor y del equipo lector. | |

### 1.4.1 Selección por comparación

Como se puede observar en la tabla anterior, comparado con otras técnicas de identificación automática, el código de barras destaca como una tecnología atractiva. Tiene los atributos deseados para cualquier tipo de tecnología, éstos son: una baja tasa de error (SER), bajo costo del símbolo de identificación (la etiqueta con el código de barras en este caso) y bajo costo y flexibilidad del equipo de identificación o rastreo. Se puede imprimir en una gran variedad de sustratos por una amplia gama de técnicas y ofrece una alta seguridad de los datos. Existe una gran variedad de equipo disponible para satisfacer cualquier aplicación imaginable.

Por este motivo, se optó por la tecnología de código de barras, la opción alterna era un lector de banda magnética, el cual se desechó por tener un mayor costo indirecto asociado con él. Este último requiere de un codificador con el cual se graba en la banda magnética la información requerida, y su precio fluctúa alrededor de los $2000 dólares. Además de que el costo de cada credencial es aproximadamente el doble al de una mica sencilla que es lo que requiere el código de barras. Desde el punto de vista de la susceptibilidad de la información codificada (ya sea en el código o en la banda), la banda magnética es susceptible de dañarse si se acerca a campos magnéticos fuertes como motores eléctricos o generadores (comunes en una planta industrial).

Por su parte, el código de barras es sumamente económico de producir ya que se pueden imprimir códigos de la mejor calidad en una impresora láser común, con un costo mínimo por código. Además sólo requiere enmicarse para su protección y en caso de pérdida, el costo por reposición es mínimo y lo puede efectuar personal secretarial si se cuenta con los códigos ya impresos.

### 1.4.2 Simbologías de código de barras

El código de barras es una representación gráfica de un número o de una cadena alfanumérica que consiste de barras y espacios alternados de dimensiones específicas. Existen diversos tipos de simbologías, las cuales utilizan un método diferente de codificar la información en las barras y los espacios. A continuación se presenta una muestra de las diferentes simbologías disponibles:

**Tabla 3 - Simbologías**

| Nombre | Tipo | No. caracteres | Densidad neta de datos (CPI)[5] |
|---|---|---|---|
| CODE 39 | Alfanumérico | Variable | 9.8 |
| CODABAR | Numérico | Variable | 12.8 |
| CODE 128 | Alfanumérico | Variable | 12.1 |
| INTERLEAVED 2 OF 5 | Numérico | Variable (par) | 18.0 |
| CODE 93 | Alfanumérico | Variable | 14.8 |
| EAN 13 | Numérico | Fijo (13) | |

[5] La densidad neta de datos refleja los caracteres por pulgada al imprimirse utilizando una dimensión X de 7.5 mil

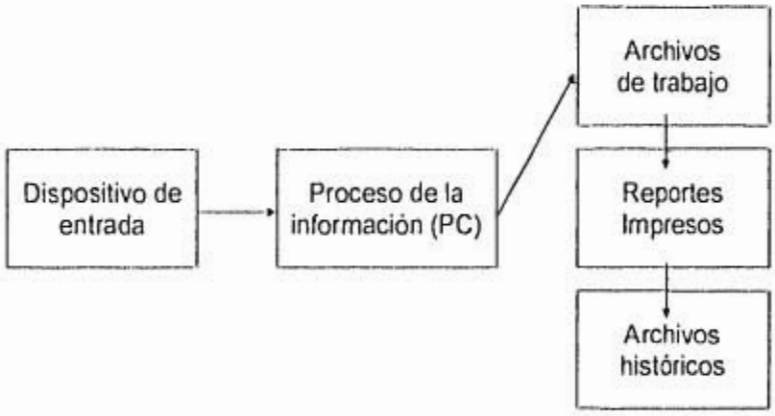
Algunas de estas simbologías son un estándar de facto en diversas industrias.

Se buscó una simbología que fuera compacta, es decir que en sus especificaciones nominales (sin reducción fotográfica o de otro tipo) codificara el mayor número de caracteres por unidad de longitud (mayor densidad de caracteres). Esto se hizo tomando como parámetro el número del trabajador que por su naturaleza es exclusivamente numérico y de longitud fija (6 dígitos).

Tomando en cuenta los requerimientos se concluyó que la simbología que mejor llenaba los requisitos es la **Interleaved 2 of 5**, debido a su facilidad de impresión y su alta densidad.

## 2 Diseño de una alternativa

Inicialmente, el sistema se concibió de la siguiente manera:




Una vez que se tomó la decisión de utilizar la tecnología de código de barras como instrumento de captura automática, el siguiente paso era determinar la forma de interrelacionar éste con las demás partes del sistema. En este aspecto se consideraron las siguientes opciones:

1. Lector de código de barras para gafetes (*slot reader*) conectado a través de un emulador de teclado (*keyboard wedge*) a una computadora tipo PC. La PC interpretaría los datos provenientes del lector como si éstos hubieran sido tecleados directamente en el teclado de la PC. La computadora, mientras tanto, estaría corriendo un programa de captura del número de trabajador (éste sería recibido fracciones de segundo después de que el trabajador hubiera deslizado su gafete). Una vez

recibido el dato, le anexaría la fecha y hora de la transacción (llevada por el reloj interno de la PC) y los datos serían almacenados en un archivo local.

2. Lector de código de barras para gafetes (igual al del punto 1) conectado a través de una terminal con capacidad de transmisión bidireccional de datos con la PC. Esta terminal cuenta con una pequeña memoria (128 bytes) y pantalla de LCD de 2 líneas por 10 caracteres. Las características de este tipo de terminal permitirían presentar mensajes específicos en su pantalla, tales como informar al trabajador la hora y fecha de su registro, y si éste fue recibido correctamente en la PC. Todos estos mensajes serían enviados en tiempo real por el programa de la PC de acuerdo a la información recibida.

3. Terminal de recolección automática de datos (DCU[6]). Esta terminal consiste básicamente de un lector de código de barras, una unidad central de procesamiento y un módulo de entrada y salida, todo agrupado en un gabinete que lo protege del medio ambiente. La unidad analizada tiene las siguientes características: pantalla LCD de 16 caracteres, 64 kb RAM, teclado tipo membrana de 12 teclas, batería recargable, módulo de entrada y salida tipo RS485.

### 2.1 *Análisis de las opciones consideradas.*

1. Esta opción implica tener una PC dedicada el 100% del tiempo, dejándola disponible para respaldos y el proceso en si de la información sólo el tiempo entre la entrada y salida de algún turno; además el trabajador no tiene manera de saber con qué hora se está registrando su movimiento o si la transacción fue recibida por la computadora o no, a menos que pudiera ver físicamente la pantalla de la PC para corroborar el hecho. Este tipo de configuración sólo permite un lector por cada PC[7].

2. Esta opción también requiere de la PC dedicada aunque tiene algunas ventajas sobre la anterior:

- La interfase RS485 permite una configuración *multidrop* (multipunto) en la cual se pueden conectar en serie hasta 16 terminales a una sola PC.
- La pantalla permitiría desplegar la fecha y la hora con la que se registra el movimiento y comunicar a la persona que está checando si la lectura fue exitosa o no, incluso podría hacer que el programa enviara un mensaje específico a un trabajador en particular en el momento que éste pasara el gafete por el lector.

3. Esta opción tiene las siguientes ventajas sobre las anteriores:

- Permite conectar la red RS485 (hasta 32 terminales) directamente al puerto serial de la PC.
- Al ser "inteligente", permite la captura autónoma del número de trabajador, "estamparle" la fecha y hora a la transacción y almacenarlo internamente.

---

[6] *Data Collection Unit*
[7] Pueden conectarse más terminales si la PC cuenta con más de un puerto serial.

- Permite validación local. La terminal cuenta con una base de datos que contiene los números de gafetes válidos, esto permite rechazar los gafetes dados de baja sin tener que consultar a la computadora anfitriona (HOST PC).
- Al contar con una batería interna recargable, asegura la operación total de la terminal hasta por cuatro horas sin energía eléctrica externa.
- Permite cierto grado de programación, haciendo con esto más flexible la operación de la terminal, pudiendo incluso ser utilizada para otras funciones además de la de control de tiempo y asistencia.

## 2.2 *Costo de cada alternativa*

1. $250 dólares
2. $500 dólares
3. $1000 dólares

Aunque la alternativa no. 3 es la de mayor costo, también es la que ofrece mayores ventajas sobre las otras, principalmente el hecho de poder trabajar de manera independiente de la PC y de poder seguir trabajando sin energía eléctrica. Por otro lado, una vez implantado el sistema, la información recabada por el lector de las horas de entrada y salida de cada trabajador es muy importante para la empresa, y claro para el trabajador, pues de ello depende el pago justo y correcto del tiempo trabajado normal y extra. Este equipo es el que ofrece mayor seguridad de los datos capturados.

### *2.3 Análisis específico de la terminal seleccionada.*

Ya que la empresa en la que se implantará este sistema cuenta con 320 trabajadores, la memoria utilizada por el lector para almacenar los códigos de los trabajadores sería:

$$Memoria\ requerida = n(L_1 + 3)$$

donde: $n$ es el número de trabajadores

$L_1$ es la longitud del código de barras

La memoria requerida está expresada en bytes.

Para este caso en particular la memoria requerida es entonces:

$$Memoria\ requerida = 320(6 + 3)$$

$$Memoria\ requerida = 2880 \text{ bytes.}$$

Del total de 65536 bytes (64 kb) la terminal distribuye la memoria según la siguiente tabla:

| AREA DE COLAS |
|---|
| BLOQUE DE HORARIOS |
| BLOQUE DE VALIDACIÓN |
| BLOQUE DE MENSAJES |
| BLOQUE DE PROMPTS<br>BLOQUE DE CONTROL DE TERMINAL<br>BLOQUE DE BIFURCACIÓN<br>BLOQUE DE SELECCIÓN DE CODIGO DE BARRAS<br>BLOQUE DE ACTIVIDAD<br>BLOQUE DE ENLACES<br>RAM DEL SISTEMA |

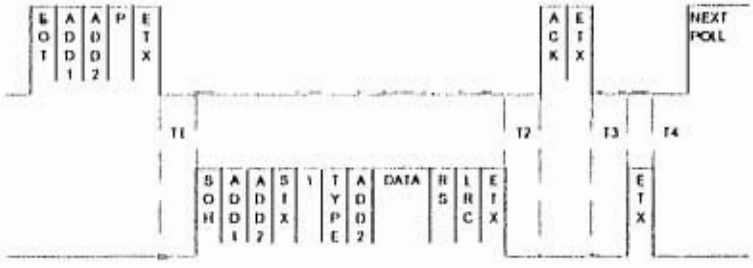
Como se puede ver en la tabla de distribución de memoria, el espacio asignado a la cola de transmisión es dinámico, es decir puede aumentar o disminuir dependiendo de cómo se asigne la memoria a los otros conceptos. Esto repercute en el caso estudiado de la siguiente manera:

Memoria disponible = 65535 - 2880 - 4000 (usados por el sistema).
Memoria disponible neta = 58655 bytes.

### *2.4 Protocolo de comunicación*

El protocolo de comunicación entre la PC y la terminal tiene por objeto mantener un alto grado de confiabilidad y rendimiento. Los tiempos de espera representan los intervalos entre el fin de la transmisión de un dispositivo y el inicio de la respuesta del dispositivo que recibe. Los tiempos mínimos especificados son los requeridos para evitar la pérdida de caracteres en el dispositivo receptor. Los tiempos máximos permiten al protocolo ciclar sin tener una excesiva demora en espera de una respuesta[6].

[6] *CMI POLLED PROTOCOL, CONTROL MODULE INC. 1991. pag 6*




Este protocolo ofrece la ventaja de la seguridad de la información, si se observa su implementación se deduce que un registro determinado se encuentra en el lector hasta que la PC notifique que fue recibido.

Para que la PC compruebe la integridad del dato o registro recibido, el lector lo formatea de la siguiente manera antes de enviarlo:

\ta *gs*<mPppp><data,*rs*><data,*rs*>*us*<timestamp,*rs*>

donde

ta es el tipo (t) y la dirección (a) de la terminal
*gs* es el indicador de inicio de grupo de datos
*rs* es el separador de registros
*us* es el indicador de inicio de unidad

A este "paquete" de transmisión, la terminal le agrega al final un carácter verificador tipo LRC[9]. Al recibir la PC el paquete, calcula el LRC y lo compara contra el recibido, si los caracteres coinciden, significa que el paquete fue recibido correctamente. La PC informa de la recepción a la

terminal, enviando un carácter especial (ACK), por su parte, la terminal borrará el registro sólo hasta haber recibido el ACK, de lo contrario lo retransmitirá la próxima vez que la PC solicite un registro.

Este proceso se lleva a cabo registro por registro hasta que la cola de transmisión del lector se encuentre vacía.

### 2.4.1 Archivo de Registro

Todo este flujo de información es capturado por la PC, decodificado y almacenado en una base de datos de las siguientes características:

| Dato | Tipo | Longitud | Observaciones |
|---|---|---|---|
| No. de gafete | Numérico | 6 | |
| Fecha del movimiento | Fecha | 8 | |
| Hora del movimiento | Carácter | 5 | |
| Tipo del movimiento | Carácter | 1 | Entrada o Salida |
| Origen | Numérico | 1 | Dirección de la terminal de origen |

**Tabla 4 - Estructura del archivo de registro**

A partir de estos datos el programa deberá generar toda la información requerida, tomando en cuenta los siguientes casos especiales:

- Que alguna de las dos checadas requeridas (entrada o salida) haya sido omitida por el trabajador.

[9] *Longitudinal Redundancy Check*

- Que el trabajador marque equivocadamente entrada por salida, o haga dos movimientos consecutivos del mismo tipo.
- Que un trabajador marque cualquier número de veces su entrada o salida (movimientos duplicados).

## 2.5 *Diseño del sistema*

### 2.5.1 Módulo de entrada

Tomando en cuenta estas consideraciones, el proceso de la información del tiempo trabajado podría diseñarse de la siguiente manera:


Este diseño permite hacer todas las tomas de decisión y cálculos una sola vez, dejando la información ya procesada disponible para la emisión de los diferentes reportes, los cuales la analizarán desde diversos puntos de vista. Con esto se ahorrará significativamente en tiempo de cálculo ya que algunas de las operaciones a efectuar involucran la toma de un número considerable de decisiones.

El primer paso en el diagrama de bloques anterior, consiste en transferir los datos de entradas y salidas que residen en la terminal lectora hacia la

computadora. La transferencia se debe llevar a cabo de acuerdo al protocolo descrito en la página 21.

#### *2.3.1.1 Comunicación*

Independientemente del lenguaje en que se desarrolle el resto de la aplicación, la manera más eficiente de controlar el puerto serial RS-232 (mediante el cual se comunican la terminal lectora y la computadora) es hacerlo directamente en lenguaje ensamblador o en lenguaje C. En este caso se utilizó lenguaje C ya que es un poco más versátil y más fácil de depurar. Por el lado de la aplicación en sí se optó por archivos de datos con formato DBASE. Esto fue debido a los requerimientos de las empresas donde se instaló el sistema y al hecho de que este formato está muy difundido y es fácil de mantener. Para la codificación de la aplicación se optó por el lenguaje Clipper. Éste tiene la ventaja de ser compilado (a diferencia de ser interpretado o pseudo-compilado como otros), lo cual aumenta el rendimiento al disminuir considerablemente los tiempos de ejecución. Como Clipper cuenta con un API para incorporar funciones y procedimientos escritos por el usuario en lenguaje C, estas funciones pueden incorporarse directamente dentro del código y ser llamadas desde la aplicación como cualquier otra función nativa de Clipper, basta compilar el módulo C y generar un OBJ, el cual será encadenado o ligado[10] junto con los OBJ's de la aplicación para generar el ejecutable final.

[10] *linked*

Para este propósito se diseñó una pequeña rutina que utiliza un buffer circular de 4 kb de longitud, el cuál será suficiente para esta aplicación. Debido a la velocidad requerida de transmisión y recepción y para asegurarse de que no se pierda ninguno de los caracteres recibidos, se programó bajo el esquema de manejador de interrupciones del CPU[11], colgándose la rutina a la cadena de interrupciones del sistema. Para este efecto el programa toma una dirección no asignada y la sustituye por un vector que apunta a esta rutina, al terminar la rutina, el vector original (sin asignar) es regresado a la dirección utilizada.

Para hacer más flexible esta rutina, se dividió en varias funciones :

| | Nombre función | Descripción |
|---|---|---|
| 1 | rs232up() | Abrir el puerto serial, asignando los parámetros deseados tales como: velocidad de transmisión (baud rate), número de bits, número de bits de parada y tipo de paridad. |
| 2 | CommInt() | Sustituye vector de interrupción por uno que apunta a la dirección de commISR() |
| 3 | commISR() | Rutina de captura de caracteres del UART-8250[12] al buffer, controlada por interrupciones. |
| 4 | xmit() | Transmitir un carácter |
| 5 | rdchr() | Recibir un carácter (es decir, leerlo del buffer circular). |
| 6 | crdy() | Devuelve TRUE si hay un carácter pendiente de lectura |
| 7 | clrcomm() | Regresa vector de interrupción a valor original y cierra el puerto serial |
| 8 | calclrc() | Calcula el LRC de una cadena |

**Tabla 5 - Funciones en C**

[11] *Interrupt handler*
[12] *Universal Asynchronous Recieve-Transmit* : chip controlador del puerto serial RS232 de las PC's

Código en lenguajo C de las rutinas necesarias para transmisión y recepción do datos a través del puerto serial, descritas arriba:

```
#include "c:\clipper5\include\extend.h"
#include <dos.h>

#define UART_THR        0x00
#define UART_RBR        0x00
#define UART_IER        0x01
#define UART_IIR        0x02
#define UART_LCR        0x03
#define UART_MCR        0x04
#define UART_LSR        0x05
#define UART_MSR        0x06
#define DTARDY   0x01
#define RDR      0
#define CTS      0x10
#define MASK     0x7f
#define IMR      0x21
#define THRRDY   0x20

#define MAXBUF   4095    /* Longitud máxima del buffer */
#define RS232    0x14
#define B9600    0xe0
#define MARK     0x28
#define ODD      0x08
#define EVEN     0x18
#define WORD7    0x02
#define STOP2    0x04

void interrupt (*isave) ();
void interrupt commISR ();

byte buffer [MAXBUF];
byte over;
unsigned int buffer_in,buffer_out,irq;
unsigned int COMBASE;

CLIPPER rs232up ()
{
   union REGS regs;
   unsigned char setup;
   unsigned char puerto;

   puerto=_parni (1);

   switch (puerto) {
     case 1 : COMBASE=0x3f8;
              break;
     case 2 : COMBASE=0x2f8;
              break;
```

```
      default : COMBASE=0;
   }

   setup=0;
   setup|=B9600;
   setup|=WORD7;
   setup|=ODD;
   setup|=STOP2;

   regs.x.dx=puerto-1;    /* com port 0=COM1; 1=COM2 etc. */
   regs.x.ax=setup;
   int86 (RS232,&regs,&regs);
   _storni (regs.h.ah,2);  /* port status */
   _storni (regs.h.al,3);  /* modem status */
}

CLIPPER CommInt ()
{
   int i,m;

   if (COMBASE==0) {
      _retni (-1);
      return;
   }

   for (i=0;i<=MAXBUF;buffer [i++]=0);  /* inicializa el buffer de rec. */

   irq=(COMBASE >> 8) +1;
   if (inportb (UART_IIR+COMBASE) & 0x00F8) {
      _retni (irq);
      return;
   }
   buffer_in=0; buffer_out=0; over=0;

   isave=getvect (irq+8);   /* guarda vector original */
   setvect (irq+8,commISR); /* el vector ahora apunta a nuestra función */
   disable ();
   outportb (UART_LCR+COMBASE,inportb (UART_LCR+COMBASE) & MASK);
   i=inportb (UART_LSR+COMBASE);
   i=inportb (UART_RBR+COMBASE);
   i=inportb (IMR);
   m=(1 << irq) ^ 0x00ff;
   outportb (IMR,i & m);              /* habilita irq en el 8250 */
   outportb (COMBASE+UART_IER,0x01);  /* habilita Data Ready Int */
   i=inportb (COMBASE+UART_MCR);      /* habilita OUT2 en el 8250 */
   outportb (COMBASE+UART_MCR,i | 0x08);
   enable ();
   _retni (0);
}

void interrupt commISR ()
{
   enable ();
```

```
   /* almacena carácter en el buffer */
   buffer [buffer_in]=inportb (COMBASE+UART_RBR);
   if (++buffer_in > MAXBUF) buffer_in=0;
   disable ();
   outportb (0x20,0x20);    /* emite un EOI */
}


CLIPPER rdchr ()
{
   byte c;

   while (buffer_in==buffer_out) {}
   c=buffer [buffer_out];
   disable ();
   if (++buffer_out > MAXBUF) buffer_out=0;
   enable ();
   _retc (&c);
}


CLIPPER crdy ()
{
   _retl ((buffer_in==buffer_out) ? FALSE : TRUE);
}


CLIPPER circomm ()
{
   int i,m;

   disable ();
   i=inportb (IMR);
   m=1 << irq;
   outportb (IMR,i | m);
   outportb (UART_IER+COMBASE,0);
   outportb (UART_MCR+COMBASE,0);
   enable ();
   /* regresa al IRQ su vector original */
   setvect (irq+8,isave);
}


CLIPPER xmit ()
{
   int cnt;
   byte ch;

   ch = _parni (1);
   cnt=0;
/* Espera a que esté vacio el Transmit Hold Register */
   while (!(inportb (COMBASE+UART_LSR) & THRRDY) && cnt < 10000)
cnt++;
```

```
   if (cnt>=10000) {
      _retni (-1);
      return;
   }

 /* transmite el carácter */
   disable ();
   outportb (COMBASE+UART_THR,ch);
   enable ();
   _retni (0);
}

CLIPPER calclrc ()
{
   char *cmd;
   unsigned char i, x, l;

   cmd=_parc (1);
   l=_parclen (1);
   x=0;
   for (i=0;i < l;x^= cmd [i++]);
   x|= 0x40;
   _retni (x);
}
```

Para poder contar con un OBJ utilizable en Clipper, este segmento de código se debe compilar con el compilador C utilizando el modelo grande (*large model*) e indicarle que no se desea generar un EXE.

Con este conjunto de funciones escritas en C, liberamos a la aplicación de la labor de verificar a cada momento el estado del puerto serial, y podemos escribir funciones de alto nivel para hacer la interfase propiamente con la terminal de código de barras.

Ya que con frecuencia se hará referencia dentro del código a caracteres de control específicos de la terminal referida, es conveniente crear definiciones para ellos de manera que se les pueda referir por una clave

*mnemonica*[13], en lugar de su valor decimal o hexadecimal. Tomando esto en cuenta, se creará el siguiente archivo de definiciones:

```
#define NUL   0
#define SOH   1       // Start of Header
#define STX   2       // Start of TeXt
#define ETX   3       // End Of Text
#define EOT   4       // End of Transmission
#define ENQ   5       // Enquiry
#define ACK   6       // Acknowledge
#define BEL   7       // Bell
#define LF   10       // Line Feed
#define CR   13       // Carriage Return
#define NAK  21       // Not Acknowledged
#define EM   25       // End of Message
#define ADD1 48       // Address 1
#define POLL 80       // Poll character
#define  F_RS  chr(30)     // Record separator
#define  F_GS  chr(29)     // Group separator
#define  F_US  chr(31)     // Unit separator
```

Usando estas definiciones, se escribieron un par de funciones en Clipper que realizan la comunicación con la DCU apegándose al protocolo de comunicación descrito con anterioridad. Estas funciones llamadas "PollReader" y "ETXwait", realizan este proceso, basándose en parte en las rutinas escritas anteriormente en C:

```
Static Function PollReader (UAC)
Local TmpStr:='',res,ctr:=0,c,done:=.f.,t,Lt,LRC
Local UBUAC:=UAC+1
   xmit (EOT)
   xmit (ADD1)
   xmit (ADD1+UAC) // ADD2
   xmit (POLL)
   xmit (ETX)
   cdelay (100)  // milisegundos
   while !done .and. ctr <= MyWait
      if crdy ()
```

[13] *mnemonic code*: abreviatura del nombre de una función que facilita su memorización.

```
            c=Left (rdchr (),1)
            t=asc (c)
            if t=ETX
               done=.t.
            else
               TmpStr+=c
            endif
         else
            ctr++
         endif
      enddo
      if !done
         SetColor ('w+/r')
         @ MaxRow(),(UAC)*16+12 SAY 'OUT'
      endif
      Lt=Len (TmpStr)
      if Lt > 0
         LRC=chr (calcLRC (Left (TmpStr,Lt-1)))
         if Right (TmpStr,1)==LRC
            xmit (ACK)
            xmit (ETX)
            cdelay (100) // milisegundos
            ETXwait (UAC)
            SetColor ('w+/g')
            @ MaxRow(),(UAC-1)*16+12 SAY '   '
         else
            TmpStr=''
            SetColor ('w+/r')
            @ MaxRow(),(UAC-1)*16+12 SAY 'NAK'
            Tone (2000,1)
            xmit (NAK)
            xmit (ETX)
            cdelay (100)  // milisegundos
            ETXwait (UAC)
         endif
      endif
   return TmpStr


   /****
   *  ETXwait
   *     -> Espera hasta recibir un carácter ETX por el puerto serial
   *        este cáracter indica el fin de la transmisión.
   */
   Static Procedure ETXwait (UAC)
   local done:=.f.,c,sTime:=Seconds ()
      while (Seconds ()-sTime) < 1 .and. !done
         if crdy()
            c=Left (rdchr(),1)
            if Asc (c)=ETX
               done=.t.
            endif
         endif
      enddo
```

```
  * Si pasó mas de 1 seg. y no responde el lector, manda mensaje
de TIMEOUT
  * y aumenta la pausa al lector que no está respondiendo.
    if (Seconds ()-sTime) > 1 .and. !done
        @ MaxRow(),(UAC)*16+12 SAY 'OUT' color 'w+/r'
        ++Pausa [UBUAC]
    endif
return
```

Debido a que los registros que provienen del lector tienen un formato especifico como se mencionó en la página 21, sería conveniente escribir una función que efectúe la segmentación[14] de la cadena de entrada y la separe en sus componentes individuales. Como los componentes son de longitud variable, la función debe tomar como referencia las marcas delimitadoras de los campos para efectuar la segmentación. La función "Parse" toma como parámetro a "t" que es la cadena de entrada recibida de la terminal y extrae de ésta los componentes relevantes como: Número de la terminal de origen, hora de la lectura, fecha de la lectura, así como el número del prompt del que proviene la lectura. Si la lectura no proviene de un prompt, significa que el mensaje fue generado autónomamente por la terminal; éste puede ser el caso de que la terminal haya sido abierta o que su batería interna fuera requerida por falla de la energía eléctrica externa. Durante el proceso de segmentación, el valor de la fecha es convertido de tipo carácter a una variable de tipo "Date", así mismo la fecha es convertida del formato *hhmmss* al formato *hh:mm:ss*

```
Static Procedure Parse (t)
Local Com,Usp,Tim3,Aut
```

[14] *Parse*

```
   Nomb:=Tmov:=OriL:=Data:=GMaes:=''
   if ''=1
      Pnn=0
      return
   endif
 * Localiza marcas delimitadoras
   Com=At ('\J',1)+2
   Iqr=At (F_GS,1)
   Pnn=At ('P00',1)+4
   Edt=At (F_RS,1)
   Usp=At (F_US,1)
 * Extrae la info. segun marcas
   Term_Addr=Subs (1,Com,Iqr-Com)
   TimS=Subs (1,Usp+1)
   Fech=Left (TimS,6)
   Fech=CtoD (Right (Fech,2)+'-'+Subs (Fech,3,2)+'-'+Left (Fech,2))
   HorL=Stuff (Subs (TimS,7,At (F_RS,TimS)-7),3,0,':')
   HorL=Stuff (HorL,6,0,':')
   Nomb=''
   if Pnn=4  // La info. no viene de un prompt
       Data=Subs (1,Iqr+1,Edt-1-Iqr)
       do case
          case Data='AA00'
             Nomb='Utilizando bater¡a propia'
          case Data='AA01'
             Nomb='Corriente reestablecida'
          case Data='AA02'
             Nomb='Bater¡a agotada'
          case Data='AA10'
             Nomb='Lector abierto/cerrado'
          case Data='AG00'
             Nomb='Arranque en fr¡o'
             DownLoad (Subs (1,2,1),Subs (1,3,1),Term_Addr)
          other
             Nomb='Terminal control info.'
       endcase
    else
       Aut=At ('BP009G',1)+6
       if Aut > 6
          GMaes=Subs (1,Aut,6)
          Data=Subs (1,At (F_RS+'T',1)+2,6)
          TMov=Subs (1,At (F_RS+'M',1)+2,1)
       else
          Data=Subs (1,Pnn,Edt-Pnn)
          Tmov=Left (Data,1)
          Data=Subs (Data,2)
       endif
    endif
return
```

#### *2.5.1.2 Parámetros Generales*

Dentro de las empresas analizadas se detectaron varios parámetros que eran susceptibles de sacarlos del código, por este motivo se decidió tomar

estas variables y dejarlas accesibles al usuario para que las pudiera modificar en un futuro de ser necesario, aumentando con esto la flexibilidad del programa.

Algunos de los parámetros a los que se refiere el párrafo anterior son:

| Parámetro | Descripción |
|---|---|
| Constructor del código de barras | El código de barras del empleado puede estar formado sólo por su número de nómina, o por un campo adicional, como el número de departamento. |
| Tiempos de gracia de inicio del tiempo extra. | Determina cuántos minutos antes o después del turno deben transcurrir para que ese tiempo se considere como extra |
| Redondeo del tiempo extra. | En algunas empresas el tiempo extra se redondea a múltiplos de 5, 10 ó 15 min. |
| Retardo | Determina cuántos minutos deben transcurrir después del inicio del turno, para que se considere como retardo. |
| Rango de terminales | El programa debe conocer las direcciones lógicas (UA[15]) de la primera y última terminal en la red. |
| Tiempo entre checadas. | Determina cuántos minutos deben transcurrir para que dos checadas del mismo tipo (entradas o salidas) se consideren como duplicadas. |
| Puerto serial | Indica qué número de puerto serial debe utilizar el programa para comunicarse con las terminales. |
| Longitud del código de barras | El programa espera recibir y enviar códigos de barra de esta longitud. |
| Nombre de la compañía. | |

**Tabla 6-Parámetros generales**

[15] *Unit Address*

Para facilitar el acceso dentro del programa a todos estos parámetros, se utilizará un arreglo que contenga los valores de dichos parámetros. Éste arreglo tiene la siguiente estructura:

```
Public BC_Info:= { {NIL,'0.00',-1,NIL},;
                    NIL,;
                   {NIL,NIL,NIL,NIL},;
                    NIL,;
                    NIL,;
                   {NIL,NIL},;
                    NIL,;
                   {NIL,NIL},;
                    NIL,NIL,NIL,NIL,NIL,NIL,NIL,;
                   {NIL,NIL},;
                    NIL;
                 }
/* estructura de BC_Info :
 no. elem  Contenido
     1     {Serie,Version,Nivel,Digito-Codificador}
     2     Nombre cia
     3     {Clave1,clave2,clave3,clave4}
     4     Constructor (macro) del codigo de barras
     5     No. puerto serial
     6     {Min UAC, Max UAC} {1er. y ultima direccion de
lectores}
     7     Long. cod. barras
     8     {Tiempo de gracia inicio T. Extra (de la salida en
adelante),
            T.G.I.T.E. de la entrada para atras }
     9     Redondeo tiempo extra a x Min.
    10     Retardo despues de x Min.
    11     Factor septimo dia
    12     Factor descanso trabajado
    13     Tiempo minimo entre checadas repetidas
    14     Calculo por Horas o por Dias
    15     Paga solo Horas Extra autorizadas ?  <- boolean
    16     {H. Ini. Prima Noc.,H Fin Prima Noc.}
    17     Ruta a dbf's e indices
   (18)    Tiempo de gracia inicio T. Extra (antes de la entrada)
   (19)    Permite T. extra "flotante" <- boolean
*/
```

Para facilitar la lectura y depuración del código, se utilizará el siguiente conjunto de definiciones, de tal manera que no sea necesario recordar las posiciones de los elementos dentro del arreglo:

```
*******
* Definiciones globales:
*
#define BC_NO_SERIE 1,1
#define BC_NO_VERSN 1,2
#define BC_SECLEVEL 1,3
#define BC_DIGITCOD 1,4
#define BC_NOMBRE   2
#define BC_CLAVE1   3,1
#define BC_CLAVE2   3,2
#define BC_CLAVE3   3,3
#define BC_CLAVE4   3,4
#define BC_CONSTR   4
#define BC_RS232C   5
#define BC_MINUAC   6,1
#define BC_MAXUAC   6,2
#define BC_CODBARL  7
#define BC_TGITE    8,1
#define BC_TGITA    8,2
#define BC_REDTE    9
#define BC_RETDESP  10
#define BC_FSEPTD   11
#define BC_FDESTR   12
#define BC_MINREP   13
#define BC_HOD      14
#define BC_AUTHE    15
#define BC_INIPNOC  16,1
#define BC_FINPNOC  16,2
#define BC_PATH     17
#define BC_TGITEA   18
#define BC_OKTEFLOT 19
```

Este archivo de definiciones puede guardarse en un solo archivo junto con las definiciones mencionadas antes. Este archivo es llamado por el compilador utilizando una instrucción `#include` dentro del código.

### *2.5.1.3 Captura de Datos*

Una vez definido el proceso de intercomunicación entre los dos equipos, contando con las herramientas necesarias, se puede escribir en pseudo-código el proceso de captura de los datos de entradas y salidas:

-› Abrir puerto de comunicación
-› (☆)Envía solicitud de registro
-› Recibe y decodifica paquete
-› Calcula LRC
-› si LRC correcto envia ACK
    -› si no, envía NAK y vuelve a (☆)
-› Si la lectura es válida, graba datos en registro nuevo
-› Vuelve a (☆)

es decir:

```
// Abre puerto de comunicación
Sts=RS232up (BC_INFO [BC_RS232C])
// Instala controlador de interrupts
res=CommInt ()
if res # 0
   Avisa ('Error al tratar de instalar programa de comunicación',;
          ErroClr)
   return
endif

// Lee los registros de la terminal hasta que el usuario cancele
while .t.
   xKey=Inkey ()
   if xKey==K_ESC
      exit
   endif

 * Lee datos de la terminal
   Line=PollReader (BC_INFO [BC_MINUAC])
   Parse (Line)
   if Len (Line) > 0 .and. !Empty (Fech)
    * Despliega informacion
      SetColor (NormClr)
      do case
         case Tmov=='E'
            nIn++
            @ nReng,59 say 'Entrada'
         case Tmov=='6'
            nIn++
            @ nReng,59 say 'Entrada Aut/'+GMaes
         case TMov=='7'
            nOut++
            @ nReng,59 say 'Salida Aut/'+GMaes
         case TMov=='S'
            nOut++
            @ nReng,59 say 'Salida'
      endcase

      lEntr=(Tmov $ 'E6')    // App. specific !!!

      @ nReng, 1 say Data
      @ nReng,13 say Nomb
```

```
@ nReng,39 say HorL
@ nReng,49 say Fech
@ nReng,77 say Term_Addr

if Pun > 4
   ErrCode=Valida (Data)
   do case
      case ErrCode=0
         sele Trab01
         @ nReng,13 say Nombre
         nCheck++
         Turn=Turno

         sele Trab01
         while .t.
            if Rlock ()
               exit
            endif
            @ MaxRow(),70 say 'Espere' color ErroClr
         enddo
         En_Planta:=lEntr
         unlock

         sele InOut
         append blank
         repl Cod_Barra with Data,Fecha with Fech,Hora with HorL,;
              Hora_Dec with Str2Dec(HorL),Entrada with lEntr,;
              Turno with Turn,Tipo with Tmov,Autorizado with GMaes
         unlock

      case ErrCode=-1  // invalido
         @ nReng,59 say 'INVALIDO !'
      case ErrCode=-2
         @ nReng,13 say 'NO EXISTE' color ErroClr

      case ErrCode=-3
         @ nReng,59 say 'BAJA !' color AtenClr

      case ErrCode=-4
         @ nReng,67 say ''dupl'' Color ErroClr

      case ErrCode=-5
         // vacio, no hacer nada
   endcase

   if ErrCode # -5
    * Muestra estadisticas
      SetColor (AtenClr)
      @ MaxRow()-3,1 say 'Total registrados :'+Str (nCheck,5)
      @ MaxRow()-3,col()+2 say 'Entradas :'+Str (nIn,5)
      @ MaxRow()-3,col()+2 say 'Salidas :'+Str (nOut,5)
      SetColor (NormClr)
   endif  // errcode # -5
endif // mov. de trab.

if ++nReng > MaxRow()-5
   Scroll (6,1,MaxRow()-5,MaxCol()-1,1)
   RestScreen (MaxRow()-5,1,MaxRow()-5,MaxCol(),cLin)
   nReng=MaxRow()-5
```

```
      endif
   endif  // Parse (Line)
enddo
SetColor (NormClr)
ClrComm ()
SetCursor (1)
RestScreen (1,0,10,79,Scrn)

*-*-*-*-* (MAIN)

/*****
* Str2DecHr (cadena)
*   -> Convierte una hora en formato de cadena "HH:MM"
*      a formato decimal H.MM
*/
Function Str2DecHr (sHour)
return Round (Val (sHour)+Val (Subs (sHour,4))/60,3)

*-*-*-*-*

Function Valida (nt)
  * vacio ?
    if ''==nt
       return -5
    endif

  * revisa que sea la long correcta
    if Len (nt) # 6  // 6 es la longitud legal del codigo de barras
       return -1
    endif

   * revisa que el trabajador exista
    sele Trab01
    seek Val (nt)
    if !found ()
       return -2
    endif

   * si existe, dado de baja ?
    if Status='B'
       return -3
    endif

   * Revisa re-checado, tiempo predeterminado= archivo config.
    sele InOut
    seek Dtos (Fech)+Data
    if found ()
       while Fecha=Fech .and. Data=Trim (Cod_Barra)
         * checo en la misma fecha, revisar la hora
         if Secs (ElapTime (Hora,Hor1)) <= BC_Info [BC_MINREP]*60*60 ;
            .and. Tipo=Tmov // si es menor del t. de rechecado y es el mismo
mov.
            return -4
         endif
         skip
       enddo
    endif
return 0
```

El código anterior lleva a cabo las funciones básicas de extracción, validación y almacenamiento de la información proveniente de la terminal. Como se puede observar, se incluyeron dos funciones de apoyo "Valida" y "Str2Decim". La primera toma como parámetro el número de gafete leído y lo valida contra la base de datos de trabajadores, devolviendo un valor numérico de acuerdo al resultado de la validación. La segunda función, toma como parámetro una cadena que representa una hora de registro y la convierte en una variable tipo real (numérica) que contiene la hora en formato decimal. Este dato es almacenado en el archivo de registro de movimientos y es sumamente importante ya que en base a éste se harán los cálculos de tiempo trabajado y tiempo extra.

En caso de pasar toda la validación, los datos provenientes de la terminal son almacenados en el archivo de registro descrito en la pág. 23. Este archivo se encuentra indexado por 3 llaves que se actualizan simultáneamente al momento de grabar cada registro. Estas tres llaves permitirán recorrer el archivo en órdenes diferentes, dependiendo del proceso que se desee efectuar, siendo el más importante el índice con la llave:

```
Dtos (Fecha) + Cod_Barra
```

este índice ordena a los registros en orden ascendente de fechas y a su vez en orden ascendente de número de gafete dentro de cada fecha, de

modo que todas las checadas de la misma persona en una fecha dada, se encuentran en registros lógicos contiguos. La función "Dtos (Fecha)" convierte al campo "Fecha" de una variable *Date* a una cadena alfanumérica con formato *YYYYMMDD* de modo que por ejemplo, la fecha "01-05-95" se convierte en "19950501".

Tener todas las checadas de una misma persona en registros lógicos consecutivos es sumamente útil en el procesamiento sistemático de la información como se podrá ver en la siguiente sección.

### 2.5.2 Módulo de Procesamiento

El módulo de procesamiento es la parte más importante del sistema ya que en él se llevan a cabo todos los cálculos referentes a tiempos trabajados y extras. Es de suma importancia que su diseño sea lo más confiable y robusto posible para que los demás módulos cuenten con información confiable y expedita.

Uno de los datos más importantes para el cálculo del tiempo trabajado, es el de los horarios oficiales de trabajo. Se utiliza el término "*horarios oficiales*" para diferenciarlo del horario de trabajo, que es el horario real en el que la persona checó de entrada y salida. Para facilitar el acceso a esta información dentro del sistema se creó el archivo de turnos, el cual tiene la siguiente estructura:

| Dato | Tipo | Longitud | Observaciones |
|---|---|---|---|
| No. de horario | Numérico | 3 | Hasta 999 horarios |
| Entrada1 | Carácter | 5 | Hora de entrada "HH:MM", día 1 |
| Salida1 | Carácter | 5 | Hora de Salida "HH:MM", día 1 |
| ... | | | ... |
| Entrada7 | Carácter | 5 | Hora de entrada "HH:MM", día 7 |
| Salida7 | Carácter | 5 | Hora de Salida "HH:MM", día 7 |

**Tabla 7-Estructura del archivo de horarios**

En la estructura anterior, cada registro corresponderá a un horario completo y contendrá la información de una semana completa con sus horas de entrada y salida.

#### *2.5.2.1 Cálculo de tiempos*

Durante el análisis se buscó el mejor método para calcular tiempos transcurridos entre dos eventos. El tiempo que transcurre entre dos eventos y su comparación contra un patrón predeterminado (i.e. las horas de checadas y las horas de entrada y salida oficiales del turno) es un proceso relativamente simple e intuitivo para una persona, pero representa un problema un poco más complejo para un programa de computadora. Esto es especialmente notorio cuando las horas de entrada y salida pueden darse a cualquier hora del día y cuando pueden no tener nada que ver con lo que indica el horario "oficial" de la persona en cuestión. Estas

características aumentan considerablemente el número de toma de decisiones que debo llevar a cabo el programa durante su ejecución.

Tomemos el siguiente ejemplo para ilustrar este punto:

**Horario "oficial" de la persona:**

| *Dom* | *Lun* | *Mar* | *Mie* | *Jue* | *Vie* | *Sab* |
|---|---|---|---|---|---|---|
| -*- | 06:45 (e) | 06:45 (e) | 06:45 (e) | 16:00 (e) | 07:45 (e) | 09:00 (e) |
| -*- | 15:45 (s) | 15:45 (s) | 15:45 (s) | 23:15 (s) | 16:30 (s) | 14:00 (s) |

(e)=Entrada (s)=Salida

**Horas reales de checado:**

| *Dom* | *Lun* | *Mar* | *Mie* | *Jue* | *Vie* | *Sab* |
|---|---|---|---|---|---|---|
| -*- | 06:39 (e) | 06:45 (e) | 06:50 (e) | 13:21 (e) | 00:19 (s) | 08:55 (e) |
| -*- | 15:45 (s) | 15:40 (s) | 15:50 (s) | | 07:40 (e) | 13:55 (s) |
| | | | | | 16:31 (s) | |

(e)=Entrada (s)=Salida

Como se puede observar de los datos anteriores, existen varios patrones identificables de los datos anteriores, dentro de los cuales caerán el 100% de las checadas de los trabajadores. Este proceso de identificación representa el primer paso para la solución del cálculo del tiempo trabajado. Estos patrones se pueden resumir de la siguiente manera y cada trabajador puede tener uno o más de ellos cada día:

1. Tiempo extra por entrada antes de la hora oficial
2. Tiempo extra por salida después de la hora oficial
3. Retardo por entrada después de la hora oficial
4. Descuento de tiempo por salida antes de la hora oficial

Además de estos cuatro casos básicos se observó que existe un quinto caso que se da más esporádicamente. Éste consiste en pares de checadas entrada/salida que no están ligados o que no tienen correspondencia con ningún horario oficial. A este se le denominó "tiempo extra flotante". Un ejemplo de este caso tomando como referencia el horario oficial descrito anteriormente sería si el trabajador hubiera realizado las siguiente checadas el día miércoles:

| *Mie* |
|---|
| 06:25 (e) |
| 15:50 (s) |
| 19:00(e*) |
| 20:30 (s*) |

En este caso el tiempo trabajado entre las 19:00 y las 20:30 hrs. no está ligado a ningún horario, ya que el horario oficial para este turno en cuestión y para ese día en específico es exclusivamente de 06:45 a 15:45. A esas 1.5 hrs. se les denominó tiempo extra flotante. En el análisis que se hizo de las empresas, se observó que mientras 2 de ellas permitían ese tipo de movimientos las otras 3 no lo hacían, salvo en casos específicos, requiriendo autorización para su pago. Por este motivo se decidió incluir la opción de pago automático de tiempo extra flotante, dentro de los parámetros generales del sistema de modo que fuera definible por el usuario.

*2.5.2.2 Procedimientos de Cálculo*

Si se observan solamente los tres primeros días de las checadas reales del trabajador, el procedimiento para determinar y calcular los 5 puntos mencionados arriba, es bastante simple. Se restan las horas en formato decimal de la entrada real y oficial, si el número es positivo existe un retardo, si es negativo, existe un posible tiempo extra[16]. El mismo procedimiento se efectúa para la hora de salida.

| | Lunes | | Martes | | Miércoles | |
|---|---|---|---|---|---|---|
| | Hora de entrada | Hora de salida | Hora de entrada | Hora de salida | Hora de entrada | Hora de salida |
| Real | 6.65 | 15.75 | 6.75 | 15.667 | 6.833 | 15.833 |
| Oficial | -6.75 | 15.75 | 6.75 | 15.750 | 6.750 | 15.750 |
| Tol. decimal | -0.10 | 0.00 | 0.00 | 0.083 | 0.083 | 0.083 |
| Total en minutos: | -0:06 | 0:00 | 0:00 | -0:05 | 0:05 | 0:05 |

Como se puede ver en la tabla, el trabajador incurrió en los 4 posibles casos de checado en el periodo de lunes a miércoles.

Para este caso, podríamos escribir el procedimiento en pseudo-código de la siguiente manera:

→ Abrir archivo indexado
→ Pregunta día inicial para los cálculos, almacena en *d*
→ ① Busca en el día *d* la primera entrada
→ Si no se encontró
  *d*←*d*+1, vuelve a ①
→ *her*←Hora de entrada real (decimal)
→ *heo*←Hora oficial de entrada para *d* del archivo de turnos (decimal)

[16] La empresa puede determinar si el tiempo que exista antes de la hora de entrada oficial es extra o se descarta.

→ $t \leftarrow (hrr\text{-}hro)$ // calcula tiempo en la entrada
→ si $t > 0$ es un retardo
→ si $t < 0$ es tiempo extra
→ $d \leftarrow d+1$, vuelve a ①

En el procedimiento arriba descrito se procesan las entradas comenzando en una fecha $d$ determinada por el usuario. El algoritmo simplemente busca la primera entrada del día y la compara contra el patrón (el horario oficial). Este paso es importante ya que si se tomara como inicial el primer movimiento del día, éste podría ser una salida. En el caso anterior, la simpleza del algoritmo se debe a que se presupone que ambas horas corresponden al mismo día. Como esto en la realidad no es una constante, el algoritmo debe modificarse para contemplar estos casos. Aunque primero se debe completar el algoritmo para procesar el turno completo, es decir entradas y salidas:

→ Abrir archivo indexado
→ Pregunta día inicial para los cálculos, almacena en $d$
→ ① $dc \leftarrow d$
→ $tdif \leftarrow 0$
→ ② Busca en el día $dc$ la primera entrada
→ Si no se encontró el movimiento deseado en $dc$
$dc \leftarrow dc+1$, vuelve a ②
→ $hrr \leftarrow$ Hora de registro real (decimal)
→ Si se trata de una entrada entonces
$hi \leftarrow hrr$
→ si no
$hf \leftarrow hrr$
→ $hro \leftarrow$ Hora oficial de registro para $dc$ del archivo de turnos (decimal)
→ $t \leftarrow (hrr\text{-}hro)$ // calcula tiempos faltantes o sobrantes
→ si el tipo de movimiento es una salida
$t \leftarrow -t$

*tdif* ← *tdif* + (*ht*-*hi*)
→ si *t* > 0 es un retardo o salida anticipada, en su caso.
→ si *t* < 0 es tiempo extra
→ Salta al siguiente registro (de entrada a salida)
→ si cambió el número de trabajador
vuelve a ①
→ vuelve a ②

Nótese que en el procedimiento anterior, se está calculando, además de los retardos (o faltantes) y tiempos extras, el tiempo trabajado. El algoritmo descrito arriba es más flexible ya que permite cualquier número de entradas y salidas durante el día. El programa recorre todos los pares de entradas y salidas y calcula el tiempo neto trabajado, almacenando este dato en la variable *tdif*.

Como se puede ver, el procedimiento fallará en cuanto cualquier salida corresponda al día siguiente, ya que la hora será menor que la de la entrada, dando la resta un resultado negativo. Peor aún, si la salida corresponde a otro día, entonces el dato de la hora de salida que fue leído del archivo de turnos no es correcto, o es ambiguo, en el mejor de los casos.

Para solucionar esto, se diseñó un método diferente de calcular el tiempo transcurrido entre dos eventos, con lo cual se eliminó la necesidad de tener que revisar si la hora corresponde a otro día. A este método se le denominó **horario lineal absoluto** y facilita enormemente el proceso de cálculo.

#### 2.5.2.3 *Horario Lineal Absoluto*

Este método consiste simplemente en eliminar la fecha de los horarios o mejor dicho incorporarlos dentro de la misma hora. Para convertir cualquier par fecha/hora al horario lineal absoluto, se aplica la siguiente fórmula al par.

$$hla = \left( \left( F_r - F_i \right) \cdot 24 + hd \right) \cdot 3600$$

donde:

$F_r$ es la fecha inicial o fecha de referencia
$F_i$ es la fecha para la hora que se desea calcular
$hd$ es la hora (expresada en decimal) que se desea convertir

El resultado $hla$, estará expresado en segundos y se puede definir como el número de segundos que han transcurrido desde el segundo cero de $F_r$ hasta la hora que se está calculando en la fecha $F_i$.

La siguiente función de Clipper toma como parámetros la fecha de referencia, la fecha y la hora a convertir y devuelve un número $hla$.

```
/**********
*
*  Function AbsTime (FechaRef,FechaAct,Hora) -> nSegundos
*
*    Convierte una hora dada a una cantidad absoluta de segundos;
*    donde el segundo 0 es el primer segundo contado desde FechaRef
*/
Static Function AbsTime (dF,dA,cHora)
Local tmp
   tmp=Secs (cHora+':00') + (dA-dF)*24*60*60
return tmp
```

Utilizando este método se puede almacenar en memoria todo el periodo que se desea calcular ya sea semanal o quincenalmente. Este arreglo

contendría toda la información respecto al patrón (horario oficial) y será mucho más sencillo compararlo contra los registros reales.

Para poder hacer comparaciones rápidas en memoria, de la hora de registro del trabajador contra el horario oficial, se debe crear un arreglo que le sea fácil al programa recorrer y que al mismo tiempo contenga la información relevante.

```
/****
*   Crea arreglo que contenga toda una semana de horas de entrada
*   y salida en tiempos absolutos por turno.
*   El primer elemento es cero para checadas antes de la 1er hora
*   del 1er dia.
*/
use turnos new readonly shared
SemaTur:=Array (LastRec(),(ff-fi+2)*2+1)  // 1 semana mas 1 dia max.

while !eof()
   SemaTur [Recno(),1]=0  // 1er elem. 0 como referencia
   j=2
   for r=fi to ff+1
    * calcula el no. de campo a leer  (1=dom, 2=lun, etc.)
      cmp := dow (r) % 7     // % = módulo
      if cmp==0  // controla "wrap-around"
         cmp:=7
      endif
    * ofs controla turnos que entran en un dia y salen en otro
      ofs=if (FieldGet (cmp*2) < FieldGet (cmp*2-1),1,0)  // salida <
entrada ?
      SemaTur [Recno(),j++] = AbsTime (fi,r    ,FieldGet (cmp*2-1))
      SemaTur [Recno(),j++] = AbsTime (fi,r+ofs,FieldGet (cmp*2  1)
   next r
   skip  // sig. turno
enddo
```

La función anterior crea una matriz que contiene n renglones (donde n es el número de turnos definidos por la empresa) por m columnas (donde m=2*Número_de_días+1). Con esta disposición, las columnas nones contienen la hora de entrada y las columnas pares la hora de salida, ambas expresadas en formato *hla*.

De manera general se puede decir que el dato de la hora de entrada de un turno $T$, en el día de proceso $d$, se encuentra utilizando la expresión $SemaTur[T,d\cdot 2-1]$, mientras que los datos de salida se encuentran con la expresión $SemaTur[T,d\cdot 2]$.

Con las consideraciones hechas hasta este punto, se puede escribir una rutina en Clipper que lea las horas de entrada y salida del archivo de registro y efectúe el cálculo de un día en particular. En el siguiente segmento de código, la Cod_Barra representa el dato del código de barra del gafete del trabajador. Este campo es parte de la estructura del archivo mencionado.

```
cb=Cod_Barra
while cb=Cod_Barra .and. Fecha==r .and. !eof()
   if Tipo=='P'   // ignora permisos con goce
      skip
      loop
   endif
   if !Entrada
      AddObs (cb,r,'-E',.f.)
      skip
      loop
   endif
   hi=AbsTime (fi,fecha,hora)
   skip
   if Entrada .or. (Fecha > r+1) .or. cb != Cod_Barra .or. eof()
   // 2 entradas seguidas o salto 1 dia o cambio trab.
      AddObs (cb,r,'-S',.t.)
      loop
   endif
   // se brinca los permisos con goce
   locate for Tipo != 'P' while cb=Cod_Barra .and. (fecha <= r+1)

   // Coloca al archivo de turnos en su posicion correcta
   oDbf:=Select ()
   sele Turnos
   go turn
   Select (oDbf)
```

```
      cInPares
      hi=AbsTime (ti,fecha,hora)
      tDif = (hf-ht) / 3600     // a H.hh

    // Calcula si el punto medio entre la entrada y la salida
    // cae dentro del turno
    TipE=TipoMov (hi,turn)
    TipS=TipoMov (hf,turn)
    TipM=TipoMov ((hi+hf)/2,turn)
    // condicion especial para checado antes y salir despues
    if TipM=='F'
       if hi <= hEntr .and. hf > hSale
          TipM:='T'
       endif
    endif

    do case
       // Caso 1 : Todo dentro de turno (es t. normal)
       case TipE='T' .and. TipM='T' .and. TipS='T'
          hNorm += tDif
          hTarde += cRetardo (nPares,hi,hEntr)
       // Caso 2 : Todo fuera de turno (es t. extra)
       case TipE='F' .and. TipM='F' .and. TipS='F'
          hExtra += tDif

       // Caso 3 : Entro en turno salio fuera de turno
       case TipE='T' .and. TipS='F'
          hExtra += (hf - hSale) / 3600
          hNorm += (hSale - hi) / 3600
          hTarde += cRetardo (nPares,hi,hEntr)

       // Caso 4 : Entro antes, salio despues
       case TipE='F' .and. TipM='T' .and. TipS='F'
          hNorm += TurNorm
          hExtra += (hEntr - hi) / 3600  // t.e. anterior
          hExtra += (hf - hSale) / 3600  // t.e. posterior

       // Caso 5 : Entro antes, salio antes
       case TipE='F' .and. TipS='T'
          hExtra += (hEntr - hi) / 3600
          hNorm += (hf - hEntr) / 3600
    endcase
    skip
enddo
```

En este bloque de código se hace referencia a la función `TipoMov`, esta función acepta dos parámetros: el primero es una hora en formato *hh.a* y el segundo es el turno de la persona para ese día específico. La función simplemente devuelve el valor "T" si la hora a probar cae dentro del turno (entre la entrada y la salida oficial), de lo contrario devuelve "F". Como se

puedo ver en el código, se están tomando en cuenta los 5 casos posibles de combinaciones de checado, y se está calculando el tiempo normal y extra independientemente para cada uno de ellos.

Una vez procesados los datos en esta primera fase inicial, ya se pueden tomar otro tipo de consideraciones; por ejemplo: generalmente el día de descanso de la persona se toma todo como tiempo extra, en este caso, una parte posterior del programa verifica esta situación y en caso de cumplirse esta condición, simplemente se hace que *hn* += *hn*; *hn*=0, es decir se acumula al tiempo extra el tiempo normal y el normal se hace cero. En esta parte del programa se agrega cualquier otra condición específica que tenga la empresa en cuanto al manejo de los retardos o el tiempo extra.

Una vez calculados los tres principales datos (tiempo normal, tiempo extra y tiempo de retardo), la información se graba en un archivo, permitiendo que al emitir los diferentes reportes que se deseen, la toma de decisiones y la complejidad de los mismos se verá reducida significativamente, aumentando con esto el desempeño y asegurando la uniformidad de los cálculos.

### 2.5.3 Módulo de excepciones.

Para poder contar con información completa de todos los eventos que afectaron la asistencia e inasistencia del trabajador, y con esto poder emitir informes completos y detallados, el sistema requiere de la captura de las excepciones asociadas a cada trabajador.

Aunque existe un número de excepciones que todas las empresas tienen, el sistema se diseñó de tal manera que el usuario pudiera definir un número ilimitado de ellas, siendo las más comunes:

| |
|---|
| FALTA JUSTIFICADA |
| FALTA INJUSTIFICADA |
| INCAPACIDAD POR ACCIDENTE DE TRABAJO |
| INCAPACIDAD POR ENFERMEDAD O MATERNIDAD |
| DIA DE DESCANSO |
| CAMBIO DE HORARIO |
| VACACIONES |

**Tabla 8-Excepciones más usuales**

Para poder llevar un buen control de esta información se crearon dos archivos con la siguiente estructura:

| CATÁLOGO DE MOVIMIENTOS | |
|---|---|
| TIPO Y LONGITUD | NOMBRE DEL CAMPO |
| N, 3 | TIPO_MOV |
| C,20 | DESCRIPCION |
| N, 1 | TIPO_APLIC |
| L, 1 | BAJA |

| ARCHIVO DE MOVIMIENTOS | |
|---|---|
| TIPO Y LONGITUD | NOMBRE DEL CAMPO |
| N, 6 | NO_TRABAJADOR |
| N, 3 | TIPO_MOV |
| D, 8 | FECHA_INICIAL |
| D, 8 | FECHA_FINAL |
| L, 1 | BAJA |
| D, 8 | F_PAGA_VAC |
| N, 3 | NUEVO_TURNO |
| L, 1 | DESCONTAR |

**Tabla 9-Estructura de excepciones**

Como se puede observar, se añadieron algunos campos especiales al archivo de movimientos, éstos son:

DESCONTAR: Campo tipo lógico que indica si la excepción debe descontarse (tomarse como falta) o no.

NUEVO_TURNO: Se incorporó este campo para manejar la excepción "Cambio de turno" dentro del catálogo de movimientos. Con esta opción se le puede indicar al sistema el cambio de turno de un trabajador inclusive a mitad de la semana o de la quincena. Como el módulo de procesamiento lee el turno día por día, se pueden hacer tantos cambios de turno como se desee dentro del período de cálculo.

F_PAGA_VAC: Este campo tipo fecha, permite definir en qué semana o quincena se deben pagar las vacaciones (las cuales no siempre se pagan en el período que se están disfrutando).

Como se puede apreciar, hasta este punto el programa puede manejar una gran variedad de casos de checado, lo cuál lo hace prácticamente adaptable a cualquier empresa que cuente con cualquier esquema de turnos o prácticas de checado, éstas son:

- Capacidad de dar de alta un número ilimitado de trabajadores
- Capacidad de dar de alta un número ilimitado de turnos (horarios de checado).
- Capacidad de asignarle un turno fijo a una persona o especificar que la persona cambia de turno en una fecha determinada.
- Capacidad de definir un catálogo de excepciones.
- Capacidad de asignarle a una persona cualquier número de excepciones por período de pago.

# 3 Implementación

Durante la implementación del sistema se pudieron hacer varias observaciones importantes que inciden directamente sobre el desempeño de todo el sistema.

Por un lado, se encuentra la función de registro, por parte de los trabajadores, en la terminal de código de barras.

Como les fue informado a las empresas que instalaron el sistema, era de esperarse que durante los primeros días, la fila fuera un poco más larga de lo deseado, ya que el personal no tenía experiencia en este tipo de funciones y en algunos casos existía algo de "temor" hacia el nuevo sistema. Al paso de dos semanas como máximo, se observó que la curva de aprendizaje se estaba equilibrando y la fila comenzó a hacerse bastante más corta.

En promedio, de las observaciones hechas, se obtiene que el proceso de registro por persona es de 4 segundos. Este tiempo es medido a partir de que la persona anterior deja libre la terminal, hasta que el trabajador que está siendo medido, la deja libre al siguiente. El tiempo incluye los movimientos que debe realizar el trabajador para realizar el registro:

- Avanzar de la fila hasta la terminal
- Oprimir la tecla correspondiente a su movimiento (entrada o salida)
- Deslizar el gafete por la ranura
- Corroborar tanto visual como auditivamente que el movimiento fue registrado y aceptado.

Esta tasa de lecturas, da como resultado un promedio de 15 lecturas por minuto, lo cual quiere decir que en el periodo de 10 minutos antes de la hora del turno en el cual se registran el 95% de los trabajadores, se pueden atender 150 de ellos.

Debido a que la mayoría de las empresas analizadas cuentan con horarios escalonados, en la mayoría sólo se requirió de una terminal. Las únicas excepciones fueron cuando se esperaban más de 150 trabajadores registrándose para una hora determinada, o cuando la entrada de un turno coincidía con la salida de otro. En estos casos fue necesario instalar una terminal adicional para dar mayor fluidez al registro de los trabajadores.

Otro aspecto de preocupación para las empresas fue el hecho inevitable de que había trabajadores que olvidaban su gafete, y por esto, su forma de registrar su entrada y salida. En este aspecto se observó una gran variedad de políticas de manejo de este tipo de situaciones. La siguiente lista de acciones corresponde a políticas reales en caso de olvido del gafete:

- Ya que el gafete no sólo es el medio para registrar la entrada, sino la identificación del trabajador, la entrada es negada (y el día descontado, en su caso).
- El guardia de la entrada cuenta con una libreta donde se anotan las personas que lleguen sin gafete. Posteriormente, estos datos son vaciados en el sistema de forma manual.
- Los gafetes permanecen en tarjeteros (del tipo de las antiguas tarjetas checadoras), de tal manera que el trabajador la usa para registrarse y la vuelve a colocar en el tarjetero.

La opción sugerida por nosotros fue la creación de un "gafete maestro". Esta opción consiste en la generación de un gafete con un número especial almacenado en su código de barras. Debido a que la terminal utilizada tiene la capacidad para utilizar varias listas de validación, los números de "gafetes maestros" se almacenaban en una lista diferente a la de los números de trabajadores. Asimismo, se asignó una de las teclas programables de la terminal para implementar estos casos. Los gafetes maestros se distribuyeron a los diferentes supervisores de personal o de producción (dependiendo de la empresa) y ellos eran los responsables de registrar al trabajador que hubiera olvidado su gafete. Para realizar esta función, el supervisor oprimía una tecla predefinida en la terminal (distinta de las de entrada o salida) y ésta generaba una secuencia interactiva en la cuál se le solicitaba:

1. Deslizar el gafete maestro
2. En caso de ser válido, se solicita el número de trabajador.
3. El supervisor teclea manualmente el número de trabajador, y al oprimir ENTER, la terminal valida el dato contra su lista de trabajadores.
4. En caso de ser correcto el número de trabajador, la terminal solicita el tipo de movimiento (entrada o salida).
5. Enseguida, el movimiento se graba en la cola de salida de la terminal, con una marca para distinguirlo como un movimiento autorizado o registrado por un gafete maestro.

En el sistema, se diseñó una pantalla de consulta de movimientos por trabajador con claves en color. De esta manera, las entradas, salidas y movimientos

autorizados por gafete maestro, tenían cada una un color diferente, siendo más sencillo el detectar movimientos faltantes o inusuales

Otro caso observado fue el de los permisos para salir del trabajo durante el turno. Como el trabajador, por norma debe checar sus entradas y salidas siempre, independientemente del motivo, era necesario idear una manera de evitar que el sistema le descontara el tiempo que estuvo fuera, en caso de que el permiso le fuera autorizado. Para este efecto se utilizó uno de los campos disponibles en el archivo de movimientos denominado "Tipo". Hasta antes de esta petición, los únicos tipos posibles eran "E" y "S" para entrada y salida, respectivamente. Se agregó el tipo "P" para los permisos, nótese que existe otro campo de tipo lógico llamado "Entrada" que indica si el movimiento es hacia el interior o hacia el exterior de la empresa. Para autorizar los permisos, se entra a la pantalla de consulta de movimientos y se marca el par Entrada/Salida correspondiente al permiso. Internamente el sistema almacenaba "P" en el campo mencionado y el proceso de contabilización de tiempo trabajado se modificó para ignorar los movimientos que tuvieran esa letra en el campo Tipo.

En general todas las empresas e instituciones que implementaron el sistema vieron una reducción importante en sus tiempos de procesamiento, el caso mejor documentado ya que se realizó un análisis para medir el impacto del sistema, fue el de Bimbo. Esta empresa logró bajar el tiempo de procesamiento de datos para la nómina de 3 horas a sólo 15 minutos, ya que por un lado se distribuyó la responsabilidad de captura de los eventos extraordinarios (cambios de turno,

permisos, etc ) de una sola persona a cada supervisor de línea. Por otro lado, se elaboró un programa que exporta la información que requiere el sistema de nómina y se envía directamente por la red local de computadoras de la empresa. Este paso cierra el ciclo entre los movimientos de los trabajadores en la línea y el procesamiento en sí de la nómina, acortando además el periodo requerido para todo el ciclo.

## CONCLUSIONES

Aunque el sistema fue un éxito en todos los lugares donde se implementó, queda la inquietud de realizar un análisis profundo del diseño para poder hacerlo todavía más flexible, sobre todo en lo referente a las políticas propias de la empresa. Esta mejora permitiría instalarlo en un mayor número de empresas y reduciría considerablemente el costo del sistema ya que las modificaciones directas al código serían mucho menores y muy localizadas.

La inclusión de un reporteador "inteligente" en el cuál el propio usuario pudiera crear los informes que mejor le convengan, sería de gran ayuda y daría gran flexibilidad al sistema.

Otra inquietud queda en el punto de migrarlo a un ambiente multi-tareas y gráfico, ya que este tipo de ambiente cada vez desplaza más al ambiente DOS. Esta mejora podría permitir, por ejemplo, el estar bajando las lecturas de la terminal de código de barras al mismo tiempo que se capturan los movimientos sobre vacaciones o permisos, o se imprime el reporte de la semana pasada.

## NOTA FINAL

El sistoma fue registrado ante la Dirección de Derechos de Autor de la Secretaría de Educación Pública a nombre del suscrito.

## BIBLIOGRAFIA

*Baltasar Cavazos Flores, [et. al.]* NUEVA LEY FEDERAL DEL TRABAJO: TEMATIZADA Y SISTEMATIZADA. 25a. ED. Trillas, México, 1990.

*Roger C. Palmer*, THE BAR CODE BOOK–Reading, Printing And Specification Of Bar Code Symbols, Helmer Publishing Inc. 1991

*John A. Gainsborough*, PERSONAL COMPUTING AND C, Ashton-Tate, U.S.A., 1985

*Terry Dettman*, DOS PROGRAMMER'S REFERENCE, 2nd EDITION, Que Corporation, U.S.A. 1989

*Control Module Inc.*, CMI POLLED PROTOCOL, CMI 1990

# Indice de Tablas