INSTITUTO TECNOLOGICO Y DE ESTUDIOS SUPERIORES DE MONTERREY

CAMPUS MONTERREY

DIVISION DE COMPUTACION, INFORMACION Y COMUNICACIONES

PROGRAMA DE GRADUADOS EN COMPUTACION, INFORMACION Y COMUNICACIONES

MAESTRIA EN CIENCIAS EN INGENIERIA ELECTRONICA
(ESPECIALIDAD EN SISTEMAS ELECTRONICOS)

T E S I S

DESARROLLO DE UN MODULO DE COMUNICACION SERIAL-ETHERNET PARA APLICACIONES DE MONITOREO Y CONTROL REMOTO

ENRIQUE ELIAS RAMOS NORIEGA

MONTERREY, N. L. DICIEMBRE DE 2002

INSTITUTO TECNOLOGICO Y DE ESTUDIOS SUPERIORES DE MONTERREY

CAMPUS MONTERREY

DIVISION DE COMPUTACION, INFORMACION Y COMUNICACIONES

PROGRAMA DE GRADUADOS EN COMPUTACION, INFORMACION Y COMUNICACIONES

MAESTRIA EN CIENCIAS EN INGENIERIA ELECTRONICA
(ESPECIALIDAD EN SISTEMAS ELECTRONICOS)

TESIS

DESARROLLO DE UN MODULO DE COMUNICACION SERIAL-ETHERNET PARA APLICACIONES DE MONITOREO Y CONTROL REMOTO

ENRIQUE ELIAS RAMOS NORIEGA

MONTERREY, N. L. DICIEMBRE 2002

DEDICATORIA:

A mis padres, María del Socorro y Enrique, y a mis hermanas
"Kitita" y Luisa, con todo mi cariño!

Enrique ☺

AGRADECIMIENTOS.

A Dios por su infinito amor y generosidad, por el don de la vida!

A mis padres por su total e incondicional apoyo, por su cariño, y por su ejemplo, por todo lo que son.

A mis maestros, especialmente a mi maestro asesor, Dr. Macías, por su confianza, por haber sido guía durante mis estudios y en la realización de esta tesis, por su trato siempre cordial en el estudio y el trabajo.

De manera especial agradezco a mi fiel compañero matutino de juego, Padre Pepe, por su amistad y su apoyo, por todos esos gratos juegos de 'racquet'.

A todos mis compañeros de escuela, de trabajo, deporte, y diversión. Por haber enriquecido esta hermosa e inolvidable etapa de mi vida.

Y a todos los que de alguna forma me apoyaron y contribuyeron en el logro de este trabajo. A todos ustedes, Muchas Gracias!

EERN ☺

RESUMEN.

Esta tesis presenta a un módulo de comunicación capaz de acoplar una comunicación serial asíncrona con una comunicación de red en una red ethernet. El módulo está orientado a acoplarse a dispositivos como microcontroladores, procesadores digitales de señales, controladores lógicos programables, o cualquier otro dispositivo con capacidad de comunicación serial, de tal forma que dichos dispositivos adquieran la facultad de publicar datos a una red ethernet y/o de recibir datos a través de la misma red. Se expone la importancia de este trabajo a partir del amplio uso de los microcontroladores y su actual dificultad para conectarse a una red ethernet e internet, esta última, a su vez, considerada como un medio de comunicación confiable, de enorme potencial y en constante crecimiento.

El módulo de comunicación propuesto, se construyó a partir de un una tarjeta de desarrollo RCM2100 de la compañía "RabbitSemiconductor". Las dimensiones del módulo son lo suficientemente pequeñas (aproximadamente: 9 x 5 x 2 cm) como para resaltar su portabilidad y facilidad de conexión a cualquier nodo de una red ethernet. El programa del módulo de comunicación opera bajo un esquema de multitarea cooperativa; con esto, el módulo logra dar atención simultáneamente a múltiples usuarios (o clientes) de red además del usuario del puerto serial asíncrono.
El almacenamiento de datos en el módulo puede efectuarse internamente en memoria RAM, o también externamente mediante un módulo externo de memoria flash, esta última con capacidad de 4 u 8 Megabytes. La memoria externa a su vez funciona como un recurso compartido entre el módulo de comunicación y múltiples usuarios externos; la memoria se comparte mediante un BUS serial síncrono que permite el acceso directo a la memoria.

Por otra parte, el protocolo utilizado para la comunicación entre el módulo de comunicación y sus clientes es el TCP ("Transfer Control Protocol"), un protocolo de red confiable que garantiza la entrega ordenada de un flujo datos de un punto a otro.
A pesar de que el enfoque primordial del módulo es la comunicación de datos, se aprovechó también la facilidad de comunicación remota y algunos de los recursos del módulo para extender su uso hacia algunas operaciones de monitoreo y control. El monitoreo se realiza a través de la lectura de un puerto digital, y el control se efectúa mediante la manipulación de los siguientes recursos: un puerto digital de salida, el puerto serial asíncrono de salida, y el puerto de red ethernet, que en este caso utiliza el protocolo UDP ("User Datagram Protocol") para el envío de datos.

Por último, se efectuaron distintas pruebas para evaluar el correcto funcionamiento y desempeño del módulo. Para la realización de las pruebas se construyeron programas que ponen en marcha las distintas facilidades del módulo de comunicación. Los programas abarcan aplicaciones para PC y para la familia de microcontroladores HC16. El núcleo de los programas de prueba esta compuesto por funciones que después quedan como librerías del usuario para el desarrollo de programas y de nuevas aplicaciones del módulo de comunicación.

INDICE GENERAL

DEDICATORIA i
AGRADECIMIENTOS ii
RESUMEN iii
INDICE GENERAL iv
INDICE DE FIGURAS vii
INDICE DE TABLAS x

Capítulo 1. Introducción 1
1.1. Uso de los microcontroladores. 1
1.2. El uso del microcontrolador contra la PC en aplicaciones determinísticas 1
1.3. Características generales del desarrollo de los microcontroladores en el mercado. 2
1.4. Comunicación Serial. 2
1.4.1. Comunicación Serial Síncrona. 3
1.4.2. Comunicación Serial Asíncrona. 3
1.4.3. El estándar RS-232 y RS-485. 4
1.5. Redes de Computadoras. 6
1.6. Planteamiento del problema. 6
1.7. Objetivos y Justificación del caso de estudio. 8

Capitulo 2. Redes de Computadoras 10
2.1. Conceptos Básicos de Redes. 10
2.1.1 Partes fundamentales de una red 10
2.1.2. Topología de Redes 11
2.1.3. Clasificación de Redes según su extensión Geográfica. 12
2.2. Red de Ethernet. 12
2.2.1. Antecedentes 12
2.2.2. Método de Acceso (CSMA/CD) 13
2.2.3. Direcciones físicas. 14
2.2.4. Trama de Ethernet. 15
2.3. Internet 15
2.4. Modelo de capas de una red TCP/IP 16
2.4.1. La capa física 17
2.4.2. La capa de enlace 17
2.4.3. La capa de red. 17
2.4.4. La capa de Transporte 18
2.4.5. La capa de Aplicación. 18
2.5. Direcciones IP. 18
2.5.1. La máscara de red ("netmask") 19
2.5.2. Direcciones IP especiales. 19
2.5.3. Direcciones IP Privadas. 20
2.5.4. Direcciones IP dinámicas y fijas 20
2.6. Los Protocolos de capas de una red TCP/IP 20
2.6.1. Capa física 21
2.6.2. Capa de enlace. 21
2.6.3. Capa de red. 22
2.6.4. Capa de Transporte. 23
2.6.5. Capa de Aplicación 24
2.7. Enrutamiento de paquetes IP. 25
2.7.1. Entrega en forma directa 25
2.7.2. Entrega en forma indirecta 25
2.7.3. Compuerta ("gateway") 26
2.7.4. Ejemplo de ruteo 26
2.7.5. Servicio de Nombre de Dominios 27

Capítulo 3. El Módulo de Comunicación 28
3.1. La Tarjeta de Desarrollo RCM2100. 28
3.1.1. El Procesador. 30
3.1.2. Memoria Interna. 32
3.1.2.1. Memoria RAM. 32
3.1.2.2. Memoria Flash. 33
3.1.3. Memoria Externa. 33
3.1.4. Puerto de Comunicación Serial Asíncrono. 35
3.1.5. Puerto de Comunicación Serial Síncrono. 36
3.1.5.1. Líneas auxiliares de control. 37
3.1.5.2. Detección de colisiones. 38
3.1.6. Puerto de Red Ethernet. 39
3.1.7. Puerto de Entrada/Salida Digital. 40
3.2. Estructura del Programa. 42
3.2.1. Multitarea Preventiva ("Preemptive Multitasking") 43
3.2.2. Multitarea Cooperativa ("Cooperative Multitasking") 43
3.2.3. Estructura en base a tareas. 45
3.2.4. "Costatements" y "Cofunciones" 45
3.2.5. Tareas Principales del Programa. 46
3.2.6. La Tarea "Ejecutar Funciones de Memoria Externa" ("Cost_ext SF1000") 47
3.2.7. Activación / Desactivación de las Tareas Principales. 48
3.2.8. Servidor Ethernet para Conexión Múltiple. 51
3.2.9. Resumen. 53
3.3. Funcionamiento del Programa del Módulo de comunicación. 54
3.3.1. El inicio del programa. 54
3.3.2. Etapa de recepción de comandos 55
3.3.2.1. Comandos para Manejo de Memoria: 58
3.3.2.2. Comandos para Manejo de Puertos: 58
3.3.2.3. Comandos de Configuración y Prueba: 59
3.3.3. Modelo de Servidor. 59
3.3.4. El Servidor Serial Asíncrono. 63
3.3.4.1. Configuración Inicial del Servidor Serial Asíncrono. 63
3.3.4.2. Funcionamiento del Servidor Serial Asíncrono 63
3.3.4.3. Protocolo del Servidor Serial Asíncrono. 64
3.3.4.4. Envío de un Comando Serial. 65
3.3.4.5. Respuesta a un Comando Serial. 66
3.3.5. El Servidor Ethernet. 67
3.3.5.1. Configuración Inicial del Servidor Ethernet. 67
3.3.5.2. Funcionamiento del Servidor Ethernet. 67
3.3.5.3. Protocolo del Servidor Ethernet. 69
3.3.5.4. Envío de un Comando Ethernet. 69
3.3.5.5. Respuesta a un Comando Ethernet. 70
3.3.6. Arbitraje de recursos. 70
3.3.7. Operaciones Cíclicas del Módulo de Comunicación. 74
3.4. Funcionamiento de la Memoria Externa. 75
3.4.1. Niveles de voltaje de entrada y salida. 76
3.4.2. La estructura interna de la memoria. 76
3.4.3. Identificación del dispositivo de memoria. 77
3.4.4. Direcciones físicas y direcciones lógicas de memoria. 78
3.4.5. Rutina de escritura del módulo de comunicación. 80
3.4.6. Rutina de lectura del módulo de comunicación. 82
3.4.7. Operaciones Misceláneas sobre la Memoria Externa. 83
3.5 Ejemplo de una aplicación real. 84

Capítulo 4. Pruebas y Resultados 86
4.1. Pruebas con depurador 86
4.1.1. Atención a un usuario de red. 88
4.1.2. Acceso desde navegador de red. 89
4.1.3. Atención a múltiples usuarios de red. 91
4.1.4. Arbitraje con recurso en conflicto. 93
4.2. Pruebas con interfase serial 94
4.3. Pruebas con interfase de red 98
4.3.1. Pruebas de velocidad. 99
4.3.2. Pruebas de operación de algunos comandos. 101
4.3.3. Envío de una página HTTP 102
4.3.4. Obtención del estado de las funciones cíclicas. 104
4.3.5. Lectura de mensajes UDP 105
4.3.6. Operación cíclica del Puerto digital B de salida. 105
4.3.7. Función cíclica del puerto serial B de salida. 107
4.3.8. Pruebas Adicionales. 108
4.4. Pruebas con usuario de Memoria Externa. 108
4.4.1. Lógica de control del BUS. 109
4.4.2. Librerías de usuario externo. 111
4.5. Pruebas de memoria externa 113

Capítulo 5. Conclusiones y Observaciones Finales 117
5.1. Conclusiones. 117
5.2. Contribuciones 117
5.3. Características Importantes 117
5.4. Recomendaciones Futuras. 118

APENDICE A. Foto y Conectores de la Tarjeta RCM2100 120
APENDICE B. Diagrama Eléctrico de la Tarjeta RCM2100 121
BIBLIOGRAFIA 124

INDICE DE FIGURAS

Figura 1.1. Esquema que muestra el amplio uso de los Microcontroladores (μC). 1
Figura 1.2. Diagrama de tiempo de una comunicación serial síncrona. 3
Figura 1.3. Esquema de una comunicación serial asíncrona. 4
Figura 1.4. Manejo de los estándares RS-485, y RS-232 a partir de un puerto serial asíncrono (UART)..... 5
Figura 1.5. El concepto de internet, una red de redes a nivel mundial. 6
Figura 1.6. Esquema que muestra el objetivo de este trabajo, la integración de un microcontrolador a una red Ethernet, mediante un módulo de comunicación de bajo costo. 9
Figura 2.1. Elementos básicos de una red 10
Figura 2.2. Vista hacia el exterior de una tarjeta de red Ethernet que soporta los 3 tipos de cableado. 13
Figura 2.3. Formato de una dirección física de Ethernet de 48 bits. 14
Figura 2.4. Formato de un paquete de Ethernet. 15
Figura 2.5. Concepto de internet como una sola red virtual, y como una red compuesta de redes TCP/IP.. 16
Figura 2.6. Modelo de capas de una red TCP/IP 17
Figura 2.7. Esquema de la distribución de los 32 bits de una dirección IP de acuerdo a la clase de dirección, A, B, o C. 19
Figura 2.8. Modelo de capas para una red TCP/IP que incluye los protocolos y componentes de cada capa. 21
Figura 2.9. Ejemplo de la operación ping para detectar la presencia del mismo anfitrión local mediante la dirección IP especial: 127.0.0.1 22
Figura 2.10. Ejemplo de una red global compuesta por 4 subredes y 3 ruteadores. 26
Figura 3.1. Esquema general de la arquitectura de la tarjeta de desarrollo RCM2100. 28
Figura 3.2. Partes principales, puertos, y líneas de conexión de la tarjeta RCM2100 29
Figura 3.3. Esquema general del Procesador Rabbit 2000. 30
Figura 3.4. Diagrama a bloques del Procesador Rabbit 2000 31
Figura 3.5. Registros internos del Procesador Rabbit 2000 32
Figura 3.6. Foto del módulo de memoria externo, y sus dimensiones 34
Figura 3.7. Pinout del módulo de memoria externo de la serie SF1000. 34
Figura 3.8. Uso de un circuito manejador "MAX232" para el acoplamiento de los voltajes de la comunicación serial entre la tarjeta RCM2100 y una PC. 36
Figura 3.9. Esquema del acceso directo a la memoria externa por medio de un bus serial síncrono 37
Figura 3.10. Esquema de conexión de las líneas de auxiliares (LIU, RIU) para coordinar el control del BUS serial síncrono 38
Figura 3.11. Conexión requerida para los pines de interrupción externa PE4 y PE5 del procesador Rabbit 2000. 39
Figura 3.12. Esquema del módulo de ethernet de la tarjeta RCM2100. 40
Figura 3.13. Representación de la ejecución de un programa con 3 tareas, utilizando un esquema de multitarea preventiva. 43
Figura 3.14. Representación de la ejecución de un programa con 3 tareas utilizando un esquema de multitarea cooperativa. 44
Figura 3.15. Diagrama de Flujo General del Programa del Módulo de Comunicación. 45
Figura 3.16. Descomposición de las tareas principales del programa en "costatements" y "cofunciones".. 46
Figura 3.17. Diagrama de las tareas principales del módulo de comunicación. 47
Figura 3.18. Diagrama que muestra algunas operaciones del "Servidor Ethernet" que se incluyen en un "costatement" al nivel de una tarea principal. 48
Figura 3.19. Casos en los que la tarea principal "Inicializar memoria externa" se activa y desactiva. 50
Figura 3.20. Activación y desactivación de la tarea principal "Cost_ext SF1000". 50
Figura 3.21. Activación y desactivación de la tarea "ejecutar operaciones cíclicas". 51
Figura 3.22. Esquema real del programa principal del módulo de comunicación, habilitado para atender "N" conexiones simultáneas de red. 51
Figura 3.23. Esquema que muestra el uso del mismo código del servidor ethernet para atender a "N" conexiones, mediante el intercambio de un juego de variables con "N" copias del mismo juego de variables 53
Figura 3.24. Esquema de los recursos y periféricos del módulo de comunicación 54

Figura 3.25. Operación del módulo de comunicación a través del puerto serial asíncrono (servidor serial).55
Figura 3.26. Operación del módulo de comunicación a través del puerto ethernet (servidor ethernet)..... 56
Figura 3.27. Modelo de la operación general de un servidor..... 59
Figura 3.28. Variante de la etapa de "espera por un comando"..... 60
Figura 3.29. Variante de la etapa "Ejecución del Comando"..... 61
Figura 3.30. Variante de la etapa "Regresa resultado"..... 63
Figura 3.31. Diagrama de flujo del servidor serial..... 64
Figura 3.32. Formato del protocolo del servidor serial..... 65
Figura 3.33. Diagrama de flujo del servidor ethernet..... 68
Figura 3.34. Formato del protocolo del servidor ethernet..... 69
Figura 3.35. Diagrama de tiempos que muestra la ejecución de tareas fraccionadas y tareas completas..... 71
Figura 3.36. Efecto del arbitraje entre operaciones que entran en conflicto al intentar hacer uso del mismo recurso..... 72
Figura 3.37. Ejemplo de operaciones simultáneas en una misma memoria con riesgo y sin riesgo de conflicto..... 73
Figura 3.38. Diagrama de flujo para el arbitraje de memoria..... 73
Figura 3.39. Diagrama de Flujo de las tareas "Operaciones Cíclicas"..... 74
Figura 3.40. La salida del módulo de memoria ("SF1000") está invertida con respecto al circuito integrado de memoria flash..... 76
Figura 3.41. Estructura interna del circuito de memoria de 4 MB..... 77
Figura 3.42. Diagrama de tiempo de una de las formas de ejecución del comando de lectura del registro estatus..... 78
Figura 3.43. Direcciones físicas y direcciones lógicas del Módulo de memoria de 4MB..... 78
Figura 3.44. Area de almacenamiento de datos de control de cada página de la memoria de 4 MB..... 79
Figura 3.45. Pasos de la rutina de escritura a memoria externa por parte del programa del módulo de comunicación..... 81
Figura 3.46. Pasos de la rutina de lectura de la memoria externa por parte del programa del módulo de comunicación..... 83
Figura 3.47. Diagrama a bloques de un sistema de medición..... 85
Figura 4.1. Distintas formas de prueba efectuadas al módulo de comunicación..... 86
Figura 4.2. Pantalla de mensajes emitidos al inicio del programa del módulo de comunicación..... 87
Figura 4.3. Lista de mensajes emitidos por el programa del Módulo de Comunicación al recibir una solicitud del comando "Get_Mem_cfg" (m) para obtener la información de los tamaños de los distintos tipos de memoria..... 89
Figura 4.4. Acceso al módulo desde navegador de red..... 90
Figura 4.5. Lista de mensajes emitidos por el programa del módulo de comunicación, durante el acceso simultáneo de 3 usuarios distintos..... 92
Figura 4.6. Prueba con dos operaciones simultáneas que se empalman con riesgo de conflicto..... 93
Figura 4.7. Lista de mensajes emitidos por el programa del módulo de comunicación, durante el acceso simultáneo de 2 usuarios distintos, los cuales entran en conflicto al hacer uso del mismo recurso..... 94
Figura 4.8. Configuración para la prueba con interfase serial..... 95
Figura 4.9. Gráfica de velocidad de transferencia serial para los distintos tipos de memoria interna del módulo de comunicación..... 96
Figura 4.10. Cálculo del tiempo de ejecución de un comando serial..... 97
Figura 4.11. Configuración para la prueba con interfase red..... 98
Figura 4.12. Modelo de un programa cliente y su relación con el modelo de un programa servidor..... 99
Figura 4.13. Prueba de velocidad de transferencia de datos del módulo de comunicación..... 99
Figura 4.14. Cálculo del tiempo de respuesta para un comando del servidor..... 100
Figura 4.15. Panel frontal de la interfase de entrada de comandos de red..... 102
Figura 4.16. Transmisión de una sección específica de memoria ante la conexión de un navegador..... 103
Figura 4.17. Panel frontal del programa que obtiene la configuración de las funciones cíclicas..... 104
Figura 4.18. Panel frontal del programa que atiende un puerto UDP para la recepción de mensajes y/o datos..... 105
Figura 4.19. Señales de salida de los bits 6 y 7 del puerto B del módulo de comunicación..... 106
Figura 4.20. Señales donde se aprecia el error de precisión sobre los cambios de estado de la operación cíclica "PortB_cycle"..... 107

Figura 4.21. Ejemplo del resultado de la operación cíclica "SerB_cycle". 107
Figura 4.22. Configuración para la prueba de la lógica de control del BUS serial síncrono. 109
Figura 4.23. Relación de algunas señales en la operación del BUS serial síncrono. 110
Figura 4.24. Relación entre la señal de entrada RIU y la señal de salida de reloj. 111
Figura 4.25. Configuración para la prueba de la operación del BUS serial síncrono. 112
Figura 4.26. Ejemplo de la documentación de la función "SH_SF1000ERASEBLOCK" para el HC16. . 113
Figura 4.27. Configuración para la prueba de la memoria externa. 114
Figura 4.28. Panel frontal de las funciones "SF1000init". 115
Figura 4.29. Paneles frontales de las funciones "CheckWrites" y "EraseBlock". 115
Figura 4.30. Paneles frontales de las funciones "SF1000_Read" y "SF1000_Write". 116

INDICE DE TABLAS

Tabla 1.1. Tabla comparativa entre los estándares RS-485 y RS-232 5
Tabla 2.1. Tipos de direcciones IP con sus respectivos números de redes y anfitriones posibles. 19
Tabla 2.2. Algunos servicios de red comunes y sus respectivos puertos asociados por defecto. 24
Tabla 2.3. Tabla del ruteo para los ruteadores A, B, y C. 26
Tabla 3.1. Función de los pines PC4 y PC5 como puerto serial B asíncrono, y su localización en la tarjeta RCM2100. 35
Tabla 3.2. Tabla de estado para las líneas LIU y RIU. 38
Tabla 3.3. Función de los pines PD0 a PD3 como puerto serial síncrono, y su localización en la tarjeta RCM2100. 39
Tabla 3.4. Pines del Puerto E utilizados para coordinar el acceso al bus serial, y su localización en la tarjeta RCM2100. 39
Tabla 3.5. Puerto B utilizado para la entrada de 4 líneas y la salida de 2 líneas digitales, y su localización en la tarjeta RCM2100. 41
Tabla 3.6. Uso de las líneas de los puertos del procesador Rabbit 2000 en la aplicación del módulo de comunicación. 42
Tabla 3.7. Tareas principales del programa del modulo de comunicación, con el tipo de tarea ("costatement" asociado), y su respectivo estado al inicio del programa. 49
Tabla 3.8. Secuencia de la ejecución del programa principal del módulo de comunicación. 52
Tabla 3.9. Configuración inicial de los puertos de comunicación del módulo de comunicación al inicio del programa. 55
Tabla 3.10. Lista de comandos del módulo de comunicación 57
Tabla 3.11. Ejemplo de valores devueltos por algunos comandos del servidor ethernet 62
Tabla 3.12. Reglas de arbitraje para el puerto serial. 74
Tabla 3.13. Capacidad de la memoria del módulo de memoria externo y el correspondiente número del circuito integrado de memoria. 75
Tabla 3.14. Tabla comparativa entre los parámetros de las dos capacidades de memoria. 80
Tabla 4.1. Velocidad de transferencia serial en bytes/segundo para los distintos tipos de memoria interna del módulo de comunicación. 96
Tabla 4.2. Velocidad de transferencia de datos de los comandos de red del módulo de comunicación según el tipo de memoria implicada en el comando. 100

Capítulo 1. Introducción.

1.1. Uso de los microcontroladores.

Desde su aparición, en la década de los setentas, el uso de los microcontroladores (μC) se ha expandido hacia un sinnúmero de aplicaciones, desde aparatos electrodomésticos, la industria automotriz, la industria de la aviación, proyectos militares y del espacio, en las comunicaciones, en equipos de seguridad, hasta en juguetes, calculadoras, y equipos de entretenimiento, además de una infinidad de otros usos. Para darse una idea de la magnitud del uso de estos dispositivos, se estima que el número de microcontroladores construidos hasta la fecha, sobrepasan al número de personas del total de la población mundial, es decir, sobre pasan los ¡6 mil millones de unidades!

Definitivamente el uso de microcontroladores ha tenido un gran impacto y ha revolucionando nuestra forma de vida durante las últimas 3 décadas. Una de las áreas donde los microcontroladores ponen en evidencia ese gran impacto, es en la industria. Muchas de las operaciones de los procesos industriales que anteriormente se realizaban por medios electromecánicos, fueron desplazadas por operaciones efectuadas dentro de los microcontroladores, como consecuencia, y a raíz de la capacidad y versatilidad de estos dispositivos, los procesos industriales se han hecho más eficientes, repercutiendo en la reducción de costos de operación.

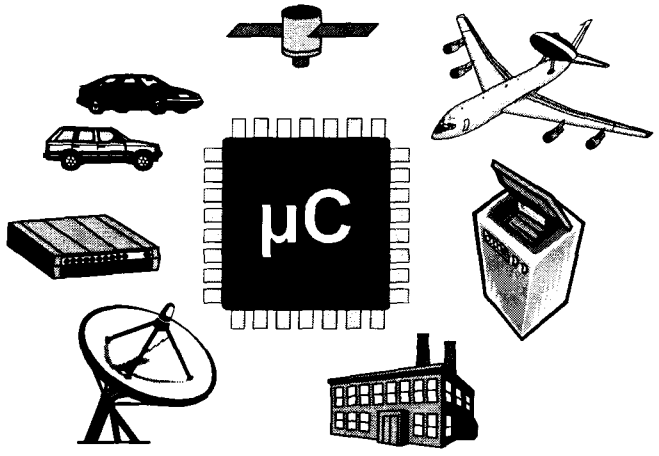


Figura 1.1. Esquema que muestra el amplio uso de los Microcontroladores (μC).

1.2. El uso del microcontrolador contra la PC en aplicaciones determinísticas.

El microcontrolador es un dispositivo vigente que sigue y augura seguir empleándose en la industria a pesar del prominente actual uso de las computadoras. Esto se debe a que en contraste con las computadoras, los microcontroladores están especialmente diseñados para llevar un control más preciso sobre los tiempos de atención a periféricos y eventos externos; en otras palabras, la operación de los microcontroladores sobre eventos externos puede ser determinística, mientras que en una PC se dificultan este tipo de operaciones relacionadas con el control directo y preciso de tiempos sobre eventos

externos, ya que el diseño de una PC realmente no esta orientado a ese tipo de tareas. Los microcontroladores en comparación con las PC's, son más reducidos en cuanto a espacio físico, a capacidad de almacenamiento, y a recursos en general, como consecuencia, en precio también son más accesibles.

1.3. Características generales del desarrollo de los microcontroladores en el mercado.

Una de las grandes ventajas de los microcontroladores es su bajo costo; actualmente es posible encontrar microcontroladores con costos por debajo de un dólar, además, existe una enorme variedad de ellos en el mercado, con distintas capacidades en cuanto a velocidad de procesamiento, almacenamiento en memoria, y periféricos. Son fáciles de adquirir y de programar. Existen varias compañías y desarrolladores independientes de "software" que ofrecen gratuitamente o venden a través de la red el acceso a sus herramientas de programación, algunas de ellas, muy prácticas y amigables, incluso, hay herramientas que soportan lenguajes de programación de alto nivel, además de librerías de funciones y notas de aplicación. Dada la sencillez y versatilidad de los μC's, muchos problemas que aparecen en la industria se resuelven, fácil y rápidamente, y a un bajo costo, mediante la aplicación de estos dispositivos.

Con las nuevas tecnologías en fabricación de circuitos integrados, en los últimos años han resurgido versiones de microcontroladores de antaño pero ahora mejoradas en cuanto a consumo de energía y velocidad de operación. Las mejoras de los μC también comprenden el incremento de la capacidad y versatilidad de sus módulos periféricos; algunos periféricos han surgido de la demanda a gran escala de aplicaciones específicas, por ejemplo, hay versiones de microcontroladores con periféricos especialmente diseñados para la industria automotriz.
Dentro de la diversidad de módulos periféricos, uno de los módulos prácticamente inseparable desde los inicios en el mercado del microcontrolador es el módulo de "Interfase de Comunicación Serial" (SCI), el cual, puede incluir tanto una comunicación serial síncrona, como también una comunicación serial asíncrona; inclusive, puede incluir ambas opciones. Otros módulos que pueden encontrarse además del SCI, son: puertos con líneas de entrada y salida digital, convertidores análogo-digital, temporizadores, generadores de pulsos, contadores de eventos, entre otros destinados a aplicaciones específicas.

1.4. Comunicación Serial.

Además de los microcontroladores, muchos dispositivos y equipos electrónicos cuentan con una interfase de comunicación serial (SCI) para comunicarse con otros dispositivos o equipos. Algunos ejemplos de equipos que comúnmente incluyen una interfase serial, son: las computadoras personales (PC's), algunos circuitos integrados como los DSP's (Procesadores Digitales de Señales), impresoras, lectoras de código de barra, PLC's (Controladores Lógicos Programables), entre muchos otros.
La comunicación serial puede dividirse en dos grandes categorías de acuerdo a la presencia o ausencia de una señal de reloj de referencia que acompaña a una comunicación de datos. Estas dos categorías son las siguientes:

- Comunicación Serial Síncrona
- Comunicación Serial Asíncrona

1.4.1. Comunicación Serial Síncrona.

La comunicación serial síncrona se caracteriza por que va acompañada explícitamente por una señal de reloj, dicha señal de reloj es común para ambas de las partes en comunicación, es decir, tanto para el transmisor como el receptor en comunicación. Las transiciones de la señal de reloj, en este caso, dan la pauta para que se lleve a cabo la comunicación de datos bit por bit entre un transmisor y un receptor. La señal de reloj puede ser enviada a través de una línea independiente a la línea de datos, o puede ser transmitida en una sola línea junto con los datos mediante una técnica de codificación(*). En esta forma de comunicación serial la velocidad de transferencia de datos puede ser muy alta, es decir, puede estar en el orden de Megabits por segundo; Además, la velocidad del reloj puede tener variaciones, o no necesariamente debe ser constante; Lo importante es que la línea de datos este siempre en sincronía con la línea de reloj. Este tipo de comunicación serial síncrona es común encontrarla en los dispositivos microcontroladores bajo el nombre de "Syncronous Serial Peripheral Interfase" (SPI).

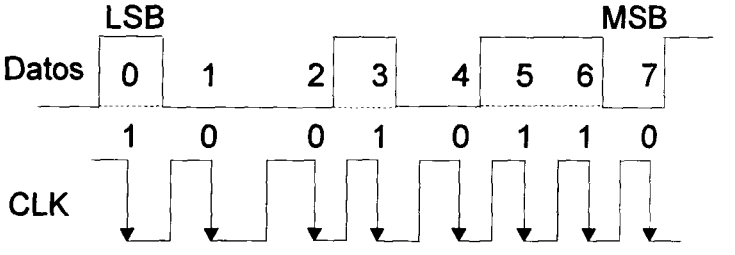


Figura 1.2. Diagrama de tiempo de una comunicación serial síncrona.

Dato enviado (10010110b). LSB = Bit menos significativo, MSB = Bit más significativo. Cada bit se registra en el flanco negativo del reloj. En este caso, el reloj no es uniforme en el tiempo, pero guarda sincronía con los datos.

1.4.2. Comunicación Serial Asíncrona.

En la comunicación serial asíncrona no existe una línea dedicada como reloj para que de la pauta en una transferencia de datos. En este tipo de comunicación tanto el transmisor como el receptor cuentan con su propio reloj interno, ambos relojes son independientes entre sí.

Para lograr con éxito una comunicación serial asíncrona es de vital importancia que tanto el receptor como el transmisor estén configurados a la misma frecuencia de reloj. Existen velocidades típicas con valores ya definidos como estándar que facilitan el acoplamiento entre las partes de la comunicación. En este tipo de comunicación, la velocidad de transferencia de datos es menor a la que se logra mediante una comunicación serial síncrona. Esto se debe a que el receptor debe de ubicar cada bit que recibe en el tiempo; y para lograrlo, el receptor realiza un muestreo de cada uno de los bits recibidos en forma

---

(*) Dieck, Graciano. Instrumentación, Acondicionamiento Electrónico y Adquisición de Datos. Pág. 386

serial. La frecuencia de muestreo del receptor debe ser varias veces mayor a la frecuencia del reloj del transmisor para validar correctamente la recepción de cada bit. Además de esto, en el protocolo serial asíncrono se acostumbra agregar a cada byte enviado de 2 a 3 bits extra para marcar el inicio y el fin de la transmisión de cada byte. Esto se hace para sincronizar al transmisor y al receptor en cada envío de un byte.
Las velocidades que pueden lograrse mediante una comunicación serial asíncrona típicamente pueden alcanzar varios cientos de Kilobits por segundo. En los microcontroladores este módulo se identifica generalmente como UART ("Universal Asyncronous Receiver Transmitter") o sencillamente como SCI ("Serial Communication Interface").

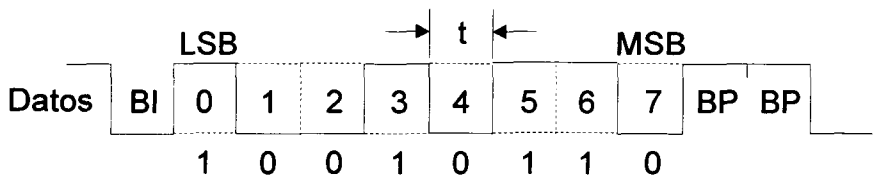


Figura 1.3. Esquema de una comunicación serial asíncrona.
Dato enviado (10010110b). BI = Bit de Inicio, BP = Bit de paro,
LSB = Bit menos significativo, MSB = Bit más significativo,
t=tiempo de bit.

1.4.3. El estándar RS-232 y RS-485.

Los estándares RS-232 y RS-485, son dos estándares de comunicación serial muy utilizados. Estos estándares se basan en la misma lógica de una comunicación serial asíncrona; La diferencia estriba en el nivel físico, es decir, en los niveles y tipos de voltaje que se utilizan para transmitir datos de un lugar a otro; también, como parte de los protocolos cada estándar utiliza una lógica de señalización para coordinar la transferencia de datos entre un transmisor y uno o varios receptores.

El estándar RS-232, es común encontrarlo sobre todo en equipos de cómputo, es un estándar mayormente utilizado en aplicaciones domésticas por así decirlo; por ejemplo para la comunicación entre una PC y distintos equipos periféricos con como lectoras de código de barra, impresoras, el "ratón" de la misma computadora, etc.
El estándar RS-485 en comparación con el estándar RS-232, en general es más robusto, además, permite crear una red de comunicación multipunto a diferencia del RS-232 que esta orientado sólo a comunicaciones punto a punto. Las características del estándar RS-485 han sido ampliamente aceptadas en aplicaciones industriales, de hecho, es un estándar que se asocia automáticamente con la industria. Muchos equipos de carácter industrial como PLC's (Controladores Lógicos Programables), medidores de parámetros eléctricos, computadoras industriales, etc. Traen incluido desde fábrica una interfase de comunicación serial bajo el estándar RS-485.

Un punto importante que conviene destacar es que a partir de un módulo de comunicación serial asíncrona (UART) puede adoptarse cualquiera de los estándares RS-232 ó RS-485. La comunicación serial asíncrona es un módulo comúnmente encontrado en varios equipos y circuitos integrados, como microcontroladores, DSP's, memorias

digitales, computadoras personales, etc. El estándar adoptado, ya sea RS-232, RS-485, u otro de mismo carácter asíncrono, depende de un circuito manejador que se acopla a la lógica del módulo UART. En el mercado existen circuitos manejadores para acoplar los niveles lógicos de un UART a los niveles y a las especificaciones de un determinado estándar. Por ejemplo, un manejador típico para el estándar RS-232 es el "MAX232" de la compañía "MAXIM", mientras que un circuito manejador típico del estándar RS-485 es el "DS3695" de la compañía "National Semiconductor".

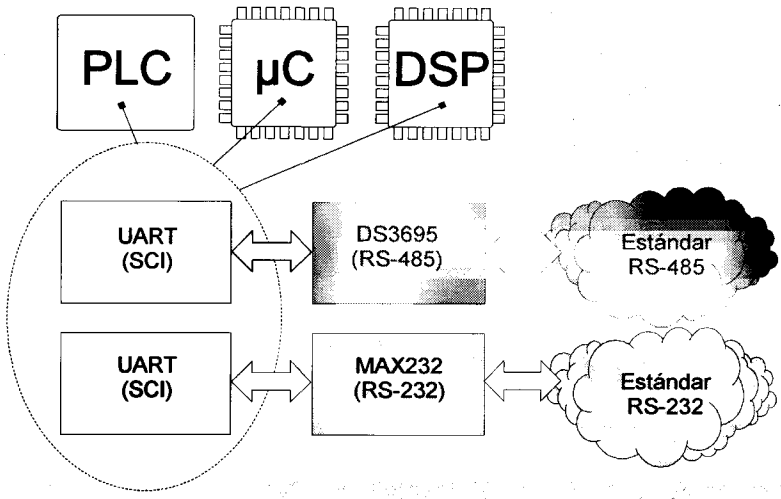


Figura 1.4. Manejo de los estándares RS-485, y RS-232 a partir de un puerto serial asíncrono (UART).

Especificación	RS-232C	RS-485
Modo de operación	tierra común	Diferencial
Transmisores en línea	1	32
Receptores en línea	1	32
Máx. Longitud de cable	50 pies / 15.2 m	4000 pies / 1219.2 m
Máx. Razón de datos	921,600 bps	10Mbps
Máx. Voltaje permitido	25 V	7 a 12 V
Voltaje de transmisión	±5 a ±15 V	±1.5 a ±5 V
Carga del Transmisor	3 a 7 KΩ	54Ω mín
Voltaje de recepción	±15V	-7 a 12 V
Sensitividad del receptor	±3V	±200 mV
Impedancia de entrada del receptor	3 a 7 KΩ	12KΩ mín

Tabla 1.1. Tabla comparativa entre los estándares RS-485 y RS-232[**]

** Dieck, Graciano. "Instrumentación, Acondicionamiento Electrónico y Adquisición de Datos". Pág. 389

1.5. Redes de Computadoras.

Por otra parte, otra de las manifestaciones actuales, producto de la evolución tecnológica, es la capacidad, cobertura, y facilidad de comunicación a nivel global. En este ámbito, la creación de redes para compartir recursos ("hardware") e información entre computadoras, se ha expandido con una enorme proyección en la que se ha buscado la forma de enlazar redes de computadoras con otras redes, hasta llegar a lo que ahora se conoce como "internet". Internet puede verse como un medio que permite la comunicación confiable de datos a largas distancias, con una gran cobertura, incluso entre continentes, con tiempos de respuesta y a costos aceptables en comparación con otros medios, como por ejemplo, el teléfono. Internet es un medio que sigue creciendo y evolucionando, cada vez con más usuarios, con mayor alcance, y con más número de aplicaciones. Inicialmente el uso de "internet" se relacionaba exclusivamente con la comunicación entre computadoras, pero dado su alcance, y su potencial, se ha buscado la forma de integrar a otros aparatos que aprovechen las ventajas del medio para la comunicación de datos. Como ya se mencionó, internet esta formada por el enlace de muchas redes. Un término muy común al hablar de redes de computadoras es el de "Redes de Area Local" (LAN de sus siglas en inglés "Local Area Network"), el cual, se refiere a una red de computadoras que se extiende sobre una área relativamente pequeña, generalmente una red que esta confinada dentro de uno o varios edificios. Una LAN, a su vez, puede permanecer aislada, o por el contrario, puede incrementar su alcance al estar suscrita a la red global de redes (internet). En la implementación de redes de área local uno de los estándares de mayor aceptación es el estándar Ethernet IEEE 802.3; hoy en día es muy común encontrar redes de computadoras tanto en las industrias, como en escuelas, hospitales, y comercios, con una infraestructura bajo este estándar de Ethernet.

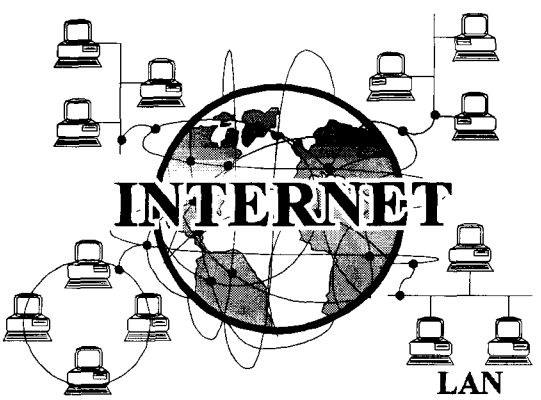


Figura 1.5. El concepto de internet, una red de redes a nivel mundial.

1.6. Planteamiento del problema.

Retomando dos de los planteamientos expuestos hasta este punto, por un lado, se ha expuesto el caso de los microcontroladores como dispositivos muy versátiles, de bajo costo y fácil programación, ampliamente utilizados en un gran número de aplicaciones, una de ellas, el área industrial donde ha sido causa de grandes transformaciones.

Por otro lado, también se ha expuesto el gran alcance de lo que ahora conocemos como "internet", con una infraestructura actual gigantesca y en constante crecimiento, resaltando que es un medio accesible y confiable para la comunicación de datos; a su vez, se ha destacado que internet surge de la interconexión de muchas redes pequeñas (LAN's) donde uno de los estándares más comunes en la implementación de LAN's es el estándar IEEE 802.3 de ethernet.

Tomando como base estos dos hechos de gran importancia, es decir, la presencia de los microcontroladores y la presencia de redes ethernet como portal de entrada a la red global de internet; se tiene la dificultad de un microcontrolador típico para conectarse directamente y establecer una comunicación con una red tipo Ethernet. La dificultad estriba principalmente en la complejidad inmersa en el medio y en los protocolos de comunicación para comunicar datos a través de la red. El acceso a la red implica, tanto capacidad de procesamiento como recursos en memoria, además de una forma de acoplamiento con el medio. El grado de procesamiento y de la capacidad de memoria, representa un requerimiento que anteriormente sólo era cubierto por computadoras o equipos similares, generalmente costosos.
Actualmente la industria electrónica ha desarrollado productos con miras de integrar a la red dispositivos de bajo costo como microcontroladores. Esta es una área de gran oportunidad que actualmente sigue en desarrollo. La mayoría de estos productos simplifican el manejo de protocolos para conectarse a la red, es decir, utilizan los requerimientos básicos de los protocolos en forma abreviada para reducir el espacio y la complejidad de algoritmos, de tal forma que puedan ser implementados y ejecutados por dispositivos pequeños y de bajo costo como microcontroladores. A pesar de que la velocidad del medio de la red no es explotada en su totalidad debido a la menor capacidad de procesamiento de los μC's y al uso de protocolos abreviados, de cualquier forma se adquiere un enorme potencial al habilitar a dispositivos compactos y de bajo costo con una comunicación a la red. Aún así, en el mercado de los microcontroladores todavía no se ha establecido un estándar considerado como un módulo periférico y que forme parte del resto de módulos disponibles para un microcontrolador, tal y como sucede por ejemplo con el módulo para manejo de comunicación serial (SCI).
Algunas de las opciones que se manejan para lograr comunicar un microcontrolador con una red ethernet son: la comunicación del μC con un circuito externo de uso específico para el manejo de los protocolos y del medio físico de la red; otra opción, es la de implementar por completo en forma de código los protocolos del manejo de red para que sean ejecutados por el procesador; en ambos casos es necesario una etapa de manejo y de acoplamiento entre el medio físico de la red Ethernet y la circuitería digital. También en ambos de estos casos, el programa principal del microcontrolador debe combinar las instrucciones de las tareas específicas para el cual esta destinada su aplicación junto con las instrucciones que manejan la parte de comunicación con la red. Como consecuencia, frecuentemente el espacio en memoria de los microcontroladores se vuelve una limitante, ya que además del código del programa principal, en la misma memoria de programa se aloja el código para el manejo de las operaciones de red. Este último puede ocupar hasta varias decenas de Kilobytes, que se traduce en un amplio porcentaje de la memoria destinada para espacio de programa.

En pocas palabras, el planteamiento del problema o caso de estudio de este trabajo, consiste en resolver la problemática de muchos de los dispositivos de procesamiento como microcontroladores (μC's) y procesadores digitales de señales (DSP's) en cuanto a la dificultad que actualmente presentan para conectarse a una red ethernet y por consiguiente para tener acceso y formar parte activa en la red mundial de internet. De esta manera se integran dos aspectos muy importantes, uno de ellos, la capacidad de control proporcionada por los μC's y DSP's, y el otro, el potencial de comunicación de una red ethernet; además, se elimina la dependencia tradicional de incluir a una computadora personal para que actúe directamente como un "puente" entre la red y un determinado sistema de control.

1.7. Objetivos y Justificación del caso de estudio.

Del planteamiento de este problema se establece el objetivo principal del trabajo, el cual consiste en: Desarrollar una herramienta de comunicación que habilite a cualquier dispositivo microcontrolador, o cualquier otro dispositivo con facilidad de comunicación serial asíncrona a establecer una comunicación con una red ethernet, y por consiguiente, a establecer una comunicación con la red de internet. Todo esto, mediante el uso de dispositivos de bajo costo, de tamaño reducido, y sin recurrir a una PC que actúe como enlace directo entre la red y el sistema digital que posee la comunicación serial.
Para lograr el objetivo principal, como parte del trabajo, se desarrollaron las siguientes partes:

- Un módulo de comunicación que funciona como enlace entre una red Ethernet y un sistema digital con capacidad de comunicación serial asíncrona (RS-232 ó RS-485).
- Librerías de software para PC para establecer una comunicación entre una PC y el módulo de comunicación.
- Librerías de software para el (μC) HC16 para establecer una comunicación directa entre el HC16 y la memoria externa del módulo.
- El acoplamiento de un módulo externo de memoria cuya capacidad incrementa las posibilidades típicas de almacenamiento del los μcontroladores y DSP's

La justificación del caso de estudio puede resumirse en pocas palabras, enunciando algunos puntos claves que ya se han mencionado hasta este punto de este documento, como lo son:

- El prominente uso de microcontroladores, tanto en el presente, como en el pronóstico del futuro, en gran número de aplicaciones especialmente en el área industrial.
- La comunicación serial asíncrona (RS-232 ó RS-485) como un módulo inseparable de los microcontroladores.
- La dificultad actual de un microcontrolador para conectarse a una red ethernet.
- El estándar de ethernet como uno de los estándares más ampliamente aceptados en la implementación de redes de área local. También, como un estándar comúnmente encontrado en la industria.
- La capacidad de expandir el alcance de una red ethernet con su integración a la red global de internet, con lo que se aprovecha toda la infraestructura de comunicación actual en cuanto a líneas, equipos, y protocolos de comunicación esparcidos por el mundo.
- La capacidad de almacenamiento reducida en μcontroladores y DSP's.

Por último, como parte del objetivo principal de este trabajo, en particular se escogió el microcontrolador HC16 para la creación de librerías de comunicación con el módulo propuesto. Esto, debido a la capacidad de procesamiento del HC16, por su amplia aceptación en aplicaciones industriales, y en particular, por el desarrollo de algunos trabajos realizados en el ITESM hasta la fecha con esta herramienta. También como una forma de otorgar a los trabajos efectuados con el HC16 la capacidad de expandir su alcance mediante la publicación de sus resultados a través de la red.

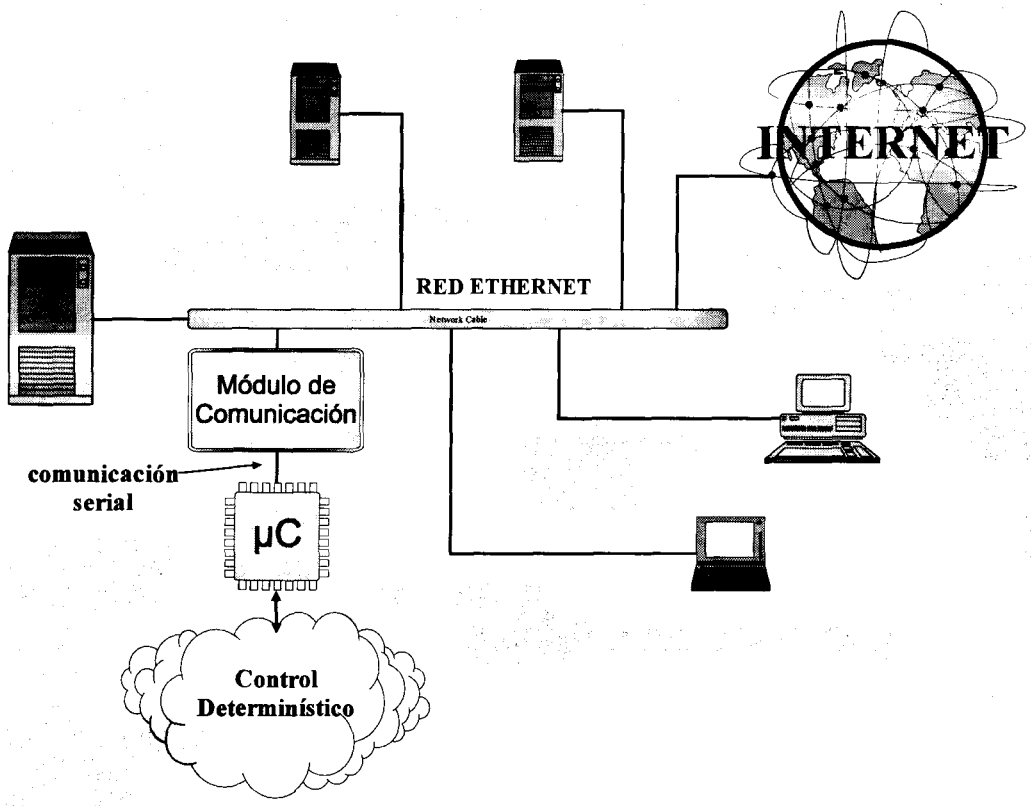


Figura 1.6. Esquema que muestra el objetivo de este trabajo, la integración de un microcontrolador a una red Ethernet, mediante un módulo de comunicación de bajo costo.

Capitulo 2. Redes de Computadoras.

2.1. Conceptos Básicos de Redes.

2.1.1 Partes fundamentales de una red.

Para comprender el funcionamiento básico de una red, primero se define el concepto de una red, y después se exponen brevemente algunas de sus partes.

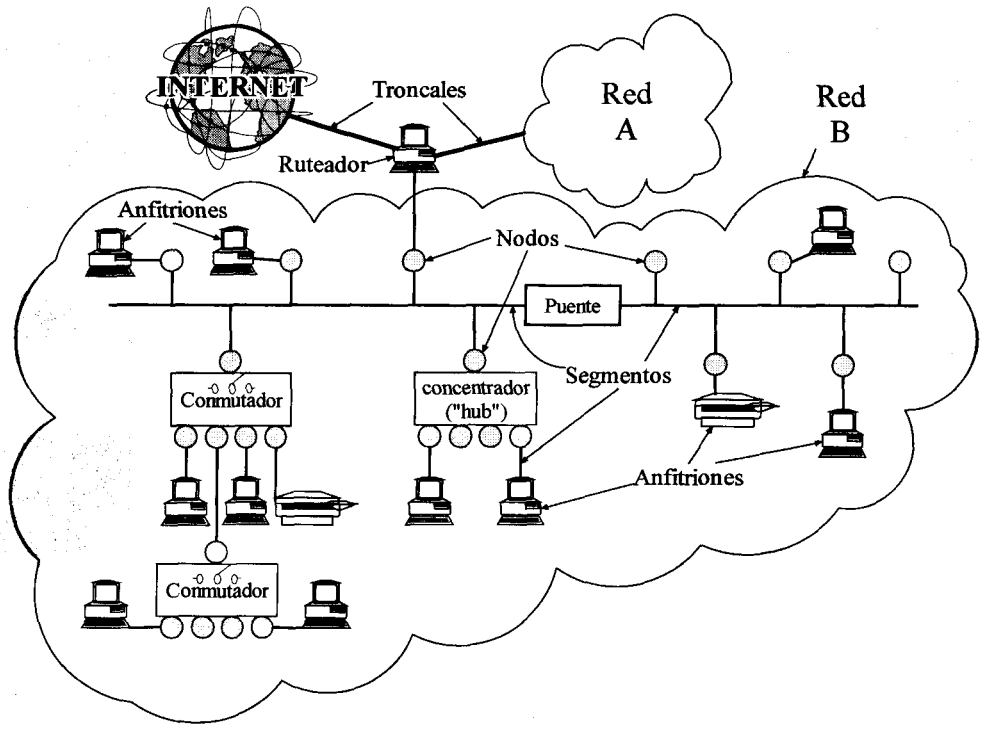


Figura 2.1. Elementos básicos de una red.

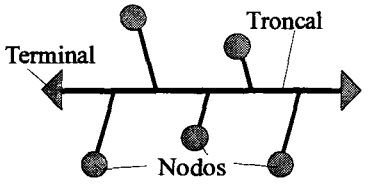
- **Red ("network")**. Una red es un grupo de computadoras conectadas con el propósito de compartir recursos ("hardware" y/o datos) entre ellas.
- **Nodo ("node")**.- un nodo se refiere a un punto de conexión con una red. Por ejemplo, una computadora se conecta a una red a través de un nodo.
- **Anfitrión ("host")**.- un usuario, computadora, o dispositivo, que se conecta a una red.
- **Protocolo ("protocol")**.- es un conjunto de reglas o convenciones predefinidas para establecer una comunicación.
- **Segmento de Red ("segments")**. Se refiere a una porción separada de una red, generalmente los segmentos de una red están separados por elementos como conmutadores, ruteadores, o puentes.
- **Conmutadores ("switches")**.- es un dispositivo que filtra y deja pasar datos entre distintos segmentos de redes.
- **Ruteadores ("routers")**.- es un dispositivo que se encarga de encontrar y de dirigir en la mejor ruta a los datos que se comunican entre distintas redes.

- **Puentes ("bridges")**.- es un dispositivo que interconecta dos segmentos de red de la misma tecnología y que utilizan el mismo protocolo.
- **Concentrador ("hub")**.- dispositivo que se encarga de unir distintos nodos de la red en un sólo punto.
- **Troncal ("backbone")**.- se refiere a la línea de comunicación principal de una red sobre la cual pueden conectarse varios segmentos. Las troncales se caracterizan por ser canales de muy alta velocidad, y comúnmente se utilizan para interconectar redes.

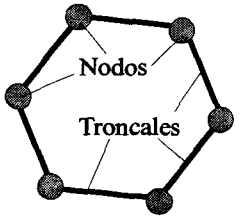
2.1.2. Topología de Redes

Topología de Red. Se refiere a la forma en que se conectan en sí los nodos de una red. A continuación se muestra la estructura de las principales topologías de redes; haciendo la aclaración de que puede haber redes que combinen distintas topologías, o tengan una estructura propia con múltiples enlaces, tal y como sucede en el caso de la red internet.

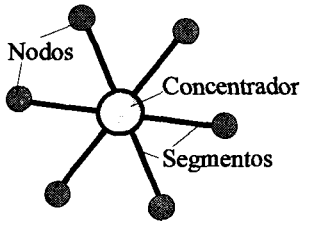
- Topología Bus.- En la topología Bus cada nodo se conecta a un mismo canal de comunicación. La información del bus viaja en ambas direcciones del canal principal de comunicación.



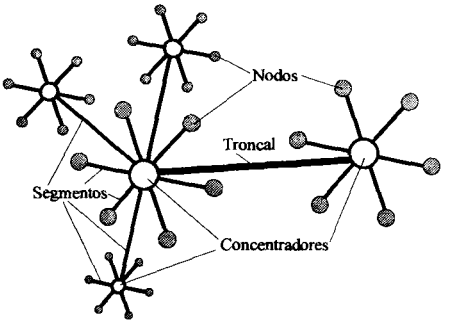
- Topología en Anillo. Esta topología no tiene conexiones terminales, es decir, forma una ruta ininterrumpida y cerrada entre los nodos de una red. La información de la red viaja en un sólo sentido a lo largo de todos sus nodos



- Estrella.- En una topología estrella existe un dispositivo central llamado "concentrador" (hub) al que se conectan todos los nodos de la red. En este tipo de topología no existe una comunicación directa entre los nodos de la red, la comunicación entre los nodos se realiza por medio del concentrador. El concentrador no realiza operaciones de filtrado o de ruteo; sólo se encarga de unir los distintos nodos de la red.



- Bus-Estrella.- Esta topología combina los elementos de las tipologías bus y estrella. Es común que varias redes estrella si interconecten mediante un bus, de tal forma que se incremente el alcance de la red, a la vez que se aprovecha la ventaja de una conexión tipo estrella. También es común que se conecten en forma anidada redes tipo estrella a redes estrella.



2.1.3. Clasificación de Redes según su extensión Geográfica.

Las redes según su extensión geográfica pueden clasificarse en dos categorías generales:

- Redes de Area Local (LAN's).
- Redes de Area Amplia (WAN's).

Aunque no existe un criterio tajante para diferenciar el límite entre una LAN y una WAN el concepto de LAN ("Local Area Network") generalmente se aplica a las redes de computadoras que se encuentra dentro de un mismo edificio, o en un complejo de varios edificios adjuntos, mientras que una WAN ("Wide Area Network") se aplica a redes que comunican a computadoras apartadas entre sí por varios kilómetros, pudiendo estar las computadoras en distintas ciudades, incluso en distintos países o hasta en distintos continentes.

2.2. Red de Ethernet.

En esta sección se hace un breve repaso de la evolución de la tecnología de Ethernet y se exponen algunas de las características principales del medio, como el tipo de acceso al medio, el manejo de direcciones físicas, y el formato de los paquetes de Ethernet.

2.2.1. Antecedentes

La tecnología Ethernet fue creada a principios de los años 70's por la compañía Xerox PARC, pero no fue si no hasta el año de 1978 que la IEEE publicó el estándar que actualmente conocemos, el IEEE 802.3. Desde entonces el estándar de Ethernet ha evolucionado. Originalmente para el cableado físico de la red se utilizaba cable coaxial de media pulgada de diámetro y módulos transceptores externos para conectar las computadoras a la red. Después el cableado de la red se sustituyó por cable coaxial delgado. Con el uso de cable coaxial delgado se eliminó el uso de transceptores externos y en su lugar se utilizaron conectores BNC para conectar las computadoras a la red. En esta etapa, los transceptores realmente se integraron a las tarjetas de red de las computadoras personales. Después, el uso de cable coaxial se sustituyó por el uso de cable de par trenzado, el cual se asemeja a los cables utilizados para los sistemas de teléfono. El par trenzado, también conocido como: "10 Base T", actualmente es la forma

de cableado más común que se utiliza en la implementación de redes Ethernet. La evolución en el tipo de cableado de las redes ethernet, también trajo consigo cambios en la topología de la red; mientras que con el uso de cable coaxial se acostumbra utilizar una topología tipo BUS, con el uso de cable de par trenzado se acostumbra utilizar una topología tipo estrella.

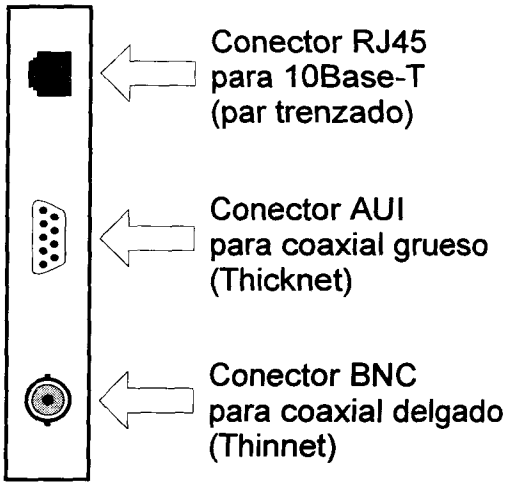


Figura 2.2. Vista hacia el exterior de una tarjeta de red Ethernet que soporta los 3 tipos de cableado.

Además, de los cambios en el tipo de cableado, la tecnología Ethernet ha evolucionado en cuanto su velocidad. Original y tradicionalmente la especificación del medio es de 10 Mbps (Mega bits por segundo), pero actualmente se manejan velocidades de 100 Mbps y hasta de 1Gbps. A pesar de las distintas velocidades del medio de la red, el principio de funcionamiento de Ethernet es el mismo. Ethernet utiliza una comunicación denominada de "conmutación de paquetes" para establecer una conexión entre computadoras. El envío de información a la red se hace en unidades denominadas "paquetes". Los paquetes a su vez incluyen un identificador que sirve de guía para que el hardware de la red pueda enrutar los paquetes hasta su correcto destino.

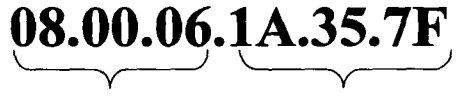
2.2.2. Método de Acceso (CSMA/CD)

La forma de acceso al medio físico de Ethernet se conoce como "Carrier Sense Multiple Access with Collision Detection" (CSMA/CD). Este término indica que varias computadoras pueden accesar a la red simultáneamente ("Multiple Access"). En el cable de red se utiliza una señal portadora para indicar que una computadora esta trasmitiendo ("Carrier Sense"), de ésta forma el resto de las computadoras escuchan la portadora y saben que el cable de la red esta en uso. Cada computadora tiene un tiempo límete para hacer uso del cable de tal forma que el resto de las computadoras conectadas al cable red también tengan oportunidad de transmitir. Aún y con la detección de la señal portadora, debido a los tiempos de propagación de las señales, es común que se presenten colisiones en el intento de 2 o más computadoras por hacer uso del cable de red ("Collision Detection"). La colisión se detecta cuando las computadoras que transmiten a la red escuchan algo distinto a lo que transmitieron. En el caso de una colisión las computadoras participantes dejan de transmitir inmediatamente y esperan un tiempo

aleatorio antes de volver a intentar hacer uso del medio. Para evitar una conglomeración de colisiones y un embotellamiento en la red, se utiliza un algoritmo de espera aleatorio, que además, por cada colisión consecutiva que se presente el algoritmo incrementa el tiempo de espera en forma exponencial. Este algoritmo se implementa dentro del hardware de la tarjeta de red de Ethernet, es decir, es propio del funcionamiento de la tarjeta de red y no tiene que ver con la programación del software de las computadoras de la red.

2.2.3. Direcciones físicas.

Las redes ethernet utilizan un esquema de 48 bits para identificar una dirección. Las direcciones físicas se refieren a un número de identificación único para cada una de las interfaces de red Ethernet. Al referirse a un número de identificación único significa que no existen dos interfaces de red Ethernet que se construyan con una misma dirección física. Para esto, cada fabricante de interfases de red ethernet recibe por parte de la IEEE un rango de números de series permitidos para que construya sus interfases de red.
Con 48 bits es posible tener entonces (2^{48}=) 2.8 x 10^{14} combinaciones distintas para la asignación de direcciones físicas de Ethernet. Para darse una idea de magnitud de este número, si se construyen interfaces de red a una razón de 1000 unidades por segundo de manera continua, es decir, durante todos los días del año sin cesar durante las 24 horas del día, se requieren de 8,925 años para agotar todos los números.
Una dirección física se acostumbra representar en formato hexadecimal por 6 números separados por puntos, cada número con una longitud de 8 bits. Los primeros 3 números hexadecimales (más significativos) identifican a la compañía de manufactura que construye a la interfase, mientras que los 3 números restantes (menos significativos) identifican el número de parte que internamente maneja la compañía de manufactura.



Los primeros (3) números hexadecimales identifican a la compañía de manufactura ej., Siemens = 08.00.06	Los segundos (3) números hexadecimales definen el # de parte interno de la compañía de manufactura N/S = 1A.35.7F

Figura 2.3. Formato de una dirección física de Ethernet de 48 bits.

Dentro de una red Ethernet el direccionamiento y envío de paquetes se hace a través de direcciones físicas. En una red ethernet hay que recordar que la información transmitida al medio, es decir, al cable de Ethernet, se hace presente en todos los nodos del segmento de la red; los paquetes enviados a la red incluyen un campo que contiene la dirección física a quien va dirigido el paquete, de tal forma que cada tarjeta de red se ocupe de filtrar los paquetes que no le corresponden y deje pasar sólo los paquetes que coincidan con su misma dirección física. Existe un caso especial cuando en un paquete se define la dirección destino como FF FF FF FF FF FF hexadecimal, en este caso, el paquete va dirigido a todas las computadoras de la red.

Por otra parte, los conmutadores o "switches" que forman parte de la infraestructura de una red ethernet también utilizan direcciones físicas para enrutar sus paquetes.

2.2.4. Trama de Ethernet.

La información que viaja a través de una red Ethernet lo hace a través de "paquetes". Un paquete de Ethernet puede ser de longitud variable pero siempre contiene los mismos "campos". Los paquetes de Ethernet esta compuestos por los 6 campos que se muestran en la figura 2.4.

Header	Dirección destino	Dirección fuente	Longitud del paquete	Tipo de Paquete	Checksum
8 bytes	6 bytes	6 bytes	2 bytes	64-1500 bytes	4 bytes

Figura 2.4. Formato de un paquete de Ethernet.

El campo de "header" se utiliza para efectos de sincronía en el envío de paquetes a la red. El campo de la "dirección destino" (de 6 bytes = 48 bits) contiene la dirección física de la interfase de red a quien va dirigido el paquete. De igual forma, en el campo de la "dirección fuente" se incluye la dirección física de la interfase de red que envía el paquete. Como se mencionó, un paquete de Ethernet puede tener una longitud variable que fluctúa en el rango de 64 a 1518 bytes. La longitud del paquete se especifica en el cuarto campo denominado: "longitud del paquete". En el campo "tipo de paquete" se incluyen los datos a enviar a la red, y además, dentro de este campo se destinan 2 bytes para identificar el tipo de paquete; aunque el paquete por si solo sugiere que es del tipo de ethernet, en el mismo medio de la red ethernet pueden utilizarse diferentes protocolos, por ejemplo, en una red ethernet puede utilizarse protocolos de distintos fabricantes, o protocolos propios diseñados con fines de investigación o para aplicaciones específicas; los 2 bytes de identificación (dentro del campo tipo de paquete) se utilizan entonces para diferenciar los paquetes correspondientes a los distintos protocolos. Por último, todos los datos de los campos hasta ahora mencionados se suman en una representación de 4 bytes y el resultado de la suma se guardan en el último campo del paquete llamado "Checksum". Este campo se utiliza para verificar la integridad del paquete una vez que llega a su destino.

2.3. Internet.

La palabra internet surge del concepto "internetworking" que significa redes interconectadas o interconexión de redes. Internet surge a partir de un proyecto inicialmente con fines militares en Estados Unidos en el año de 1969, su investigación se extendió hasta los laboratorios de universidades en donde tuvo una gran aceptación y un gran impulso en su desarrollo hasta expandirse hoy en día en una comunicación para todo tipo de fines, desde comunicación entre corporativos, para propósitos comerciales, para investigación, entretenimiento, etc. y con un número actual de más de 50 millones de usuarios.

El concepto de internet consiste básicamente en comunicar distintos tipos de redes, independientemente de que las redes sean físicamente distintas entre sí. La parte medular de internet, donde ha radicado su éxito, consiste en el desarrollo de un conjunto de protocolos llamados (TCP/IP) que permiten que se lleva a cabo el enrutamiento de

624352

paquetes entre los distintos tipos de redes proporcionando el efecto de tratar con una sola red global. Es muy importante resaltar que internet es una red enfocada a la "conmutación de paquetes", esto significa que los paquetes enviados de un punto a otro pueden tomar distintas rutas o trayectorias alternativas para llegar a su destino. Por ejemplo, en situaciones de mucho tráfico, o incluso en la situación de ruptura en algún canal de comunicación, la comunicación entre 2 puntos aún puede llevarse a cabo a través de rutas alternas.

Otra ventaja de internet, es que es un sistema abierto con especificaciones de dominio público; prácticamente cualquier organización o persona puede suscribir y conectar su propia red a internet (ya que internet es una red de redes), y también puede desarrollar programas de comunicación que hagan uso de la red con la ventaja de poder compartir información a lo largo de todo el mundo.

Desde el punto de vista del usuario, internet es una sola red virtual aunque internamente este compuesta por muchas redes interconectadas en forma cooperativa. Un usuario tiene la capacidad de enviar datos de un punto a otro aunque los puntos no estén conectados directamente, pero sí de alguna forma a través de redes intermedias. El usuario puede hacer uso de dichas redes intermedias en forma transparente. Esta es una característica del diseño de la red hecha en base a capas funcionales.

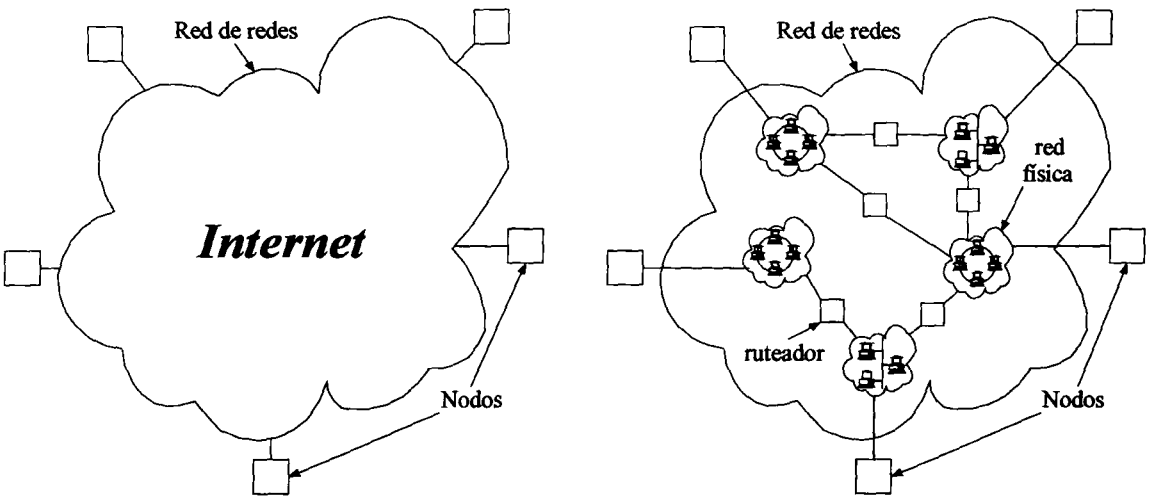


Figura 2.5. Concepto de internet como una sola red virtual, y como una red compuesta de redes TCP/IP.

2.4. Modelo de capas de una red TCP/IP

Internet es una red TCP/IP, es decir, su funcionamiento se basa en el conjunto de protocolos TCP/IP. La mayoría de las tecnologías de redes actuales soportan el manejo de TCP/IP para poder integrarse a la red global de redes. TCP/IP actualmente es un estándar adoptado universalmente en términos de manejo de protocolo de redes, y no es la excepción en el caso de la tecnología ethernet, misma que se utiliza en el desarrollo de este trabajo.

En esta sección se expone el modelo de una red TCP/IP estructurada en forma de capas explicando brevemente el funcionamiento de cada una de ellas.
Una red TCP/IP esta compuesta por 5 capas funcionales, las cuales se muestran en la siguiente figura.

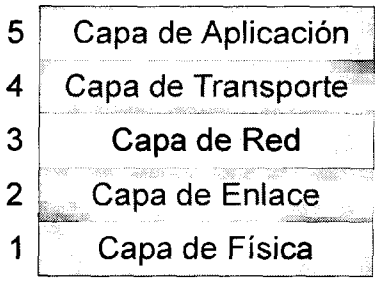


Figura 2.6. Modelo de capas de una red TCP/IP

A continuación se describe en forma breve el funcionamiento de las 5 capas del modelo de red TCP/IP.

2.4.1. La capa física

La capa física se refiere al medio a través del cual viajan los datos de una comunicación en la red. El medio puede ser un conductor eléctrico, un cable de fibra óptica, incluso el mismo ambiente por donde viajan señales de radio o de microondas. Por ejemplo, en el caso de una red ethernet, la capa física esta formada por los cables que interconectan las computadoras de la red.

2.4.2. La capa de enlace.

Como su nombre lo indica, esta capa se encarga de enlazar la capa física con la capa de red que es la capa adyacente a la capa de enlace. Utilizando como ejemplo el caso de una red ethernet la capa de enlace esta constituida por la tarjeta de red que se encarga de verificar la integridad de los datos recibidos a través de la capa física, y por otro lado, una vez ya ha verificados los datos recibidos, se encarga de entregarlos a la capa de red.

2.4.3. La capa de red.

La capa de red es la parte medular del modelo de una red TCP/IP, en esta capa se manejan las unidades básicas de datos que se comunican a través de una red TCP/IP; dichas unidades se denominan "paquetes IP" y son el motor de la comunicación a través de la red. Toda la comunicación de una red TCP/IP, independientemente de los protocolos que originen la comunicación en las capas superiores, se realiza a través de paquetes IP. En esta capa es donde se manejan las direcciones IP (ver sección 2.5), por ejemplo: 131.178.142.105. Y también donde se utiliza el protocolo IP ("Internet Protocol") para el enrutamiento de datos desde un punto origen a otro punto de destino.

2.4.4. La capa de Transporte.

En esta capa funciona como un enlace entre la capa de red y la capa de aplicación, es decir, se encarga de entregar (o transportar) los datos recibidos de la capa de red a la capa de aplicación. En esta capa se llevan operaciones de segmentación de datos para poder transmitirlos por la red, y cuando se solicita, también se realizan operaciones de control para asegurarse de que los datos lleguen a su destino íntegramente y en la secuencia correcta, tal es el caso del protocolo TCP ("Transmission Control Protocol").

2.4.5. La capa de Aplicación.

Esta capa representa a los programas o aplicaciones destinadas a un usuario y que hacen uso de la red, como por ejemplo, los programas para envío de correo electrónico, para navegar en la red, para conversar o "chatear", etc.
En esta capa a los datos comunicados por la red se les da un formato que sea entendible o tenga sentido para el usuario.

2.5. Direcciones IP.

En esta sección se expone en que consiste una dirección IP y algunos tipos de direcciones IP; esto, a manera de preámbulo para comprender el funcionamiento del protocolo IP y sus protocolos derivados, así como también para comprender cómo se lleva a cabo el enrutamiento de datos en una red TCP/IP.
Una "dirección IP" identifica a cada computadora o dispositivo conectado a una red. Por regla, no debe de haber dos o más direcciones IP iguales en una misma red. Una dirección IP es un identificador único parecido a un número telefónico de la red de teléfono, y se expresa por cuatro números decimales separados por puntos, como en los siguientes dos ejemplos:

131.178.142.105
10.10.1.7

Cada número decimal esta en el rango de 0 a 255 debido a que es representado por un entero de 8 bits; por lo tanto, para representar una dirección IP se requieren en total de 32 bits. Las mismas direcciones mostradas arriba representadas en 32 bits son las siguientes:

10000011.10110010.10001110.01101001
00001010.00001010.00000001.00000111

Una red local utiliza un grupo de direcciones IP contiguas y cercanas entre sí. Dependiendo de la magnitud de la red, es decir, del número de anfitriones (posibles usuarios) de la red, las direcciones IP tienen varias clases: A, B, C, D, y E.
Las clases se identifican mediante el valor de los primeros 5 bits de mayor orden de una dirección IP. Cada clase tiene asociado un cierto número de anfitriones junto con un cierto número de posibles redes según la forma en que se administran los 32 bits de una dirección IP. Las direcciones tipo A, B, y C se utilizan para identificar a un cierto anfitrión de red en específico. Las direcciones de la clase D se utilizan para efectos de transmisión múltiple, es decir, para transmisiones a varias computadoras de una sola red en forma simultánea, mientras que las direcciones de tipo E son de uso reservado.

Clase	valor de los bits de mayor orden (dedicados para identificar el tipo de red)	numero de bits para identificar una red / numero de redes posibles	numero de bits para identificar un anfitrion / número de anfitriones posibles
A	0 - - - -	7 / 128	24 / 16,777,216
B	10 - - -	14 / 16,384	16 / 65,536
C	110 - -	21 / 2,097,152	8 / 256
D	1110 -	para difusión múltiple	
E	11110	reservado	

Tabla 2.1. Tipos de direcciones IP con sus respectivos números de redes y anfitriones posibles.

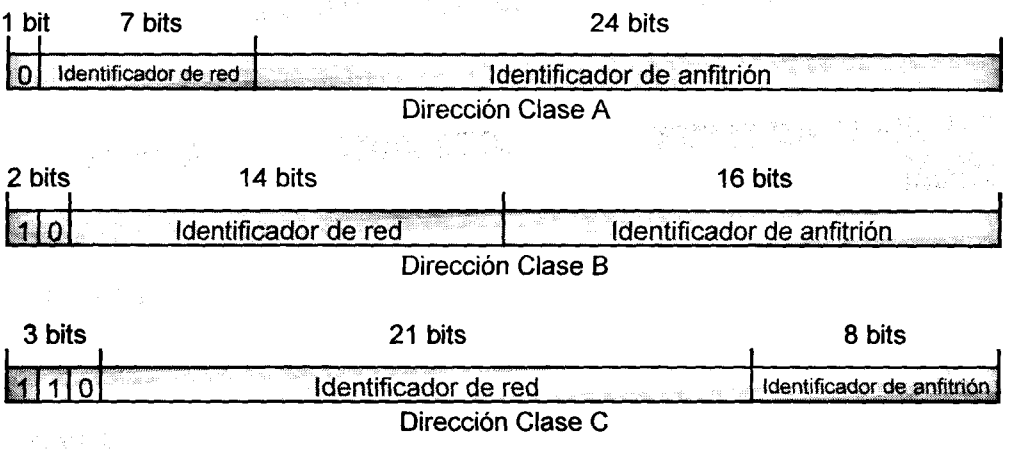


Figura 2.7. Esquema de la distribución de los 32 bits de una dirección IP de acuerdo a la clase de dirección, A, B, o C.

2.5.1. La máscara de red ("netmask").

En este rubro la "mascara de red" ("netmask") cuya representación es igual a la de una dirección IP, se utiliza para definir el número de anfitriones de una red. Por ejemplo, la máscara de red: 255.255.255.0 que tiene los 8 bits menos significativos igual a cero, indica que los último 8 bits se utilizan para definir las direcciones de los anfitriones locales de una red.

Por ejemplo, para la máscara de red: 255.255.255.0, dada una dirección de red, desde la dirección: 131.178.142.0, hasta la dirección: 131.178.142.255 son direcciones locales de la red. Cualquier dirección fuera de ese rango, por ejemplo 131.178.10.200 se trata de una dirección externa a la red local.

2.5.2. Direcciones IP especiales.

Existen algunas direcciones IP con usos especiales, las cuales se enlistan a continuación:

- **127.0.0.1** Esta dirección se refiere al mismo anfitrión local, es decir a la tarjeta de la computadora en uso. Esta dirección se utiliza para probar si la computadora local esta

conectada a la red mediante un acceso de prueba ("ping") que proporciona un mensaje de retroalimentación.

- **0.0.0.0** se utiliza para definir un anfitrión desconocido. Esta dirección es utilizada por algunos programas de red antes de que el anfitrión obtenga una dirección IP.
- **255.255.255.255** Esta dirección se utiliza para enviar un mensaje a todos los anfitriones de una red ("broadcast address"). Generalmente la dirección más alta de una red se utiliza para enviar mensajes a todos los anfitriones de la red, mientras que la dirección más baja se utiliza para identificar a la red.

2.5.3. Direcciones IP Privadas.

Además de las direcciones mencionadas existen direcciones "IP privadas". Estas direcciones se utilizan en la implementación de redes locales aisladas del exterior, es decir, que no se están conectadas a la red pública de internet. Por tal motivo, el uso y asignación de direcciones en estos rangos queda completamente libre para el usuario.
Existen 3 rangos de direcciones privadas de acuerdo a las 3 clases de direcciones IP.

- **10.n.n.n** (1 red para redes clase A).
- **172.16.n.n** a **172.31.n.n** (16 redes para redes clase B)
- **192.168.0.n** a **192.168.255.n** (256 redes para redes clase C)

2.5.4. Direcciones IP dinámicas y fijas.

En ocasiones las direcciones IP se asignan de manera temporal, el caso típico es cuando por ejemplo un usuario se conecta a la red "internet" a través de una conexión telefónica. El proveedor del servicio le asigna una dirección IP que dura mientras permanezca la conexión. Una vez que el usuario se desconecte la dirección IP se libera y puede ser asignada a otro usuario distinto. Cada vez que un usuario se conecte a la red no necesariamente recibirá la misma dirección IP. Otro ejemplo es la red del ITESM, Campus Monterrey, que asigna una dirección IP dinámicamente a los alumnos que conectan sus computadoras portátiles a la red. Aunque la dirección IP tome un valor distinto al iniciar una sesión de red, el número de IP sigue siendo una referencia para poder comunicar información de un punto a otro.
Opuestamente a las direcciones dinámicas, existen las direcciones IP fijas. Estas direcciones son asignadas a los equipos servidores que otorgan algún servicio, de tal forma que puedan ser localizados desde cualquier punto. Generalmente la dirección de un servidor no cambia, o si cambia, lo hace muy a lo largo, ya que para que un servidor pueda cumplir con su propósito en un principio debe poder ser localizado.

2.6. Los Protocolos de capas de una red TCP/IP

El modelo de red expuesto en la sección 2.4 se refiere al de una red TCP/IP, donde TCP/IP comprende una serie de protocolos. Los protocolos TCP/IP se distribuyen a lo largo de las capas del modelo de acuerdo al esquema que se muestra en la figura 2.8. En la misma figura también se indican las diferentes líneas de comunicación que existen entre los protocolos y los elementos de las capas.
En este trabajo en particular el interés principal se enfoca a los protocolos de la capa de transporte que es la capa sobre la cual se ha construido la aplicación de este trabajo de tesis. Sin embargo, a continuación se hace un repaso a lo largo de cada una de las capas para explicar brevemente el funcionamiento de sus respectivos protocolos.

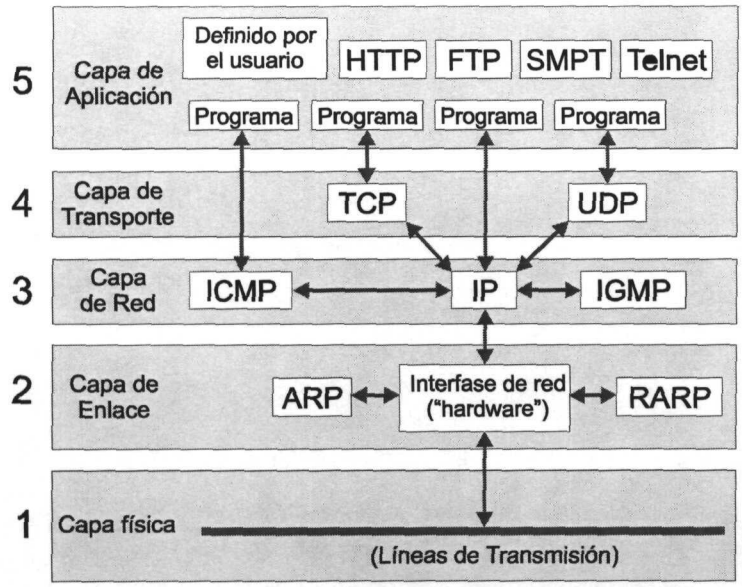


Figura 2.8. Modelo de capas para una red TCP/IP que incluye los protocolos y componentes de cada capa.

2.6.1. Capa física

En la capa física se manejan protocolos de bajo nivel que dependen de la tecnología de red utilizada. Generalmente estos protocolos de bajo nivel van acompañados de un esquema de modulación que les permite trasladar datos en forma confiable de un punto a otro. Tanto los protocolos como las técnicas de modulación varían de acuerdo al medio utilizado para la transmisión. En esta capa lo importante es que los protocolos de bajo nivel de esta capa sean capaces de trasladar la información binaria en forma de 1's y 0's de un lugar a otro.

2.6.2. Capa de enlace.

En esta capa se manejan 2 protocolos: ARP y RARP

El protocolo ARP ("Address Resolution Protocol"), se utiliza para encontrar la dirección física de una dirección IP. La dirección física corresponde a un número único de 48 bits que identifica a una interfase de red, mientras que la dirección IP corresponde a una dirección lógica dentro de una red.

El protocolo RARP ("Reverse Address Resolution Protocol") se encarga de realizar la operación inversa, es decir, se encarga de relacionar una dirección IP de la capa de red con una dirección física de la capa física.

La función de este par de protocolos se resume en enlazar la dirección física de una interfase de red con una "dirección IP" de la capa de red.

2.6.3. Capa de red.

En esta capa se manejan 3 protocolos, IP ("internet protocol"), ICMP ("Internet Control Message Protocol"), IGMP ("Internet Group Management Protocol"). De los 3 protocolos, el principal es protocolo IP, mientras que los otros 2 funcionan como apoyo.
El protocolo IP se encarga de enviar "paquetes" de datos de una dirección origen a una dirección destino. Los paquetes IP constituyen la unidad básica de una red TCP/IP de conmutación de paquetes. Incluso los protocolos ICMP e IGMP de esta misma capa se basan en el uso de paquetes IP. El formato de una paquete IP contiene un encabezado de 20 bytes en el que se incluye la dirección IP origen, la dirección IP destino, y el tipo de paquete. Los paquetes IP se transportan de la capa de red de un punto origen hasta la capa de red de un punto destino. Sin embargo, los paquetes IP enviados no tienen garantía de llegar a su destino, o de llegar en el mismo orden en que se enviaron; en otras palabras el protocolo IP es un protocolo no confiable. El protocolo IP no garantiza la llegada de un paquete enviado de un punto a otro, pero sí se encarga de checar la integridad de los paquetes recibidos mediante un "checksum"; si los paquetes recibidos no pasan la verificación de errores, entonces no son tomados en cuenta y se descartan, por eso es un protocolo no confiable. Si se desea realizar un envío de manera confiable a través de la red se recurre entonces a los protocolos de mayor nivel de las capas superiores.
Otra característica de los paquetes IP es que pueden ser fragmentados cuando alcanzan redes cuyo hardware de la capa física no puede manejar el tamaño de los paquetes; en estos casos, cada paquete IP se fragmenta en varios paquetes de menor tamaño, cada uno con su propio encabezado IP para que circulen en forma independiente en la red. Después, los paquetes vuelven a integrarse al llegar a la capa de red de su destino.
A manera de paréntesis, dentro del encabezado de un paquete IP también se incluye un campo de un byte para asignar el "tiempo de vida" (TTL) del paquete. Este campo es útil para evitar que en la red circulen paquetes indefinidamente que no tengan o no encuentran su dirección destino, evitando así congestionamientos y problemas de tráfico.
Regresando a los demás protocolos de la capa de red, el protocolo ICMP consiste en una serie de mensajes que se utilizan para verificar el estado y controlar el flujo de otros protocolos. Un ejemplo típico del uso de ICMP es la operación "ping" que se utiliza para verificar la presencia de un anfitrión en la red.

```
C:\>ping 127.0.0.1

Pinging 127.0.0.1 with 32 bytes of data:

Reply from 127.0.0.1: bytes=32 time<10ms TTL=32
Reply from 127.0.0.1: bytes=32 time<10ms TTL=32
Reply from 127.0.0.1: bytes=32 time=1ms TTL=32
Reply from 127.0.0.1: bytes=32 time<10ms TTL=32
```

Figura 2.9. Ejemplo de la operación ping para detectar la presencia del mismo anfitrión local mediante la dirección IP especial: 127.0.0.1

Si se observa la operación "ping" es un programa de la capa de aplicación que se comunica directamente con la capa de red (protocolo ICMP). La comunicación directa entre estas dos capas puede apreciarse en el esquema de la figura 2.8.
Por último, el protocolo IGMP de la capa de red se utiliza para la transmisión de mensajes múltiples a dos o más sistemas.

2.6.4. Capa de Transporte.

En la capa de transporte se utilizan los protocolos TCP ("Transmission Control Protocol") y UDP ("Universal Datagram Protocol"). Es importante no confundir el término TCP/IP que se refiere al conjunto de todos los protocolos de la red, con el término TCP que se refiere sólo a un protocolo de la capa de transporte. Sin embargo, es un hecho que el término TCP/IP surgió de la unión de los nombres de los protocolos TCP de la capa de transporte y del protocolo IP de la capa de red.
El protocolo TCP se apoya en el protocolo IP para transferir datos en una red. TCP es un protocolo orientado a conexión, es decir, para transmitir datos de un punto a otro primeramente establece una sesión que simula una conexión directa entre los puntos, después, los datos a enviar se fragmentan a una longitud apropiada para que puedan ser enviados en forma de paquetes IP; el protocolo TCP se encarga de fraccionar los mensajes, de enviar los datos, de darles seguimiento durante la transmisión para asegurarse de que lleguen a su destino libre de errores y en el mismo orden que fueron enviados. Para esto, como parte del mismo protocolo, tanto el transmisor como el receptor se envían mensajes de confirmación que certifican la recepción de los datos en forma ordenada y sin errores.
El protocolo TCP es un protocolo confiable, ya que garantiza que el envío de un mensaje llegará a su destino libre de errores y en la misma secuencia en el que fue enviado. Vale la pena destacar que TCP es un protocolo confiable a pesar de que utiliza como base un protocolo no confiable (IP) para transmitir datos.
El protocolo UDP es un protocolo "sin conexión", es decir, no establece una conexión punto a punto para transmitir datos. Aún así este protocolo para ciertas aplicaciones es más eficiente y ofrece la ventaja de poder transmitir mensajes a múltiples destinos simultáneamente, lo que se conoce como envíos "broadcast".
Como en todo, este protocolo tiene sus ventajas y sus desventajas, por una parte proporciona una comunicación más rápida ya que no utiliza mensajes de confirmación para asegurarse de establecer una conexión previa y de que los datos lleguen a su destino. UDP simplemente transmite a la red y si la dirección destino no existe o si el destinatario no esta "escuchando", o si la dirección destino no puede localizarse, los mensajes enviados se deshechan. Otra característica de UDP es que el envío de paquetes consecutivos no se garantiza que lleguen a su destino en el mismo orden en que fueron enviados (similarmente al funcionamiento del protocolo IP), también, UDP, en el envío de paquetes de datos grandes por arriba de un cierto tamaño, dependiendo del hardware de la red, no garantiza que lleguen completos o que lleguen del todo.

Tanto en el protocolo TCP como en el protocolo UDP se manejan 3 parámetros en común para llevar a cabo un envío de datos: los dos primeros de ellos: dirección IP de origen y dirección IP destino, ya han sido comentados, pero además se utiliza como tercer parámetro un "número de puerto", el cual, se representa por un entero de 16 bits, es decir,

se tienen 65,536 puertos disponibles para realizar un envío. El receptor debe de estar "sintonizado" al mismo puerto que definió el transmisor para poder recibir los datos. Existen aplicaciones conocidas que utilizan puertos asignados en forma fija, algunas de estas aplicaciones se muestran en la tabla 2.2.

Servicio	Puerto #	Descripción
ftp	21	Protocolo para transferencia de archivos ("File Transfer Protocol")
telnet	23	Acceso a Telnet
http	80	Acceso a la "World Wide Web" ("Hyper Text Transfer Protocol")
pop3	110	Recepción de correo electrónico ("Post Office Protocol - v. 3")

Tabla 2.2. Algunos servicios de red comunes y sus respectivos puertos asociados por defecto.

De hecho, existen tres rangos de puertos:

- Del 0 al 1023 son "puertos conocidos" utilizados por las aplicaciones comunes de red como las que se muestran en la tabla 2.2.
- Del 1024 al 49151 son "puertos registrados" para aplicaciones desarrolladas por programadores que quieran registrar oficialmente el uso de sus aplicaciones. Un ejemplo es la aplicación "Data Socket Protocol" (DSTP) de National Instruments que tiene registrado el puerto #3015.
- Del 49152 al 65536 son "puertos dinámicos" o "puertos privados" que por el momento son de uso libre para fines de experimentación o cualquier otro uso.

Se ha puesto especial énfasis en esta sección dedicada a los protocolos de la capa de transporte (TCP y UDP) ya que son los protocolos con los que se han trabajado para el desarrollo de este trabajo de tesis. Resumiendo algunos conceptos de estos protocolos se tiene que:

- TCP es un protocolo confiable orientado a una conexión punto a punto, que garantiza la entrega de mensajes libre de errores y en forma ordenada.
- UDP es un protocolo sin conexión que no garantiza la entrega de mensajes, pero es más ágil en cuanto a velocidad de transferencia y además permite el envío simultáneo a múltiples destinos ("broadcast").
- Ambos protocolos (TCP y UDP) para realizar un envío deben definir un número de puerto además de una dirección IP destino; en el mensaje enviado también se incluye automáticamente la dirección IP del origen del mensaje.

2.6.5. Capa de Aplicación.

En la capa de aplicación se encuentran los protocolos de las aplicaciones; puede haber protocolos de aplicaciones conocidas como: FTP, HTTP, Telnet, SMTP, POP3, etc. o también puede haber protocolos creados por el usuario para el envío de datos a través de la red. En cualquier caso, los protocolos de la capa de aplicación generalmente se apoyan

en los protocolos de la capa de transporte (TCP y UDP) para hacer uso de la red. Para este trabajo de tesis precisamente se utilizan ambos protocolos de la capa de transporte, TCP y UDP, y a partir de ellos se desarrolla otro protocolo de uso específico que es el que se encarga del manejo del módulo de comunicación propuesto en este trabajo.

2.7. Enrutamiento de paquetes IP.

El gran potencial de una red TCP/IP consiste en poder enviar datos a cualquier punto de la red con tan sólo saber la dirección IP del punto destino. Los paquetes IP enviados a una red pueden ser entregados básicamente de dos formas:

- En forma Directa.
- En forma Indirecta.

La forma en que se entreguen los paquetes dependen de 3 elementos que son: la dirección IP destino, dirección IP origen, y la máscara de red ("netmask").

Si la dirección destino se encuentra dentro de la misma red física, la entrega de paquetes se hace en forma directa; en el caso contrario, si la dirección destino se refiere a otra red física distinta el paquete se entrega en forma indirecta.

El uso de la máscara de red se utiliza como auxiliar para determinar si una dirección IP pertenece a la misma red o si pertenece a una red física distinta (ver sección 2.5.1: "máscara de red").

2.7.1. Entrega en forma directa.

La entrega en forma directa utiliza un mapeo de direcciones físicas; la red se encarga de crear una tabla de mapeo que relacione las direcciones IP locales con las direcciones físicas de la red, de esta forma, si un envío se destina a una dirección IP dentro de la misma red física, la entrega se hace en forma directa a la dirección física que corresponda a la dirección IP destino.

2.7.2. Entrega en forma indirecta.

Cuando el envío de un paquete no pertenece a la red local, entonces se realiza una entrega en forma indirecta. En la entrega indirecta se requiere del uso de un "ruteador" para hacer llegar el paquete IP a su red destino. Un ruteador es un dispositivo, generalmente una computadora, que se encarga de interconectar redes físicas para transferir información entre ellas. Las redes interconectadas inclusive pueden ser de distintas tecnologías. Los ruteadores contienen tablas de enrutamiento que identifican la siguiente dirección en la trayectoria de un paquete rumbo a su destino. Generalmente las direcciones de las tablas de un ruteador consisten en direcciones de redes y no direcciones de anfitriones; esto, con el fin de reducir la cantidad de información que se almacena en las tablas de enrutamiento. Cuando un paquete IP se envía a una red, la misma red que recibe el paquete verifica si va dirigido a un anfitrión local, de ser así hace una entrega directa, de lo contrario el paquete es enrutado nuevamente hacia otra red para que continúe su camino rumbo a su destino. En ocasiones la entrega de un paquete IP requiere pasar por varias redes intermedias y por varios ruteadores antes de llegar a su destino. Finalmente cualquier transmisión indirecta de un paquete IP termina en una entrega directa, misma que ejecuta el último ruteador cuando hace entrega del paquete IP a la computadora anfitrión destino. Esto significa que una entrega indirecta finalmente implica una entrega

directa; la diferencia entre los dos tipos de entrega radica entonces en que una entrega indirecta forzosamente hace uso de un ruteador, mientras que una entrega directa no.

2.7.3. Compuerta ("gateway")

Un ruteador tiene una dirección IP al igual que el resto de los anfitriones de la red; el acceso al ruteador se realiza mediante una dirección IP comúnmente conocida como "gateway' o compuerta. En otras palabras, el término "gateway" se utiliza para identificar la dirección IP de ruteador, mientras que el término "ruteador" se refiere al dispositivo físico que se encarga del ruteo.

2.7.4. Ejemplo de ruteo.

El tema de ruteo es muy amplio en incluye algunas variantes, inclusive el ruteo puede ser de carácter dinámico ante situaciones de tráfico con el fin de que los paquetes lleguen a su destino de una manera más rápida. La idea de esta sección no es abarcar todos esos detalles, si no más bien exponer en que consiste el principio básico de ruteo, mismo que permite que una red local se comunique con otras redes y con la red global de internet.
En la figura 2.10. se muestra un esquema de una cuatro redes unidas por tres ruteadores (A,B,C).

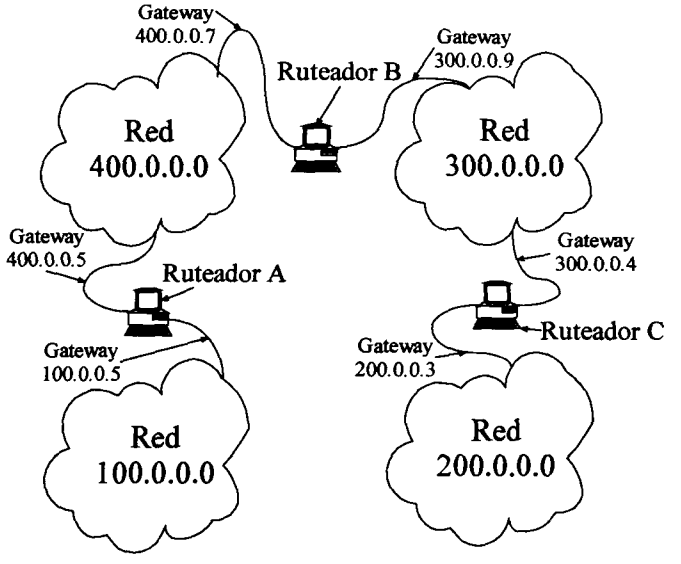


Figura 2.10. Ejemplo de una red global compuesta por 4 subredes y 3 ruteadores.

anfitrión destino de la red:	Siguiente salto del Ruteador A	Siguiente salto del Ruteador B	Siguiente salto del Ruteador C
100.0.0.0	Entrega Directa	400.0.0.7	300.0.0.4
400.0.0.0	Entrega Directa	Entrega Directa	300.0.0.4
300.0.0.0	400.0.0.5	Entrega Directa	Entrega Directa
200.0.0.0	400.0.0.5	300.0.0.9	Entrega Directa

Tabla 2.3. Tabla del ruteo para los ruteadores A, B, y C.

La tabla 2.3. muestra la operación o ruta que efectúa cada uno de los ruteadores de la red de la figura 2.10. según la dirección del anfitrión destino al cual se dirige un paquete IP. Por ejemplo suponiendo que el anfitrión 100.0.0.25 (de la red 10.0.0.0) envía un paquete al anfitrión 300.0.0.48 (de la red 300.0.0.0) la ruta del paquete sería la siguiente:
100.0.0.25 (origen) → 100.0.0.5 ("gateway" del Ruteador A) → 400.0.0.7 ("gateway" del Ruteador B) → 300.0.0.48 (entrega directa).
Para el usuario de la red, la trayectoria del paquete IP es transparente y el efecto de la transmisión es una transmisión directa del punto origen al punto destino:
100.0.0.25 (origen) → 300.0.0.48 (destino)

2.7.5. Servicio de Nombre de Dominios.

En internet existe un servicio llamado DNS ("Domain Name Service") que es proporcionado precisamente por "servidores de nombre de dominios" o servidores DNS. El servicio consiste en traducir nombres de dominios a direcciones IP. Los nombres de dominios básicamente se refieren a una serie de caracteres de texto que son más fáciles de recordar que una dirección IP. Los nombres de dominios de alguna forma sugieren algún asunto, actividad o función; en este caso, lo importante es que un nombre expresado en texto puede relacionarse y utilizarse como un equivalente a una dirección IP gracias al servicio DNS. Por ejemplo es mucho más fácil recordar la dirección:
www.cnn.com (servicio "world wide web", organización "cnn", com "fines comerciales") que recordar la dirección IP: 64.236.16.116
El sistema de nomenclatura de dominios esta formada por una base de datos distribuida, es decir, los datos que proporcionan las equivalencias entre nombres y direcciones IP, se encuentran físicamente distribuidos en varias partes de la red a través de servidores DNS (computadoras). Cualquier persona, institución u organización puede registrar un nombre de dominio en la red de tal forma que su computadora (o servicio) pueda ser localizada por cualquier usuario de la red en el mundo. Existen negocios e instituciones que otorgan este tipo de servicios. La mayoría de ellos están disponibles a través de la red, y por un costo accesible puede darse de alta el nombre de un dominio en la red, generalmente el registro de un dominio se realiza por períodos anuales.
El tema de los servidores de nombre de dominio, también es un tema amplio e interesante, pero para el enfoque de este trabajo sólo se requiere saber:

- Que cualquier nombre de dominio (que no este registrado) puede darse de alta en internet. Por ejemplo:
  www.enrique_ramos.com, correo@enrique_ramos.com, etc.
- Que el uso de un nombre de dominio equivale al uso de una dirección IP.
- Cualquier usuario de internet en el mundo puede referirse a una dirección IP mediante el nombre de dominio asociado a la dirección IP.

Capítulo 3. El Módulo de Comunicación.

3.1. La Tarjeta de Desarrollo RCM2100.

El "módulo de comunicación" esta construido a partir de una tarjeta de desarrollo de la compañía "Rabbitsemiconductor". A su vez, el núcleo de la tarjeta consiste en un procesador "Rabbit 2000", el cual, cuenta con un set de instrucciones muy similar al Z80 de Zilog pero con algunas mejoras, principalmente en cuanto a velocidad y a su arquitectura interna en la que se incluye la adición de algunos registros alternos para facilitar la programación del microprocesador en lenguaje C.
La tarjeta de desarrollo utilizada corresponde al modelo "RCM2100" cuyo esquema general se muestra en la siguiente figura:

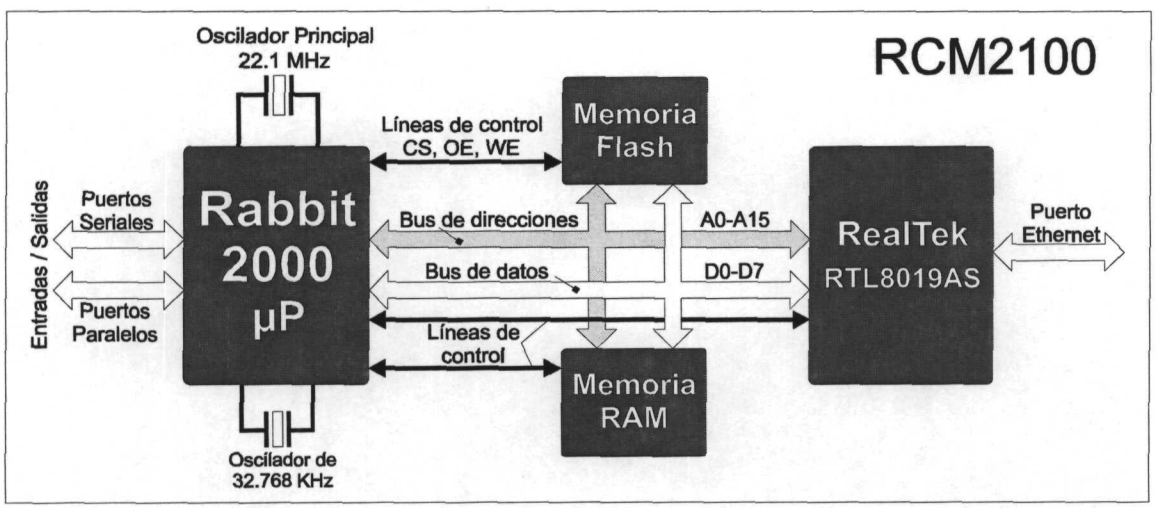


Figura 3.1. Esquema general de la arquitectura de la tarjeta de desarrollo RCM2100.

El fabricante de la tarjeta ofrece varios modelos de tarjetas que funcionan prácticamente bajo la misma arquitectura, la diferencia entre un modelo y otro estriba básicamente en la capacidad de los circuitos de memoria, en la velocidad del oscilador principal, y en la inclusión o exclusión del circuito manejador de ethernet (RTL8019AS). Dependiendo del modelo de tarjeta, también varía el número de terminales disponibles para el usuario para el manejo de puertos de entrada y salida.
Las características principales de la tarjeta RCM2100 son las siguientes:

- Microprocesador Rabbit 2000 corriendo a 22.1 MHz
- Puerto Ethernet 10 Base T con conector RJ-45
- Memoria Flash 512K (2 x 256 K)
- Memoria RAM 512K
- 5 Puertos de entrada/salida digital.
- 4 puertos seriales.
- Reloj de Tiempo Real

- 5 temporizadores de 8 bits, y un temporizador de 10 bits con dos registros de comparación.
- Soporte para Watch Dog.

La figura 3.2 muestra las partes principales de la tarjeta RCM2100, además de sus puertos y algunas otras líneas disponibles para su conexión con el exterior.

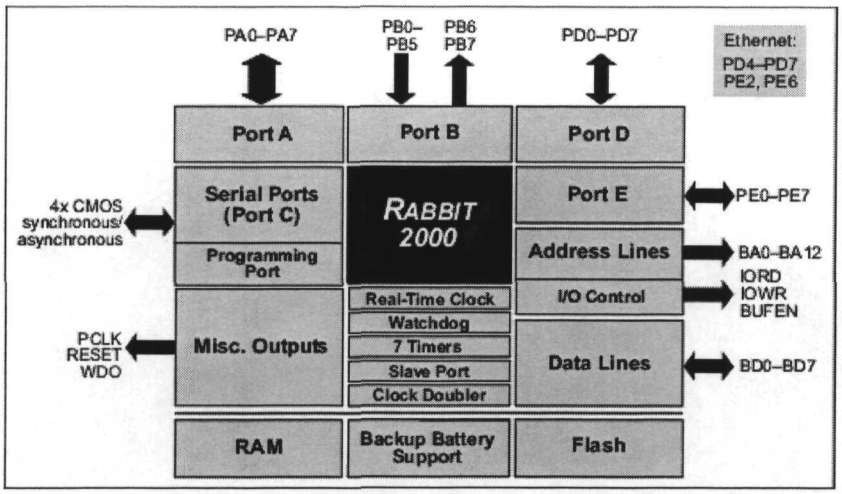


Figura 3.2. Partes principales, puertos, y líneas de conexión de la tarjeta RCM2100 (*).

En total la tarjeta RCM2100 tiene 34 líneas de entrada y salida que se agrupan en cinco puertos paralelos de 8 bits. Existen puertos que son configurables, y que pueden operar en distintos modos, por ejemplo, el puerto C puede operar como un puerto paralelo de entrada o salida, o puede configurarse para que algunas de sus terminales funcionen como puerto serial.

Para el desarrollo de la aplicación del "Modulo de Comunicación" propuesta en este trabajo de tesis se utiliza precisamente una tarjeta RCM2100.
Desde el punto de vista de la aplicación, el módulo de comunicación esta compuesto por las siguientes partes:

- El procesador.
- Memoria interna.
- Memoria externa.
- Puerto de comunicación serial asíncrono
- Puerto de comunicación serial síncrono
- Puerto de Red Ethernet.
- Puerto de entrada/salida digital.

A continuación se explica el funcionamiento de cada una de las partes del módulo de comunicación y su relación con el hardware de la tarjeta RCM2100.

* "RabbitCore RCM2100 User's Manual", pág. 7.

3.1.1. El Procesador.

El procesador es el núcleo principal de la tarjeta y corresponde a un procesador "Rabbit 2000" de 8 bits, el cual, puede correr a una velocidad de hasta 30 MHz. El procesador cuenta con algunos periféricos como puertos paralelo y serial configurables, tiene un módulo para facilitar la activación de periféricos sin necesidad de hardware externo ("Memory Chip Interface"), puede direccionar hasta 1 Megabyte de memoria; su set de instrucciones en lenguaje ensamblador es muy similar al del procesador Z80 pero tiene algunas modificaciones que facilita la misma programación del procesador en lenguaje C; cuenta con algunos temporizadores y registros de comparación, con reloj de tiempo real respaldado por batería, manejo de interrupciones externas, soporte para "Watch Dog", entre otras funciones y periféricos que se integran en un mismo chip de 100 terminales y de las siguientes dimensiones: 24 x 18 x 3mm.

El chip del procesador "Rabbit 2000" esta compuesto por una "Unidad Central de Proceso" que además esta rodeada de varios periféricos y componentes auxiliares tal y como se muestran en la figura 3.3.

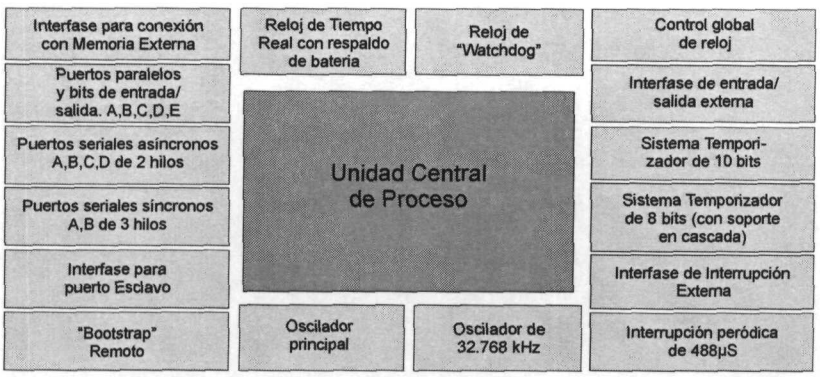


Figura 3.3. Esquema general del Procesador Rabbit 2000.

Internamente el procesador cuenta con un reloj de tiempo real que lleva registro de la hora y la fecha, en el rango desde el 1 de Enero de 1980 hasta el año 2047, con una resolución de un segundo. Este reloj se aprovecha para llevar registro de la hora y fecha como una facilidad extra para el usuario del módulo de comunicación. La marcha del reloj esta dada por el oscilador de 32.768 MHz; la operación del oscilador puede respaldarse mediante una batería de respaldo; cuando así sucede, el registro de la hora y fecha del reloj de tiempo real continúa su curso y se conserva vigente; esto, aunque se suprima la alimentación principal del módulo de comunicación. La batería de respaldo mantiene tanto al oscilador de 32.768 MHz y el reloj de tiempo real, así como también el contenido de la memoria RAM del módulo de comunicación.

Por otra parte, la figura 3.4 muestra con mas detalle en un diagrama a bloques las partes principales del circuito procesador Rabbit 2000 y la conexión y comunicación entre sus partes.

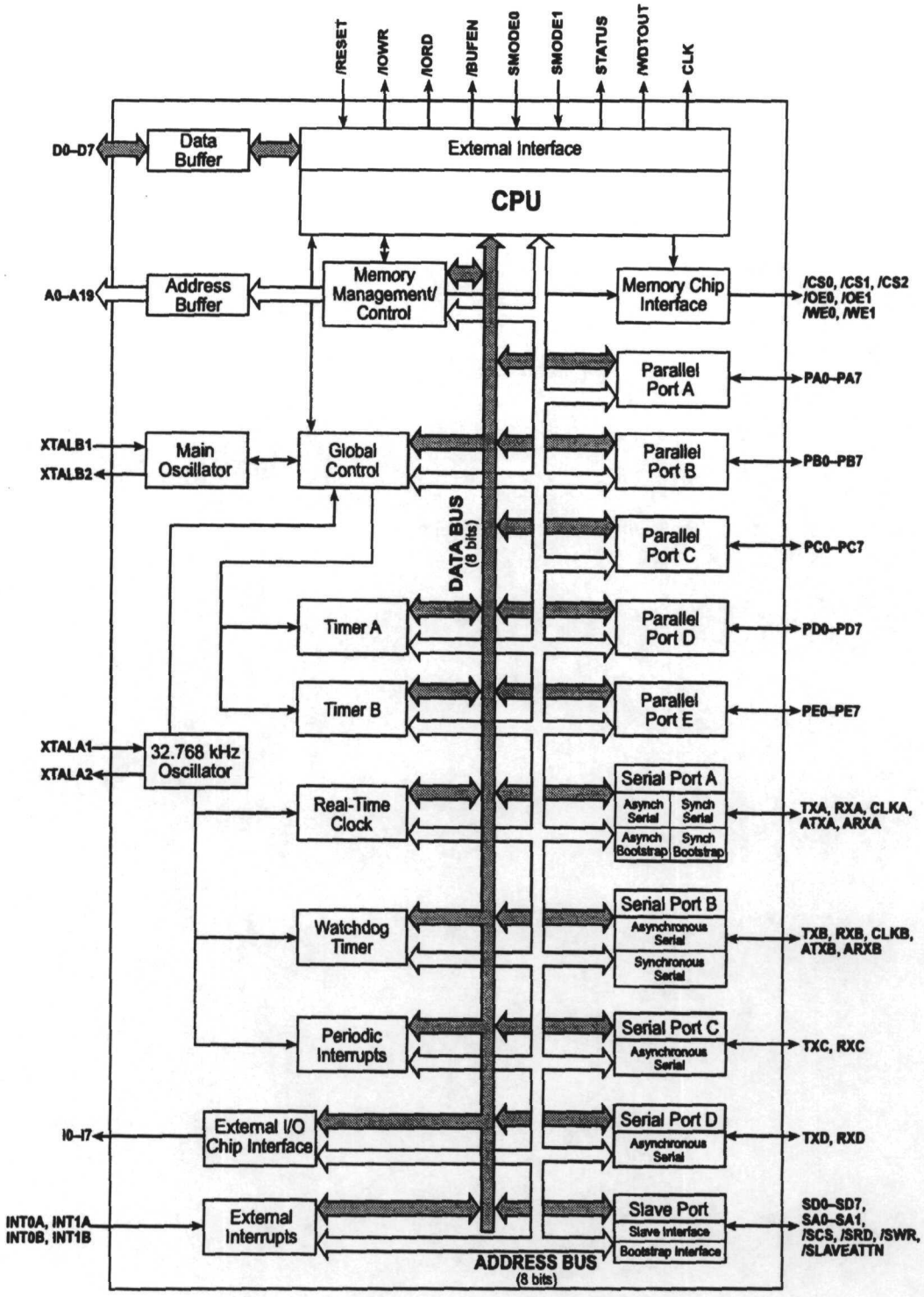


Figura 3.4. Diagrama a bloques del Procesador Rabbit 2000 [&].

[&] "Rabbit 2000™ Microprocessor User's Manual", pág. 4.

La programación del microprocesador Rabbit 2000 en lenguaje ensamblador es relativamente sencilla. El procesador cuenta con varios registros de 8 bits para efectuar operaciones, como transferencias entre registros, operaciones de manipulación de bits, operaciones aritméticas, entre otras. A continuación se muestran los registros internos del procesador con los que dispone el usuario para realizar operaciones.

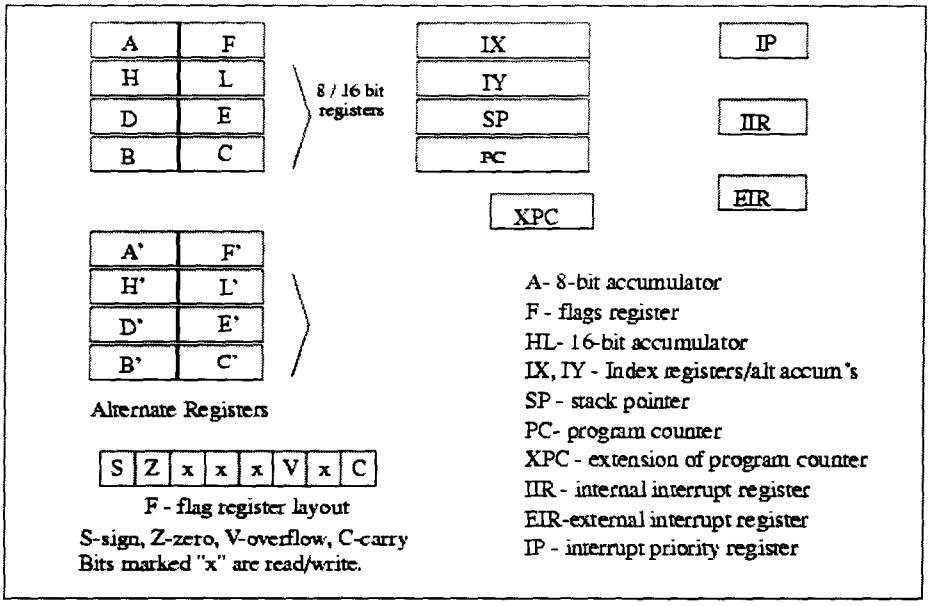


Figura 3.5. Registros internos del Procesador Rabbit 2000[Ψ].

3.1.2. Memoria Interna.

La memoria interna del módulo de comunicación a su vez se divide en:

- Memoria RAM
- Memoria Flash.

3.1.2.1. Memoria RAM.

La memoria RAM se utiliza para el almacenamiento de datos del módulo de comunicación y corresponde al circuito de memoria RAM de la tarjeta RCM2100. Aunque el circuito de memoria tiene una capacidad de 512 Kbytes, sólo 130 Kbytes se utilizan para el almacenamiento de datos del usuario. Parte del resto del espacio en RAM se utiliza para el manejo de variables de programa y del stack. Los datos de la memoria pueden conservarse mediante una batería de respaldo aún y cuando se interrumpa la alimentación principal de la tarjeta, los datos se pueden conservar hasta por más de 6 años si se respetan las especificaciones de la batería propuesta en el manual de la tarjeta RCM2100. La tarjeta esta preparada con dos terminales para la conexión de la batería de respaldo.

Dentro de la memoria RAM, a su vez, el módulo de comunicación cuenta con dos categorías de memoria, que son:

[Ψ] "Rabbit 2000™ Microprocessor User's Manual", pág. 13.

- Memoria Raíz
- Memoria Extendida

Ambas categorías se alojan dentro del chip de memoria RAM de la tarjeta RCM2100. La memoria raíz tiene una capacidad de 2 KB, mientras que la memoria extendida es de una capacidad de 128 KB. El tiempo de acceso de la memoria raíz es menor que el de la memoria extendida, de hecho, las operaciones sobre la memoria extendida internamente requieren de una transferencia intermedia a memoria raíz; esto implica que el tiempo de acceso a memoria extendida es mayor que el tiempo de acceso a memoria raíz.

3.1.2.2. Memoria Flash.

La memoria flash del módulo de comunicación se utiliza para alojar el código del programa del mismo módulo de comunicación. Además, es común que se utilice para el alojamiento de tablas y/o datos de configuración, los cuales, son de carácter no volátil.
La memoria flash esta compuesta por dos circuitos integrados de 256 Kbytes cada uno. Por lo tanto, la capacidad total de la memoria es flash del módulo de comunicación es de 512 Kbytes.

3.1.3. Memoria Externa.

Adicionalmente a la memoria RAM de la tarjeta RCM2100, se cuenta con un módulo externo de memoria flash para el almacenamiento masivo de datos no volátiles, cuya capacidad puede ser de 4 u 8 Megabytes.
La memoria externa consiste en una tarjeta portátil físicamente aparte de la tarjeta RCM2100 en la que se desarrolló la aplicación del "Módulo de Comunicación". La tarjeta de memoria externa corresponde a la serie SF1000 de tarjetas de expansión de memoria flash serial de la compañía "Rabbit Semiconductor". La idea del módulo de memoria es que pueda acoplarse opcionalmente para el almacenamiento o acceso de datos del módulo de comunicación. En otras palabras, el módulo de memoria se puede conectar, intercambiar, o simplemente no conectar al módulo de comunicación. También se puede trasladar a otro sitio para leer los datos contenidos en su interior, o incluso para cargarle datos a partir de cualquier otro dispositivo que tenga la capacidad de comunicarse con él, como puede ser una computadora o un microcontrolador además del mismo módulo de comunicación. La comunicación con el módulo de memoria se realiza en una forma serial síncrona. En total el módulo de memoria requiere del uso efectivo de 6 líneas o "pines" para establecer una comunicación; 3 de ellos dedicados a un puerto serial síncrono (Serial_In, Serial_Out, Clk); 2 para la alimentación (Vcc, Gnd); y uno para la habilitación del circuito de memoria del módulo (Chip_Select~).

Una característica importante de la memoria tipo flash es que los datos almacenados en su interior se conservan por tiempo indefinido aunque el circuito de memoria no este energizado; lo que lo hace apto para aplicaciones donde se requiera el transporte de datos en forma física. Los datos en memoria pueden ser leídos tantas veces como se desee, pero el número de escrituras esta limitado y de ello depende el tiempo de vida de la memoria flash. Por ejemplo, por mencionar algunos valores típicos, existen memorias diseñadas para soportar: 50,000, 100,000 y hasta más número de escrituras. Para ciertas aplicaciones de almacenamiento el uso de una memoria flash es suficiente a pesar de su limitación en cuanto al número de escrituras. Por otra parte, en comparación con la

memoria RAM, una memoria flash es varias veces más costosa en términos de precio por unidad de almacenamiento.

Dentro de la serie "SF1000" actualmente se tienen dos modelos con distinta capacidad de almacenamiento:

- El modelo "SF1000" de una capacidad de 8 Megabytes.
- El modelo "SF1010" de una capacidad de 4 Megabytes.

El módulo consiste en una pequeña tarjeta de circuito impreso de 44.5 x 38.4 mm de superficie, en donde se monta el circuito integrado de memoria flash (por ejemplo, para 4 MB, el AT45DB321 de Atmel), además de algunos componentes para el acondicionamiento de niveles de voltaje, y los conectores para la conexión del módulo de memoria con el exterior. También puede considerarse un dispositivo de almacenamiento de bajo consumo de potencia; con un consumo típico de 14.4mW durante un ciclo de lectura y de 10.8μW en modo de Stand-by.

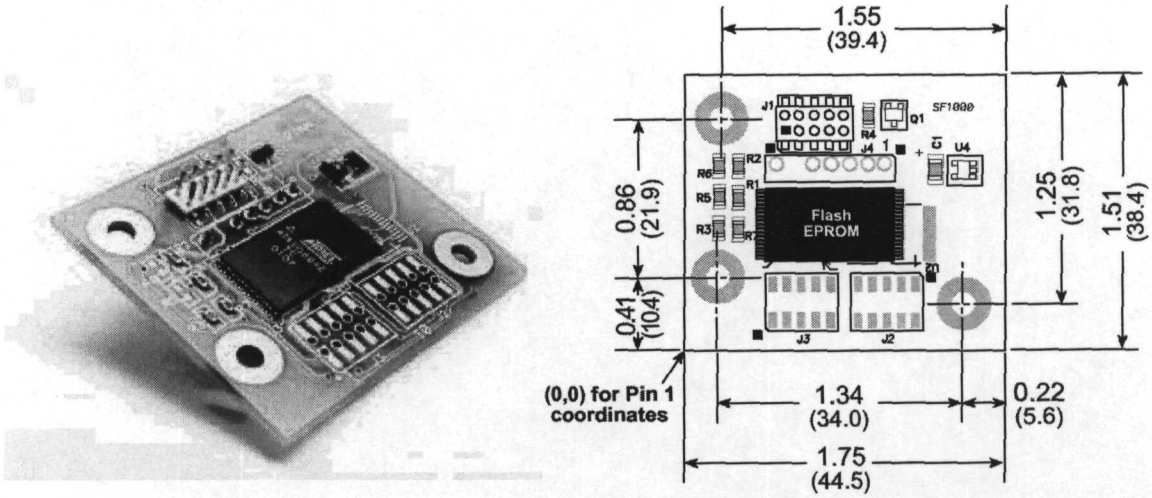


Figura 3.6. Foto del módulo de memoria externo, y sus dimensiones[℘].

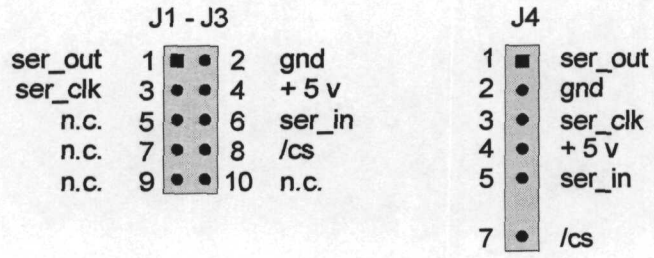


Figura 3.7. Pinout del módulo de memoria externo de la serie SF1000.

℘ "SF1000 Serial Flash Expansion Card User's Manual", portada y pág. 22

La tarjeta del módulo de memoria cuenta con un conector de 10 pines (J1) bajo el estándar de 2 mm de separación entre pines; también, esta lista con espacios para instalar hasta 3 conectores más (J2, J3 y J4) para accesar las mismas líneas de comunicación y de alimentación de la memoria en un bus compartido. A través de dicho bus pueden conectarse distintos dispositivos para el acceso a la memoria. El conector J4 tiene una separación estándar de 0.1 pulgada entre pines a diferencia de los conectores J1, J2 y J3.
En resumen, para almacenamiento de datos el módulo de comunicación tiene 3 opciones de memoria que son:

- Memoria Raíz (RAM)
- Memoria Extendida (RAM)
- Memoria Externa (Flash –opcional-)

3.1.4. Puerto de Comunicación Serial Asíncrono.

El puerto de comunicación serial asíncrono se utiliza para comunicar el módulo de comunicación con un microcontrolador, o con un procesador digital de señales; a través de este puerto, el módulo de comunicación espera recibir datos que representan un comando. La operación del módulo de comunicación se basa en la solicitud y ejecución de comandos, que es la forma como se controlan y manipulan los elementos del módulo. El puerto serial también envía datos como respuesta a la solicitud de algún comando. Existen comandos requieren de parámetros, por lo tanto, la operación del puerto serial puede resumirse como un puerto bidireccional que por un lado espera recibir datos pertenecientes a comandos acompañados de posibles parámetros, y que por otro lado entrega los datos de la respuesta del comando solicitado.
Con respecto a la tarjeta RCM2100 el puerto de comunicación serial asíncrono corresponde al puerto serial B que utiliza los pines PC4 y PC5 del puerto C. El puerto C de la tarjeta, que realmente proviene del puerto C del procesador Rabbit 2000, consta de 8 bits, de los cuales, 4 son de entrada, y 4 de salida. La función alterna del puerto C es habilitar las líneas de comunicación serial de los distintos puertos seriales del procesador que son 4 (A, B, C, D). Específicamente, los pines PC4 y PC5 tienen la función alterna de operar como las líneas de comunicación del puerto serial B. Cuando PC4 y PC5 se utilizan para el puerto serial, el resto de las líneas del puerto C pueden continuar en uso sin afectar la función del puerto serial B.
El pin PC4 corresponde a la línea de transmisión (TxB) del puerto serial B y se localiza en el pin #29 del conector J1 de la tarjeta RCM2100, mientras que el pin PC5 corresponde a la línea de recepción (RxB) del puerto serial B y se localiza en el pin #30 del conector J1 de la tarjeta RCM2100.

Bit del Puerto C	Funcion alterna (Serial B)	Conector/pin en la tarjeta RCM2100
PC4	TxB	J1 / pin #29
PC5	RxB	J1 / pin #30

Tabla 3.1. Función de los pines PC4 y PC5 como puerto serial B asíncrono, y su localización en la tarjeta RCM2100.

Los niveles de voltaje que maneja el puerto serial B están entre 0 y 5 volts, es decir, utiliza 0 volts para identificar un nivel bajo (L) y 5 volts para identificar un nivel alto (H). La comunicación serial de la tarjeta puede utilizar cualquiera de los estándares RS-485 o RS-232, pero para ello, es necesario agregar un circuito manejador que convierta los niveles digitales de 0 a 5 volts del puerto serial a los niveles de voltaje del estándar específico, ya sea a los niveles del estándar RS-485 o a los niveles de estándar RS-232.
Como se mencionó en la sección 1.4.3, tanto el estándar RS-232 como el RS-485 se basan en la misma lógica de comunicación, la diferencia radica en los niveles de voltaje que se utilizan para transmitir datos de un lugar a otro. Un circuito de uso común para acoplar niveles digitales de 0 y 5 volts a niveles del RS-485 es el DS3695 de "Dallas Semiconductor", mientras que para realizar la conversión de niveles al estándar RS-232 es común que se utilice el circuito MAX232 de "Maxim". El circuito manejador del estándar de comunicación debe instalarse en forma externa a la tarjeta RCM2100. Para efectos de prueba del prototipo del módulo de comunicación con una PC, se instaló un circuito manejador para RS-232 que acopla los niveles de voltaje del puerto serial de la tarjeta RCM2100 con los niveles del puerto serial RS-232 de la PC.
Por último, las velocidades en bits por segundo (bps) que soporta el puerto serial B de la tarjeta RCM2100 son las siguientes:

110, 300, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200, 230400.

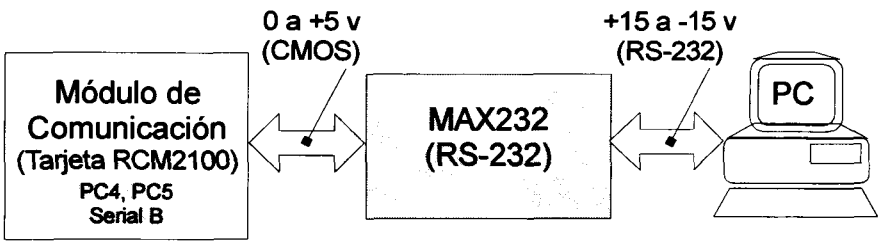


Figura 3.8. Uso de un circuito manejador "MAX232" para el acoplamiento de los voltajes de la comunicación serial entre la tarjeta RCM2100 y una PC.

3.1.5. Puerto de Comunicación Serial Síncrono.

Este puerto se utiliza para la comunicación con la tarjeta de memoria flash externa ("SF1000"). Aunque la tarjeta RCM2100 tiene la capacidad de manejar una comunicación serial síncrona por medio de hardware, en este caso la comunicación se genera a través de software manipulando el estado de algunos bits del puerto D del procesador. El hecho de generar la comunicación por software implica mayor tiempo de procesamiento que el que se requiere en una comunicación auxiliada por hardware, sin embargo, existe una razón de peso para hacerlo de esa forma. La razón es evitar que el procesador atienda la solicitud de otros usuarios externos y se encargue de los detalles para escribir o leer en la memoria externa, lo que implica todavía una mayor demanda de tiempo para el procesador.
Para evitar que el procesador atienda a las solicitudes de los usuarios externos en cuanto al uso de la memoria externa, se optó por dejar a la memoria externa ("SF1000") como un recurso compartido entre varios usuarios. Los usuarios externos se comunican directamente con la memoria mediante un bus, tal y como se muestra en la figura 3.9. El

puerto serial del módulo de comunicación debe de tener la capacidad para conmutarse a un estado de alta impedancia puesto que se conecta a un bus compartido en el que pueden participar las salidas de otros usuarios. Los puertos seriales auxiliados por hardware del procesador Rabbit no tienen la capacidad de conmutarse a alta impedancia, sin embargo, el puerto D puede configurar cada uno de sus 8 bits como salida o como entrada (en alta impedancia), además de controlar el nivel (H o L) de sus salidas. Por tal motivo, y para evitar la adición de hardware externo en la comunicación serial síncrona, se utiliza el puerto D para generar por software las señales de la comunicación serial y para conmutar el puerto a un estado de alta impedancia cuando el módulo de comunicación no este haciendo uso de la memoria.

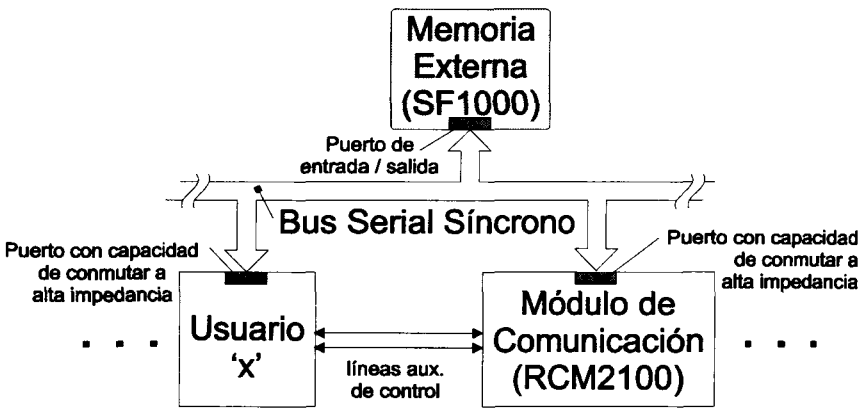


Figura 3.9. Esquema del acceso directo a la memoria externa por medio de un bus serial síncrono.

3.1.5.1. Líneas auxiliares de control.

Una limitación en cuanto al acceso de la memoria externa es que sólo un usuario a la vez puede hacer uso de la memoria. Para coordinar el acceso a la memoria entre los distintos usuarios, el puerto serial síncrono va acompañado de dos líneas auxiliares de control, que son: LIU ("Local in use") y RIU ("Remote in use"). El nombre asignado a cada una de las líneas representa el estado del bus serial desde el punto de vista del módulo de comunicación. Por ejemplo, la línea LIU, activa en nivel bajo (L), indica que el módulo de comunicación esta haciendo uso del bus para comunicarse con la memoria, mientras que el nivel alto (H) indica lo contrario, es decir, que el bus esta libre o no esta siendo utilizado por el módulo de comunicación.

La línea RIU es una señal de entrada en el módulo de comunicación que recibe el estado del bus con respecto a uno o varios usuarios remotos. Por ejemplo, cuando en la línea RIU se recibe un nivel alto (H), el módulo de comunicación es advertido de que el bus serial esta libre y que por lo tanto puede disponer del bus. Si por el contrario, en la línea RIU se recibe un nivel bajo (L), esto le indica al módulo de comunicación que el bus serial esta ocupado por otro usuario remoto, por consiguiente, el módulo de comunicación debe esperar a que el usuario remoto libere el bus.

Línea		Condicion del bus serial
LIU (Out)	RIU (In)	
H	H	Libre
L	H	En uso local
H	L	En uso remoto
L	L	Colisión

Tabla 3.2. Tabla de estado para las líneas LIU y RIU.

3.1.5.2. Detección de colisiones.

El módulo de comunicación, en la puesta a nivel bajo (L) de la línea LIU, activa automáticamente una interrupción para la detección de colisiones. La interrupción se detecta mediante la línea de entrada RIU. Cuando ocurre una colisión la interrupción se encarga de proteger el puerto SPI colocándolo en un estado de alta impedancia.

La figura 3.10 muestra el ejemplo de conexión entre 2 usuarios del mismo tipo (módulo de comunicación) que manejan la misma lógica para coordinar el acceso al bus serial. Esta forma de coordinar el acceso al bus evita la adición de hardware en el acoplamiento de 2 usuarios para el acceso a la memoria externa, sin embargo, si se desea compartir el BUS entre más de dos usuarios, es necesario diseñar y adaptar una lógica adicional para evitar colisiones en el bus.
En el ejemplo, los usuarios 1 y 2 son del mismo tipo, pero en la práctica uno de ellos puede ser un microcontrolador, o cualquier otro dispositivo digital; lo importante es que respete y maneje la convención utilizada de las líneas LIU y RIU para coordinar el acceso al bus.

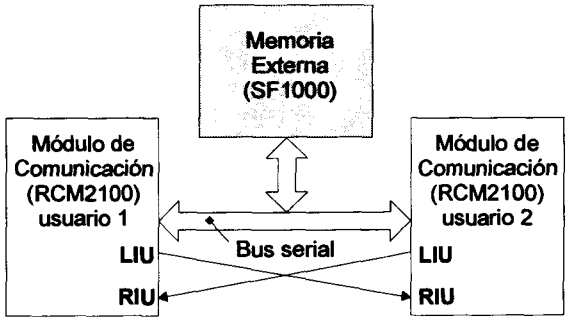


Figura 3.10. Esquema de conexión de las líneas de auxiliares (LIU, RIU) para coordinar el control del BUS serial síncrono.

La localización de los pines del puerto serial síncrono y de las líneas LIU y RIU en el hardware de la tarjeta RCM2100 se muestran en las tablas 3.3 y 3.4, respectivamente.

Bit del Puerto D	Funcion Serial Síncrona	Conector/pin en la tarjeta RCM2100
PD0	Clk (Out)	J1 / pin #33
PD1	Tx (Out)	J1 / pin #34
PD2	/CS (Out)	J1 / pin #35
PD3	Rx (In)	J1 / pin #36

Tabla 3.3. Función de los pines PD0 a PD3 como puerto serial síncrono, y su localización en la tarjeta RCM2100.

Bit del Puerto E	Funcion	Conector/pin en la tarjeta RCM2100
PE4	RIU (In)	J2 / pin #21
PE5		J2 / pin #20
PE7	LIU (Out)	J2 / pin #18

Tabla 3.4. Pines del Puerto E utilizados para coordinar el acceso al bus serial, y su localización en la tarjeta RCM2100.

Como puede apreciarse en la tabla 3.4 la señal de entrada RIU ocupa 2 pines (PE4 y PE5), esto se debe a cuestiones de hardware del microprocesador Rabbit 2000 en el manejo de interrupciones externas, las cuales, se exponen con detalle en la nota de aplicación TN301 emitida por "Rabbitsemiconductor". Entre los pines PE4 y PE5 se conecta una resistencia de 1K para desfasar la señal de entrada entre los dos pines. PE5 debe recibir la señal directa de entrada.

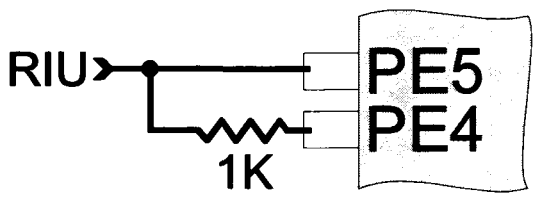


Figura 3.11. Conexión requerida para los pines de interrupción externa PE4 y PE5 del procesador Rabbit 2000.

3.1.6. Puerto de Red Ethernet.

La función del puerto ethernet es comunicar el módulo de comunicación con uno o varios usuarios a través de una comunicación por red; los usuarios pueden estar conectados directamente a la misma red ethernet donde se conecta el módulo de comunicación, o también pueden estar conectados indirectamente a través de otras redes, como por

ejemplo, internet. El puerto ethernet del módulo de comunicación se encarga de recibir y transmitir datos a la red. Los datos que recibe el módulo de comunicación llevan consigo la solicitud de un comando para que el módulo realice una determinada acción, mientras que los datos enviados a través del puerto pertenecen a la respuesta del comando solicitado.

En términos de hardware, el puerto de red ethernet consiste en un conector RJ45 hembra para la conexión de la tarjeta RCM2100 con un nodo de red ethernet. La capacidad de la tarjeta RCM2100 para el manejo de ethernet radica en el circuito RTL8019AS de "RealTek" que es un controlador full-duplex de ethernet. El circuito controlador de ethernet a su vez lo dirige el procesador Rabbit 2000 mediante 20 de sus líneas distribuidas en el bus de datos, bus de direcciones, y algunos pines de los puertos D y E, tal como se muestra en la figura 3.12.

Es muy importante advertir que cuando en la tarjeta se habilita el uso del puerto ethernet, los pines PE2, PE5, PE6 del puerto E, y los pines PD4, PD5, PD6, PD7 del puerto D, quedan dedicados para el control del circuito de controlador de ethernet, por lo tanto, dichos pines no deben utilizarse para otro propósito, ya que causaría conflicto con el circuito manejador de la red dejándolo fuera de operación. Cuando esto sucede, es necesario volver a inicializar la tarjeta para restablecer el circuito manejador de ethernet.

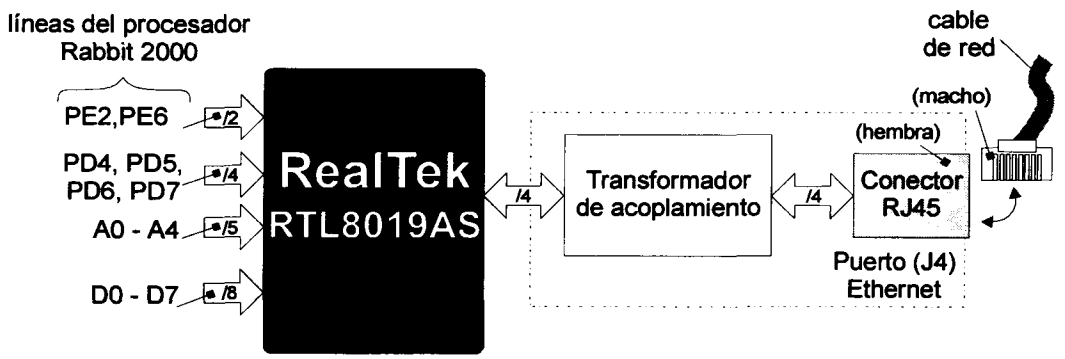


Figura 3.12. Esquema del módulo de ethernet de la tarjeta RCM2100.

3.1.7. Puerto de Entrada/Salida Digital.

Este puerto consta de 6 bits que se distribuyen en 4 bits de entrada y 2 bits de salida. Los 4 bits de entrada se utilizan para el monitoreo de aplicaciones que manejen sólo dos estados (encendido/apagado), como por ejemplo, determinar la ausencia o presencia de un objeto en algún proceso. Las 4 líneas de entrada pueden utilizarse en forma independiente o también pueden utilizarse en conjunto para identificar diferentes estados a partir de las combinaciones que se generan del estado de las 4 líneas. Esto es, hasta (2^4 =) 16 combinaciones distintas las que pueden identificarse mediante las 4 líneas de entrada.

En el hardware de la tarjeta RCM2100 este puerto corresponde a los pines de entrada PB2 a PB5 del puerto paralelo B.

Los dos bits de salida del puerto digital tienen la intención de utilizarse en aplicaciones de control "on/off"; como por ejemplo el encendido y apagado de un motor, una válvula, etc. También pueden utilizarse las dos líneas de salida en conjunto para generar hasta (2^2 =) 4 distintas combinaciones de control.
Las dos líneas de salida de este puerto corresponden a los pines PB6 y PB7 del puerto digital B del procesador Rabbit. Su localización en la tarjeta RCM2100 se muestra en la siguiente tabla.

Bit del Puerto B	Función	Conector/pin en la tarjeta RCM2100
PB2	Entrada digital	J2 / pin #3
PB3		J2 / pin #4
PB4		J2 / pin #5
PB5		J2 / pin #6
PB6	Salida digital	J2 / pin #7
PB7		J2 / pin #8

Tabla 3.5. Puerto B utilizado para la entrada de 4 líneas y la salida de 2 líneas digitales, y su localización en la tarjeta RCM2100.

Adicionalmente, en la tarjeta de desarrollo quedan algunas líneas disponibles que pueden utilizarse para expandir la capacidad de control y monitoreo del módulo de comunicación en base a salidas y entradas digitales. Por ejemplo, el puerto A de la tarjeta queda disponible para hacer uso de él en forma similar al puerto B que se explicó en esta sección.
La tabla 3.6 muestra un sumario del uso de que se les da a las líneas de los 5 puertos con los que cuenta el procesador. Dichos puertos, son los mismos con los que cuenta el usuario de la tarjeta de desarrollo.
Las líneas disponibles para expandir el alcance del módulo de comunicación son las que se muestran en la columna del extremo derecho de la tabla; también, en la tabla se muestran las líneas actualmente ocupadas por el módulo y su función.
Las líneas de los puertos que en la tabla aparecen marcadas con un asterisco (PB1, PC6, y PC7) corresponden al puerto serial A. Dicho puerto se ocupa durante el modo de programación y depuración de la tarjeta. Una vez que la tarjeta se programa para que opere en forma independiente, las líneas del puerto serial A quedan disponibles para el usuario.

Puerto	Bits		Terminales Ocupadas por el Módulo de comunicación y su función		Terminales Disponibles y su función
Puerto A	PA0	↔			Puerto de entrada/salida digital, configurable bit por bit.
	PA1	↔			
	PA2	↔			
	PA3	↔			
	PA4	↔			
	PA5	↔			
	PA6	↔			
	PA7	↔			
Puerto B	PB0	←			Entrada
	PB1	←	* CLKA (Puerto de Programación)		Entrada
	PB2	←	Puerto B (entrada/salida)	Entrada Digital	
	PB3	←			
	PB4	←			
	PB5	←			
	PB6	→		Salida Digital	
	PB7	→			
Puerto C	PC0	→			Salida / TxD
	PC1	←			Entrada / RxD
	PC2	→			Salida / TxC
	PC3	←			Entrada / RxC
	PC4	→	Serial B Asinc.	Salida serial B (TxB)	
	PC5	←		Entrada serial B (RxB)	
	PC6	→	* TxA (Puerto de Programación)		Salida / TxA
	PC7	←	* RxA (Puerto de Programación)		Entrada / RxA
Puerto D	PD0	→	Pto. Serial Síncrono	Clk (Salida SPI)	
	PD1	→		Tx (Salida SPI)	
	PD2	→		/CS (Salida SPI)	
	PD3	←		Rx (Entrada SPI)	
	PD4	→	Ethernet Chip (Realtek 8019AS)		
	PD5	↔			
	PD6	↔			
	PD7	↔			
Puerto E	PE0	↔			Entrada/salida ó entrada de Interrupción Externa A
	PE1	↔			
	PE2	→	Ethernet Chip (Realtek 8019AS)		
	PE3	↔			Entrada/Salida
	PE4	←	Entrada de Interrupción Externa B (RIU)		
	PE5	←			
	PE6	→	Ethernet Chip (Realtek 8019AS)		
	PE7	→	Salida (LIU)		

Tabla 3.6. Uso de las líneas de los puertos del procesador Rabbit 2000 en la aplicación del módulo de comunicación.

3.2. Estructura del Programa.

El programa del módulo de comunicación, tiene una estructura de procesamiento multitarea. Esto significa que el procesador (Rabbit 2000) aparenta estar ejecutando varias tareas en forma paralela. El procesador realmente sólo puede ejecutar una instrucción a la vez, sin embargo, para aparentar ejecutar varias tareas en forma simultánea, el procesador divide su atención y dedica una porción de su tiempo y de su capacidad de procesamiento a cada una de las tareas de un programa. La atención del procesador a las tareas se realiza mediante un barrido secuencial y ordenado a lo largo de cada una de las tareas. La velocidad con la que el procesador realiza el barrido es de una rapidez tal, que para una persona el procesador aparenta estar realizando varias actividades a la vez.

Existen dos esquemas principales de multitarea que son:

- Multitarea Preventiva ("Preemptive Multitasking").
- Multitarea Cooperativa ("Cooperative Multitasking").

El programa del módulo de comunicación utiliza un esquema de multitarea cooperativa, sin embargo, a manera informativa y de comparación, se exponen en forma breve los dos esquemas de multitarea.

3.2.1. Multitarea Preventiva ("Preemptive Multitasking")

En este esquema de multitarea el procesador realiza un recorrido a lo largo de un número total de tareas, dedicándoles un tiempo de procesamiento fijo a cada una de ellas. Existe una base de tiempo que es la que da la pauta para que el procesador cambie de la ejecución de un fragmento de una tarea a otra. Este esquema de multitarea es el que utilizan los sistemas operativos "Windows" por mencionar un ejemplo.

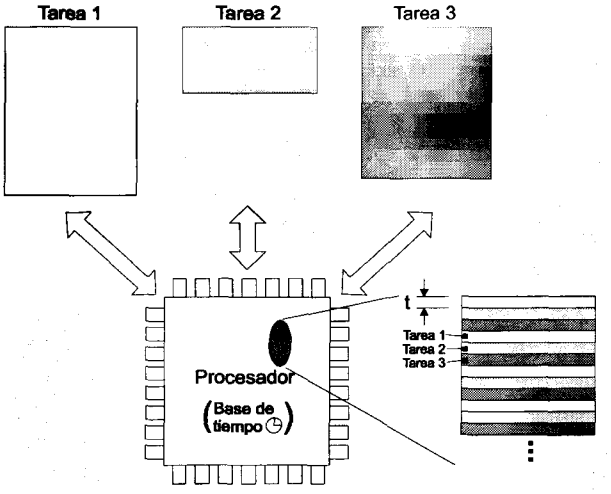


Figura 3.13. Representación de la ejecución de un programa con 3 tareas, utilizando un esquema de multitarea preventiva.

La representación de la figura 3.13, muestra a la derecha del procesador la ejecución de cada una de las tareas por segmentos. Cada segmento esta compuesto por un cierto número de instrucciones que son ejecutadas dentro de un tiempo límite "**t**". El tiempo "t" esta dado por un reloj base de tiempo.

3.2.2. Multitarea Cooperativa ("Cooperative Multitasking")

En un esquema de multitarea cooperativa cada tarea esta diseñada para ésta ceda el control voluntariamente a las demás tareas de un programa. En esta forma de programación, dentro del código de cada tarea existen instrucciones especiales para ceder el control del programa hacia la siguiente tarea en secuencia. La longitud de los segmentos de las tareas, y por consiguiente, la duración de cada segmento es variable. El programador define como parte del código en que puntos el programa cede el control de una tarea a otra. La figura 3.14 muestra la representación de la ejecución de 3 tareas distintas por un sólo procesador utilizando un esquema de multitarea cooperativa. En la parte derecha de la figura puede apreciarse la forma en que el procesador ejecuta en secuencia cada uno de los segmentos de las tareas. En la misma representación, puede

apreciarse que el tiempo de ejecución de los segmentos (t1, t2, t3, ...) es variable de acuerdo al número de instrucciones que pertenecen a cada segmento antes de ceder el control.

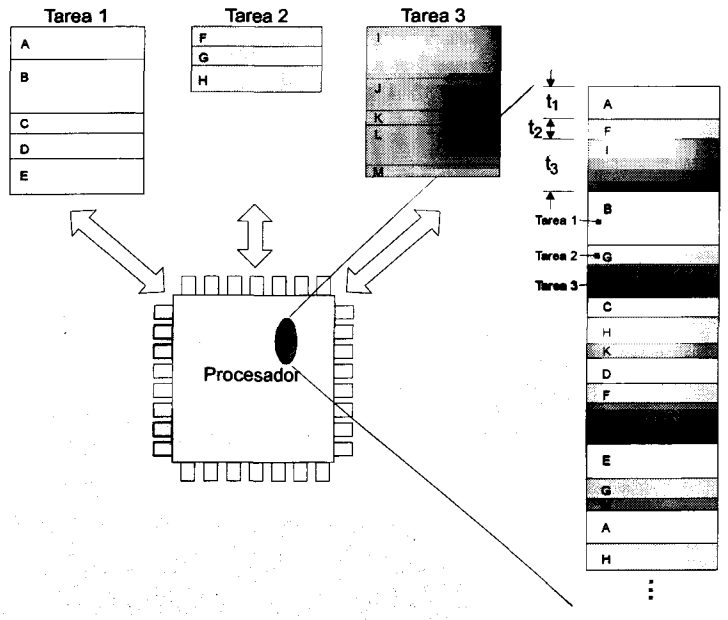


Figura 3.14. Representación de la ejecución de un programa con 3 tareas utilizando un esquema de multitarea cooperativa.

En un esquema de programación de multitarea cooperativa cada tarea de un programa principal es en realidad una máquina de estados. Las tareas están compuestas de distintos estados, por ejemplo en la figura 3.14 la "Tarea 1" esta compuesta por los estados A, B, C, D y E. Cada tarea tiene asociado un "puntero" de estado. El programa principal se auxilia de dichos punteros para saber cual es el próximo estado a ejecutar en el barrido de las tareas. En el ejemplo de la figura 3.14, por cuestiones de simplicidad, las tareas se ejecutan siempre en el mismo orden, pero en un programa real que funciona como una máquina de estados, existen saltos condicionados; es decir, saltos que pueden tener distintos destinos dependiendo de las condiciones del programa para efectuar un salto. En ocasiones dichas condiciones son de carácter aleatorio; un ejemplo típico es una tarea dedicada a detectar si un usuario ha presionado un botón desde el exterior; mientras el botón no se presione la tarea puede continuar dentro de un ciclo haciendo saltos hacia el mismo estado de lectura de un puerto y dando oportunidad a la vez de que se ejecuten otras tareas en forma paralela, si el botón se presiona, entonces la tarea para la detección del botón realiza un salto hacia la siguiente fase (estado) de la misma tarea.

En un programa con una estructura de multitarea cooperativa, el hecho de tener estados de distinta longitud, además de saltos condicionados, donde incluso las condiciones para efectuar un salto pueden ser de carácter aleatorio, dificulta poder determinar con exactitud el tiempo que tardará el microprocesador antes de volver a atender una determinada tarea. En este tipo de programación el programador es responsable de

seccionar cada una de las tareas estratégicamente para que el tiempo de atención a cada tarea sea la adecuada de acuerdo a la aplicación del programa.

3.2.3. Estructura en base a tareas.

El programa principal del módulo de comunicación, utiliza un esquema de procesamiento de multitarea cooperativa como la descrita en la sección anterior. La parte medular del programa consiste en un ciclo infinito, dentro del cual, están comprendidas las diferentes tareas que aparentan ejecutarse en paralelo. Previamente a la ejecución del ciclo infinito se ejecuta una etapa de inicialización para el ajuste del valor de algunas variables de configuración.

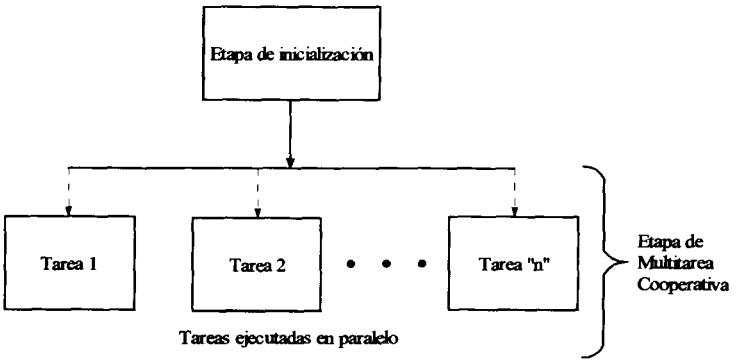


Figura 3.15. Diagrama de Flujo General del Programa del Módulo de Comunicación.

La figura 3.15 muestra a grandes rasgos la estructura del programa principal del módulo de comunicación. El programa esta hecho en lenguaje "Dynamic C" de la compañía "Z-World". Dicho lenguaje esta cimentado en la misma sintaxis del lenguaje de ANSI C, pero tiene algunas modificaciones precisamente orientadas a la programación multitarea en microprocesadores sencillos (en comparación con un Pentium por ejemplo) como es el caso del Rabbit 2000, un procesador de 8 bits.

3.2.4. "Costatements" y "Cofunciones"

Como parte de las facilidades que ofrece el lenguaje de programación "Dynamic C" para la programación en un ambiente de multitarea cooperativa se tienen dos estructuras básicas, que son:

- El uso de "Costatements"
- El uso de "Cofunciones"

En cuanto al uso de "costatements" y "cofunciones" ambos operan bajo el esquema de una máquina de estados, la diferencia es que una "cofunción" es llamada en forma similar a una función de C, y además en su llamada puede incluir parámetros; el "costatement" es una sección de código que funciona como una máquina de estados y que puede manipularse mediante llamadas de funciones especiales, dichas funciones pueden "pausar", "resumir" o continuar, y "reinicializar" un "costatement". El "costatement" no esta diseñado para recibir parámetros en las llamadas a las funciones que regulan su

operación; sin embargo, se puede hacer uso de variables globales y modificarlas a manera de pasar parámetros a un "costatement".
El "costatement" puede considerarse de mayor jerarquía que una "cofunción"; esto se debe a que dentro de un "costatement" pueden anidarse "cofunciones", incluso dentro de un "costatement" puede haber anidados otros "costatements", pero dentro de una "cofunción" no puede haber un "costatement". Sin embargo, dentro de una "cofunción" si pueden anidarse "cofunciones". Este concepto se muestra en la figura 3.16 donde las tareas principales del programa que se encuentran dentro del ciclo infinito son "costatements" (nivel 1). Cada tarea principal en su interior puede estar fraccionada por otras máquinas de estados que pueden ser "costatements" o "cofunciones", los "costatements" de cualquier nivel pueden fraccionarse sucesivamente hasta "m" niveles.

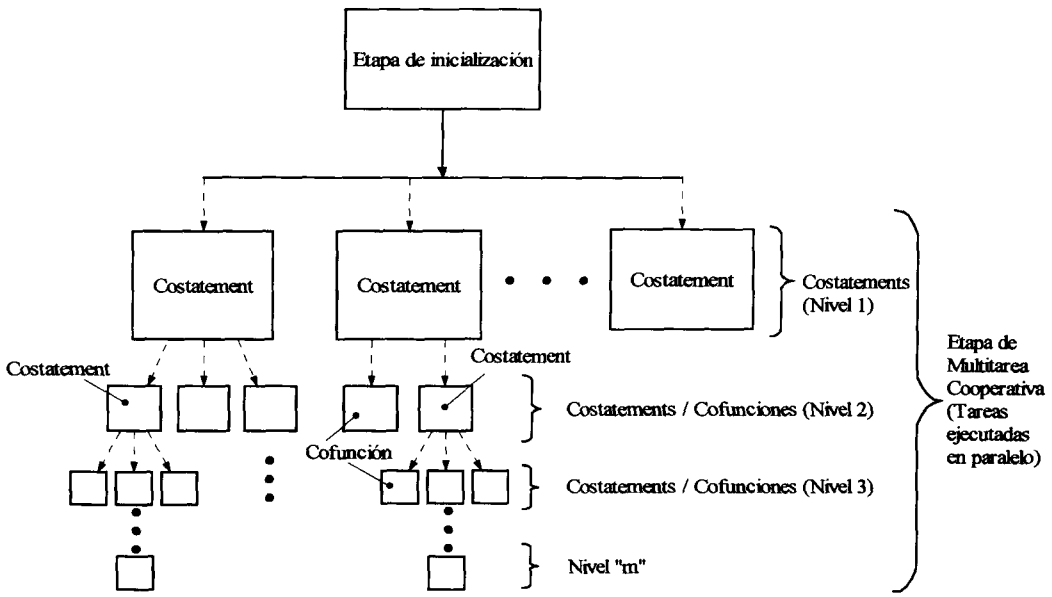


Figura 3.16. Descomposición de las tareas principales del programa en "costatements" y "cofunciones".

El programa del módulo de comunicación utiliza una estructura en base a "costatements" y "cofunciones" para lograr un ambiente de multitarea.

3.2.5. Tareas Principales del Programa.

El programa principal del módulo de comunicación esta compuesto por una etapa de inicialización y por 5 tareas principales que se ejecutan bajo un esquema de multitarea cooperativa.
Una "tarea principal" se considera aquella que se coloca en el ciclo principal (nivel 1) del programa del módulo de comunicación.

Las tareas principales con sus respectivos nombres (entre paréntesis) son las siguientes:

- Inicializar memoria externa.
- Atender una conexión serial ("Servidor Serial")
- Atender múltiples conexiones de red ("Servidor Ethernet")
- Ejecutar funciones de memoria externa ("Cost_ext SF1000")
- Ejecutar Operaciones Cíclicas.

Las tareas principales del programa se muestran en la figura 3.17. Para efectos de simplicidad en el diagrama, se hace referencia a algunas de las tareas mediante su respectivo nombre ("Servidor Serial", "Servidor Ethernet", "Cost_ext SF1000").

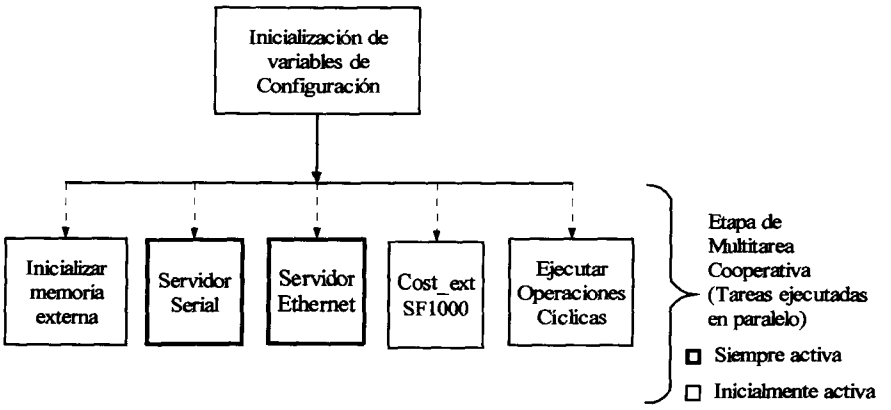


Figura 3.17. Diagrama de las tareas principales del módulo de comunicación.

3.2.6. La Tarea "Ejecutar Funciones de Memoria Externa" ("Cost_ext SF1000").

Algunas operaciones internas de la tarea principal para "atender múltiples conexiones de red" ("Servidor Ethernet") se colocaron sobre el nivel (1) del programa a manera de "costatements" dentro de la tarea principal para "Ejecutar funciones de memoria externa" ("Cost_ext SF1000"). Esto se debe a las limitaciones de la herramienta de programación ("Dynamic C") en el manejo de variables locales dentro de "cofunciones". Al menos en la versión del compilador utilizado 7.05P de "Dynamic C" las "cofunciones" sólo pueden manejar hasta 128 bytes de espacio para variables locales. Es por eso que algunas operaciones que requieren de más de 128 bytes se implementaron en forma de "costatements" (dentro de un "costatement" principal) en el ciclo principal del programa.

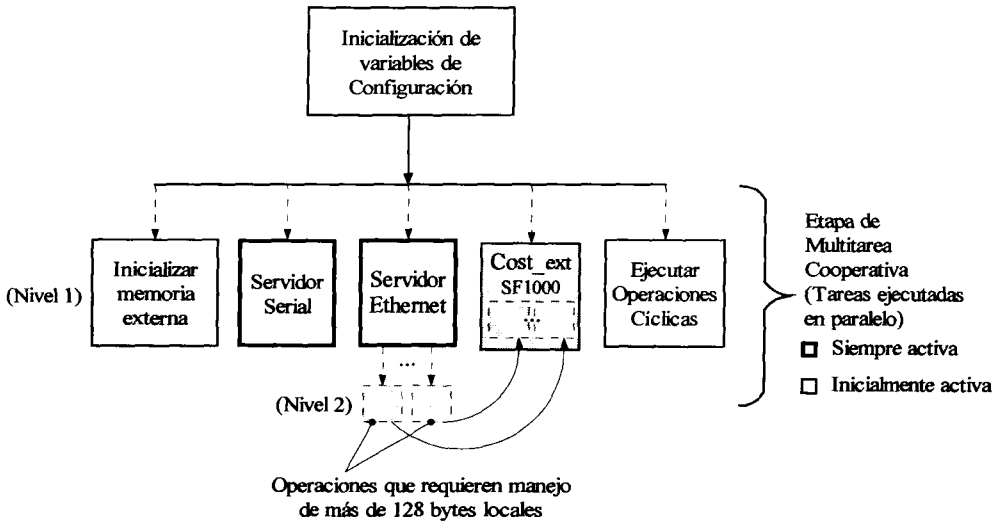


Figura 3.18. Diagrama que muestra algunas operaciones del "Servidor Ethernet" que se incluyen en un "costatement" al nivel de una tarea principal.

Las operaciones implementadas dentro del "costatement" "Cost_ext SF1000" están relacionadas específicamente con operaciones para el manejo de la memoria externa ("SF1000"). Las funciones para el manejo de la memoria externa desde un principio utilizan una tabla de 256 bytes para invertir el orden de los bits en la transmisión serial de un byte; esto hace que excedan la capacidad de una "cofunción" para el manejo de variables locales, por lo tanto, y por facilidad de programación se implementaron dentro de un "costatement" en el ciclo principal del programa.
Las operaciones dentro del "costatement" "Cost_ext SF1000" son las siguientes:

- Leer memoria flash externa.
- Leer buffer 2 RAM.
- Escribir buffer 2 RAM.
- Verificar número de escrituras sobre un bloque de la memoria flash.
- Escribir memoria flash.
- Enviar contenido de la memoria flash a través del puerto serial B.

3.2.7. Activación / Desactivación de las Tareas Principales.

Los "costatements" de las tareas principales pueden ser de dos tipos:

- Siempre activos ("always_on")
- Inicialmente activos ("init_on")

Los "costatements" siempre activos, como su nombre lo indica, están siempre activos, es decir, se ejecutan invariablemente y no pueden ser pausados, detenidos, o desactivados.
Los "costatements" inicialmente activos, se caracterizan por que pueden activarse o desactivarse según se requiera.
La siguiente tabla muestra las 5 tareas principales del programa, el tipo de tarea ("costatement"), y el estado inicial de la tarea al inicio del programa.

Tarea Principal	Tipo	Estado inicial
Inicializar memoria externa.	Inicialmente Activa	Activa
Atender una conexión serial ("Servidor Serial")	Siempre Activa	Activa
Atender múltiples conexiones de red ("Servidor Ethernet")	Siempre Activa	Activa
Ejecutar funciones de memoria externa ("Cost_ext SF1000")	Inicialmente Activa	Inactiva
Ejecutar Operaciones Cíclicas.	Inicialmente Activa	Inactiva

Tabla 3.7. Tareas principales del programa del modulo de comunicación, con el tipo de tarea ("costatement" asociado), y su respectivo estado al inicio del programa.

Las tareas del "servidor serial" y "servidor ethernet" son tareas que incondicionalmente permanecen activas. Esto, para dar atención a los usuarios del módulo de comunicación que soliciten algún servicio ya sea a través de una conexión serial o a través de una conexión de red.
Las tareas "inicialmente activas" son tareas auxiliares de las dos tareas siempre activas, ("servidor serial" y "servidor ethernet"). El "servidor serial" y "servidor ethernet" activan o desactivan las tareas auxiliares para realizar algunas funciones específicas del módulo de comunicación.

Por ejemplo, la tarea "inicializar memoria externa", al inicio del programa se activa para que busque e identifique el tipo de memoria instalada al módulo de comunicación. Una vez que se identifica el módulo de memoria externo, esta tarea automáticamente pasa o modo "inactivo". La tarea puede volver a activarse bajo dos formas generales:

1. en forma manual, cuando explícitamente lo solicita un usuario.
2. en forma automática, cuando ocurre un error en el acceso a la memoria externa que de indicios de la posible ausencia o del intercambio del dispositivo de memoria externa.

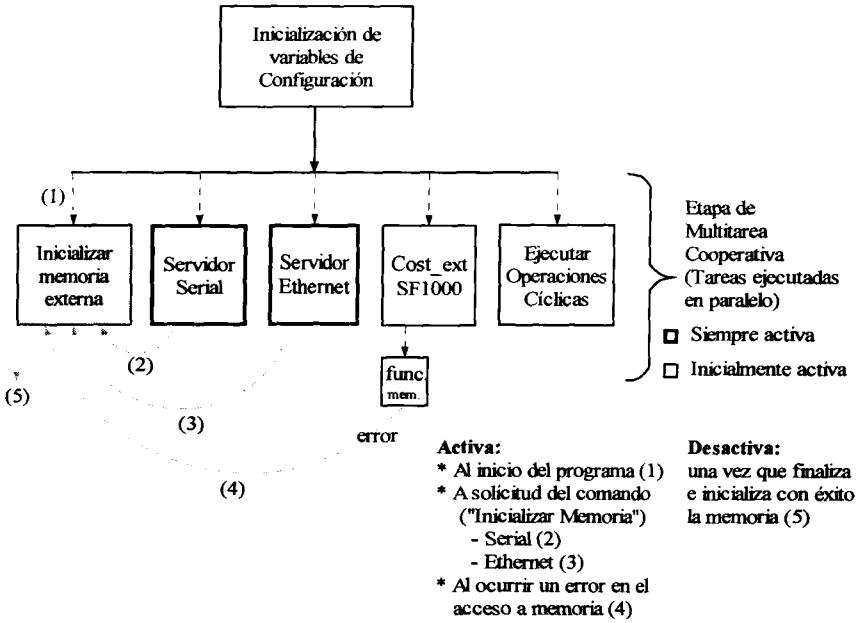


Figura 3.19. Casos en los que la tarea principal "Inicializar memoria externa" se activa y desactiva.

La tarea "Cost_ext SF1000" se activa solamente durante la ejecución de un comando del servidor ethernet relacionado con la memoria externa. Esto es, cuando un usuario del módulo de comunicación solicita realizar alguna operación con el módulo de memoria externa. Internamente dentro de esta tarea principal ("Cost_ext SF1000") se encuentran anidadas las tareas específicas que realizan una función sobre el módulo de memoria, como leer, escribir, etc. Cuando se habilita la tarea principal "Cost_ext SF1000" junto con ella también se habilita la tarea interna que realiza la operación con la memoria; una vez finalizada la operación, la tarea interna y la tarea principal "Cost_ext SF1000" se desactivan.

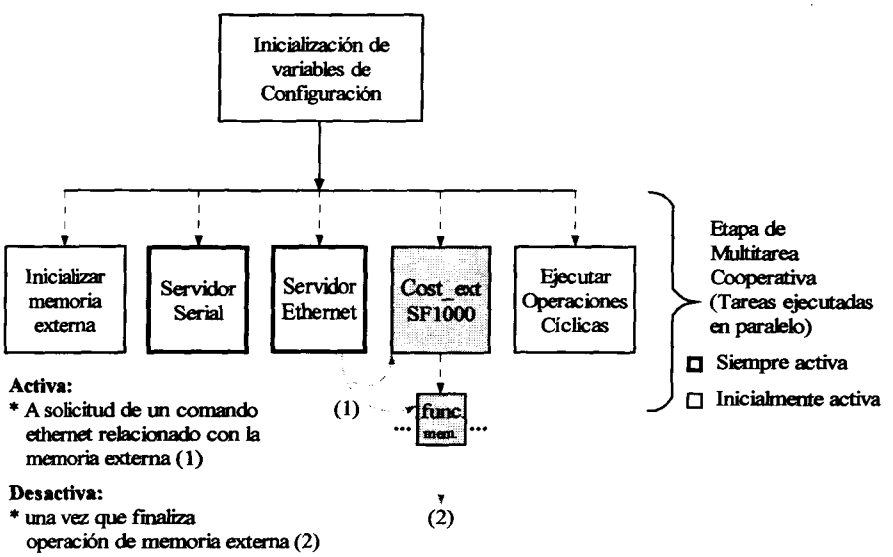


Figura 3.20. Activación y desactivación de la tarea principal "Cost_ext SF1000".

Por último, la tarea principal "Ejecutar operaciones cíclicas" realmente esta compuesta por tres tareas separadas que manejan los siguientes recursos:
- El puerto de red ethernet (puerto de red UDP).
- El Puerto Serial asíncrono de salida.
- El Puerto digital de salida.

Cada una de estas tareas se activa y desactiva solamente a solicitud de un usuario del módulo de comunicación. Esto, mediante el envío de un comando por parte del usuario ya sea a través del puerto serial o del puerto de red ethernet.

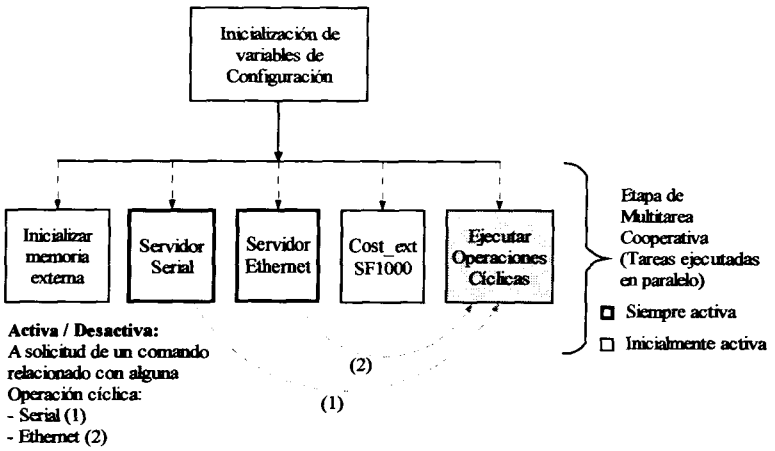


Figura 3.21. Activación y desactivación de la tarea "ejecutar operaciones cíclicas".

3.2.8. Servidor Ethernet para Conexión Múltiple.

Otra particularidad del programa del módulo de comunicación es que el servidor ethernet puede atender "N" conexiones o usuarios simultáneos. El número de usuarios "N" se define antes de compilar el programa . El programa principal esta diseñado para que por medio de una variable (definida por un macro) pueda compilarse para que atienda a un cierto número deseado de conexiones simultáneas de red. El número de usuarios "N" esta limitado por el espacio disponible en memoria RAM de la tarjeta RCM2100.

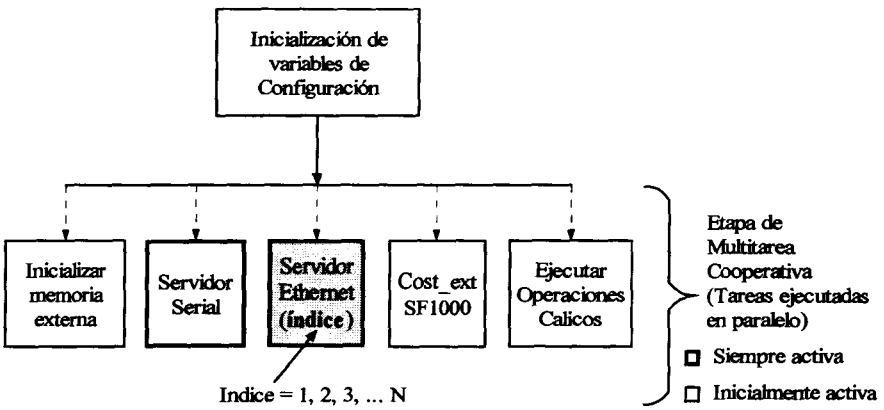


Figura 3.22. Esquema real del programa principal del módulo de comunicación, habilitado para atender "N" conexiones simultáneas de red.

Paso # →	1	2	3	4	5
Barrido # ↓	Inicialización de Memoria Externa	Servidor Serial	Servidor Ethernet (N = 3)	Cost_ext SF1000	Operaciones Cíclicas
1	√	√	√ (usuario 1)	√	√
2	√	√	√ (usuario 2)	√	√
3	√	√	√ (usuario 3)	√	√
4	√	√	√ (usuario 1)	√	√
5	√	√	√ (usuario 2)	√	√
6	√	√	√ (usuario 3)	√	√
7	√	√	√ (usuario 1)	√	√
.....	√	√	.....	√	√

Tabla 3.8. Secuencia de la ejecución del programa principal del módulo de comunicación.
(La tabla se lee de izquierda a derecha y de arriba a abajo.)

Un punto importante es que el código del programa del servidor ethernet es el mismo que se aplica para cada una de las "N" conexiones o "sockets" de red. El servidor ethernet es una máquina de estados, que como cualquier máquina de estados cuenta con dos partes fundamentales:
1.- las reglas o funciones de la máquina de estado,
2.- las variables de estado.
Las reglas o funciones son la parte fija de la máquina de estado, pertenecen en este caso al código del programa de la tarea del "servidor ethernet", mientras que las variables de estado son la parte dinámica que controla el flujo de la maquina de estados. Las variables de estado, dado su carácter variable, se alojan en memoria RAM.
Para poder atender a "N" conexiones de red el programa aloja "N" copias de variables de estado del servidor ethernet, es decir, dedica una copia de las variables de estado para cada conexión. El programa realiza un recorrido por cada una de las "N" posibles conexiones del servidor y aplica el valor de las variables de estado al código (o reglas) del servidor. Cada vez que se ejecuta un paso en la tarea del servidor, las variables de estado se actualizan y se guardan en la copia de variables de estado correspondiente. De esta forma se le da seguimiento a las "N" conexiones del servidor utilizando el mismo código para cada una de las conexiones.

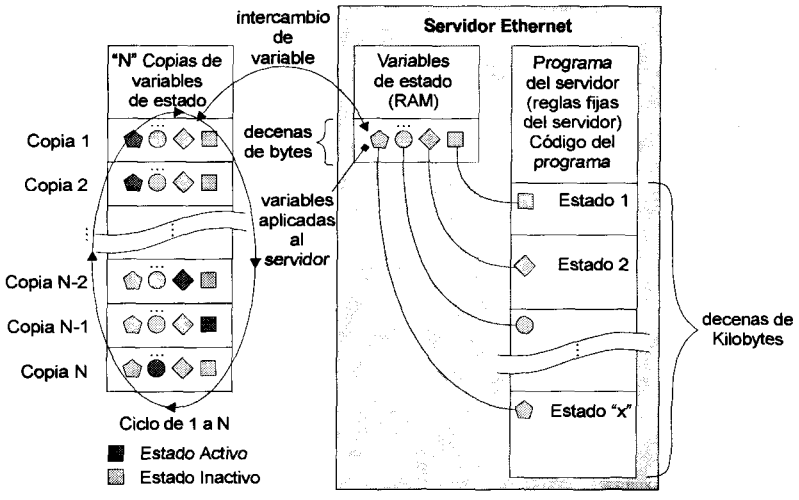


Figura 3.23. Esquema que muestra el uso del mismo código del servidor ethernet para atender a "N" conexiones, mediante el intercambio de un juego de variables con "N" copias del mismo juego de variables.

El hecho de utilizar el mismo código y utilizarlo para atender distintas conexiones, por una parte repercute en el ahorro de espacio de programa, pero por otra parte, también disminuye el desempeño del programa debido al tiempo que invierte el procesador en realizar el continuo cambio de variables. Para evitar el continuo cambio de las variables de estado, en teoría, se pueden crear varias copias del código del servidor ethernet, cada copia acompañada de sus respectivas variables de estado; sin embargo, en la práctica, el código del servidor ethernet es tan extenso (varias decenas de Kilobytes) que el compilador no acepta una segunda copia del código del servidor debido a que sobre pasa el espacio de memoria destinado para el código de programa.

Por otro lado, las variables de estado ocupan un espacio de varias decenas de bytes, y una o varias copias de dichas variables de estado no implica mayor problema de almacenamiento en la memoria RAM de la tarjeta RCM2100.

En este caso, más que por el hecho de ahorrar espacio en memoria de programa, la reutilización del código se aplica por necesidad y para dar atención a varias conexiones de red. La ventaja de aplicar esta técnica de reutilización de código radica en la atención múltiple y simultánea a varias conexiones de red. La desventaja, como ya se mencionó, repercute en la disminución del desempeño del programa debido al tiempo que invierte el procesador en realizar el continuo cambio de variables.

3.2.9. Resumen.

En resumen, la estructura del programa del módulo de comunicación utiliza un esquema de multitarea cooperativa. Al inicio del programa se ejecuta una etapa de configuración de variables, para después entrar a la parte principal del programa que consiste en la ejecución de 5 tareas principales, las cuales, se ejecutan dentro de un ciclo infinito y en forma cooperativa. El código de la tarea "servidor ethernet" se reutiliza con el auxilio de varias copias de variables de estado, esto, para atender a varias conexiones de red simultáneamente. Las tareas "Servidor serial" y "servidor ethernet" son tareas que

siempre permanecen activas, mientras que el resto de las tareas principales ("Inicialización de memoria externa", "Cost_ext SF1000", y "Operaciones cíclicas"), se activan o desactivan según el flujo y las condiciones del programa del módulo de comunicación. La idea de exponer la estructura del programa principal del módulo de comunicación es ofrecer un panorama más amplio de la forma en que opera internamente el programa, además de facilitar a quien lo desee, modificar, adaptar, o continuar con el desarrollo de este trabajo.

3.3. Funcionamiento del Programa del Módulo de comunicación.

El programa del módulo de comunicación se encarga de darle vida al hardware de la tarjeta RCM2100 para funcionar precisamente como el "módulo de comunicación" propuesto en este trabajo de tesis. La intención del módulo es habilitar a sistemas digitales sencillos, como microcontroladores y DSP's, a expandir su alcance y capacidad de comunicación a través de una red ethernet y de internet. Por lo tanto el módulo de comunicación funciona como un puente entre un dispositivo microcontrolador o DSP y una red ethernet. Por un parte el módulo de comunicación se comunica con el microcontrolador o DSP mediante una interfase serial asíncrona, y por otra parte se comunica con la red a través de un puerto de red ethernet.

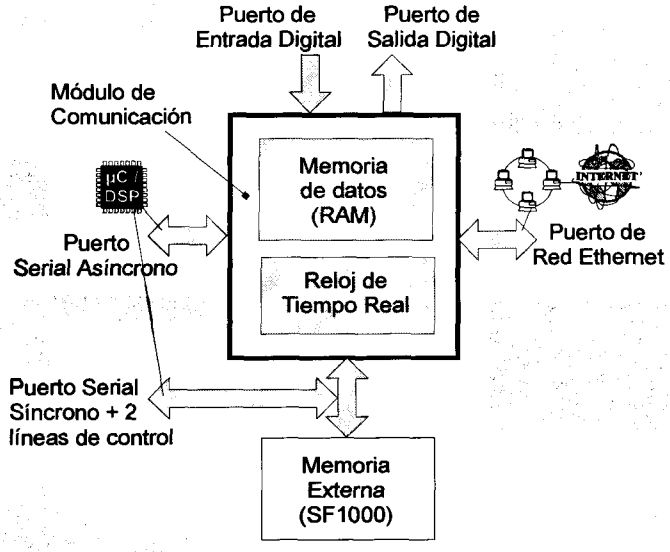


Figura 3.24. Esquema de los recursos y periféricos del módulo de comunicación.

3.3.1. El inicio del programa.

El inicio del programa sucede cuando el módulo de comunicación se enciende, es decir, cuando pasa de estar desenergizado a estar energizado. En la etapa de inicio del programa, el módulo de comunicación adquiere los valores por default de los parámetros de comunicación, es decir, los valores de configuración del puerto de comunicación serial asíncrono y del puerto de comunicación ethernet. Dichos valores se muestran en la tabla 3.9.

Configuración inical del	
puerto serial asíncrono	puerto ethernet
Bits por segundo: 9600	Dirección IP: 10.10.1.1
Bits de datos: 8	Puerto: #80
sin paridad	Netmask: 255.255.255.0
1 bit de paro	Gateway: 10.10.1.2

Tabla 3.9. Configuración inicial de los puertos de comunicación del módulo de comunicación al inicio del programa.

Durante la etapa inicial del programa el contenido de la memoria RAM y del reloj de tiempo real se conserva siempre y cuando el módulo cuente con la batería de respaldo, de lo contrario, el contenido de la memoria es aleatorio y el reloj de tiempo real empieza a contar desde el 1 de Enero de 1980.

3.3.2. Etapa de recepción de comandos

Después de la etapa de inicialización el programa entra en modo de espera para la recepción de **comandos**, ya sea a través del puerto serial asíncrono o del puerto ethernet. Esta etapa corresponde a la parte principal del programa donde se realiza la ejecución del ciclo infinito que contiene a las tareas principales del programa.

El programa del módulo de comunicación hace las veces de un servidor de datos y operaciones, es decir, constantemente esta en espera de que un usuario establezca una conexión con él para solicitar algún dato de memoria o alguna operación. Las solicitudes de los usuarios, denominados también "clientes", se realizan a través de **comandos**. Los comandos se envían al módulo de comunicación a través del puerto serial asíncrono, y del puerto de red ethernet. El módulo atiende en forma simultánea tanto al puerto serial asíncrono como al puerto de red ethernet.

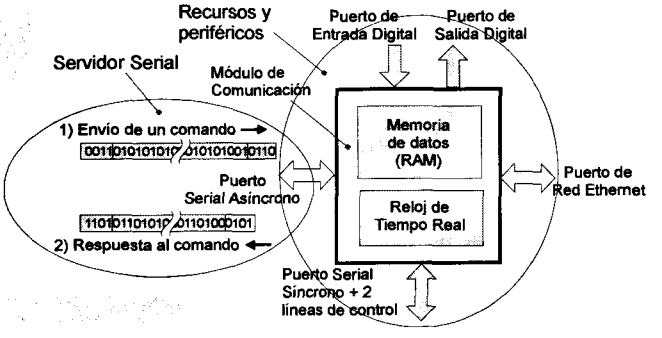


Figura 3.25. Operación del módulo de comunicación a través del puerto serial asíncrono (servidor serial).

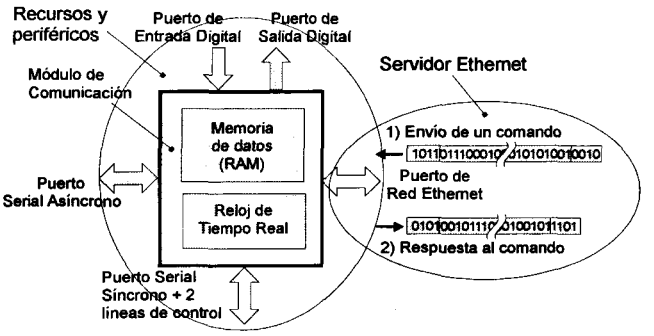


Figura 3.26. Operación del módulo de comunicación a través del puerto ethernet (servidor ethernet).

Los comandos tienen como finalidad realizar alguna operación específica sobre los recursos o periféricos del módulo de comunicación. Dentro de los recursos se considera a la "memoria de datos", el "reloj de tiempo real", y a la "memoria externa", esta última, como un módulo opcional del módulo de comunicación. Los periféricos son los que se muestran en la figura 3.24 en forma de flechas alrededor del módulo de comunicación; estos son: el "puerto de entrada digital", el "puerto de salida digital", el "puerto serial asíncrono", el "puerto de red ethernet", y el "puerto serial síncrono más 2 líneas de control".

La lista de comandos soportados por el programa se muestra en la tabla 3.10. En la lista (columnas 4 y 5) puede observarse que la mayoría de los comandos pueden solicitarse tanto a través del puerto serial como del puerto de ethernet, sin embargo, existen comandos exclusivos para cada uno de los dos puertos. La columna 1 de la tabla, se refiere al número de comando dentro de la tabla, en total el programa soporta 47 comandos. La columna 2 se refiere al identificador del comando, en su representación ASCII, que es la identificación que utiliza el programa internamente para diferenciar un comando de los demás. La columna 3 es el nombre asignado al comando de acuerdo a la función que realiza. La columna 6 muestra el nombre y la secuencia de los parámetros asociados a la solicitud de un determinado comando. Y por último, en la columna 7 se expone el tipo de dato que le corresponde a cada parámetro del comando, donde: "c" se refiere a un carácter de una longitud de 1 byte; "i" a un entero de longitud de 2 bytes, y "L" a un entero largo de 4 bytes de longitud.
Tomando como ejemplo el comando No. 1 de la tabla 3.10, de acuerdo a la misma información de la tabla, éste comando es exclusivo para los clientes que se conecten a través del puerto ethernet, y no se soporta para la solicitud a través del puerto serial. La función del comando ("ReadSF1000") se refiere a una lectura de la memoria externa ("SF1000"); para efectuar la lectura se requiere de 3 parámetros: (SrcAddr, ByteCount, T.O.).
"SrcAddr" se refiere a la dirección inicial del origen de los datos dentro de la memoria y se representa por un entero largo de 4 bytes ("L")
"ByteCount" se refiere al número de bytes que se desean leer a partir de la dirección inicial; este parámetro se representa también por un entero largo de 4 bytes ("L")

"T.O." es el tiempo límite en milisegundos que el programa puede esperar para efectuar la lectura en caso de que la memoria este ocupada. El valor de este parámetro se representa por un entero de 2 bytes ("i").

Cmd #	Clave / ID	Comando	Ethernet	Serial	Parametros	Tipo de datos
1	A	ReadSF1000	√	x	(SrcAddr, ByteCount, T.O.)	(L,L,i)
2	B	WriteSF1000	√	x	(DstAddr, ByteCount, T.O.)	(L,L,i)
3	C	CheckWritesSF1000	√	x	(StarBlock, EndBlock, T.O.)	(i,i,i)
4	D	WrRAM2	√	x	(DstAddr, ByteCount, T.O.)	(i,i,i)
5	E	RdRAM2	√	x	(SrcAddr, ByteCount, T.O.)	(i,i,i)
6	F	XMEMWrite	√	√	(XMEMSrcAddr, ByteCount)	(L,L)
7	H	XMEMRead	√	√	(XMEMDstAddr, ByteCount)	(L,L)
8	I	XMEMFill	√	√	(XMEMDstAddr, value, ByteCount)	(L,c,i)
9	J	XMEMMove	√	√	(XMEMSrcAddr, XMEMDstAddr, ByteCount)	(L,L,i)
10	K	rootWrite	√	√	(rootDstAddr, ByteCount)	(i,i)
11	L	rootRead	√	√	(rootSrcAddr, ByteCount)	(i,i)
12	M	rootFill	√	√	(rootDstAddr, value, ByteCount)	(i,c,i)
13	N	rootMove	√	√	(rootSrcAddr, rootDstAddr, ByteCount)	(i,i,i)
14	O	Serial_B_Config	√	√	(L_Bauds, int databits)	(L,i)
15	P	SerBOut_TCPstream_In	√	x	(L_ByteCount)	(L)
16	x	SerBOut_SerBin_stream	x	√	(L_ByteCount)	(L)
17	Q	SerBOut_root	√	√	(int add, int ByteCount)	(i,i)
18	R	SerBOut_XMEM	√	√	(L_add, L_ByteCount)	(L,L)
19	S	SerBOut_SF1000	√	√	(L_add, L_ByteCount, int TO)	(L,L,i)
20	T	Test_UDP	√	√	(RemIP, UDPport)	(L,i)
21	U	UDPtxRoot	√	√	(RemIP, UDPport, rootAddr, ByteCount)	(L,i,i,i)
22	V	UDPtxXMEM	√	√	(RemIP, UDPport, XMEMAddr, ByteCount)	(L,i,L,L)
23	W	UDPtxTCPstream	√	x	(RemIP, UDPport, ByteCount)	(L,i,L)
24	w	UDPtxSerBin_stream	x	√	(RemIP, UDPport, ByteCount)	(L,i,L)
25	X	PortB_Out	√	√	(value bits6,7)	(c)
26	Y	PortB_In	√	√	none	none
27	Z	root2XMEM	√	√	(rootSrcAdd, XMEMDestAdd, ByteCount)	(i,L,i)
28	a	root2SF1000	√	√	(rootSrcAdd, SFDestAdd, ByteCount, T.O)	(i,L,i,i)
29	b	XMEM2root	√	√	(XMEMSrcAdd, rootDestAdd, ByteCount)	(L,i,i)
30	c	XMEM2SF1000	√	√	(XMEMSrcAdd, SFDestAdd, ByteCount, T.O)	(L,L,L,i)
31	d	SF10002root	√	√	(SFSrcAdd, rootDestAdd, ByteCount)	(L,i,i,i)
32	e	SF10002XMEM	√	√	(SFSrcAdd, XMEMDestAdd, ByteCount, T.O)	(L,L,L,i)
33	h	Cycle_UDP_root	√	√	(RemIP, RemPort, rootAddr, ByteCount, DelayMs)	(L,i,i,i,L)
34	i	Cycle_SerB_root	√	√	(rootAddr, ByteCount, DelayMs)	(i,i,L)
35	j	Cycle_PortB	√	√	(Val1, Delay1, Val2, Delay2)	(c,L,c,L)
36	k	Get Date&Time	√	√	none	none
37	l	Set Date&Time	√	√	(struct tm)	(c,c,c,c,c,c,c,c)
38	m	Get Mem Cfg	√	√	none	none
39	n	Get Net Cfg	√	√	none	none
40	o	Set Page HTTP	√	√	(memSrc, InitAddr, ByteCount)	(c,L,L)
41	p	Get Pg HTTP_Cfg	√	√	none	none
42	q	GetCycleFunctStatus	√	√	none	none
43	r	GetSerBConfig	√	√	none	none
44	s	GetLast3CMD_result	√	√	none	none
45	t	Set Net Cfg	√	√	(IP, Netmask, Gateway)	(L,L,L)
46	u	Restart_SF1000	√	√	none	none
47	v	storeNetcfg	√	√	none	none

Tabla 3.10. Lista de comandos del módulo de comunicación

La intención en este capítulo no es describir la función detallada de cada uno de los comandos de tabla 3.10, pero si explicar en términos generales las funciones principales del módulo. Para simplificar la explicación de la función de los comandos, podemos asociarlos en tres distintas categorías:

- Comandos de Manejo de Memoria.
- Comandos de Manejo de Puertos.
- Comandos de Configuración y prueba.

3.3.2.1. Comandos para Manejo de Memoria:

Los comandos de manejo de memoria abarcan las operaciones básicas de lectura y escritura para las 3 distintas categorías de memoria (raíz, extendida, y externa). Además de las funciones básicas, se implementaron todas las combinaciones posibles para hacer transferencias o movimientos de datos entre los distintos tipos de memoria.
También hay funciones aisladas para los distintos tipos de memoria, como la función de rellenado, que esta disponible tanto como para la memoria raíz como la memoria extendida; otro ejemplo es la función de utilería "CheckWritesSF1000" que se utiliza para verificar el número de escrituras efectuadas en memoria flash externa.

3.3.2.2. Comandos para Manejo de Puertos:

Estos comandos efectúan operaciones de lectura y/o escritura sobre los distintos puertos. Algunas operaciones que pueden efectuarse son las siguientes:

- Transmitir datos a la salida del puerto serial asíncrono, los cuales tengan su origen en cualquiera de las siguientes fuentes:
  - Memoria Raíz.
  - Memoria Extendida.
  - Memoria Externa.
  - Flujo de entrada de datos provenientes del puerto serial asíncrono.
  - Flujo de datos provenientes de una conexión TCP ("Transfer Control Protocol").
- Transmitir datos a la red mediante el protocolo UDP ("User Datagram Protocol"), a partir de cualquiera de las siguientes fuentes:
  - Memoria Raíz.
  - Memoria Extendida.
  - Flujo de datos provenientes del puerto serial asíncrono.
  - Flujo de datos provenientes de una conexión TCP ("Transfer Control Protocol").
- Lectura de un puerto digital de 4 bits.
- Escritura de un puerto digital de 2 bits.

Dentro de esta categoría de "Comandos para Manejo de Puertos" existen operaciones cíclicas que pueden programarse sobre los puertos; tal es el caso de los comandos 33, 34, y 35. Las operaciones cíclicas, mientras se activen a solicitud de un usuario, se repiten indefinidamente dentro de un ciclo automático. Son 3 operaciones cíclicas y cada una de ellas puede activarse o desactivarse de manera independiente. Un parámetro en común que se define para estas funciones, es el intervalo de tiempo que separa a dos eventos consecutivos dentro de la ejecución cíclica.

Por ejemplo, el comando 33 de la lista ("Cycle_UDP_root") transmite a la red mediante el protocolo UDP una porción de la memoria raíz cada cierto tiempo de acuerdo a los parámetros del comando: (RemIP, RemPort, rootAddr, ByteCount, DelayMs)
"RemIP" y "RemPort", definen la dirección y el Puerto UDP del destinatario en la red.
"rootAddr" y "ByteCount" definen la porción de memoria raíz que se desea enviar.
"DelayMs" es el intervalo de tiempo que separa a cada uno de los constantes envíos que se efectúan hacia el destinatario.
Similarmente el comando 34 ("Cycle_SerB_root") envía cíclicamente una porción de la memoria raíz a través del puerto serial asíncrono de salida.
El comando 35 ("Cycle_PortB") alterna cíclicamente entre 2 estados de salida para el puerto digital. La duración de cada uno de los estados también se define a través de los parámetros del comando.

3.3.2.3. Comandos de Configuración y Prueba:

Los comandos de configuración abarcan desde la configuración de la velocidad para el puerto serial asíncrono, la dirección IP del módulo de comunicación, la hora y fecha del reloj de tiempo real, etc.; incluso, en esta categoría también pueden incluirse las funciones cíclicas mencionadas en la sección anterior, ya que son funciones que se ejecutan a partir de una configuración previa. Cada comando de configuración tiene asociado un comando de lectura que permite obtener la configuración vigente del módulo; este tipo de comandos de lectura puede utilizarse como una forma de verificación a la ejecución de un comando de configuración previo, o como base para tomar una decisión antes de realizar una operación de configuración. Como ejemplo tenemos el comando No. 14 ("Ser_B_Config") que configura el puerto de comunicación serial asíncrono, y el comando No. 43 ("GetSerBConfig") que obtiene la configuración del mismo puerto serial.

3.3.3. Modelo de Servidor.

La figura 3.27 muestra el modelo del funcionamiento general de un servidor. Este modelo es en el que se basan tanto el "servidor serial" como el "servidor ethernet" del módulo de comunicación.

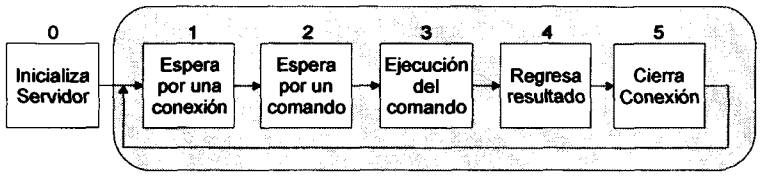


Figura 3.27. Modelo de la operación general de un servidor.

El servidor cuenta con una etapa de inicialización que se ejecuta sólo una vez al inicio del suministro de la alimentación del módulo de comunicación; esta etapa en la figura 3.27. esta señalada con el número "0". Después, el servidor entra a su ciclo de operación normal, el cual esta compuesto por 5 etapas que son:

1. Espera por una conexión.
2. Espera por un comando.
3. Ejecución del comando.

4. Regresa resultado.
5. Cierra conexión.

En el módulo de comunicación, tanto el servidor serial como el servidor ethernet utilizan un esquema de funcionamiento similar al que se expone en la figura 3.27. Sin embargo, la misma variedad de comandos y de operaciones del servidor de datos, hace que algunas etapas del modelo cuenten con algunas variantes.

Variantes en la solicitud de comandos.
Las variantes en la solicitud de comandos involucran a las etapas 2, 3 y 4 del modelo de servidor expuesto en la figura 3.27, es decir, a las etapas:

- Espera por un comando.
- Ejecución del comando.
- Regresa resultado.

Variantes en la espera por un comando.
Dentro de la etapa "Espera por un comando" existen categorías de comandos que reciben parámetros y comandos que no requieren parámetros. De acuerdo al comando enviado al módulo de comunicación el programa identifica si el comando requiere o no requiere de parámetros. En caso de que el comando implique el uso de parámetros, el mismo programa identifica el orden, tipo, y número de parámetros a recibir. Ver tabla 3.10 (columna 6).

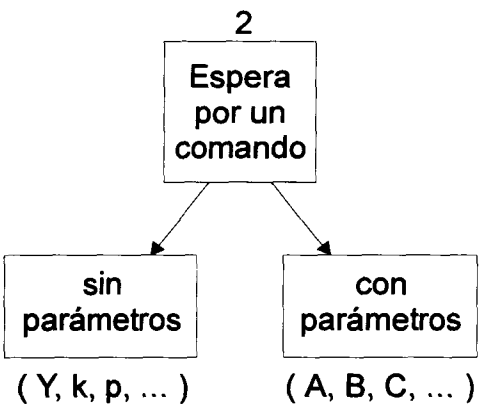


Figura 3.28. Variante de la etapa de "espera por un comando".

Variantes en la ejecución del comando.
En la etapa de "Ejecución del comando", cada comando realiza una operación distinta, por lo tanto, la variedad en esta etapa es evidente desde el punto de vista funcional. Por ejemplo, existen comandos que además de recibir parámetros, esperan recibir un flujo de datos, tal es el caso de una escritura a memoria; donde los parámetros definen las localidades de memoria a escribir, y el flujo de datos pertenece a los datos destinados a escribirse en memoria. Similarmente, existen comandos cuya ejecución consiste en entregar un flujo de datos, tal es el caso de una lectura a memoria, etc.
Por otra parte, desde el punto de vista de ejecución, los comandos pueden tener una de las siguientes dos formas de ejecución:

- Ejecución completa.
- Ejecución fraccionada.

Para comprender la diferencia entre estas dos formas de ejecución es conveniente recordar que el programa principal funciona como una máquina de estados en un ambiente de multitarea; el programa realiza un barrido por cada una de las distintas tareas, y ejecuta sólo un estado por cada tarea. Durante la ejecución de un estado el procesador esta dedicado solamente a la ejecución de las instrucciones del estado en cuestión; en otras palabras, puede decirse que el procesador mientras ejecuta las instrucciones de un estado permanece momentáneamente "bloqueado" para el resto de las tareas.

Las operaciones de "ejecución completa" generalmente se caracterizan por que son operaciones sencillas y que no consumen mucho tiempo de procesamiento, por ejemplo, la lectura o escritura del puerto digital, tan sólo requiere de un par de instrucciones que se ejecutan en cuestión de microsegundos. Las operaciones de "ejecución completa" en el programa principal se ejecutan de principio a fin como parte de un estado, y sólo cuando se completa su ejecución el programa cede el control hacia las demás tareas principales del programa.

Las operaciones de "ejecución fraccionada" generalmente se refieren a operaciones largas que para completarse requieren de un tiempo considerable (desde fracciones de segundo hasta minutos). Las operaciones se ejecutan paulatinamente en segmentos hasta completarse; entre la ejecución de un segmento de la operación y otro, el procesador atiende al resto de las tareas del programa principal; de esta forma, el procesador continúa dando atención a todos los usuarios, y a la vez, continúa ejecutando la operación pendiente del comando en curso, con lo que da la impresión de estar ejecutando varias operaciones en paralelo.

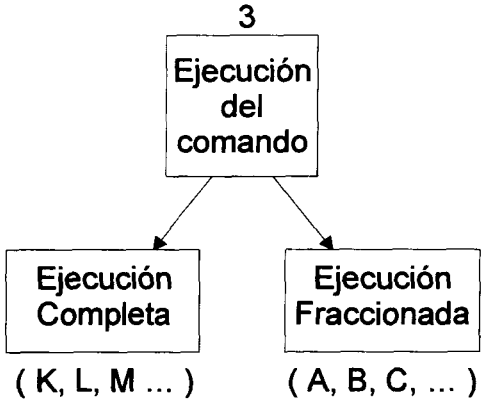


Figura 3.29. Variante de la etapa "Ejecución del Comando".

Variantes en el regreso de resultado.

Tanto en el servidor serial como en el servidor de ethernet se tiene la opción de enviar o no enviar la respuesta de la ejecución de un comando. Vale la pena recalcar que existen comandos que retornan datos, por ejemplo, un comando de lectura a memoria lleva implícito el retorno de los datos contenidos en la memoria, sin embargo, el retorno de los

datos corresponde a la etapa de la ejecución del comando y no a la etapa de regreso del resultado. En la etapa de regreso de resultado se incluye la respuesta de la ejecución de un comando, es decir, se indica si el comando se ejecutó correctamente libre de errores, o si ocurrió un error durante su ejecución. A manera de ejemplo, la tabla 3.11 muestra el listado de los valores devueltos por algunos comandos después de haber sido ejecutados.

Comando	Nombre del Comando	Valores Devueltos despues de ejecutado el comando (Modo "SafeMode")
A	ReadSF1000	0 = Comando Ejecutado Exitosamente -1 = Error de Acceso a Memoria, "Bytecount" fuera de rango -2 = Direccion de Mem. Flash Ilegal (fuera de rango) -4 = BUS SPI Ocupado -5 = Colision en el BUS SPI -6 = Conexion cerrada o reinicializada por el usuario (cliente)
C	CheckWritesSF1000	0 = Comando Ejecutado Exitosamente -1 = Valor Ilegal del prefijo de memoria ("illegal prefix value") = Bloque en Blanco -2 = Error durante la transmision del resultado de la funcion "SF1000CheckWrites()". -4 = BUS SPI Ocupado -5 = Colision en el BUS SPI -6 = Conexion cerrada o reinicializada por el usuario (cliente)
D	WrRAM2	0 = Comando Ejecutado Exitosamente -1 = Falla en la escritura de RAM2, o en la recepcion (a traves de la red) de los datos a ser escritos en RAM2. -4 = BUS SPI Ocupado -5 = Colision en el BUS SPI -6 = Conexion cerrada o reinicializada por el usuario (cliente)

Tabla 3.11. Ejemplo de valores devueltos por algunos comandos del servidor ethernet

La etapa de "Regreso de Resultado" es una etapa que puede omitirse o ejecutarse de acuerdo a la decisión del usuario. La opción para solicitar la respuesta de un comando se realiza dentro de la misma solicitud del comando. Dentro de la trama de solicitud del comando se incluye un campo especial ("SafeMode") para habilitar o deshabilitar el envío de la respuesta del comando. Este punto se expone con más detalle en la explicación de los protocolos de los servidores serial y ethernet de la sección 3.3.4.3 y 3.3.5.3. Por lo pronto, es suficiente saber que dentro de la solicitud de un comando se tiene la opción para que el módulo envíe o no envíe la respuesta de la ejecución del comando. En otras palabras, el servidor puede operar bajo dos modos de operación, que son:

- Sin respuesta (modo normal)
- Con respuesta (modo seguro -"SafeMode"-)

El habilitar la opción en "modo seguro" tiene como utilidad asegurarse de la correcta ejecución de un comando o de rastrear la raíz del error en la ejecución del comando principalmente para las operaciones que están expuestas a errores durante su ejecución.

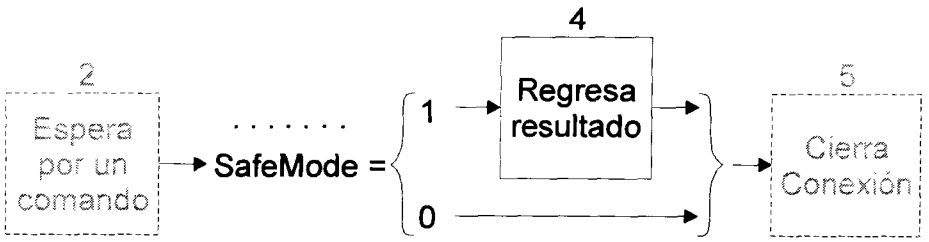


Figura 3.30. Variante de la etapa "Regresa resultado".

Aclaración sobre de los diagramas de flujo.
A partir de este punto, en diferentes secciones se utilizan diagramas de flujo para describir el funcionamiento de algunas tareas del programa del módulo de comunicación. Es importante tener en mente y poner énfasis en que el programa del módulo de comunicación funciona en un ambiente de multitarea cooperativa bajo el principio de una máquina de estados. Es decir, cada tarea del programa cede el control hacia la siguiente tarea en un orden preestablecido; con esto, el programa principal da la impresión de estar ejecutando varias tareas en paralelo.
En los diagramas de flujo de las siguientes secciones los bloques de los diagramas indican operaciones generales relativas a una tarea en particular. Es importante recalcar que las operaciones de los bloques no representan precisamente los estados que conforman a una tarea. Los bloques tan sólo se utilizan para mencionar las fases primordiales de la ejecución una tarea en particular; de hecho, la mayoría de los bloques, internamente están seccionados para ceder el control hacia las demás tareas.

3.3.4. El Servidor Serial Asíncrono.

La función del servidor serial hace que el módulo de comunicación permanezca siempre atento a la solicitud de un servicio por parte de un usuario a través del puerto serial asíncrono; es decir, el módulo de comunicación siempre esta escuchando y busca interpretar lo que recibe a través del puerto serial para posteriormente ejecutar alguna acción. El servidor serial esta diseñado para atender a un sólo usuario.

3.3.4.1. Configuración Inicial del Servidor Serial Asíncrono.

El servidor serial al inicio del programa asume la configuración indicada en la tabla 3.9, es decir, espera establecer una comunicación a 9600 bits por segundo, con 8 bits de datos, sin paridad, y un bit de paro. Para cambiar la configuración del puerto de comunicación serial existe un comando para tal propósito, el cual, puede ejecutarse tanto a través del puerto serial asíncrono como a través del puerto ethernet.

3.3.4.2. Funcionamiento del Servidor Serial Asíncrono.

El funcionamiento del servidor serial se describe en el diagrama de flujo de la figura 3.31. La función del servidor serial consiste en escuchar constantemente el puerto serial de entrada para identificar la solicitud de un comando. Una vez identificado y validado el comando, el módulo realiza la operación asociada al comando, y dependiendo del tipo de

operación solicitada, el módulo de comunicación puede emitir o no emitir una respuesta a través del puerto serial asíncrono de salida.

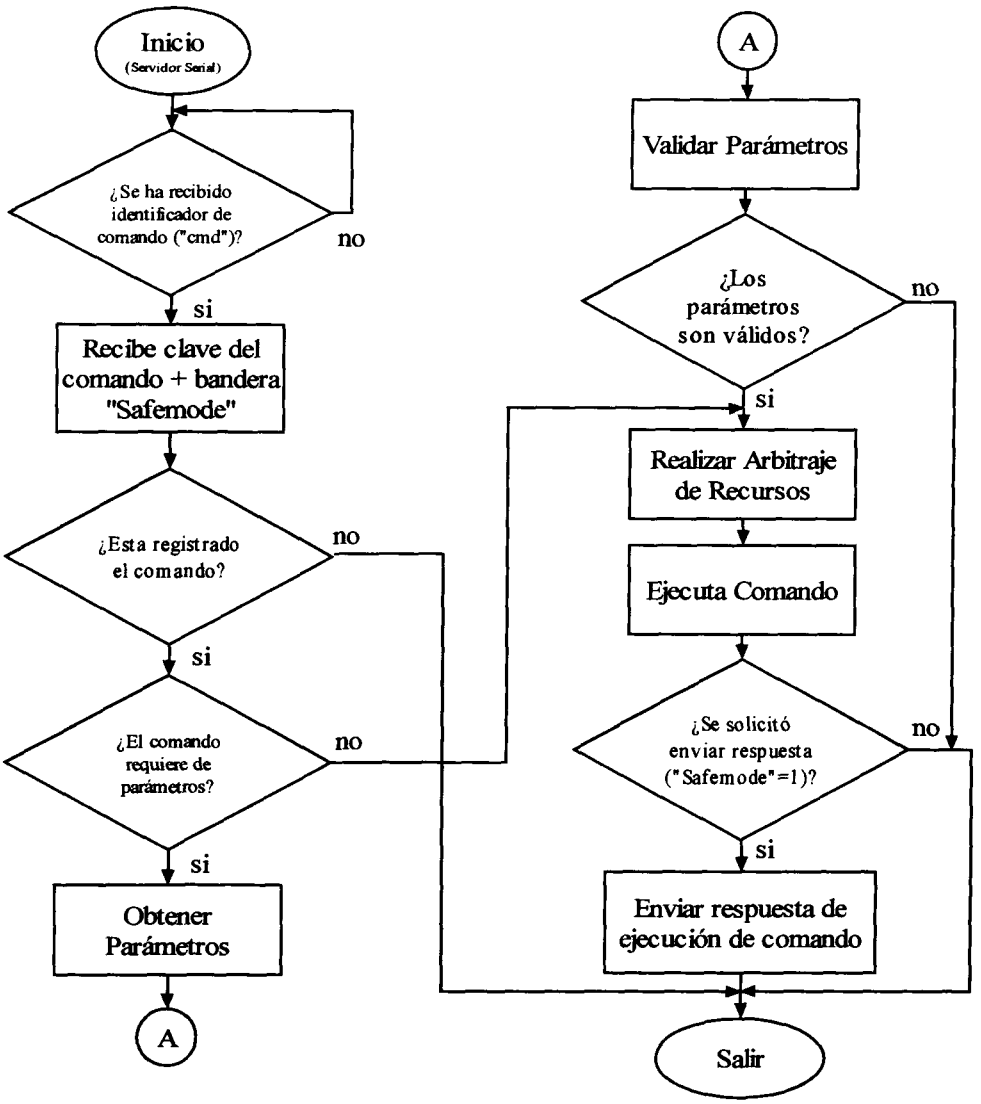


Figura 3.31. Diagrama de flujo del servidor serial.

El bloque que aparece en el diagrama como: "Realizar arbitraje de recursos" se explica a detalle en la sección 3.3.6 ("Arbitraje de Recursos").

3.3.4.3. Protocolo del Servidor Serial Asíncrono.

La figura 3.32 muestra el esquema del protocolo del servidor serial para el envío y la recepción de comandos. Cabe destacar que la comunicación serial se efectúa de acuerdo a la configuración establecida en el puerto serial asíncrono, es decir, a una velocidad 'X' (9600) en bits por segundo, con 8 bit de datos, sin paridad, y un bit de paro. Dicha configuración serial esta inmersa en un protocolo serial de bajo nivel, el cual, es

susceptible a errores de comunicación. Por lo tanto, para agregar la capacidad de detección de errores en la comunicación serial del módulo de comunicación, sobre el protocolo serial de bajo nivel se implementa un protocolo sencillo en una etapa de mayor nivel; este nuevo protocolo incluye un campo de "checksum" para la validación de una trama de datos. La trama se considera como válida sólo si el "checksum" recibido en el destino corresponde efectivamente a la suma de todos los bytes de la misma trama.

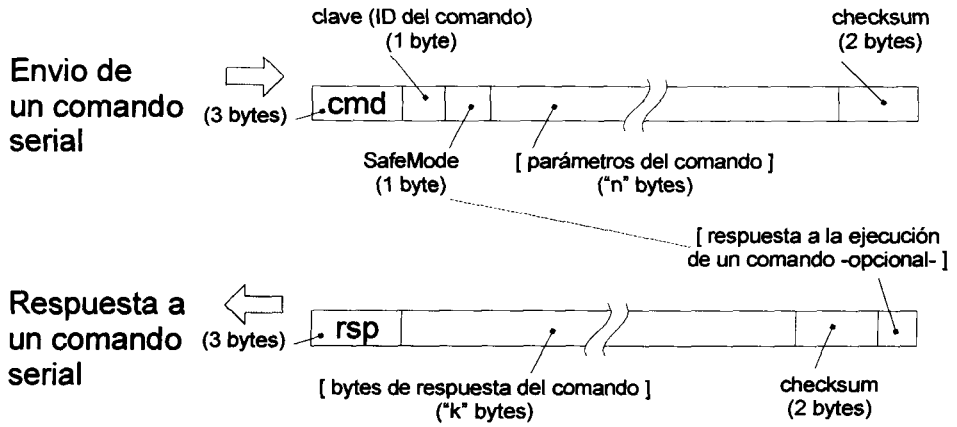


Figura 3.32. Formato del protocolo del servidor serial.

3.3.4.4. Envío de un Comando Serial.

El formato del envío de un comando para el servidor serial consta de los siguientes campos:

- Encabezado de inicio.
- Clave del comando.
- Modo Seguro ("SafeMode").
- Parámetros del comando (dependiente).
- Checksum.

Encabezado de inicio.

El encabezado de inicio de la trama para la solicitud de un comando consta de la secuencia de tres caracteres ASCII que conforman la cadena "cmd". Dicha cadena ocupa 3 bytes; el valor decimal de cada uno de los bytes de la cadena es el siguiente: 99 ('c'), 109 ('m'), y 100 ('d').

El servidor serial constantemente esta escuchando el puerto serial en búsqueda de la secuencia de caracteres del encabezado de inicio de un comando. Una vez que identifica la cadena que da inicio a un comando el siguiente byte de la trama representa la "clave del comando".

Clave del comando.

La "Clave del comando" identifica y diferencia un comando de los demás. Este campo tiene una longitud de 1 byte, por lo tanto, pueden diferenciarse hasta 256 comandos

distintos, a pesar de que actualmente sólo se tienen 47 comandos disponibles. La clave de cada comando se muestra en la columna 3 de la tabla 3.10.

Modo Seguro ("SafeMode").
El campo "Modo Seguro" se utiliza como una bandera para indicarle al módulo de comunicación si debe enviar o no debe enviar respuesta de la ejecución de un comando. Este campo tiene una longitud de un byte; cuando la bandera tiene el valor decimal de 49, es decir, equivale al caracter ASCII '1', el servidor regresa el resultado de la ejecución del comando; para cualquier otro valor de la bandera "modo seguro" el servidor serial omite la etapa de respuesta de la ejecución de un comando.

Parámetros del comando.
Seguidamente, a partir del comando solicitado, el servidor espera u omite la recepción de parámetros. El servidor cuenta con una tabla interna que relaciona cada comando con el número y tipo de parámetros esperados.
Por mencionar un ejemplo de un comando que requiere de parámetros, está el comando para fijar la hora y fecha del reloj de tiempo real, que requiere de los datos de la hora y fecha deseada a manera de parámetros. También existen comandos que no requieren de parámetros, como por ejemplo, el comando para la lectura del puerto digital de entrada, que para su ejecución no requiere de ninguna otra condición.
El campo de parámetros por lo tanto puede o no puede aparecer en la trama de la solicitud de un comando serial; todo depende del comando solicitado.

Checksum.
Por último, en la solicitud de un comando al final se utiliza un checksum de 2 bytes que consiste en la suma de todos los bytes de la trama de la solicitud de un comando, desde el encabezado de inicio hasta el último byte antes del checksum. Por una parte, el servidor recibe 2 bytes de checksum, y por otra parte, el mismo servidor calcula el checksum a partir de todos los bytes recibidos. Después, compara el checksum recibido con el checksum calculado y si ambos coinciden se considera que se realizó una recepción de comando exitosa, de lo contrario, el comando no se ejecuta o se reporta un error en la ejecución del comando.

3.3.4.5. Respuesta a un Comando Serial.

La respuesta a un comando serial esta compuesta por los siguientes campos:

- Encabezado de inicio.
- Bytes de respuesta (dependiente).
- Checksum.
- Respuesta a la ejecución (opcional)

Encabezado de inicio.
El encabezado de inicio de la respuesta de un comando esta compuesta por la cadena de tres caracteres "rsp". El valor decimal de cada uno de los tres bytes de la cadena es el siguiente: 114 ('r'), 115 ('s'), y 112 ('p').
Esta cadena tiene como utilidad facilitar la sincronización del usuario con el módulo de comunicación.

Bytes de respuesta.
Seguidamente, dependiendo de la naturaleza del comando, el servidor puede enviar o no enviar una serie de caracteres como parte de la ejecución del mismo comando. Por ejemplo, un caso en el que el módulo realiza un envío de datos hacia el usuario es en los comandos para lectura de memoria, los cuales, implican la transmisión del contenido de la memoria a través del puerto serial de tal forma que el usuario se entere de dicho contenido memoria. Por otra parte, hay casos como en los comandos de escritura a memoria en que el módulo de comunicación como parte de la ejecución del comando no requiere enviar datos al usuario. En resumen, el campo de "bytes de respuesta" es un campo cuya ausencia o presencia depende del comando solicitado al módulo de comunicación.

Checksum.
El campo "checksum" consiste en la suma de todos los bytes de la trama de la respuesta a un comando desde el encabezado de inicio hasta el último byte antes del checksum. La longitud de este campo es de 2 bytes y tiene como finalidad que el receptor pueda comprobar la integridad de los datos recibidos como respuesta.

Respuesta a la ejecución
Por último, en forma opcional se tiene el campo de "respuesta a la ejecución" del comando. Este campo se habilita (o deshabilita) previamente en la misma solicitud del comando. La longitud de este campo es de un byte y su valor indica el resultado exitoso o fallido (incluyendo el tipo de fallo) de la ejecución del comando solicitado.

3.3.5. El Servidor Ethernet.
El "servidor ethernet" esta diseñado para atender hasta "N" usuarios simultáneos que solicitan algún servicio u operación. Para ello, constantemente está atento al puerto de red ethernet en espera de una conexión de red. El parámetro "N" que equivale al número de conexiones simultáneas se define al momento de compilar el programa.

3.3.5.1. Configuración Inicial del Servidor Ethernet.
Al momento de la puesta en marcha el módulo de comunicación, el servidor ethernet adquiere los siguientes valores por defecto, mismos que se muestran en la tabla 3.9.
Dirección IP: 10.10.1.1
Puerto: #80
Netmask: 255.255.255.0
Gateway: 10.10.1.2
De estos 4 parámetros, todos a excepción del número de puerto (#80) pueden editarse mediante la solicitud de un comando. Dicho comando puede solicitarse tanto a través del puerto serial asíncrono como a través de una conexión de red.

3.3.5.2. Funcionamiento del Servidor Ethernet.
El diagrama de flujo de la figura 3.33 describe la lógica del funcionamiento del servidor ethernet. El servidor ethernet utiliza la misma lógica para atender a los "N" distintos usuarios.

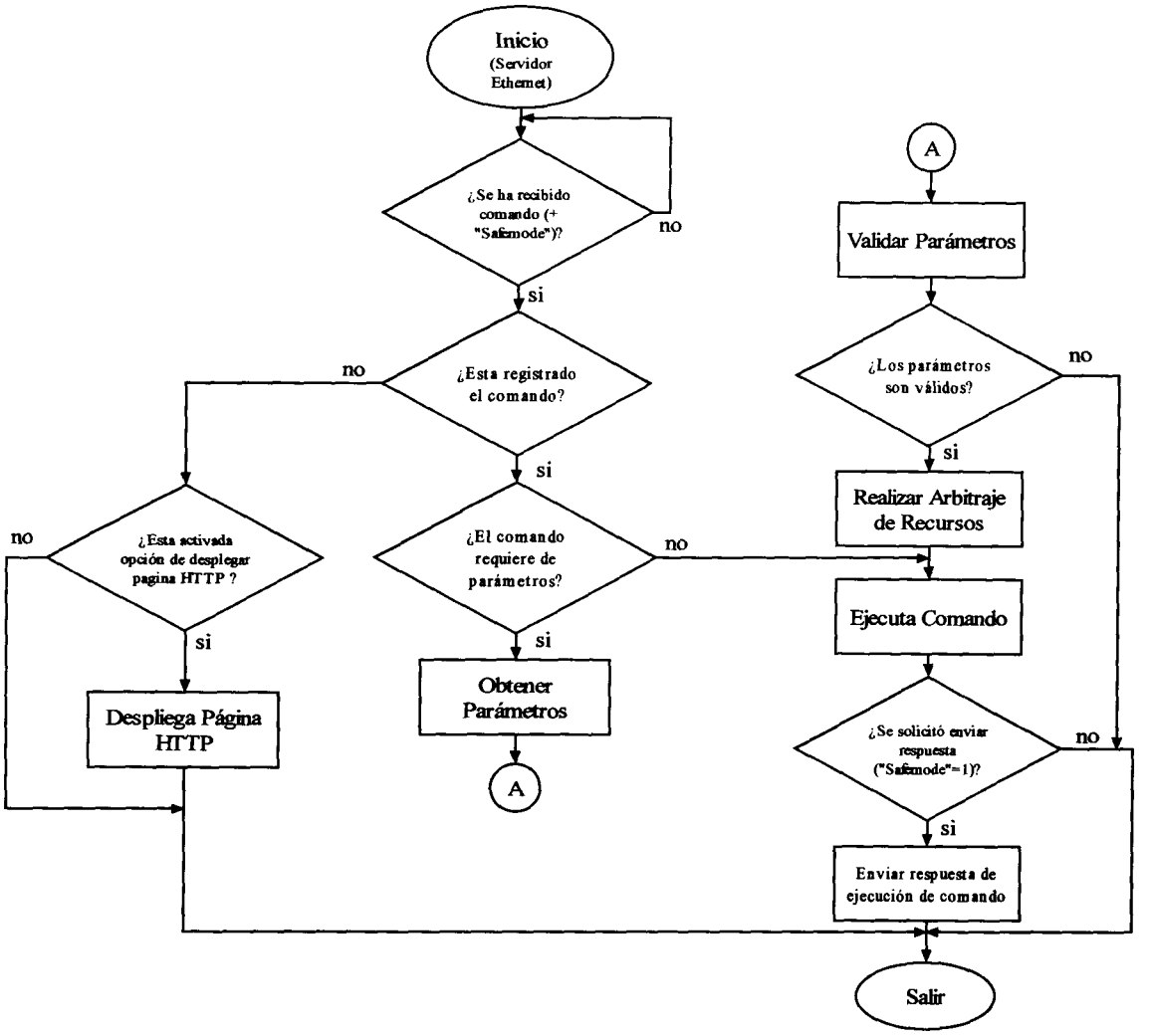


Figura 3.33. Diagrama de flujo del servidor ethernet.

La lógica de la operación del servidor ethernet es muy semejante a la lógica del servidor serial. Sin embargo una diferencia que resalta, además del medio de comunicación que utiliza cada servidor, es la decisión que toma el programa cuando un comando recibido no esta registrado. Esto es, cuando el identificador del comando recibido no corresponde a ninguno de los comandos enlistados como válidos en la tabla 3.10. En el caso del servidor serial cuando se recibe un comando no registrado el programa sencillamente ignora el comando y vuelve al estado inicial en espera de la solicitud de un nuevo comando; mientras que en el servidor ethernet la recepción de un comando no registrado se interpreta como una solicitud hecha por un navegador de internet (o "browser"). El usuario puede configurar si desea o no desea enviar una respuesta a la solicitud de un comando no registrado. Como parte de la misma configuración para el envío de una respuesta, el usuario define la localidad y la sección de memoria que será enviada. La sección de memoria puede estar ubicada en Memoria Raíz (RAM), Memoria Extendida (RAM), o Memoria Externa (Flash); dentro de esta sección de memoria se ubica la información que se quiere mostrar en la pantalla del navegador; la información puede ir

acompañada de comandos HTTP para darle un formato más vistoso al momento de ser desplegada en el navegador, también puede estar siendo actualizada o modificada directamente en memoria a través de una o varias conexiones del módulo con distintos usuarios.
La función de otros bloques sobre los cuales no se ha hecho mención hasta este punto son la "Validación de Parámetros", y el "Arbitraje de Recursos".
La validación de parámetros consiste en verificar que los parámetros tengan coherencia y que estén dentro de los rangos permitidos. Por ejemplo, en el caso de un comando de lectura a memoria se verifica que los parámetros queden dentro de los límites del tamaño de la memoria antes de realizar la lectura.
El arbitraje de recursos se encarga de evitar conflicto entre 2 o más usuarios que requieran utilizar simultáneamente el mismo recurso. El arbitraje de recursos se explica a detalle en la sección 3.3.6.

3.3.5.3. Protocolo del Servidor Ethernet.

La figura 3.34 muestra el esquema del protocolo propuesto del servidor ethernet tanto para el envío como para la respuesta de un comando. El protocolo propuesto se apoya en el protocolo de red TCP ("Transmission Control Protocol") que es un protocolo confiable, por lo tanto, no requiere de una fase de corrección de errores, ya que la llegada de todos los campos que conforman el protocolo propuesto esta garantizada.

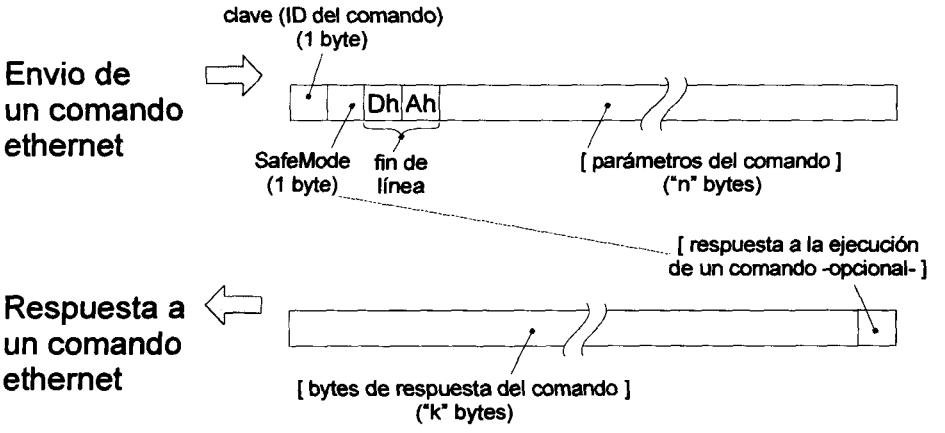


Figura 3.34. Formato del protocolo del servidor ethernet.

3.3.5.4. Envío de un Comando Ethernet.

El formato del envío de un comando para el servidor ethernet consta de los siguientes campos:

- Clave del comando.
- Modo Seguro.
- Fin de línea.
- Parámetros del comando.

Los campos del envío de un comando ethernet son muy similares a los campos del servidor serial, de hecho, los campos: "Clave del comando", "Modo Seguro", y "Parámetros del comando" cumplen con la misma descripción de la sección del servidor serial (ver sección 3.3.4.4)
El único campo distinto es el de "fin de línea" y es la forma en que el programa identifica el límite que divide la clave y el modo de operación del resto de los datos de un comando. El "fin de línea" consta de 2 bytes, cuyo valor en hexadecimal es: 0D0A (3338 decimal) y en código ASCII representan la secuencia de un comando de retorno de carro y de un comando de retorno de línea.

3.3.5.5. Respuesta a un Comando Ethernet.

La respuesta a un comando ethernet esta compuesta por los siguientes campos:

- Bytes de respuesta (dependiente).
- Respuesta a la ejecución (opcional)

Bytes de respuesta.

Este campo de bytes de respuesta, es análogo al del protocolo del servidor serial. Es decir, es un campo cuya presencia o ausencia depende de la naturaleza del comando solicitado. Existen comandos que por ejemplo envían un flujo de datos como parte de su ejecución, por ejemplo, un comando de lectura a memoria implica el envío de los datos contenidos en el interior de la memoria. En el lado opuesto, también hay casos de comandos no envían "bytes de respuesta" como parte de su ejecución, tal es el caso de un comando de configuración o de un comando de escritura.

Respuesta a la ejecución

El campo de "respuesta a la ejecución" del comando es de carácter opcional. Este campo se habilita (o deshabilita) en la misma solicitud del comando mediante el estado del campo "safemode" ('1'=habilitado, cualquier otro valor = deshabilitado). La longitud de este campo es de dos bytes y su valor indica el resultado exitoso o fallido (incluyendo el tipo de fallo) de la ejecución del comando solicitado.

3.3.6. Arbitraje de recursos.

El hecho de tener múltiples solicitudes en forma simultánea sobre los mismos recursos obliga en el programa principal a ejecutar una fase de arbitraje para evitar conflicto en el uso de los recursos.
El arbitraje de recursos se encarga de evitar conflicto entre 2 o más usuarios que solicitan utilizar simultáneamente el mismo recurso. Los recursos que pueden entrar en conflicto son la memoria, y el puerto serial de salida. Por ejemplo, si dos (o más) usuarios requieren al mismo tiempo enviar datos por el puerto serial de salida, el primero que sea detectado por el servidor es atendido, mientras que el segundo usuario y los demás usuarios que en ese momento llegaran a solicitar el puerto serial entran en una cola de espera para después ser atendidos uno a uno en el orden que se registraron las solicitudes para el uso del puerto serial.
La misma estructura y el flujo del programa principal del módulo de comunicación en forma normal detecta y da prioridad a las solicitudes de los usuarios. Esto se logra gracias al proceso de barrido del esquema de multitarea cooperativa. Es importante recordar que

el programa principal está estructurado y funciona como una máquina de estados y que existen dos tipos de operaciones relacionadas con los comandos que son (ver sección 3.3.3):

- De ejecución completa.
- De ejecución fraccionada.

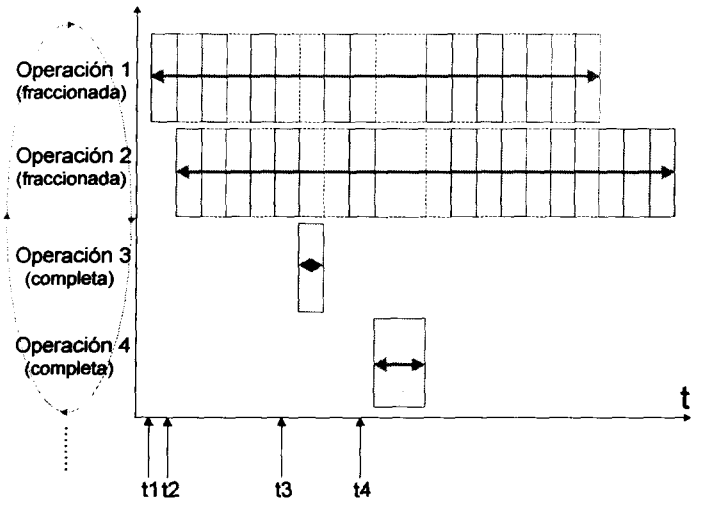


Figura 3.35. Diagrama de tiempos que muestra la ejecución de tareas fraccionadas y tareas completas.

La figura 3.35 muestra la forma en que el procesador realiza la ejecución de 4 distintas operaciones. El eje horizontal de la figura representa el tiempo; t1 a t4 indica el momento de la solicitud de cada operación correspondiente (1 a 4). Las flechas horizontales con puntas en ambos extremos indican la duración de cada operación. Las operaciones 1 y 2 para completarse se ejecutan en forma segmentada, mientras que las operaciones 3 y 4 de ejecución completa se ejecutan de principio a fin en cuanto el procesador las atiende.

Las operaciones de "ejecución completa" no entran explícitamente a una etapa de procesamiento de arbitraje puesto que se ejecutan y finalizan en el momento que son detectadas y antes de que el programa preste atención al resto de los usuarios del módulo de comunicación.
Sólo las operaciones de "ejecución fraccionada" se someten a una etapa de arbitraje para evitar que ocurra conflicto de recursos entre distintas solicitudes de comandos simultáneas.
Los recursos que implican "operaciones fraccionadas" y que se someten a una etapa de arbitraje son los siguientes:

- Memoria extendida.
- Memoria externa.
- Puerto serial asíncrono de salida.

La figura 3.36 muestra el ejemplo de la ejecución de 4 operaciones bajo el esquema de multitarea cooperativa, en donde dos de ellas (operaciones 1 y 2) entran en conflicto al intentar hacer uso del mismo recurso simultáneamente. La operación 2 entra en una cola

de espera puesto que se detectó después del inicio de la operación 1, ambas operaciones (1 y 2) con solicitud de uso del mismo recurso. En la figura se indica el tiempo de espera que ocasiona el arbitraje. El inicio de la de la operación 2 ocurre en la primera oportunidad una vez que la operación 1 finaliza y libera el recurso en conflicto.

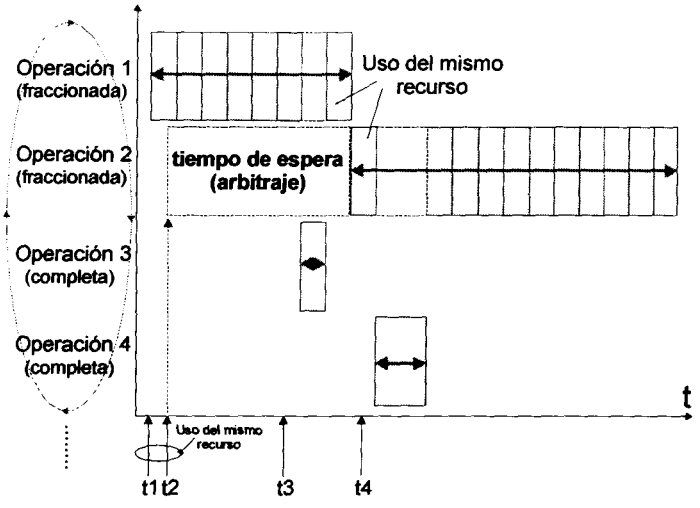


Figura 3.36. Efecto del arbitraje entre operaciones que entran en conflicto al intentar hacer uso del mismo recurso.

Reglas de arbitraje.

Las reglas de arbitraje son distintas para el puerto serial y para los dos tipos de memoria. La razón es que en una transmisión serial no deben mezclarse los datos de salida de dos o más comandos distintos, ya que la secuencia de los datos en una transmisión serial es un factor básico.

En el caso de acceso a una memoria, el orden en que los datos son leídos o escritos no es relevante. Lo que sí es importante es que los datos de dos o más operaciones simultáneas no se mezclen. Esto ocurre únicamente cuando dos o más operaciones realizan un acceso a la misma localidad de memoria y al menos una de las operaciones es de escritura. Es importante tomar en cuenta que cuando todas las operaciones simultáneas son de lectura, no existe ningún riesgo de modificar los datos contenidos en memoria, por lo tanto, en este caso se garantiza la integridad de los datos de la memoria y el acceso múltiple a memoria es permitido.

La única operación que modifica el contenido de memoria es una escritura. Cuando sobre un mismo bloque de memoria se presenta una escritura y cualquier otra operación simultánea existe el riesgo de que los datos de la segunda operación se mezclen con los datos de la escritura en transición. Es por ello que se realiza el arbitraje de memoria, y en el arbitraje se consideran los siguientes puntos:

- Tipo de operación: Lectura o Escritura

- Rango de la operación: Dirección de memoria inicial y dirección de memoria final.

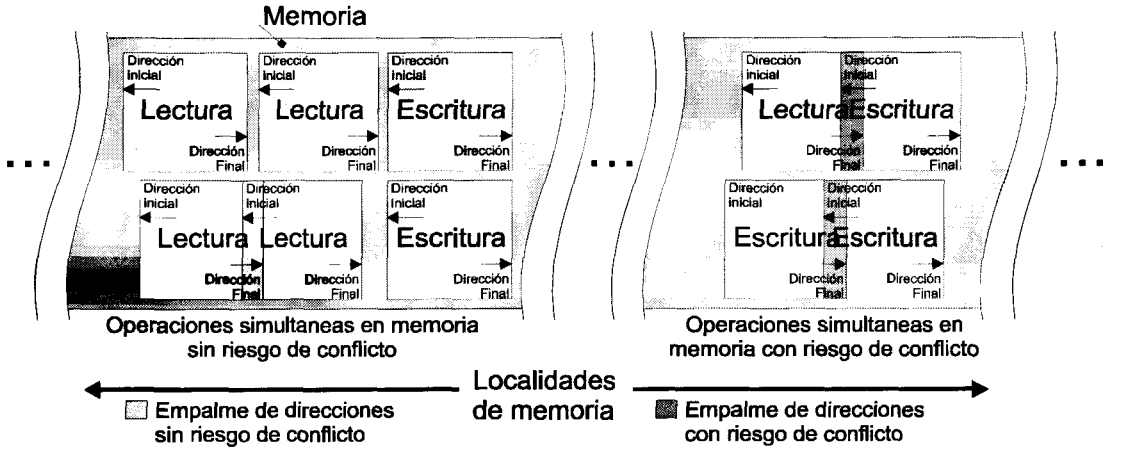


Figura 3.37. Ejemplo de operaciones simultáneas en una misma memoria con riesgo y sin riesgo de conflicto

En resumen las regla de arbitraje para la memoria se reduce a verificar si el rango de direcciones de la última solicitud de la memoria se empalma con el rango de direcciones de cualquier operación de escritura en curso. En otras palabras, la escritura de un determinado bloque de la memoria no debe empalmarse con otra operación simultánea, ya sea escritura o de lectura. Las reglas de arbitraje de memoria se describen mediante el siguiente diagrama de flujo.

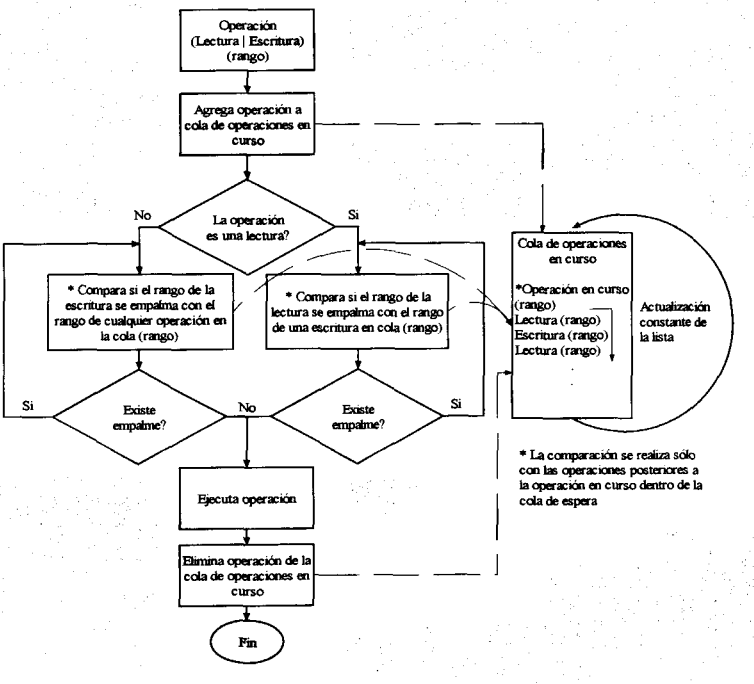


Figura 3.38. Diagrama de flujo para el arbitraje de memoria.

Por otra parte, la regla de arbitraje para el puerto serial se basa en el estado de la salida del mismo puerto serial al momento de recibir una nueva solicitud por parte de un usuario. El puerto serial puede estar en dos estados que son: "ocupado" o "libre".
Si el puerto serial esta ocupado, la solicitud entra a una cola de espera para ser atendida posteriormente en cuanto se libere el puerto serial.
En el caso contrario, si el puerto serial esta libre, la operación se ejecuta tomando posesión del puerto serial.

Estado del puerto serial	Accion de Arbitraje
libre	Se ejecuta la operación y se hace uso del puerto serial
ocupado	La operación entra en cola de espera → ▥▥▥→

Tabla 3.12. Reglas de arbitraje para el puerto serial.

3.3.7. Operaciones Cíclicas del Módulo de Comunicación.

Las operaciones cíclicas del módulo de comunicación consisten en tres tareas independientes entre sí, y que en el programa principal se ejecutan en forma paralela.
El diagrama de flujo que muestra el funcionamiento de las tres tareas se muestra en la siguiente figura.

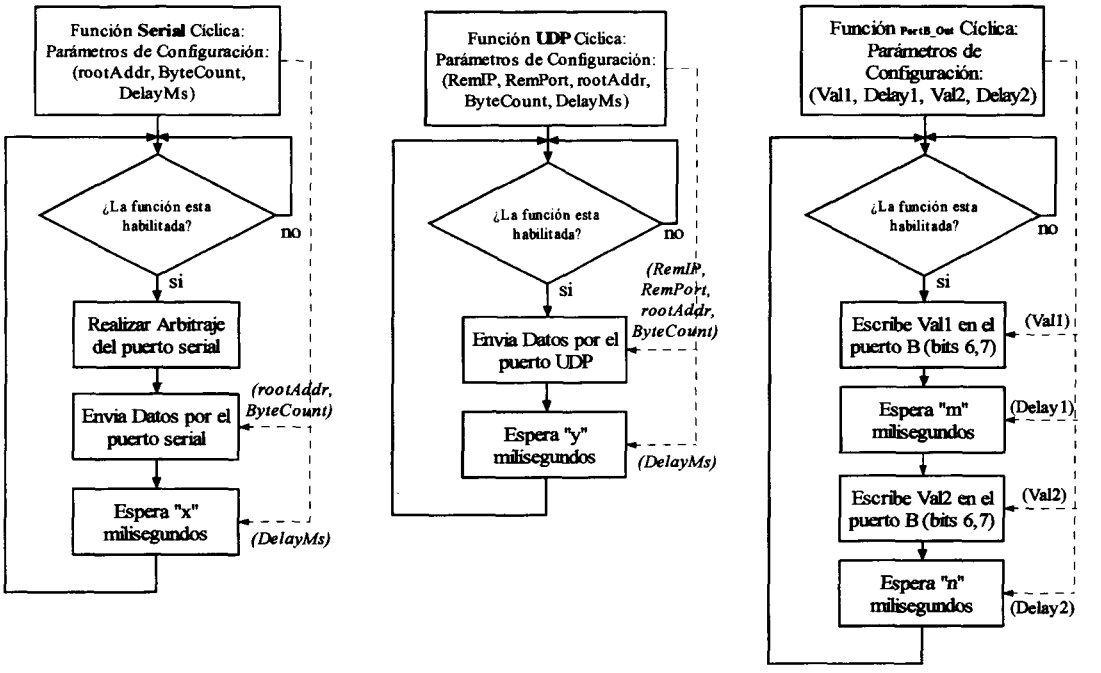


Figura 3.39. Diagrama de Flujo de las tareas "Operaciones Cíclicas".

Estas tres tareas pueden habilitarse y configurarse selectivamente mediante la solicitud de los comandos correspondientes (#33, 34, y 35 de la tabla 3.10). La validación de los parámetros de configuración de estas tareas se realiza durante la solicitud del mismo comando que las configura y habilita. La configuración de las tareas queda guardada en una área específica de memoria RAM, a partir de la cual, cada tarea obtiene los parámetros para regular su función.
Algunos detalles sobre las operaciones cíclicas también se exponen en la sección (3.3.2.2. "Comandos para Manejo de Puertos")
En el caso de la tarea cíclica para el puerto serial de salida también se entra en un proceso de arbitraje, tal y como sucede en las tareas de los servidores; esto, para evitar conflictos con otras posibles operaciones derivadas de los comandos que simultáneamente requieran hacer uso del puerto serial.

3.4. Funcionamiento de la Memoria Externa.

Hasta este punto se ha expuesto a la memoria externa como un módulo opcional de acceso múltiple y que puede ser compartido entre dos o más usuarios. También se expuso la lógica de control que utiliza el módulo de comunicación por medio de las líneas de control LIU y RIU para coordinar el acceso de la memoria entre el módulo de comunicación y uno o más usuarios externos.
En esta sección se exponen algunos detalles internos sobre el módulo de memoria externo que pueden ser útiles para introducir al programador de aplicaciones sobre la estructura y funcionamiento de la memoria. De igual forma, puede ayudar al usuario a prolongar la vida del dispositivo y hacer un uso más eficiente de la memoria.
El módulo de memoria externo básicamente consiste en un chip de memoria flash fabricado por la compañía ATMEL. En la fabricación del módulo de memoria externo, se utilizan dos circuitos de memoria de distinta capacidad pero que funcionan bajo el mismo principio y el mismo protocolo.

Capacidad de la memoria flash (bytes)	No. de parte del fabricante (Atmel)
4 M	AT45DB321
8 M	AT45DB642

Tabla 3.13. Capacidad de la memoria del módulo de memoria externo y el correspondiente número del circuito integrado de memoria.

Para el uso de la memoria externa se puede hacer referencia a las hojas de datos del fabricante de los circuitos integrados de memoria (AT45DB321 y AT45DB642). En dichas hojas de datos aparece la estructura interna de la memoria, los distintos comandos para la manipulación de la memoria, el protocolo de comunicación serial, sus distintas variantes, etcétera.
Para efectos prácticos, los datos expuestos en esta sección se basan en la hoja de datos del circuito de memoria AT45DB321 de 4 Megabytes. El funcionamiento de este circuito es

análogo al del circuito de 8 MB, la diferencia entre ambos es su capacidad de almacenamiento.

Tanto las lecturas como escrituras a memoria se realizan a través de un puerto de comunicación serial síncrono. Las operaciones en memoria se efectúan a través de comandos. La memoria cuenta con su propio protocolo para recibir y dar respuesta a un comando. Por ejemplo, una escritura en memoria requiere del envío en forma serial del comando de escritura, seguido de los parámetros y de los datos a escribir en memoria. Existen distintos comandos para la manipulación de la memoria, y es común que durante una rutina de escritura o lectura del programa del módulo de comunicación se lleve a cabo un proceso en el que se ejecutan varios comandos.

3.4.1. Niveles de voltaje de entrada y salida.

El circuito integrado de memoria del módulo de comunicación opera con voltajes en el rango de 0 a entre 2.7V y 3.6V. La tarjeta del módulo de comunicación (de la serie "SF1000") utiliza una etapa de acondicionamiento tanto para acoplar la entrada como la salida del circuito integrado a niveles compatibles con TTL, entre 0 y 5 volts, que son los niveles que maneja el resto de la lógica del módulo de comunicación. En la entrada de la memoria se utiliza un divisor de voltaje para reducir el nivel de 5 volts, y a la salida se utiliza un transistor que funciona como inversor, y que a la vez, acopla los niveles del circuito integrado de la memoria con los niveles TTL.
Es muy importante considerar entonces que la salida del módulo de comunicación ("SF1000") está invertida con respecto a la salida que proporciona el circuito integrado de memoria.

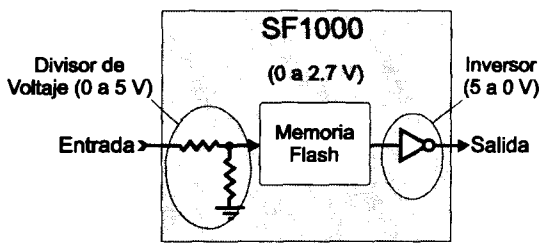


Figura 3.40. La salida del módulo de memoria ("SF1000") está invertida con respecto al circuito integrado de memoria flash.

3.4.2. La estructura interna de la memoria.

Internamente el circuito de memoria esta compuesto por un las siguientes partes:

- Un arreglo de memoria flash
- Dos "buffers" de memoria RAM
- Una interfase de comunicación serial.

El arreglo de memoria flash.

El arreglo de memoria flash esta organizado en forma de páginas. Cada página tiene una cierta capacidad de almacenamiento. El número de páginas por la capacidad de la página, proporciona la capacidad de almacenamiento total de la memoria.

Los buffers de memoria RAM.
Los buffers se utilizan para la transferencia intermedia de datos entre la memoria principal y la interfase de comunicación de datos.

Una interfase de comunicación serial.
La interfase de comunicación serial detecta la solicitud de comandos para el manejo de la memoria. Además de los comandos, a través de la interfase serial se comunican los datos a leer o escribir en memoria.

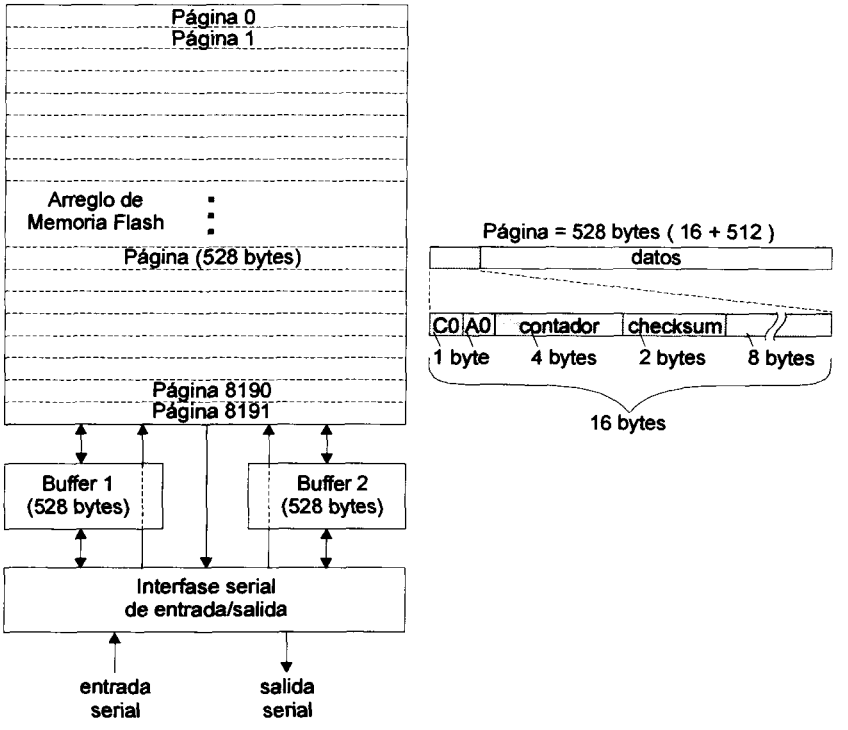


Figura 3.41. Estructura interna del circuito de memoria de 4 MB.

Tanto la memoria de 4MB como la de 8 MB cuentan con 8192 páginas, la diferencia entre ellas es el tamaño de las páginas (o "sectores"); mientras que para la memoria de 4MB el tamaño de cada página es de 528 bytes, para la de otra es de 1056 bytes.

3.4.3. Identificación del dispositivo de memoria.

Dentro de la variedad de comandos con los que cuenta el circuito integrado de la memoria, existe un comando de "estatus", el cual, además de entregar el estado ("ready" / "busy") de la memoria, entrega un número "identificador de densidad". El identificador de densidad se utiliza para identificar el tipo de circuito integrado de memoria, y por lo tanto, sus características internas, y su capacidad de almacenamiento.
El software del módulo de comunicación esta diseñado para operar con cualquiera de las dos capacidades de memoria, es por ello que el programa ejecuta una rutina de inicialización de memoria, para identificar la capacidad de la misma. De acuerdo al valor

del "identificador de densidad" el programa define los valores de los parámetros que utilizan las rutinas del programa del módulo de comunicación para el manejo de memoria. Cuando la respuesta del comando "estatus" es ilógica o desconocida, el programa asume la ausencia del módulo de memoria externo.

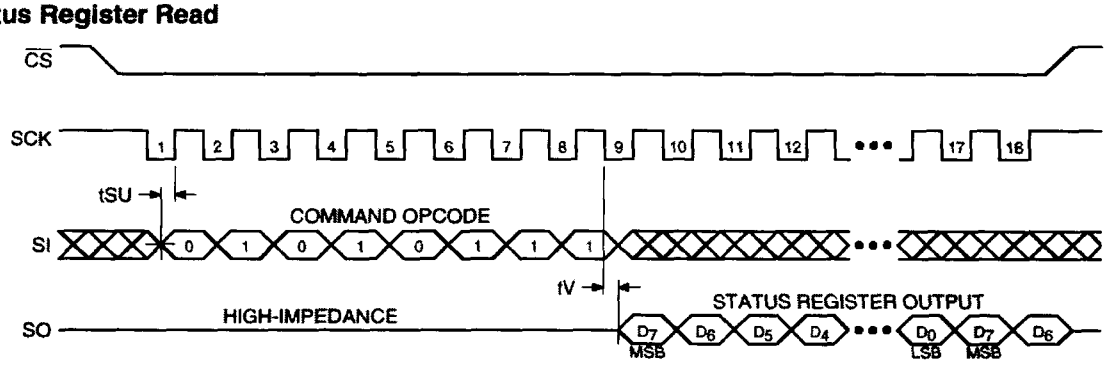


Figura 3.42. Diagrama de tiempo de una de las formas de ejecución del comando de lectura del registro estatus(*).

3.4.4. Direcciones físicas y direcciones lógicas de memoria.

Las direcciones físicas de memoria se refieren al total de las localidades del arreglo de memoria flash. Por ejemplo, si el arreglo de memoria flash tiene páginas de 528 bytes, de la dirección 0 a la dirección 527 se encuentran en la página 0; la de la dirección 528 a la dirección 1055 en la página 1, y así sucesivamente.

Por otra parte, el programa del módulo de comunicación dedica una porción de cada página para el almacenamiento de datos de control sobre el uso de la memoria; el resto del espacio de la página lo dedica para el almacenamiento de datos del usuario. Los datos del usuario se almacenan en direcciones lógicas. El programa del módulo de comunicación utiliza direcciones lógicas para la escritura y lectura de datos de la memoria externa. Los datos de control permanecen ocultos para las operaciones que utilizan direcciones lógicas.

Esto significa que en la aplicación del módulo de comunicación no toda la capacidad de la memoria física se utiliza para el almacenamiento de datos del usuario.

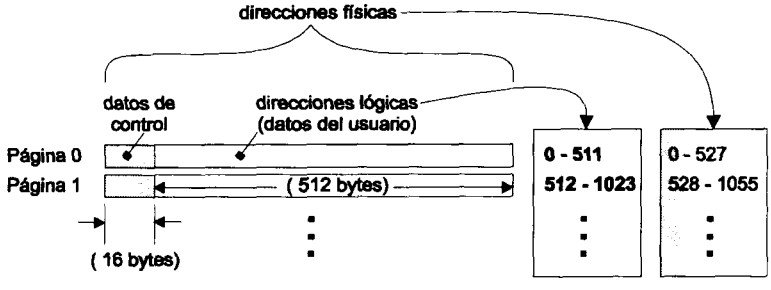


Figura 3.43. Direcciones físicas y direcciones lógicas del Módulo de memoria de 4MB.

* "32-megabit 2.7-volt Only Serial DataFlash", pág. 14.

En el área de memoria dedicada al almacenamiento de datos de control cuenta con los siguientes campos:

- Byte de sincronía
- Byte de Versión
- Contador de escrituras
- Checksum

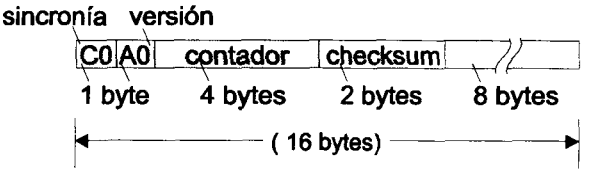


Figura 3.44. Area de almacenamiento de datos de control de cada página de la memoria de 4 MB.

El byte de sincronía se utiliza como referencia para saber si la página en cuestión ha sido escrita previamente o si nunca ha sido escrita. Si el valor del byte de sincronía equivale a C0 hexadecimal (192 decimal) significa que la página de memoria ha sido escrita previamente; de lo contrario, el programa asume que la página nunca ha sido escrita.
El byte de versión identifica la versión del software y la convención utilizada en el manejo de la memoria. Para este caso el valor del byte de versión es A0 hexadecimal (160 decimal).
El contador lleva registro del número de veces que ha sido escrita la página en cuestión. El contador tiene una longitud de 32 bits (4 bytes) con lo que es posible contar hasta 4.29 x 10^9.
El espacio de "checksum" por el momento no se utiliza al igual que los siguientes 8 bytes adyacentes.
En total, para la memoria de 4 MB, la longitud del área de datos de control es de 16 bytes por página. Esto significa que del total del espacio de memoria física (16 bytes x 8192 páginas =) 131,072 bytes no se utilizan para el almacenamiento de datos.

La tabla 3.14 muestra la comparación entre las especificaciones de las memorias de 4 y 8 MB. La columna del "tamaño de sector o página" se refiere al espacio físico de cada página de la memoria. La columna del "tamaño de bloque" se refiere al número de localidades lógicas por página destinadas para el almacenamiento de datos del usuario. La diferencia entre el "tamaño de sector" y "tamaño de bloque" representa el espacio destinado por página para el "almacenamiento de datos de control".
En ambas capacidades de memoria el porcentaje del espacio físico total que se aprovecha para el almacenamiento de datos del usuario es el mismo: 96.97%.
Mientras que el porcentaje restante (3.03%) se utiliza para el "almacenamiento de datos de control".

Capacidad de la memoria flash (bytes)	Identificador de Densidad	Tamaño de Sector (página)	Tamaño de Bloque	Número de Bloques
4 M	6	528	***512***	8192
8 M	7	1056	***1024***	8192

Tabla 3.14. Tabla comparativa entre los parámetros de las dos capacidades de memoria.

Términos del software del módulo de comunicación.
En el software del módulo de comunicación se utilizan los siguientes términos:
"Sector": equivale a una página.
"Bloque": que equivale al área de cada página destinada para el almacenamiento de datos del usuario. El acceso al espacio del bloque se realiza mediante direcciones lógicas.
"Prefijo": equivale al área de almacenamiento de datos de control.
Para la memoria externa, el número de páginas, de sectores y de bloques, es el mismo.

3.4.5. Rutina de escritura del módulo de comunicación.

Un punto muy importante a considerar en la escritura de la memoria del módulo externo es que la escritura se efectúa únicamente por página. Es decir, si se desea modificar tan solo un byte del contenido de la memoria, la modificación implica volver a escribir la página completa en donde se encuentra el byte en cuestión.
La memoria tiene un tiempo de vida limitado a un cierto número de escrituras, por eso es importante considerar y tomar en cuenta este hecho.
Idealmente, para prolongar el tiempo de vida de la tarjeta de memoria, el usuario debe procurar realizar escrituras que abarquen direcciones de bloques completos; y/o evitar tanto como sea posible realizar escrituras segmentadas (y que puedan evitarse) sobre direcciones que se encuentren dentro de un mismo bloque. Por ejemplo, la escritura de un bloque de 512 bytes desde la dirección 0 a la dirección 511 de la memoria flash. Escrita byte por byte implicaría 512 escrituras sobre la página 0 de la memoria flash, mientras que escrita en forma de un bloque de 512 bytes, implica solamente una escritura sobre la página 0 de la memoria flash.

La rutina del módulo de comunicación para la escritura de la memoria externa recibe como parámetros direcciones lógicas, además del bloque de datos a escribir. La rutina se encarga de escribir los datos en las direcciones físicas correspondientes y de actualizar el contador de escrituras en cada una de las páginas de la memoria donde se efectúa una escritura. Una característica importante de la rutina de escritura es que todas las operaciones de escritura en bajo nivel, se reducen a escribir el contenido íntegro del "buffer 1" de memoria RAM a una determinada página de la memoria flash. En otras palabras, en la rutina de escritura del módulo de comunicación, todos los datos que se escriben en la memoria flash invariablemente pasan por el "buffer 1".
En la rutina de escritura del módulo de comunicación para la escritura de la memoria externa, se realizan tres operaciones básicas:

- Transferir un Sector a RAM
  La transferencia de un sector a RAM consiste en copiar la información íntegra de una determinada página ("sector") de la memoria flash al "buffer 1" de memoria RAM.
- Transferir la RAM a un sector.
  La transferencia de la RAM a un sector consiste en escribir el contenido del "buffer 1" a una determinada página completa de la memoria flash.
- Escribir bloque de datos en "buffer 1" a través del puerto serial.
  Con este comando se puede escribir una sección o completamente el "buffer 1" mediante una comunicación serial. El comando recibe serialmente como parámetro la dirección inicial del buffer donde se empiezan a escribir los datos seguido de flujo serial de datos a escribir en el buffer.

Para comprender el proceso de la rutina de escritura del módulo de comunicación, se presenta el siguiente ejemplo de una escritura a memoria externa. En este ejemplo la escritura abarca el bloque de las direcciones lógicas entre la dirección 340 y 1439. Dicho bloque de datos tiene una longitud de 1,100 bytes y abarca las primeras 3 páginas (0, 1 y 2) de la memoria, tal como se muestra en la figura 3.45.

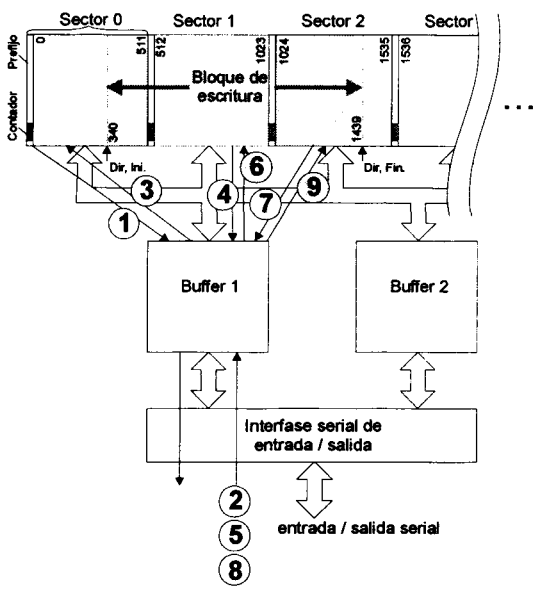


Figura 3.45. Pasos de la rutina de escritura a memoria externa por parte del programa del módulo de comunicación.

Los pasos de la rutina de escritura del programa del módulo de comunicación se desglosan a continuación:

1.- Transfiere Sector 0 a "buffer 1".
2.- Escribe bloque parcial de datos (dir. 340 a 511) en "buffer 1" a través del puerto serial.
3.- Transfiere "buffer 1" a Sector 0.

4.- Transfiere Sector 1 a "buffer 1".
5.- Escribe bloque de datos (dir. 512 a 1023) en "buffer 1" a través del puerto serial.
6.- Transfiere "buffer 1" a Sector 1.
7.- Transfiere Sector 2 a "buffer 1".
8.- Escribe bloque parcial de datos (dir. 1024 a 1439) en "buffer 1" a través del puerto serial.
9.- Transfiere "buffer 1" a Sector 2. (Fin)

Los pasos de la escritura de la memoria flash (pasos 3, 6, 9) se resaltan en la figura con color gris. Dichos pasos corresponden al comando de transferencia del "buffer 1" a un sector de la memoria flash. Es importante recalcar que la escritura de la memoria se realiza en forma interna, ya que el "buffer" y sus datos se encuentran en el interior del mismo circuito integrado de la memoria flash. La escritura de la memoria se dispara mediante el comando de transferencia del "buffer 1" a una determinada página.
En la misma transferencia del buffer a un sector de la memoria, el programa del módulo de comunicación, previamente se encarga de incrementar el campo del "contador" del prefijo de la página sobre la cual se realiza la escritura. Para ello, el programa verifica el valor del campo de "sincronía", si este último campo equivale al valor predefinido C0 (hexadecimal) el contador se incrementa en uno; de lo contrario el programa asume que la página nunca ha sido escrita, y por lo tanto, inicializa el campo del contador al valor de 1 y el campo de "sincronía" al valor C0 (hexadecimal).

3.4.6. Rutina de lectura del módulo de comunicación.

El proceso de lectura de la memoria es más sencillo en comparación con el proceso de escritura. En el caso de la lectura, la memoria no esta limitada a un cierto número de lecturas que defina el tiempo de vida útil de la memoria.
En el proceso de lectura las transferencias de datos de la memoria hacia el exterior también se efectúan a través del "buffer 1" del circuito integrado de memoria.
En la rutina de lectura del módulo de comunicación se realizan las siguientes operaciones básicas:

- Transferir un Sector a RAM
  La transferencia de un sector a RAM consiste en copiar la información íntegra de una determinada página ("sector") de la memoria flash al "buffer 1" de memoria RAM.
- Leer bloque de datos del "buffer 1" a través del puerto serial.
  Este comando recibe serialmente como parámetro la dirección inicial del buffer a partir de la cual se empiezan a leer los datos de la memoria; seguidamente, a la salida se obtiene el flujo serial de datos contenidos en el buffer.

A continuación se presenta un ejemplo de lectura para describir el proceso de la rutina de escritura del módulo de comunicación. En este ejemplo se utiliza el mismo bloque de direcciones que en el ejemplo de la escritura, es decir, el bloque de direcciones lógicas que comprende 1,100 bytes entre la dirección 340 y 1439, y el cual abarca las páginas 0, 1 y 2 de la memoria, tal como se muestra en la figura 3.46.

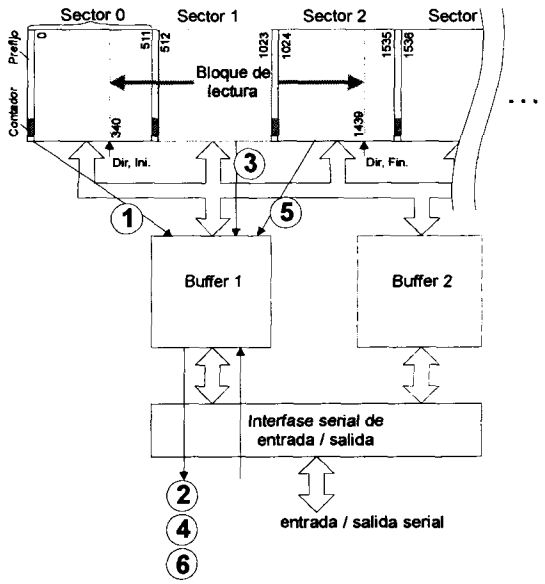


Figura 3.46. Pasos de la rutina de lectura de la memoria externa por parte del programa del módulo de comunicación.

Los pasos de la rutina de lectura del programa del módulo de comunicación son los siguientes:

1.- Transfiere Sector 0 a "buffer 1".
2.- Lee bloque parcial de datos del "buffer 1" (dir. 340 a 511) a través del puerto serial.
3.- Transfiere Sector 1 a "buffer 1".
4.- Lee bloque de datos del "buffer 1" (dir. 512 a 1023) a través del puerto serial.
5.- Transfiere Sector 2 a "buffer 1".
6.- Lee bloque parcial de datos del "buffer 1" (dir. 1024 a 1439) a través del puerto serial.
(Fin)

Es importante mencionar que las operaciones de cada uno de los 6 pasos de la rutina de lectura de la memoria externa implican el envío de un comando a la memoria. El comando junto con su(s) respectivo(s) parámetro(s) lo envía el programa a través de una comunicación serial. Por ejemplo, para transferir un sector al "buffer 1", el programa envía un comando junto con el parámetro que indica el número de sector a transferir al "buffer 1". La memoria recibe los comandos a través de la interfase de entrada serial. Para efectos de simplicidad, en las figuras 3.45 y 3.46 no se muestra en forma explícita el envío de comandos a través del puerto de entrada serial. Pero es importante tomar en cuenta que todos y cada uno de los pasos expuestos de las rutinas de escritura y lectura implican el envío previo del comando que da origen a la operación específica del paso.

3.4.7. Operaciones Misceláneas sobre la Memoria Externa.

Además de las rutinas básicas de escritura y lectura de la memoria, el módulo de comunicación cuenta con la facilidad de realizar otras rutinas auxiliares en el manejo de la memoria. A continuación se enlistan y describen brevemente dichas rutinas:

- Inicializar Memoria.
  Esta rutina es la misma que se expuso en la sección 3.4.3 ("Identificación del dispositivo de memoria"), y consiste en identificar el tipo de circuito integrado del módulo de memoria externo. En base al tipo de circuito integrado, el programa ajusta y adapta los parámetros de las rutinas para el manejo de la memoria. Los parámetros que ajusta el programa son: el tamaño de un "sector", tamaño de un "bloque", el tamaño del "prefijo", y el número de "bloques".
- Verificar Número de Escrituras.
  Esta rutina se encarga de leer el contador del número de escrituras de un "sector" (n) de la memoria flash. Para ello, la rutina realiza los siguientes pasos:
  1.- Transfiere Sector 'n' a "buffer 1".
  2.- Lee campo "contador" y lo envía a través del puerto serial.
- Borrar Sector.
  Esta rutina ejecuta un comando interno de la memoria que se encarga de borrar un "sector" completo (o "página") de la memoria. Cuando se borra un sector, todos las localidades (bytes) del sector adquieren un valor de FF hexadecimal. Las localidades de un sector comprenden tanto los datos almacenados por el usuario, como los datos del "prefijo", es decir, los datos de control en donde se incluye el contador de escrituras del sector. Al borrar un sector toda esa información se pierde.
- Escribir / leer "buffer 2"
  Todas las rutinas para el manejo de la memoria externa utilizan el "buffer 1" para la transferencia de datos. El "buffer 2", por lo tanto, queda disponible para cualquier otro uso. En el programa del módulo de comunicación se crearon un par de rutinas para la escritura y lectura del "buffer 2"; de esta forma se aprovecha el buffer como una área de memoria RAM compartida entre el módulo de comunicación y uno o más usuarios externos. El tamaño del "buffer 2" equivale al tamaño de un sector de la memoria flash, es decir, 528 bytes para un circuito de memoria de 4 MB, y 1056 bytes para un circuito de memoria de 8 MB.

3.5 Ejemplo de una aplicación real.

Una aplicación que se le puede dar al módulo de comunicación, es expandir la capacidad de un sistema de medición local para convertirlo en un sistema de monitoreo remoto con acceso multipunto. La figura 3.47 muestra el diagrama a bloques general de un sistema de medición. En el ejemplo se supone un sistema de medición de parámetros eléctricos a partir de la adquisición de las formas de onda de voltaje y corriente de un sistema de potencia trifásico.
En este caso el bloque de la "interfase de comunicación" se encarga de enviar los datos procesados a la interfase gráfica que es donde se despliegan dichos datos en un formato práctico y comprensible para un usuario. En un sistema de medición local la interfase de comunicación generalmente acopla la etapa de procesamiento con un panel único de despliegue.

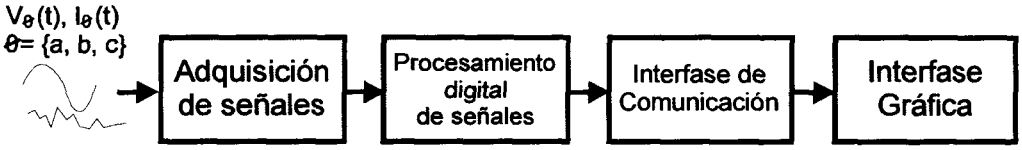


Figura 3.47. Diagrama a bloques de un sistema de medición[*].

El módulo de comunicación propuesto bien puede sustituir la función del bloque "Interfase de Comunicación" para comunicar los resultados del procesamiento hacia cualquier nodo de una red. La "Interfase Gráfica" ocuparía el lugar entonces de una o varias computadoras conectadas a la red, las cuales, pueden simultáneamente monitorear el resultado del "Procesamiento Digital de Señales". Con esto se extiende el alcance de la medición tanto en distancia como en el número de posibles puntos de monitoreo.

[*] Villanueva, Rafael . "Procesamiento Digital con Compensación de Error para el Cálculo de Parámetros Eléctricos en Sistemas de Potencia"

Capítulo 4. Pruebas y Resultados.

Las pruebas realizadas al módulo de comunicación tienen como finalidad evaluar el desempeño general del módulo y caracterizar su funcionamiento.
El módulo abarca gran diversidad de aspectos que pueden ser evaluados; el mismo módulo combina la operación simultánea de sus periféricos, y de la atención a múltiples usuarios, todo esto, inmerso en el desempeño del programa del módulo de comunicación.
Para efectos de prueba del módulo se realizaron varios tipos de pruebas, primero, con la intención de verificar el correcto funcionamiento del programa, y después, para evaluar el desempeño del módulo de comunicación.

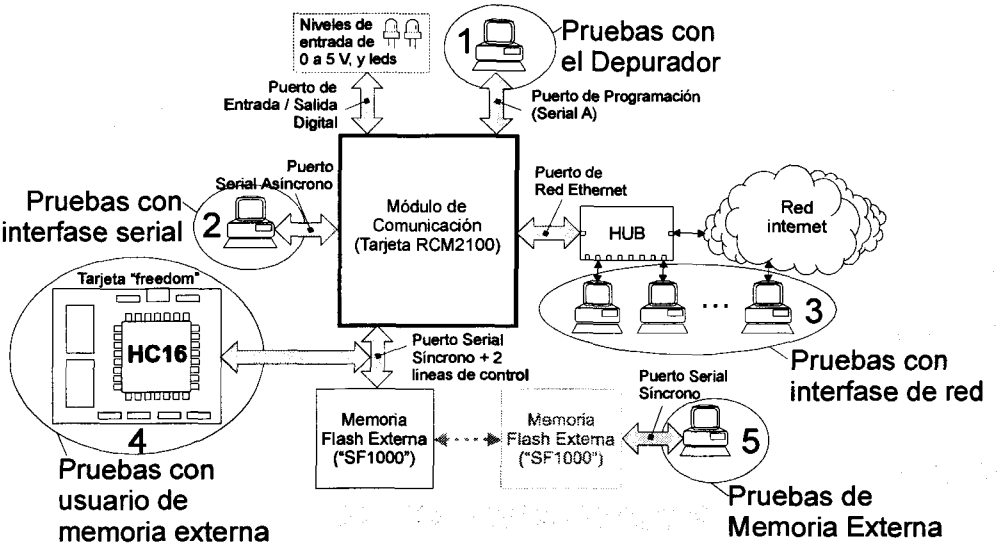


Figura 4.1. Distintas formas de prueba efectuadas al módulo de comunicación.

La figura 4.1 muestra las distintas formas de prueba que se utilizaron para verificar y evaluar la operación del módulo de comunicación. Para la realización de las pruebas se crearon programas que hacen las veces de los usuarios del módulo de comunicación. Los programas se ejecutan en una PC y manipulan, según sea el caso, la tarjeta de red o el puerto serial de la PC de acuerdo a la convención que requiere el módulo de comunicación para establecer contacto con un usuario. Los programas están hechos en el lenguaje de programación gráfica "LabView". El registro de algunos de los tiempos mostrados como resultados de las pruebas se realizaron como parte del código de los programas (llamados, "Vi's" de las siglas en Inglés de "Instrumentos Virtuales") hechos en la PC. A continuación se describe en que consiste, y los aspectos más importantes que se evaluaron en cada una de las pruebas.

4.1. Pruebas con depurador

Las pruebas con el depurador están orientadas a calificar más que cuantificar el funcionamiento del programa. Para estas pruebas fue de gran ayuda la herramienta de programación "Dynamic C" de la tarjeta RCM2100, la cual, consiste en un programa que

se ejecuta en una PC y que permite editar, compilar, cargar, y depurar programas destinados para la tarjeta de desarrollo RCM2100. Dicha herramienta ofrece la posibilidad de enviar mensajes a la pantalla de la PC mientras la tarjeta de desarrollo esta en modo de programación. Esta facilidad se aprovechó para incluir mensajes dentro del código del programa y verificar el desarrollo de algunos puntos en la ejecución del programa. En la versión del programa final, la tarjeta funciona de manera independiente, por lo tanto los mensajes no tienen sentido y se eliminan.
Las pruebas con el depurador fueron de gran ayuda y determinantes durante el desarrollo del programa del módulo de comunicación. Con el depurador se detectaron y corrigieron detalles de programación de bajo y alto nivel, se verificó el correcto funcionamiento de la lógica del programa del módulo de comunicación, que incluye el arbitraje de recursos, la recepción y validación de parámetros, la operación de las funciones para el manejo de memoria y de comunicación, entre otras cosas.

A manera de ejemplo se muestra la pantalla de mensajes de la herramienta de programación y depuración "Dynamic C". Los mensajes son emitidos por el módulo de comunicación al inicio del programa.

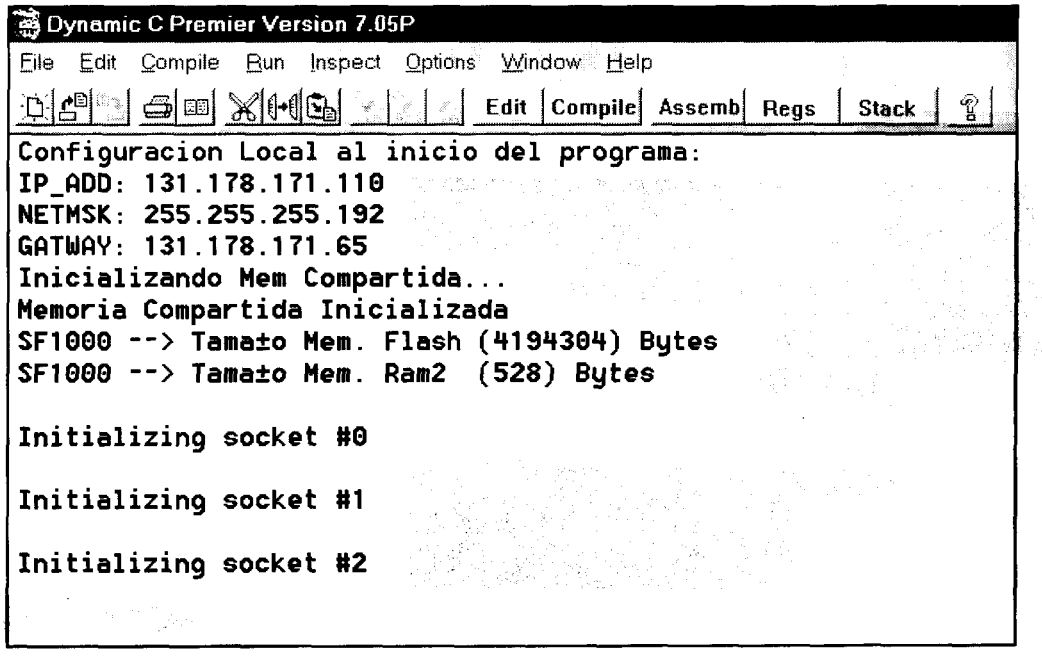


Figura 4.2. Pantalla de mensajes emitidos al inicio del programa del módulo de comunicación.

En la pantalla de la figura 4.2 se muestran algunos mensajes de la "etapa de inicialización", es decir, la etapa donde algunas variables se configuran y adquieren su valor por "default". Los mensajes emitidos en la pantalla muestran la configuración inicial de red (IP, Netmask, y Gateway). Para efectos de prueba se utilizó una configuración de red acorde a la subred del campus (ITESM, Campus Monterrey) en la

que se efectuaron las pruebas, sin embargo, la configuración inicial del programa final es la siguiente:
Dirección IP: 10.10.1.1
Netmask: 255.255.255.0
Gateway: 10.10.1.2
Después de inicializar la configuración de red, el programa ejecuta una rutina para detectar la presencia y la capacidad del circuito de memoria externa ("SF1000"). En este caso, la capacidad del módulo de memoria flash es de 4MB y la de su "buffer2" interno (memoria RAM) es de 528 bytes. Seguidamente el programa entra en un ciclo infinito en espera a la solicitud de un comando. En la misma pantalla de prueba puede apreciarse también que el programa esta compilado para atender a 3 usuarios de red simultáneamente ("sockets" 0, 1, y 2), además de atender al usuario del puerto serial asíncrono.

4.1.1. Atención a un usuario de red.

La figura 4.3 muestra los mensajes emitidos en la pantalla del depurador por el módulo de comunicación al recibir un comando a través del puerto de red ethernet. Por una parte, en la esquina superior derecha de la figura, se muestra el panel frontal del programa ("Vi") que se ejecuta en la PC, y por otra parte, en el fondo de la figura, se muestra la lista de mensajes emitidos por el programa durante la ejecución del comando.
En el panel frontal del "Vi" se recibe (en "ENTRADAS") como parámetro la dirección IP del módulo de comunicación, y a cambio (en "SALIDAS") se obtiene como respuesta la capacidad de los distintos tipos de memoria que maneja el módulo de comunicación. En la lista de mensajes de prueba se aprecia que el comando solicitado ("m") no implica la recepción de parámetros por parte del módulo de comunicación. El módulo, al recibir el comando "m", sabe que el comando no requiere de parámetros, pero también sabe que debe enviar como respuesta la información solicitada, es decir, los datos del tamaño de los distintos tipos de memoria.
También en la solicitud del comando se puede apreciar que la bandera de retroalimentación ("SafeMode") esta apagada; esto le indica al módulo de comunicación que no envíe el resultado (=0) de la ejecución del comando; en la pantalla de depuración se muestra que el resultado ha sido exitoso, esto es, igual a cero; sin embargo, en este caso el resultado no se envía a través de la red.
Es importante diferenciar entre la respuesta de un comando y la respuesta de la ejecución de un comando. El primer caso depende de la naturaleza del comando, y puede o no puede existir, por ejemplo, en un comando de lectura a memoria se espera una respuesta, mientras que en una escritura no se espera respuesta; en el segundo caso, la respuesta de la ejecución de un comando indica si el comando se ejecutó con éxito o si hubo un error durante su ejecución, esta respuesta se habilita o deshabilita mediante el estado de la bandera "SafeMode" del mismo protocolo de comunicación de red del módulo de comunicación.

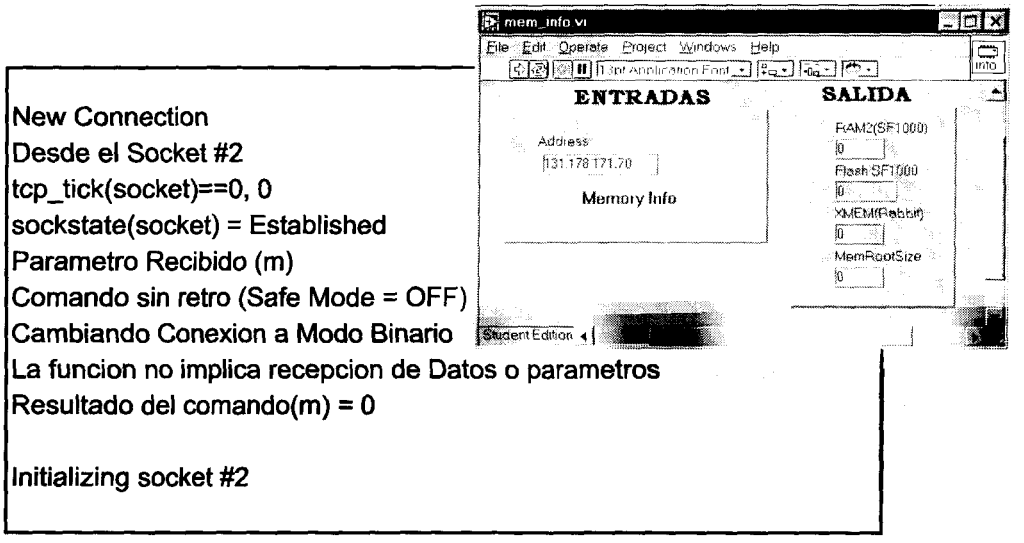


Figura 4.3. Lista de mensajes emitidos por el programa del Módulo de Comunicación al recibir una solicitud del comando "Get_Mem_cfg" (m) para obtener la información de los tamaños de los distintos tipos de memoria.

4.1.2. Acceso desde navegador de red.

Esta prueba consiste en direccionar el módulo de comunicación desde un programa "navegador de red" como "Internet Explorer" o "Netscape". El módulo de comunicación por default envía una página en HTTP ("HyperText Transfer Protocol") que contiene solamente texto. El resultado de la prueba se muestra en la figura 4.4 en donde en el fondo se observa la lista de mensajes emitidos por el módulo de comunicación; arriba a la derecha, aparece lo que observa el usuario del módulo a través del navegador; por último, en la parte de abajo de la figura se muestra el código fuente de la misma página de prueba que se despliega en el navegador.

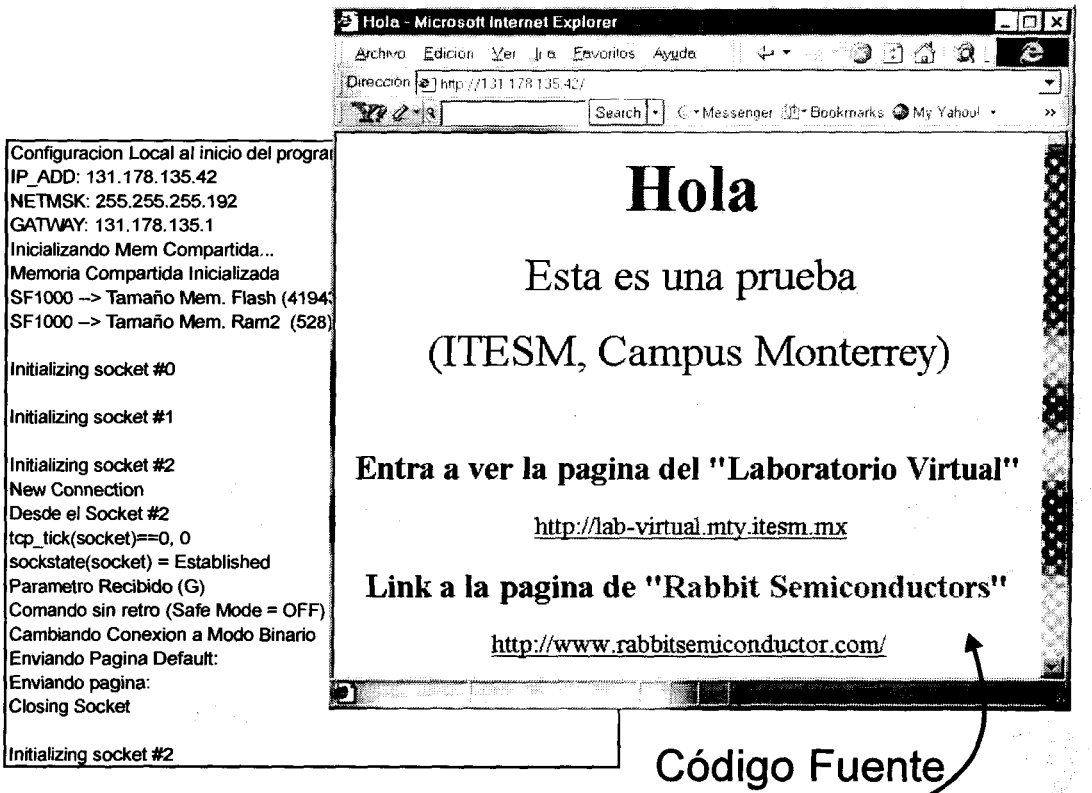


Figura 4.4. Acceso al módulo desde navegador de red.

Es importante recordar que el módulo de comunicación utiliza el puerto #80 del protocolo TCP. La razón de utilizar ese puerto en específico es precisamente tener la facilidad de acceso al módulo a través de un navegador de red; el navegador utiliza por default el puerto #80. Aparte, el navegador es un programa de uso muy común y casi imprescindible en cualquier computadora con facilidad de conexión a una red. El hecho de tener acceso al módulo de comunicación a través de un navegador de red representa una ventaja y un atractivo extra que incrementa las alternativas de uso del mismo módulo de comunicación.

4.1.3. Atención a múltiples usuarios de red.

También como parte de las pruebas hechas con el depurador se muestra la pantalla de salida que resulta del acceso simultáneo de tres usuarios del módulo de comunicación a través de una conexión de red. En la salida de mensajes se aprecia claramente la operación del programa bajo un esquema de multitarea. Esto se muestra en la figura 4.5, a la izquierda de la figura se indica el número de "socket" (o usuario) asociado a cada uno de los mensajes de la pantalla. En la figura puede apreciarse que los mensajes entre los distintos usuarios están entrelazados. Es decir, durante el proceso de la ejecución de principio a fin de un comando de un determinado usuario, existen operaciones entrelazadas que pertenecen a los demás comandos de los distintos usuarios simultáneos. El hecho de alternar la ejecución del programa entre los procesos de los distintos usuarios es lo que produce el efecto de un ambiente de multitarea.

En el ejemplo de la figura 4.5 los tres distintos usuarios solicitan el mismo comando con los mismos parámetros, es decir, una lectura a memoria externa desde la dirección: 0, hasta la dirección: 1'000,000 decimal. Según las reglas de arbitraje, la ejecución de los comandos puede realizarse en paralelo, puesto que se trata de tres lecturas las cuales no modifican el contenido de la memoria, y no afectan la integridad del resultado de las lecturas.

Socket #	Mensaje Emitido por el programa
2	Initializing socket #2
2	New Connection
2	Desde el Socket #2
2	tcp_tick(socket)==0, 0
2	sockstate(socket) = Established
2	Parametro Recibido (A)
2	Comando sin retro (Safe Mode = OFF)
2	Cambiando Conexion a Modo Binario
1	New Connection
2	# de bytes esperados [10]
1	Desde el Socket #1
1	tcp_tick(socket)==0, 0
1	sockstate(socket) = Established
1	Parametro Recibido (A)
1	Comando sin retro (Safe Mode = OFF)
1	Cambiando Conexion a Modo Binario
0	New Connection
1	# de bytes esperados [10]
2	Parametros Recibidos en Validate Correctamente
2	0 0 0 0 A0 86 1 0 0 0
2	Comprobando parametros:
2	Addr: 0, ByteCount 100000
0	Desde el Socket #0
0	tcp_tick(socket)==0, 0
0	sockstate(socket) = Established
0	Parametro Recibido (A)
0	Comando sin retro (Safe Mode = OFF)
0	Cambiando Conexion a Modo Binario
2	Etapa de Arbitraje (1), socket #2
2	ejecutando commando SF1000Read_TCP0(0, 100000, 0)
0	# de bytes esperados [10]
1	Parametros Recibidos en Validate Correctamente
1	0 0 0 0 A0 86 1 0 0 0
1	Comprobando parametros:
1	Addr: 0, ByteCount 100000
1	Etapa de Arbitraje (1), socket #1
1	ejecutando commando SF1000Read_TCP0(0, 100000, 0)
0	Parametros Recibidos en Validate Correctamente
0	0 0 0 0 A0 86 1 0 0 0
0	Comprobando parametros:
0	Addr: 0, ByteCount 100000
0	Etapa de Arbitraje (1), socket #0
0	ejecutando commando SF1000Read_TCP0(0, 100000, 0)
2	Etapa de Post_Arbitraje (2) ejecutada, socket #2
2	Resultado del comando(A) = 0
1	Etapa de Post_Arbitraje (2) ejecutada, socket #1
1	Resultado del comando(A) = 0
2	Initializing socket #2
0	Etapa de Post_Arbitraje (2) ejecutada, socket #0
0	Resultado del comando(A) = 0
1	Initializing socket #1
0	Initializing socket #0

Figura 4.5. Lista de mensajes emitidos por el programa del módulo de comunicación, durante el acceso simultáneo de 3 usuarios distintos.

4.1.4. Arbitraje con recurso en conflicto.

Esta prueba consiste en efectuar dos operaciones distintas sobre el mismo recurso, de tal forma que se produzca un conflicto entre las operaciones y se tenga que entrar en una etapa de arbitraje donde la segunda operación entra en cola de espera para su posterior ejecución hasta que la primera operación libere el recurso.

Para esta prueba el recurso que entra en conflicto es el módulo de memoria externa. Por una parte se tiene la primera operación que corresponde a una lectura sobre la memoria externa, desde la dirección inicial: 3'000,000 y con una longitud de 300,000 bytes. Y por otra parte, la segunda operación corresponde a una escritura de 10 bytes a partir de la dirección inicial: 3'000,000. Para que exista un conflicto entre las dos operaciones la segunda operación debe presentarse mientras la primera todavía este en curso.

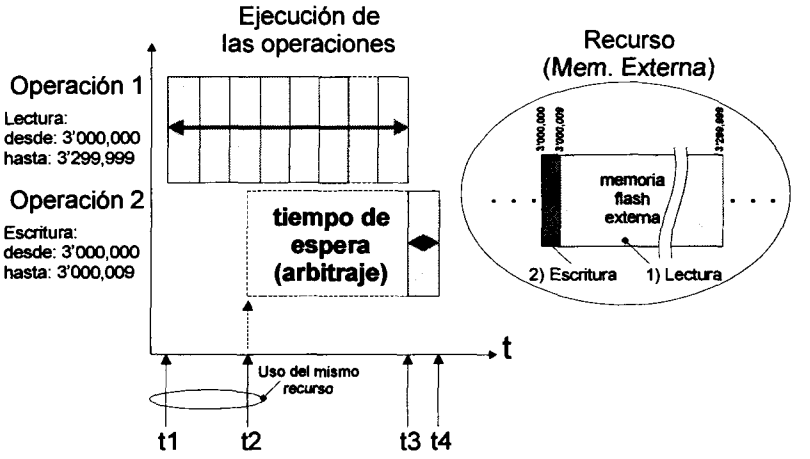


Figura 4.6. Prueba con dos operaciones simultáneas que se empalman con riesgo de conflicto.

En este caso la segunda operación de escritura se empalma sobre el rango de direcciones de la primera operación de lectura; esto causa que en la etapa de arbitraje la segunda operación entre en una cola de espera hasta que finaliza la primera operación de lectura.

La figura 4.7 muestra el listado de mensajes producto de las dos operaciones mencionadas. En la figura es posible apreciar en forma sombreada el punto donde el programa detecta el conflicto entre las dos operaciones, también se resalta que la segunda operación de escritura no empieza sino hasta después que la primera operación finaliza (en t3).

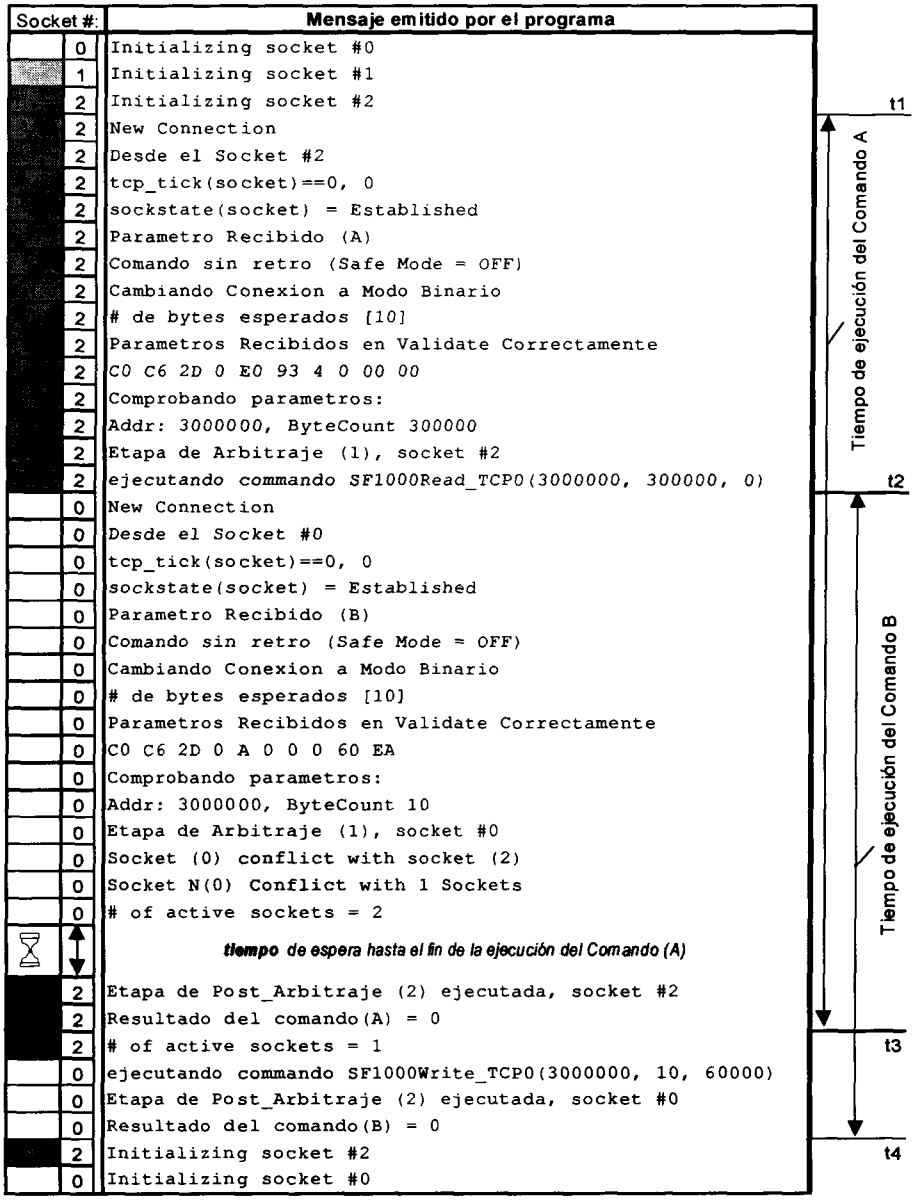


Figura 4.7. Lista de mensajes emitidos por el programa del módulo de comunicación, durante el acceso simultáneo de 2 usuarios distintos, los cuales entran en conflicto al hacer uso del mismo recurso.

4.2. Pruebas con interfase serial

Para las pruebas con la interfase serial se construyeron programas en LabView; los programas se ejecutan en una PC y están diseñados para comunicarse con el módulo de comunicación a través de un puerto serial asíncrono. Los programas manejan el protocolo serial del módulo de comunicación y simulan la operación de un dispositivo serial de acceso al módulo, como puede ser un microcontrolador o un DSP.

El propósito de estas pruebas es evaluar la velocidad y caracterizar el funcionamiento del módulo a través del puerto de comunicación serial. El núcleo de los programas de prueba

serial se aprovechó para crear una librería de funciones que queda como una facilidad extra para comunicar una PC con el módulo de comunicación a través de una conexión serial asíncrona.

Para realizar las pruebas de desempeño del módulo de comunicación se modificó el programa principal. En el programa principal se incluyeron instrucciones para cambiar el estado de salida de un puerto digital; dichas instrucciones se colocaron estratégicamente en el código del programa para medir los intervalos de tiempo que corresponden a los siguientes aspectos:

- El tiempo de procesamiento de un comando serial.
- El tiempo de transferencia de datos de la comunicación serial.

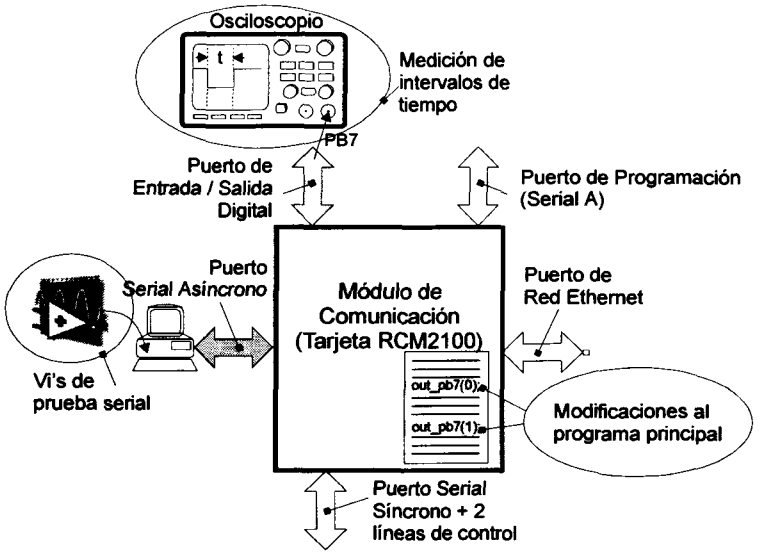


Figura 4.8. Configuración para la prueba con interfase serial.

Posteriormente a la realización de las pruebas las instrucciones que modifican el estado de salida del puerto digital se eliminaron para restaurar la operación normal del programa del módulo de comunicación.

El tiempo de procesamiento de un comando serial se refiere a la parte del programa que realiza la identificación y validación de un comando, además de la validación de los parámetros del comando. Este tiempo de procesamiento fluctúa entre 12.8 a 34 mS dependiendo del comando solicitado.

El tiempo de transferencia de datos de la comunicación serial se refiere al tiempo que tarda el módulo en recibir un dato por el puerto serial y transferirlo a la memoria del módulo, o viceversa, es decir, el tiempo que tarda el módulo en transferir un dato desde la memoria hasta la salida del puerto serial del módulo en forma de un flujo bits. El tiempo de transferencia de datos esta en función de dos factores que son:

- La velocidad de la comunicación serial en bits por segundo (bps).
- El tipo de memoria (raíz o extendida) involucrada en la transferencia de datos.

Cabe mencionar que para la realización de las pruebas se utilizó una configuración serial de 8 bits de datos, sin paridad, y con un bit de paro.

La tabla 4.1 muestra los resultados de las pruebas en función de la velocidad de la comunicación serial y del tipo de memoria (raíz o extendida) involucrada en el comando. Las velocidades utilizadas para la obtención de la tabla son las velocidades más comunes de uso estándar que aparecen en la mayoría de los programas de comunicación serial para PC. En la figura 4.9 se muestra en forma gráfica la misma información de la tabla.

Vel. De Transferencia (bytes/segundo)								
	Mem. Raíz	241	482	966	1,923	3,828	5,769	6,502
	Mem. Extendida	241	482	966	1,923	1,905	1,905	1,914
		2,400	4,800	9,600	19,200	38,400	57,600	115,200
		Velocidad serial (bps)						

Tabla 4.1. Velocidad de transferencia serial en bytes/segundo para los distintos tipos de memoria interna del módulo de comunicación.

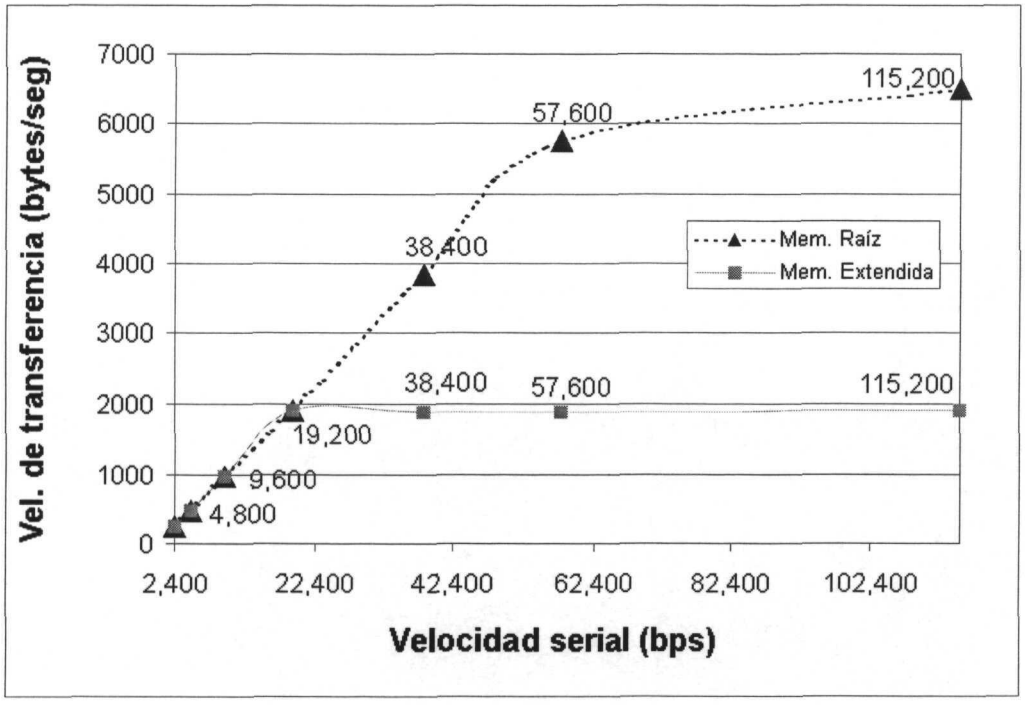


Figura 4.9. Gráfica de velocidad de transferencia serial para los distintos tipos de memoria interna del módulo de comunicación.

Es importante recalcar que el término "transferencia", utilizado en esta sección de "pruebas con interfase serial", se refiere tanto a comandos de lectura como de escritura.

Los resultados de las pruebas que se muestran en la tabla 4.1 y la figura 4.9 se aplican entonces tanto para los comandos de lectura serial, como de escritura serial.
En cuanto al comportamiento de la transferencia serial de datos, idealmente se espera que la transferencia sea linealmente proporcional a la velocidad de la comunicación serial en bps. En la gráfica puede observarse el comportamiento lineal se conserva hasta el punto en el que la velocidad es de 19,200 bps para los comandos que involucran transferencias a memoria extendida, y aproximadamente 57,600 bps para los comandos que involucran transferencias a memoria raíz.
El comportamiento de la gráfica deja de ser lineal debido al tiempo que requiere el procesador para transferir los datos del buffer serial a memoria. Cuando dicho tiempo de procesamiento es mayor que el tiempo de la comunicación serial, el comportamiento de la gráfica deja de ser lineal, y la transferencia de datos del buffer a memoria (o viceversa) se convierte en un "cuello de botella". Por otra parte, el buffer de comunicación serial en el programa del módulo de comunicación tiene una capacidad de 32 bytes; debe cuidarse que el buffer no se desborde durante una transferencia de datos. Por ejemplo, el módulo puede recibir ráfagas de datos a una velocidad serial de 115,200 bps siempre y cuando las ráfagas no sobrepasen 32 bytes de longitud (que es la capacidad del buffer serial) y la tasa de transferencia en bytes/segundo de la comunicación serial no sobrepase los 6.5 KB/s para un comando de memoria raíz y los 1.9 KB/s para un comando de memoria extendida, que son los datos que se muestran en la gráfica. Si estas condiciones no se cumplen la comunicación serial con el módulo no es efectiva.
En la gráfica puede observarse también que la velocidad de transferencia para los comandos que involucran a la memoria extendida es menor que la de los comandos de memoria raíz. Esto se debe a que en el proceso de una transferencia a memoria extendida, existe un paso intermedio que transporta los datos de memoria raíz a memoria extendida. Es por ello que las operaciones de memoria extendida consumen más tiempo o son más lentas.
A partir de los resultados obtenidos de las pruebas es posible efectuar el cálculo de la duración de cualquier comando serial del módulo de comunicación. El tiempo de ejecución esta en función de la velocidad de la comunicación serial y del tipo de memoria involucrada en el comando.

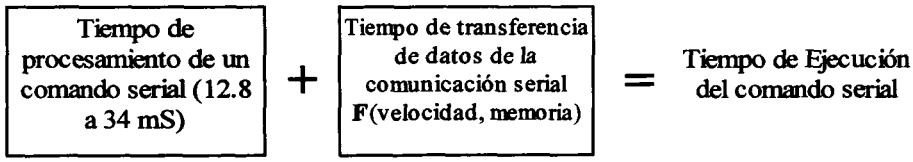


Figura 4.10. Cálculo del tiempo de ejecución de un comando serial.

Por ejemplo, considerando una velocidad de comunicación serial igual a 9600 bps; para determinar el tiempo de ejecución del comando serial: XMEMRead (0, 8191) que realiza la lectura de un bloque de 8192 bytes de la memoria extendida se requiere de un tiempo igual a:

34mS + (20 bytes) / 966 bytes/seg + (8192 bytes) / 966 bytes/seg = 8.53 segundos

Los 34 mS se obtienen del tiempo máximo registrado para el procesamiento de un comando. Los 20 bytes corresponden a los datos del protocolo serial desglosados de la siguiente forma: 3 del identificador "cmd" + 1 del identificador del comando ('H') + 1 de la bandera "safemode" + 8 de los parámetros (long, long) + 2 de checksum + 3 del identificador de respuesta de un comando "rsp" + 2 de checksum de respuesta.
Todos los datos pertenecientes al protocolo serial se procesan en memoria raíz por eso en el cálculo del tiempo total se calculan por separado y en base a la velocidad de transferencia en memoria raíz. Por último, se calcula y se agrega el tiempo de la transferencia del bloque de datos en memoria extendida. En este caso la velocidad de transferencia de memoria raíz y extendida es igual, es decir, 966 bytes/ segundo. El tiempo total que requiere la ejecución del comando es de 8.53 segundos.
En el ejemplo del cálculo de la ejecución del comando no se considera el tiempo de espera que puede originarse debido al conflicto entre distintas operaciones simultáneas sobre la memoria extendida; sin embargo, es importante tener presente que puede presentarse el caso en el que el tiempo de espera de arbitraje se añada al tiempo de ejecución de un comando.

4.3. Pruebas con interfase de red

Las pruebas con interfase de red son similares a las pruebas de con interfase serial; también consisten en una serie de programas que se ejecutan en una PC; los programas están hechos en el lenguaje de programación gráfico "LabView". El propósito de los programas es evaluar la velocidad y el funcionamiento del módulo a través del puerto de red ethernet, de tal forma que el usuario tenga idea de los tiempos de respuesta del módulo de comunicación y pueda considerarlos en las aplicaciones para las que destine el uso del módulo. Los programas de prueba, salvo pequeñas modificaciones posteriores, se utilizan como librerías para el manejo del módulo de comunicación con una PC a través de una comunicación por red.

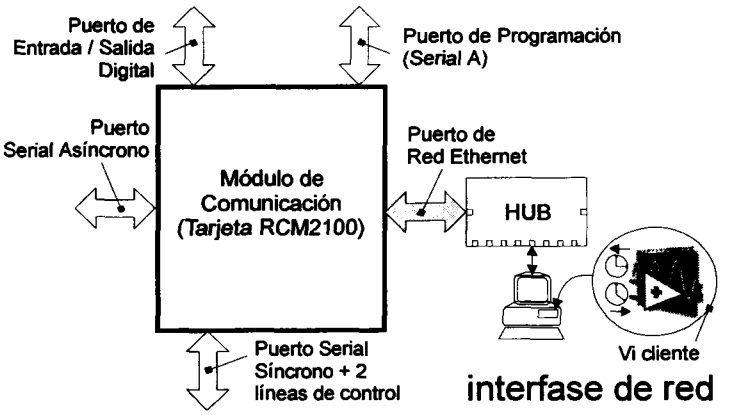


Figura 4.11. Configuración para la prueba con interfase red.

El conjunto de programas de prueba funciona bajo el esquema de operación de un cliente del módulo de comunicación. La figura 4.12 muestra las etapas de ejecución de un programa cliente y la relación que guarda con las etapas de ejecución del programa modelo de un servidor.

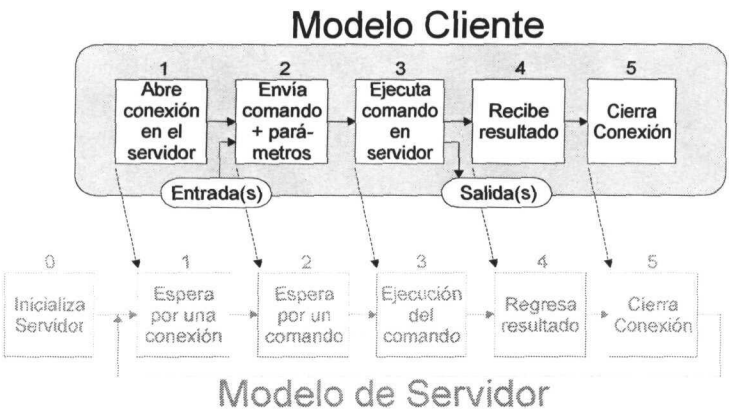


Figura 4.12. Modelo de un programa cliente y su relación con el modelo de un programa servidor.

En términos generales, para medir la velocidad de transferencia de datos del módulo de comunicación, se midió el tiempo de ejecución de una determinada tarea y se dividió entre el número de bytes transferidos en el transcurso de la tarea. De esta forma se obtuvo la velocidad dada por la razón (bytes/segundo) de las distintas operaciones del módulo de comunicación.

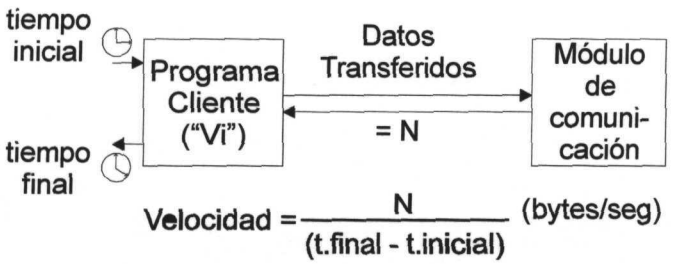


Figura 4.13. Prueba de velocidad de transferencia de datos del módulo de comunicación.

Los tiempos de ejecución (tiempo final – tiempo inicial) se miden internamente, dentro y como parte del programa ("Vi") de prueba. Posteriormente la parte que hace la medición del tiempo de ejecución se elimina del programa, quedando solamente como un simple programa cliente.

4.3.1. Pruebas de velocidad.

Para estas pruebas se conectó el módulo de comunicación y la PC a una misma red ethernet a través de un concentrador ("hub"). Se evaluaron los siguientes puntos:

- Velocidad de respuesta del servidor
- Velocidad de transferencia de datos.

En la prueba de velocidad de respuesta del servidor ethernet se observó inicialmente que en el establecimiento de una conexión del servidor con un usuario externo se consume un tiempo a simple vista apreciable. Este tiempo se refiere a la etapa #1 del modelo cliente-servidor de la figura 4.12 donde el servidor espera una conexión. Durante este tiempo ambas partes (la PC y el módulo de comunicación) entran en un proceso de acuerdo para establecer una comunicación. El tiempo de conexión registrado fluctuó en el rango de 0.5 a 0.8 segundos. La magnitud de este tiempo y las variaciones que presenta hace que los tiempos de ejecución de las otras etapas del funcionamiento del servidor de datos sea despreciable, por ejemplo, la etapa de "validación de parámetros", cuya ejecución toma unas fracciones de milisegundos es totalmente despreciable en comparación con el tiempo de conexión del servidor.
Para efectos prácticos el tiempo total de respuesta del servidor se representa entonces como la suma del tiempo de conexión más la velocidad de transferencia de datos a memoria correspondiente a un comando.

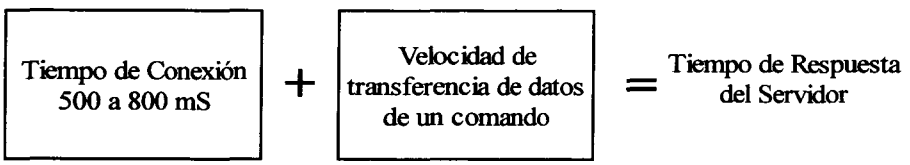


Figura 4.14. Cálculo del tiempo de respuesta para un comando del servidor.

Los comandos de red pueden estar dirigidos los tres tipos de memoria con las que cuenta el módulo de comunicación: memoria raíz, memoria extendida, o memoria externa. Todos los comandos que explícitamente no estén dirigidos a memoria extendida o memoria externa, están dirigidos a memoria raíz.
Se efectuaron pruebas de velocidad de transferencia para los comandos de lectura y escritura a cada tipo de memoria y los resultados se muestran en la tabla 4.2.

Tipo de Memoria	Operación	Velocidad de Transferencia (Kbytes/seg)
Raíz	Lectura	100
	Escritura	100
Extendida	Lectura	50
	Escritura	50
Externa	Lectura	21.7
	Escritura	14

Tabla 4.2. Velocidad de transferencia de datos de los comandos de red del módulo de comunicación según el tipo de memoria implicada en el comando.

Con los resultados de la tabla puede determinarse el tiempo de ejecución de prácticamente cualquier comando. Por ejemplo, la lectura completa de la memoria raíz

cuyo tamaño es de 2KB, tomaría un tiempo de: 0.8 + (2048 / 100,000) = .82 segundos, prácticamente el tiempo de conexión del servidor. La lectura completa de la memoria externa de 4MB tomaría un tiempo de: 0.8 + (4'194,304 / 21,730) = 193 segundos, es decir 3 minutos con 13 segundos.

Un punto importante es la comparación de velocidad de transferencia entre los comandos de comunicación por red y los comandos de comunicación serial, tanto para memoria raíz como para memora extendida. La velocidad máxima de transferencia para la comunicación serial es considerablemente menor que la velocidad de transferencia que se logra a través de una comunicación por red. Incluso, la velocidad de transferencia de comunicación serial cuenta con un "cuello de botella", el cual, se debe a la forma en que se transfieren los datos internamente a memoria; mientras que la comunicación serial realiza transferencias a memoria byte por byte, la comunicación por red lo hace mediante bloques de 256 bytes, haciendo más ágil la transferencia datos a memoria.

Los comandos se evaluaron primeramente dedicando la atención del módulo a un solo usuario, en estas pruebas se registraron los tiempos de respuesta del módulo y se calculó la tasa de transferencia de datos de la comunicación de módulo con el exterior. Después, para algunos comandos se realizaron pruebas con varios usuarios conectados simultáneamente. Esto como una forma de evaluar el desempeño del módulo en condiciones de atención a múltiples usuarios. Cuando el módulo realiza 2 o más operaciones simultáneas, el tiempo que el procesador le dedica a cada tarea sigue siendo el mismo, la diferencia es que el procesador divide su atención a cada una de las tareas. Por ejemplo, si dos usuarios simultáneamente solicitan realizar la lectura completa de la memoria flash, el tiempo total para finalizar ambas lecturas es de (2 x 193 =) 386 segundos, es decir, el doble de tiempo de lo que tarda en realizar la lectura completa de la memoria un solo usuario. Lo que sucede en esta caso, es que el tiempo de ejecución para cada usuario se distribuye en forma compartida, pero finalmente, cada operación de lectura consume un tiempo de procesamiento de 193 segundos.

4.3.2. Pruebas de operación de algunos comandos.

Estas pruebas consisten en ejecutar los programas ("Vi's") para la manipulación del módulo de comunicación a través de la red y desde una PC.
A continuación se presentan varias figuras donde se muestra el panel frontal de algunos de los programas que se utilizan para efectuar las pruebas de operación del módulo de comunicación. Algunos de los programas están diseñados para la configuración específica de un aspecto del módulo, sin embargo, se diseñó un programa principal en el que se incluyen (en un menú) la mayoría de los comandos disponibles para la manipulación del módulo a través de la red. Similarmente, existe otro programa para la manipulación del módulo a través del puerto serial asíncrono en el que se incluyen en un menú la mayoría de los comandos disponibles en forma serial.

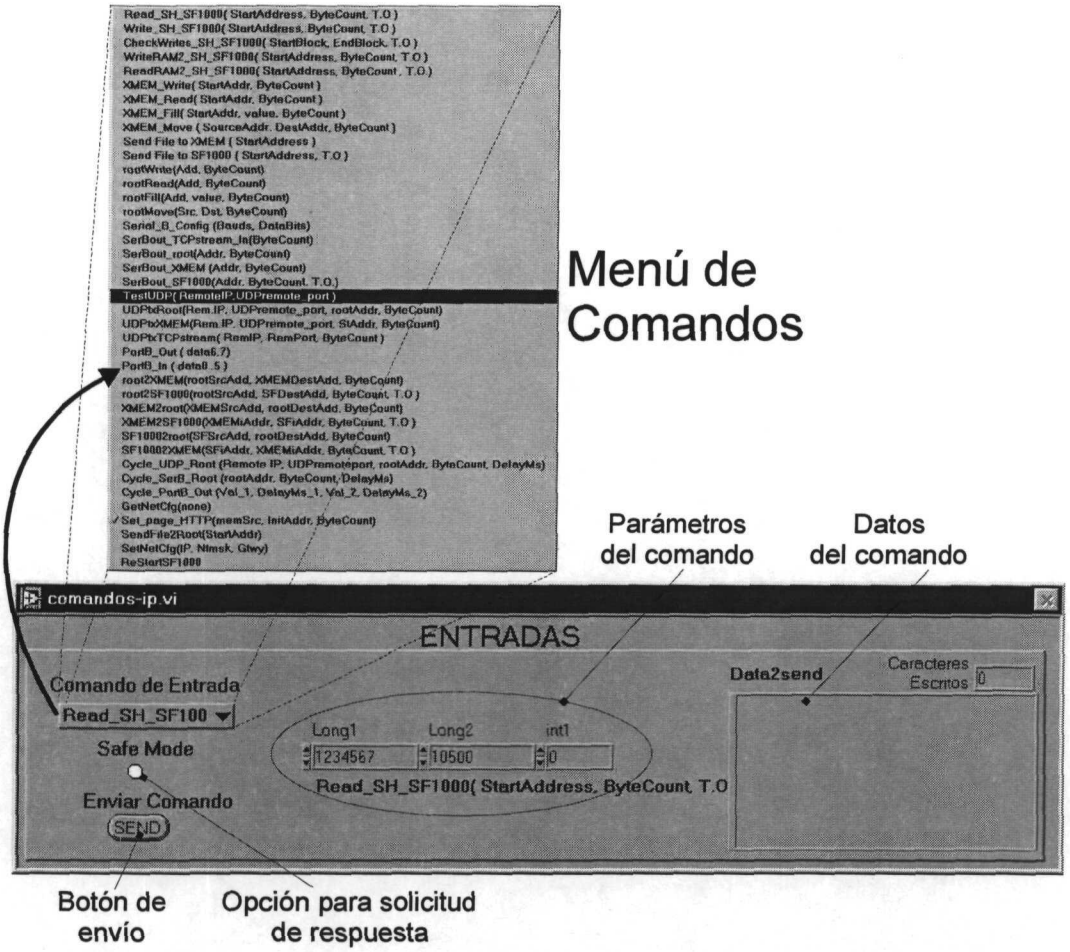


Figura 4.15. Panel frontal de la interfase de entrada de comandos de red.

4.3.3. Envío de una página HTTP.

El módulo de comunicación puede programarse para emitir el contenido de una sección de memoria, en vez de la página de prueba asignada por default. El contenido de la memoria puede estar localizado ya sea en la memoria raíz, la memoria extendida, o memoria externa del módulo de comunicación. En la siguiente figura se muestra el ejemplo de la configuración del módulo para que envíe un bloque de datos contenido en memoria extendida; el bloque de datos tiene una longitud de 405 bytes y está contenido a partir de la dirección 250. En la parte de arriba de la figura se muestra el panel del programa que obtiene la configuración del módulo de comunicación en el que a su vez se muestra la sección de memoria que transmite el módulo al detectar la conexión de un navegador de red. Previamente la configuración se define mediante el comando "Set_Page_HTTP". En la parte central de la figura se muestra el resultado de la conexión del navegador de acuerdo al contenido de la memoria (XMEM: 250 a 654 decimal); en la memoria extendida, anteriormente se carga el contenido del archivo de la página HTTP; esto, mediante el comando "XMEMWrite". El contenido del archivo ("código fuente") de la página es el que se muestra sombreado en la parte inferior de la figura.

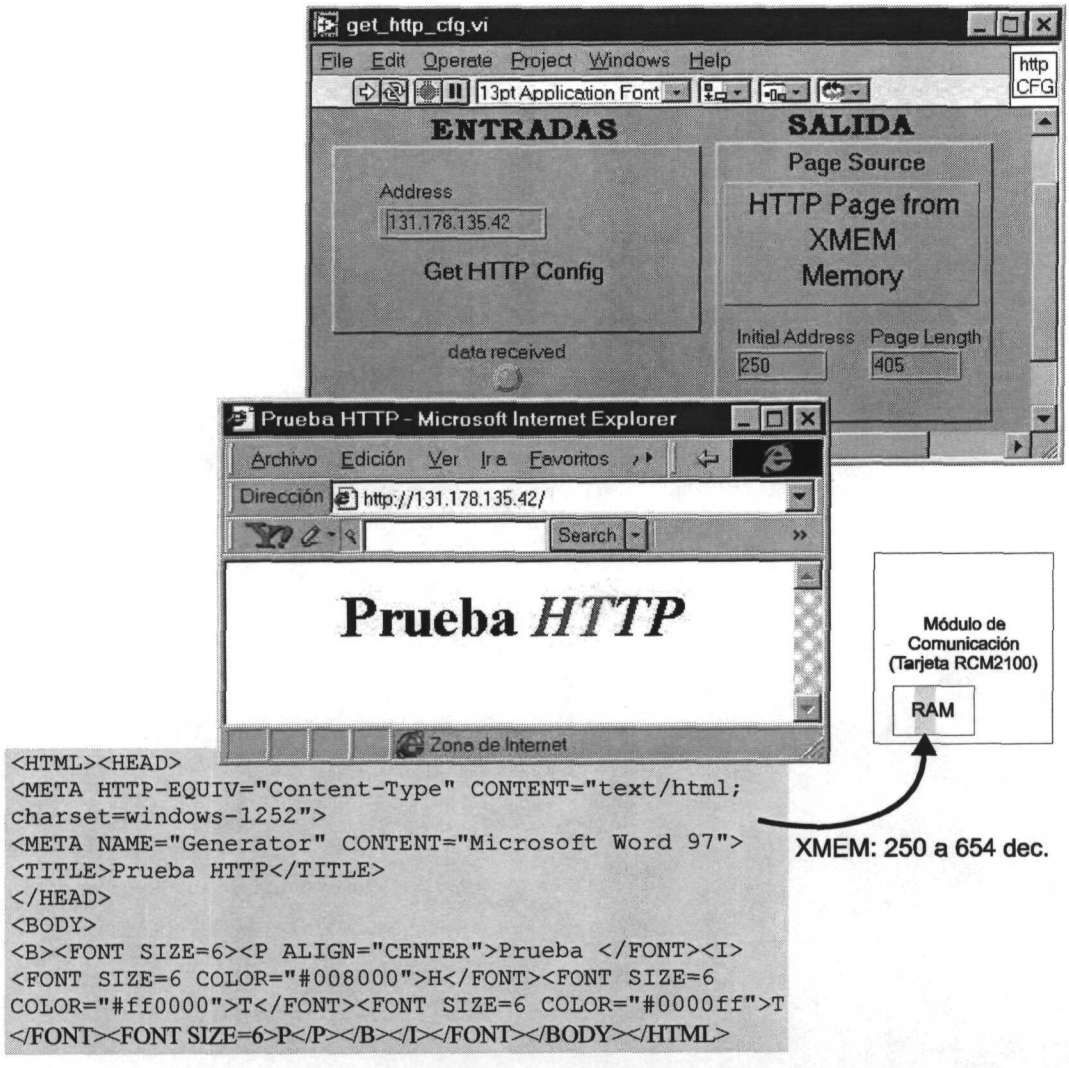


Figura 4.16. Transmisión de una sección específica de memoria ante la conexión de un navegador.

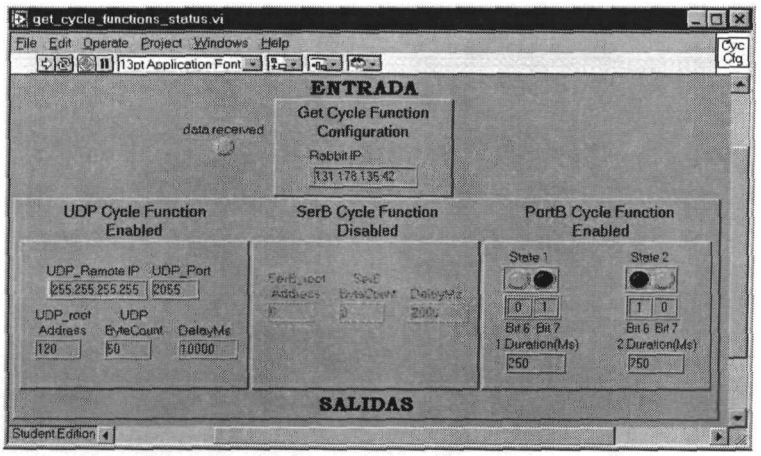


Figura 4.17. Panel frontal del programa que obtiene la configuración de las funciones cíclicas.

4.3.4. Obtención del estado de las funciones cíclicas.

En la figura 4.17 se muestra el panel frontal del programa que obtiene la configuración de las funciones cíclicas del módulo de comunicación. En la parte de arriba de la figura se muestra el campo de entrada donde se especifica la dirección IP destino del módulo de comunicación. El led indicador "data received" parpadea después de ejecutar el programa y una vez que el programa establece contacto con el módulo de comunicación. En la parte inferior del panel se muestra el estado de la configuración actual de las tres operaciones cíclicas. La operación "UDP_cycle" esta habilitada para que envíe una sección (dir: 120 a 169) de la memoria raíz a través del puerto #2055 del protocolo UDP. El envío del contenido de la memoria se realiza en forma periódica e ininterrumpida cada 10 segundos (10,000 mS). La dirección destino del envío de datos (255.255.255.255) corresponde a la dirección especial "broadcast" que esta dirigida a todos los anfitriones de la red local donde se encuentra instalado el módulo. La operación "SerB_cyle" funciona de manera similar a la operación "UDP_cycle", sólo que la salida de los datos de una sección de memoria esta dirigida al puerto serial B de salida. En este caso la operación "SerB_cyle" esta apagada. Por último, la operación "PortB_cycle" esta configurada para que alterne entre dos estados opuestos para los bits 6 y 7 del puerto digital B de salida. El primer estado (0,1) tiene una duración de 250 mS, y el segundo estado (1,0) una duración de 750 mS. El resultado de la operación de las dos funciones cíclicas activas se muestran a continuación.

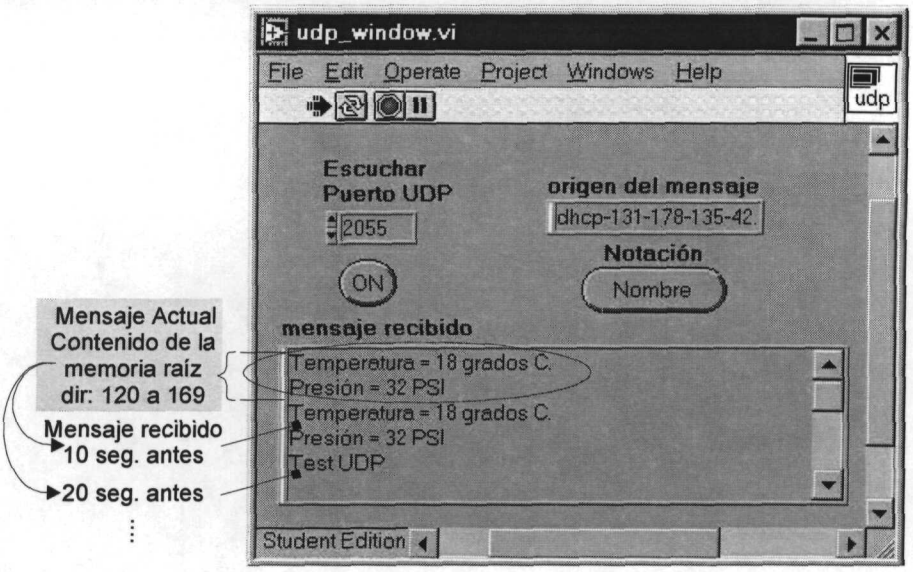


Figura 4.18. Panel frontal del programa que atiende un puerto UDP para la recepción de mensajes y/o datos.

4.3.5. Lectura de mensajes UDP.

La figura 4.18 muestra el panel frontal de un programa de aplicación que atiende un determinado puerto UDP para recibir datos y desplegarlos en forma de texto en el campo "mensaje recibido". En la figura pueden apreciarse los últimos tres mensajes recibidos producto de la operación cíclica que se muestra activada en la figura 4.17. El contenido de los datos en memoria que se envían, incluso puede cambiar si un usuario así lo solicita. Esto puede apreciarse en el mensaje recibido 20 segundos antes del actual, cuyo contenido: "Test UDP" difiere del contenido de los siguientes mensajes que indican valores de temperatura y presión a manera de ejemplo.

4.3.6. Operación cíclica del Puerto digital B de salida.

La figura 4.19 presenta el comportamiento de la función cíclica "PortB_cycle" que se muestra activada en la figura 4.17. En la figura aparecen las señales reales de los bits 6 y 7 del puerto B del módulo de comunicación, las cuales se derivan de la activación de la operación cíclica del puerto B

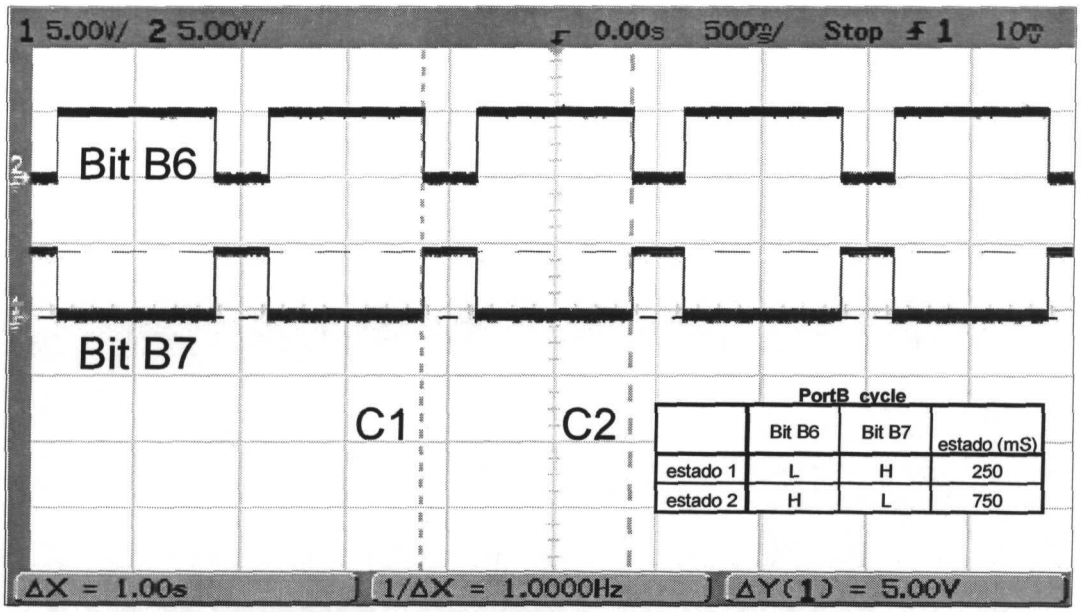


Figura 4.19. Señales de salida de los bits 6 y 7 del puerto B del módulo de comunicación.

Es importante mencionar que la duración de los tiempos de estado de la función cíclica del puerto B no es del todo exacta. Esto se debe a que los cambios de estado del puerto B se efectúan en base a una comparación "mayor que" (>) a partir del tiempo del cambio de estado anterior; y en donde el reloj base de tiempo tiene un período de 488 μS. Por ejemplo, si se solicita un cambio de estado cada milisegundo (utilizando la misma base de tiempo de 488 μS), el cambio de estado se efectúa cada vez que la cuenta del reloj base de tiempo es mayor que la cuenta equivalente al siguiente múltiplo de 1 mS con respecto al cambio de estado anterior. Este comportamiento se aprecia en la siguiente figura, en la que se supone debe efectuarse un cambio de estado cada milisegundo. Como puede apreciarse en la figura, los cambios de estado realmente no suceden cada milisegundo, sino cada vez que la cuenta del reloj de referencia (de 488 μS) avanza en al menos 1mS sobre la cuenta relativa al cambio de estado anterior. Esta situación se aprecia cuando la duración de los tiempos de estado se aproxima a la duración del reloj de referencia o base de tiempo.

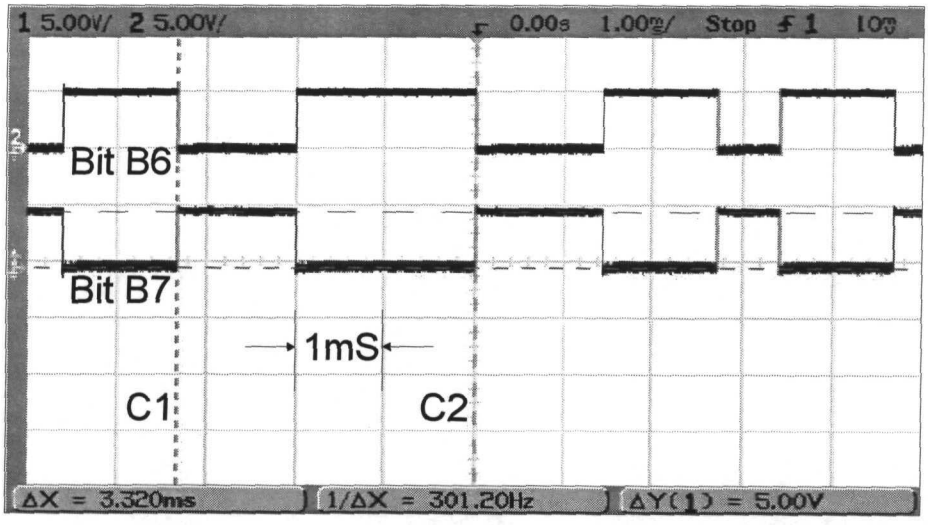


Figura 4.20. Señales donde se aprecia el error de precisión sobre los cambios de estado de la operación cíclica "PortB_cycle".

4.3.7. Función cíclica del puerto serial B de salida.

Por último se habilitó la función cíclica "SerB_cycle" para que envíe una sección de memoria raíz a través del puerto serial. Según la configuración de la función cíclica, el módulo envía 50 bytes a partir de la dirección 120 decimal de la memoria raíz y con un tiempo de espera de 2 segundos (2000mS) entre el final de un envío de 50 bytes y el inicio del próximo envío.

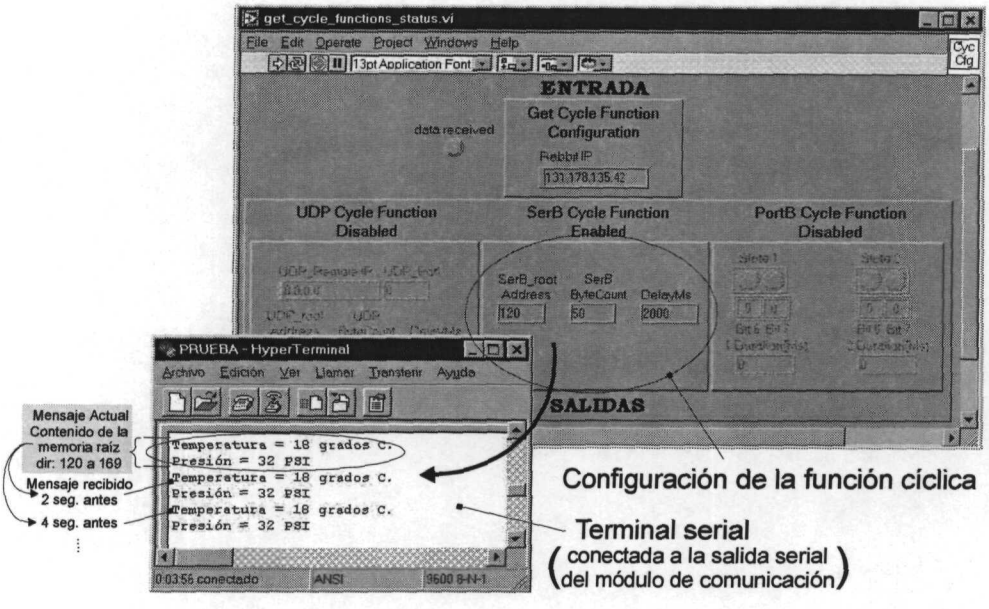


Figura 4.21. Ejemplo del resultado de la operación cíclica "SerB_cycle".

El módulo se conectó al puerto serial de una PC para recibir los datos de la comunicación serial. En la parte inferior de la figura 4.21 se muestra la pantalla del programa "terminal serial" en el que se despliegan los datos recibidos a través de la comunicación serial. Dichos datos, son los mismos que se utilizaron en el ejemplo de la "Lectura de mensajes UDP".

4.3.8. Pruebas Adicionales.

Además de los ejemplos hasta ahora expuestos en esta sección se realizaron pruebas para cada uno de los comandos disponibles en el módulo de comunicación. Esto, para cerciorarse de la correcta operación, tanto del módulo de comunicación, como de los programas ("Vi's") para la manipulación del módulo. Entre las pruebas realizadas están los comandos para realizar transferencias internas de datos entre los distintos tipos de memoria, también los comandos para la configuración y obtención de la fecha del módulo de comunicación, la configuración del puerto serial asíncrono, la configuración de red (IP, netmask, gateway) del módulo, entre otras pruebas. Todas las pruebas se efectuaron con éxito.

Otro tipo de prueba efectuada al módulo fue interrumpir una conexión en proceso. Esto con la intención de verificar la robustez del programa contra posibles fallas en el proceso de una conexión. Para ello se produjeron errores intencionales, primeramente desconectando el cable de red del módulo durante una operación en curso, y después interrumpiendo la ejecución de un programa de prueba que esta en comunicación con el módulo. En ambos casos, cuando el módulo detecta la pérdida de la comunicación, automáticamente cierra la conexión y vuelve a un estado de espera por una conexión. También se probó el módulo contra accesos de red que lleven alteraciones en el protocolo de red propuesto para el uso del mismo módulo de comunicación. La intención de esta prueba es asegurarse de que el módulo no deje de funcionar a pesar de que existan fallas o alteraciones en la forma de acceso al módulo. Dichas fallas pueden ocurrir durante el diseño de nuevas aplicaciones o de programas en otros lenguajes que hagan uso de las funciones del módulo. El módulo de comunicación espera recibir una conexión de acuerdo a la convención definida por el protocolo propuesto; de igual forma, en el protocolo propuesto se define la forma en que el módulo recibe y envía datos de acuerdo a cada comando. Ante alteraciones en el protocolo, si la conexión no se pierde o no se cierra explícitamente por el cliente, entonces el módulo detecta un error de "time out" y produce el cierre de la conexión; posteriormente, el módulo regresa al estado de espera por una nueva conexión. Los resultados de estas pruebas fueron exitosas, es decir, no se logró interrumpir la operación del módulo a pesar de las alteraciones efectuadas en el protocolo de red propuesto.

4.4. Pruebas con usuario de Memoria Externa.

Para las pruebas con usuario de memoria externa se construyó un grupo de funciones para el microcontrolador HC16. Las funciones se encargan de manipular el módulo de memoria externo, en forma similar a como lo hace el módulo de comunicación, es decir, las funciones están diseñadas para escribir y leer la memoria externa, respetando la convención, protocolo, y organización interna de la memoria flash. Además, las funciones se encargan de respetar el acceso al BUS serial síncrono y de utilizar la lógica de las líneas de control del BUS, "LIU" y "RIU". La intención de estas pruebas es

precisamente evaluar tanto la lógica de control del BUS serial síncrono, como la efectividad de comunicación del BUS en un régimen compartido, con dos usuarios reales, físicamente conectados al BUS, los cuales hagan uso del mismo módulo externo de memoria ("SF1000").

4.4.1. Lógica de control del BUS.

Antes de realizar las pruebas del BUS con el usuario microcontrolador HC16, se efectuaron pruebas con un generador de funciones, el cual, simula el estado de la señal RIU que proviene de un usuario remoto. Estas pruebas tienen como objetivo probar la lógica de control del BUS serial síncrono del módulo de comunicación antes de conectar un usuario externo al mismo BUS. Si la lógica de control no funciona correctamente y el BUS no se protege contra colisiones de acceso entre los usuarios, entonces se corre el riesgo de dañar los puertos de acceso al BUS de los usuarios. Por tal motivo, es conveniente realizar esta prueba antes de conectar un usuario externo al BUS serial síncrono.

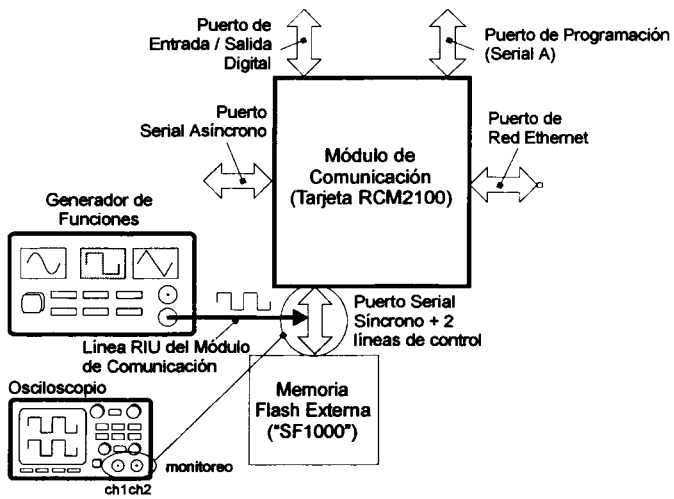


Figura 4.22. Configuración para la prueba de la lógica de control del BUS serial síncrono.

Para efectuar la prueba, en el módulo de comunicación constantemente se efectúa una llamada a una función que haga uso de la memoria externa, de tal forma que el puerto serial síncrono del módulo de comunicación este en constante actividad.

La línea RIU se conecta a la salida de un generador de funciones, de tal forma que simule la intervención de un usuario remoto para el acceso de la memoria externa. Para el módulo de comunicación, la señal del generador de funciones representa un evento asíncrono y aleatorio, el cual, debe ser tratado con precaución para que no exista un conflicto en el BUS serial síncrono, o incluso daño en el hardware de los dispositivos que participan en el BUS.

El objetivo de la lógica del BUS es que por ningún motivo existan dos o más líneas de salida activas a la vez. Es decir, que no estén activas dos o más líneas comunes que realicen una misma función y que provengan de distintos usuarios; como por ejemplo, la salida de transmisión serial de dos usuarios distintos, ya que al estar ambas activas y

conectadas a un mismo punto en común, en este caso la entrada serial de la memoria, pueden dañarse si tienen niveles de voltaje opuestos.
Los resultados de las pruebas se muestran en las siguientes figuras.

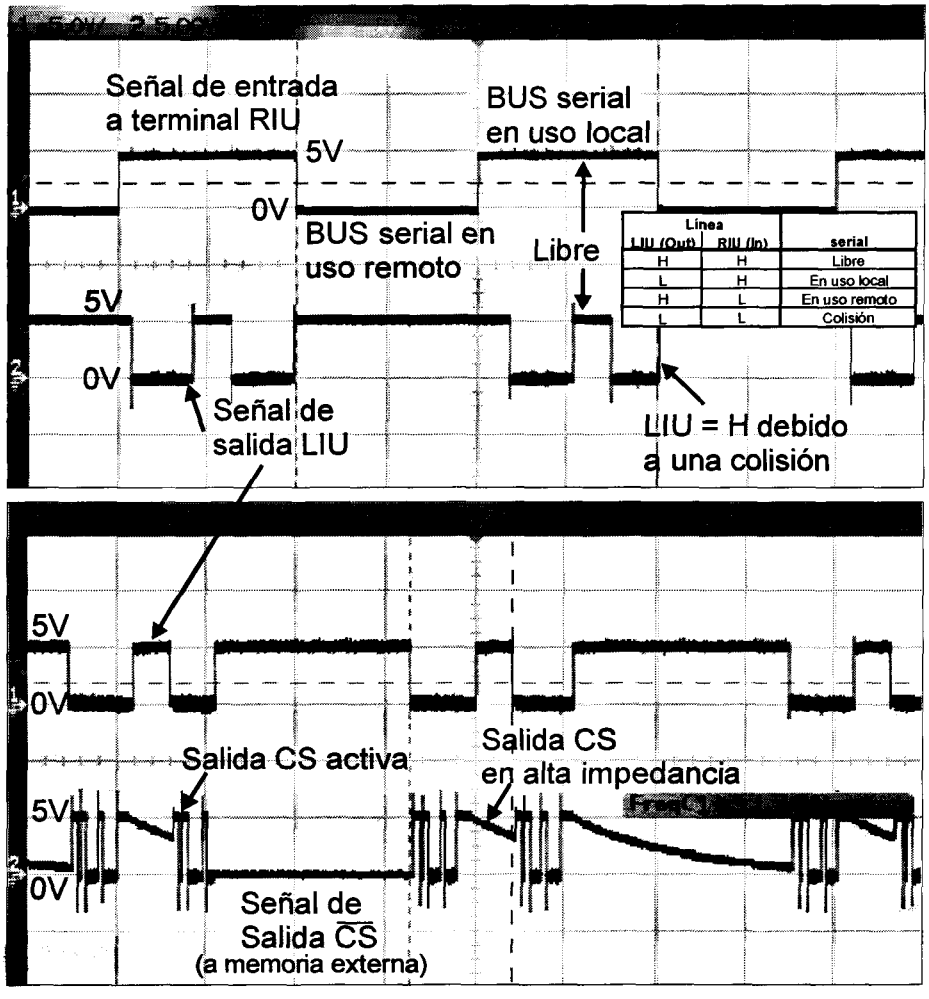


Figura 4.23. Relación de algunas señales en la operación del BUS serial síncrono.

En la parte de arriba de figura 4.23 puede observarse que la señal LIU y RIU nunca están ambas en estado bajo. El módulo de comunicación al hacer uso del BUS serial coloca la señal LIU en nivel bajo. Estando LIU en nivel bajo, si el módulo de comunicación detecta un nivel bajo en la entrada RIU, ocurre una interrupción que automáticamente libera las salidas del BUS serial del módulo de comunicación colocándolas en alta impedancia. También, en la parte de abajo de la figura se observa que cuando el módulo no hace uso del BUS serial (LIU=H), las salidas (en este caso, la salida "CS" –"chip select"-) se encuentran en alta impedancia. Esto, para dar oportunidad a que cualquier otro usuario externo pueda hacer uso del BUS.

La figura 4.24 muestra también que cuando el módulo tiene control del BUS serial, la salida de reloj se encuentra en estado activa; por el contrario, cuando el control del BUS lo tiene un usuario externo, la salida de reloj del módulo se encuentra en estado de alta impedancia. En esta figura también se muestra la naturaleza de la señal de reloj del BUS serial que genera el módulo de comunicación; dicha señal es generada por software, es por ello que puede tener variaciones en frecuencia; sin embargo, puede apreciarse que la frecuencia del reloj es aproximadamente 454KHz.

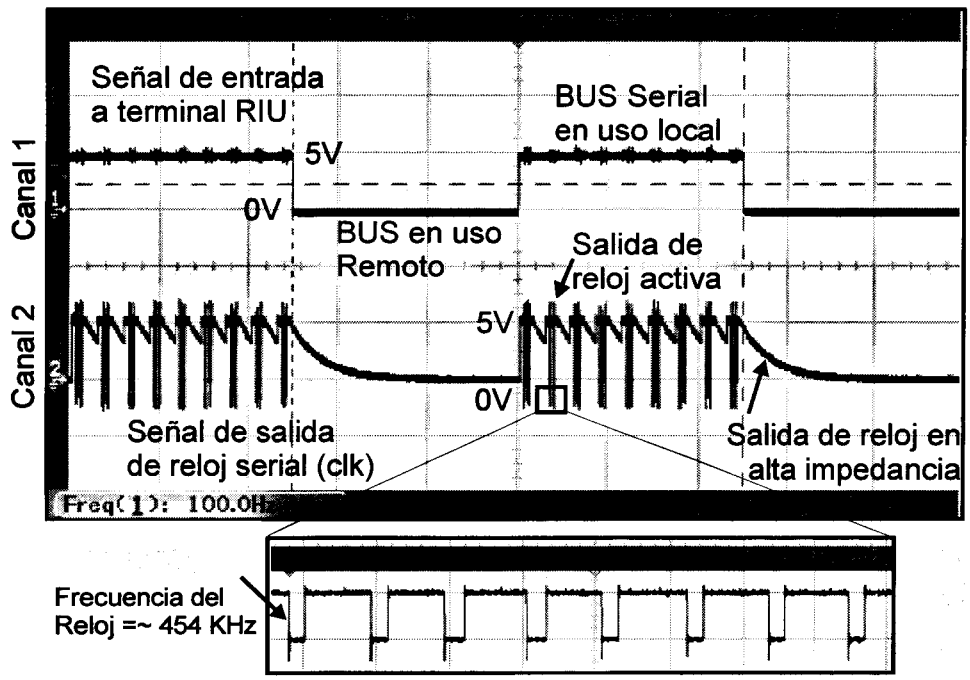


Figura 4.24. Relación entre la señal de entrada RIU y la señal de salida de reloj.

El comportamiento de las tres señales de salida del BUS serial síncrono (CS, clk, serial_out) con respecto a la señal de entrada RIU es similar; es decir, todas las salidas del BUS serial síncrono se liberan en un estado de alta impedancia, ya sea cuando:

- Ocurre una colisión
- El BUS esta en uso remoto
- El BUS esta localmente libre

4.4.2. Librerías de usuario externo.

Una vez comprobado el funcionamiento de la lógica para coordinación del BUS serial síncrono, se probó la operación del BUS con el acceso simultáneo de dos usuarios distintos; uno de los usuarios es el módulo de comunicación, y el otro usuario es una tarjeta de desarrollo que tiene como núcleo un microprocesador HC16 de Motorola.

Para el HC16 se desarrolló una librería que contiene un conjunto de funciones destinadas al manejo compartido de la memoria externa.
Las funciones son las siguientes:

- SH_SF1000INIT. Verifica el tipo y el correcto funcionamiento del circuito de memoria.
- SH_SF1000WRITE. Escribe un bloque de datos en memoria flash.
- SH_SF1000READ. Lee un bloque de datos de la memoria flash.
- SH_SF1000CHECKWRITES. Lee el número de veces que ha sido escrito un bloque.
- SH_SF1000ERASEBLOCK. Borra un determinado bloque.
- SH_WRITEBUFFER2. Escribe un bloque de datos en el Buffer2 (RAM) del circuito de memoria flash.
- SH_READBUFFER2. Lee un bloque de datos del Buffer2 (RAM) del circuito de memoria flash.

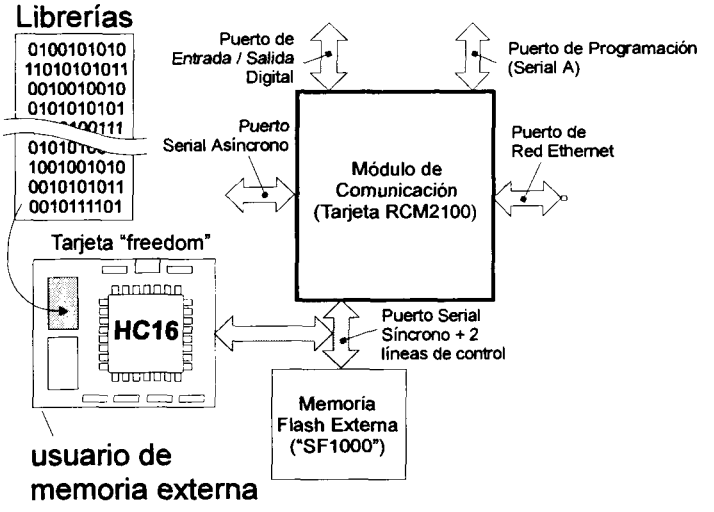


Figura 4.25. Configuración para la prueba de la operación del BUS serial síncrono.

Las funciones están hechas en lenguaje ensamblador, y documentadas en un archivo de texto. A continuación se muestra un ejemplo de la forma en que están documentadas las funciones en donde también se incluye la muestra de la llamada a la función.

```
;*******************************************************
SH_SF1000ERASEBLOCK;      [ Acc. A == TimeOut in mS ]        *
;                         [ E--> Block (Sector) Number ]     *
;(Returns)        Acc. A =[ 0-->Success                       *
;                         , 1--> Error Reading Device Status *
;                         , 2--> Error Ereasing the Block]   *
;                         , 4--> Time Out                    *
;                         , 5 --> Bus Collision]             *
;*******************************************************
; If TimeOut different from zero (#$00) this function will
; Block until is able to get response from Flash or until
; Time Out Expires. Modifies E, D(A:B) Registers
;SYNTAX (Call Example):
;       LDAA #05        ; Delay 5 mS (0 for no Delay)
;       LDE #2134       ; Erase Block # 2134
;       JSR SH_SF1000ERASEBLOCK
;       TSTA
;       BEQ Success     ; jump to success
;       BRA Error       ; Error
;DESCRIPTION:   Erase a single block
;
;PARAMETER1:    Time Out in milliseconds
;PARAMETER2:    block number (Acc. E)
;
;RETURN VALUE: (Acc. A)
        ;0 = success
        ;1 = error reading device status
        ;2 = error erasing the block
        ;4 = Time Out/BUS SPI Busy
        ;5 = Bus Collision
```

Figura 4.26. Ejemplo de la documentación de la función "SH_SF1000ERASEBLOCK" para el HC16.
(En lo sombreado, el ejemplo de la llamada a la función)

Todas las funciones del HC16 para el manejo de la memoria externa se probaron en conjunto con la operación del módulo de comunicación. Los resultados fueron satisfactorios. Por una parte el módulo de comunicación se dejó haciendo accesos continuos a la memoria externa y por otra parte mediante el debugger de la tarjeta del HC16 se cargaron programas de prueba con cada unas de las funciones para el manejo de la memoria. A través del mismo módulo de comunicación fue posible revisar el contenido de la memoria para verificar la acción de los programas de prueba del HC16. En todos los casos, la comunicación con la memoria fue correcta al verificar los datos leídos y/o escritos en memoria.

4.5. Pruebas de memoria externa

Las pruebas de memoria externa consisten en una serie de programas hechos en "LabView" que se ejecutan en una PC. Los programas incluyen las funciones básicas para la lectura y escritura de la memoria flash externa. La PC se comunica con la memoria a través de un puerto digital de una tarjeta de adquisición de datos. El puerto digital se manipula para emular una comunicación serial síncrona y de esa forma se logra comunicar la PC con la memoria.

El propósito de estas pruebas fue familiarizarse con el protocolo y con las rutinas para el manejo de la memoria externa; esto, aprovechando la ventaja de la rapidez de programación que ofrece un lenguaje de alto nivel como "LabView", en comparación con un lenguaje de bajo nivel como el lenguaje ensamblador. Una vez que se conoce el funcionamiento de la memoria, se facilita el transporte de las rutinas a un lenguaje de bajo nivel, tal es el caso de las rutinas del HC16 que se crearon para las pruebas con un usuario externo del módulo externo de memoria.

Los programas de prueba para el manejo de la memoria externa también quedan como un agregado al sistema del módulo de comunicación. Dichos programas forman en conjunto una librería de funciones para el manejo de la memoria externa en forma independiente desde una PC. Con los programas, a través de la PC, se pueden cargar o leer datos del módulo de memoria; también los datos puedan cargarse o descargarse en forma manual, o desde un archivo o a un archivo en el disco de la PC.

Para efectuar las pruebas al BUS serial síncrono se conectó el módulo de memoria externo ("SF1000") directamente a una PC mediante una tarjeta de adquisición de datos que emula la operación de un puerto serial síncrono. Las pruebas se efectuaron con la tarjeta de adquisición de datos debido a que en el laboratorio ("Laboratorio Virtual") donde se construyeron los programas de prueba, las computadoras están equipadas con una tarjeta de adquisición de datos que facilita la conexión de líneas digitales de control con el exterior; sin embargo, la operación del puerto serial síncrono también puede emularse a través del puerto paralelo de la PC de tal forma que pueda prescindirse del uso de la tarjeta de adquisición de datos y los programas puedan ser ejecutados en cualquier computadora equipada con un puerto paralelo. Para transportar los programas para que hagan uso del puerto paralelo, sólo se requiere modificar los programas ("Vi's") de bajo nivel que manipulan las líneas del puerto serial síncrono.

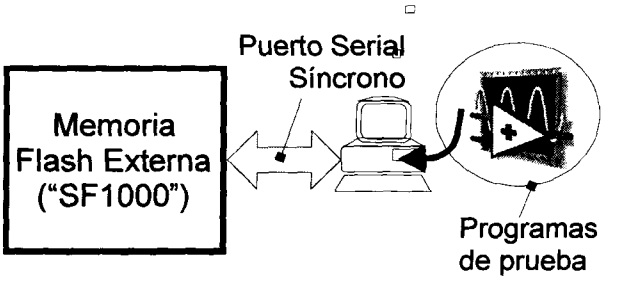


Figura 4.27. Configuración para la prueba de la memoria externa.

Las pruebas efectuadas corresponden a las operaciones básicas para el manejo de la memoria, y son las siguientes:

- SF1000Init. Verifica el tipo y el correcto funcionamiento del circuito de memoria.
- SF1000Write. Escribe un bloque de datos en memoria flash.
- SF1000Read. Lee un bloque de datos de la memoria flash.
- SF1000CheckWrites. Lee el número de veces que ha sido escrito un bloque.
- SF1000EraseBlock. Borra un determinado bloque.

La operación de todas las funciones fue satisfactoria, es decir, se pudo identificar correctamente el dispositivo de memoria mediante la función SF1000Init, también leer y borrar en el área de la memoria flash mediante las funciones respectivas SF1000Read y SF1000Write, verificar el número de escrituras sobre un determinado bloque

(SF1000CheckWrites), y por último borrar completamente los datos de un sector de la memoria (SF1000EraseBlock).

A continuación se muestran algunas imágenes de los paneles frontales de las funciones principales para el manejo de la memoria desde una PC. Las funciones son las mismas que se desarrollaron para el módulo de comunicación, como para el microprocesador HC16.

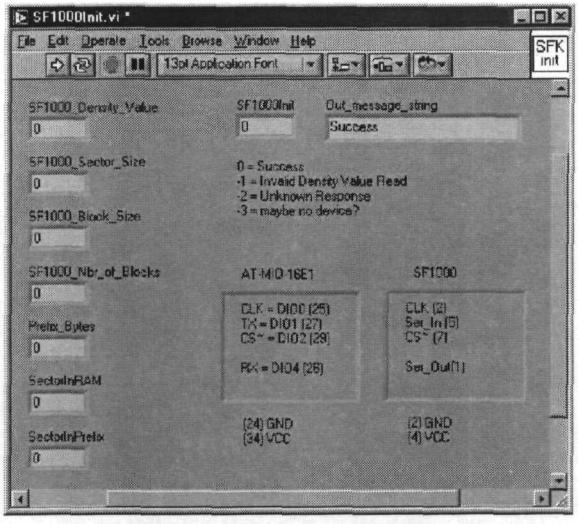


Figura 4.28. Panel frontal de las funciones "SF1000init".

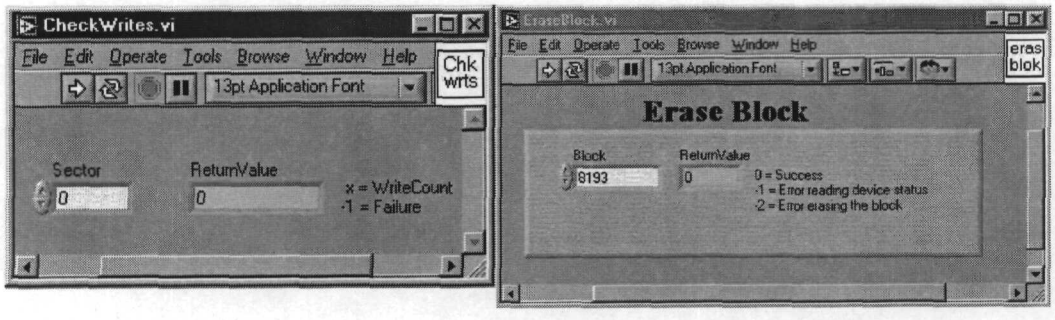


Figura 4.29. Paneles frontales de las funciones "CheckWrites" y "EraseBlock".

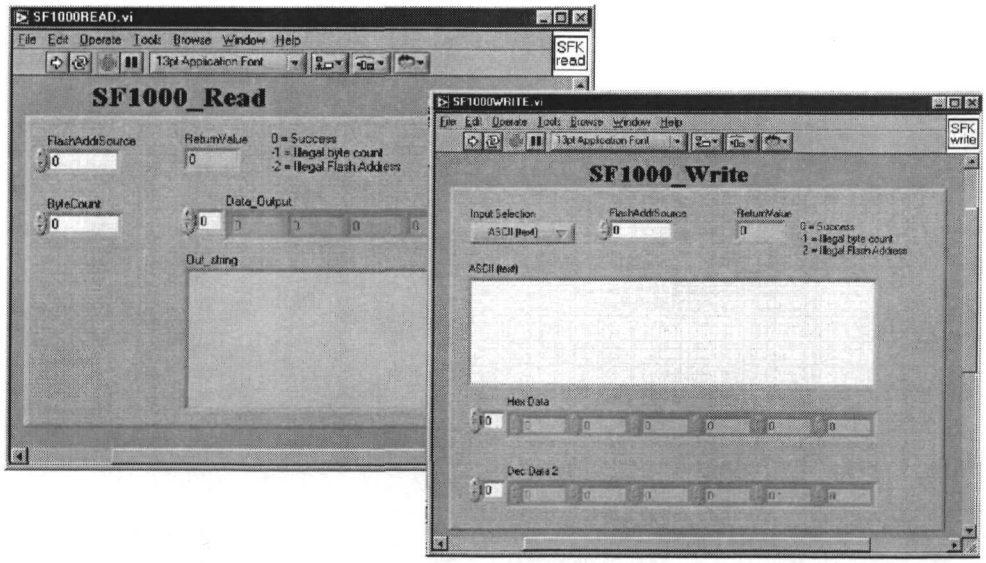


Figura 4.30. Paneles frontales de las funciones "SF1000_Read" y "SF1000_Write".

Capítulo 5. Conclusiones y Observaciones Finales.

5.1. Conclusiones.

Algunas inferencias que se derivan de la realización de este trabajo son las siguientes:

- El módulo de comunicación propuesto en este trabajo de tesis ofrece una solución a muchos dispositivos de procesamiento y con capacidad de comunicación serial, como microcontroladores (μC's) y procesadores digitales de señales (DSP's), para conectarse y comunicar datos a través de una red ethernet.
- El módulo de comunicación opera como un puente entre un dispositivo con capacidad de comunicación serial asíncrona y una red ethernet, donde el punto en común entre los dos extremos de la comunicación serial-ethernet es la memoria interna del módulo de comunicación.
- La memoria externa del módulo de comunicación también actúa como un puente entre una comunicación serial síncrona y un usuario de red del módulo de comunicación.
- El alcance de las operaciones del módulo de comunicación puede extenderse hasta los límites de internet, siempre y cuando la red ethernet donde se conecta el módulo este conectada a su vez a internet.
- Con el uso del módulo de comunicación se elimina la dependencia tradicional de utilizar una PC como elemento de enlace entre la red y cualquier otro sistema electrónico, de esta forma se reducen costos.

5.2. Contribuciones.

Algunas contribuciones de este trabajo son las siguientes:

- La utilización del mismo código de programa (en memoria flash), auxiliado por el almacenamiento de "n" copias de variables de estado (en memoria RAM), permite atender a "n" usuarios de red simultáneamente.
- La adición de un módulo externo de memoria flash para incrementar la capacidad de almacenamiento del módulo de comunicación.
- El arreglo y diseño de la lógica para compartir el módulo externo de memoria en forma directa con el módulo de comunicación y con uno o varios usuarios externos más, como por ejemplo, la tarjeta "freedom", cuyo núcleo es un μcontrolador HC16.
- La creación de librerías de funciones, tanto para el manejo del módulo de comunicación desde una PC, como para el manejo del módulo externo de memoria desde un μcontrolador HC16. Estas librerías contienen funciones ya listas y verificadas, para que en base a ellas el usuario pueda crear sus propias aplicaciones específicas.

5.3. Características Importantes.

También, algunas características importantes acerca del funcionamiento del módulo de comunicación se enlistan a continuación:

- Utiliza el protocolo TCP, que es un protocolo confiable, para el envío de comandos y para la recepción de datos que se involucran en la solicitud de los comandos. La aplicación del protocolo TCP garantiza la integridad de los datos transferidos del módulo, y al módulo.

- Utiliza el protocolo UDP para la emisión de mensajes "broadcast"; es decir, para emitir mensajes a todos los usuarios de una misma red local. Con el protocolo UDP también pueden transmitirse mensajes a un anfitrión específico que esté dentro del alcance de la red global.
- El programa del módulo de comunicación opera bajo un esquema de multitarea cooperativa, lo cual, hace posible dar atención simultánea a múltiples usuarios de red y a un usuario del puerto serial asíncrono.
- La programación de operaciones cíclicas permite que el módulo de comunicación efectúe operaciones en forma local y en base a una referencia de tiempo local. Es decir, una vez que el módulo se programa para efectuar una tarea cíclica, el módulo se desliga de cualquier usuario externo para generar la operación solicitada.
- El hecho de permitir el acceso simultáneo de distintos usuarios obliga a realizar un arbitraje de los recursos del módulo de comunicación; específicamente los recursos que entran en arbitraje son: la memoria externa y extendida, y el puerto serial asíncrono de salida. En el arbitraje las operaciones que llegan y entran en conflicto entran en una cola de espera para ser atendidas posteriormente cuando el recurso en conflicto se libera.
- Además de poder efectuar un control de carácter remoto con el uso del módulo, también es posible conservar la capacidad de un control determinístico utilizando un µcontrolador o procesador digital de señales acoplado al módulo de comunicación, de tal forma que el µC o DSP se encargue directamente del control determinístico, a la vez que el módulo de comunicación se encarga de atender la recepción y transmisión de datos de control que se comunican a través de la red.

5.4. Recomendaciones Futuras.

Algunas recomendaciones futuras para este trabajo.

- El programa principal cuenta ya con una estructura definida, sobre la cual es posible incorporar nuevos comandos de uso específico. En otras palabras, el esqueleto del programa ya está hecho, y sobre dicho esqueleto pueden agregarse, modificarse, o reemplazar comandos por otros nuevos y que por el momento no están disponibles.
- Otra idea que surge a raíz de la programación de las operaciones cíclicas, es que el módulo pueda incorporar funciones programables. Por ejemplo, las funciones pueden tomar ciertas acciones sobre el estado de un cierto puerto o recurso si ocurre una cierta condición. De esta forma, se establece remotamente un esquema de control que se ejecuta localmente en el módulo de comunicación; con esto, las acciones de control se realizan a velocidades de respuesta propias del procesador del módulo de comunicación, sin depender directamente de la velocidad de la comunicación del módulo con los usuarios.
- Se puede incorporar al módulo alguna forma de seguridad sobre el acceso de los usuarios; como por ejemplo, utilizar un esquema de permisos para restringir el uso de los recursos, donde los permisos se otorguen en base al nombre de un usuario y su respectivo "password". También puede restringirse el acceso a los usuarios de tal forma que sólo permita la solicitud de comandos de red provenientes de un IP o de una lista de direcciones IP específicas.

- Otra recomendación es aprovechar las líneas de los puertos que quedan libres y disponibles para incorporarlas en los comandos del módulo de comunicación. Como por ejemplo, en el módulo esta libre el puerto digital A, que incluso puede configurarse bit por bit como entrada o salida. También están libres los puertos seriales A, C, y D, entre otros bits de otros puertos a los cuales se les puede dar uso. De igual forma, la memoria RAM interna para el almacenamiento de datos por el momento no se utiliza en toda su capacidad; se recomienda que para una aplicación final los tamaños de la memoria interna (raíz y extendida) se incrementen para dar uso al espacio disponible en el circuito integrado de memoria RAM.
- Algunas rutinas internas del programa, sobre todo algunas rutinas relacionadas con la transferencia de datos a memoria, pueden rediseñarse para que con el auxilio de "buffers" en RAM mejoren la velocidad de transferencia y por lo tanto el desempeño general del programa del módulo de comunicación.
- En la sección de pruebas y resultados, las pruebas efectuadas al módulo consisten en pruebas sobre operaciones aisladas. Aunque el módulo ha sido probado funcionalmente y con éxito ante situaciones donde intervengan múltiples operaciones, en este trabajo no se documentan pruebas de desempeño y velocidad donde intervengan en forma combinada varias operaciones distintas. Las combinaciones y las variantes que pueden generarse de dichas combinaciones son muchas; sin embargo, puede evaluarse el desempeño del módulo ante cierto tipo de situaciones que se prevea aparezcan con frecuencia y de esta forma predecir su funcionamiento ante tales situaciones. Con respecto a la fase de pruebas, existe todavía un gran número de pruebas con operaciones simultáneas combinadas que pueden efectuarse al módulo de comunicación.

APENDICE A. Foto y Conectores de la Tarjeta RCM2100

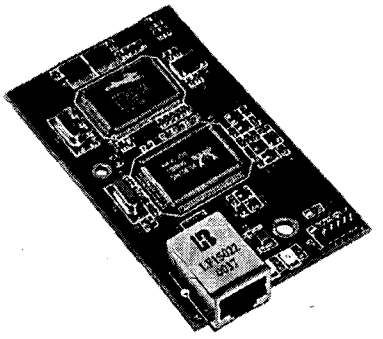
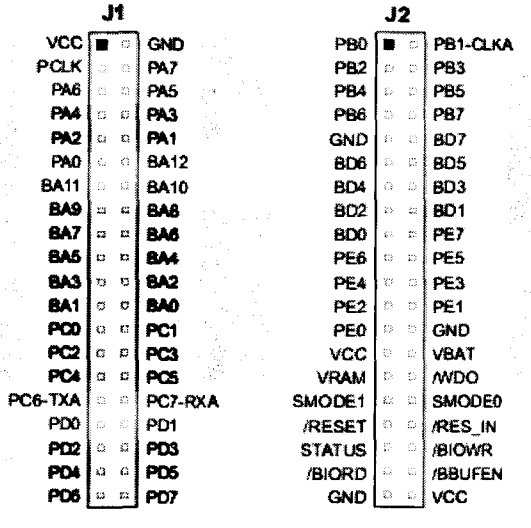
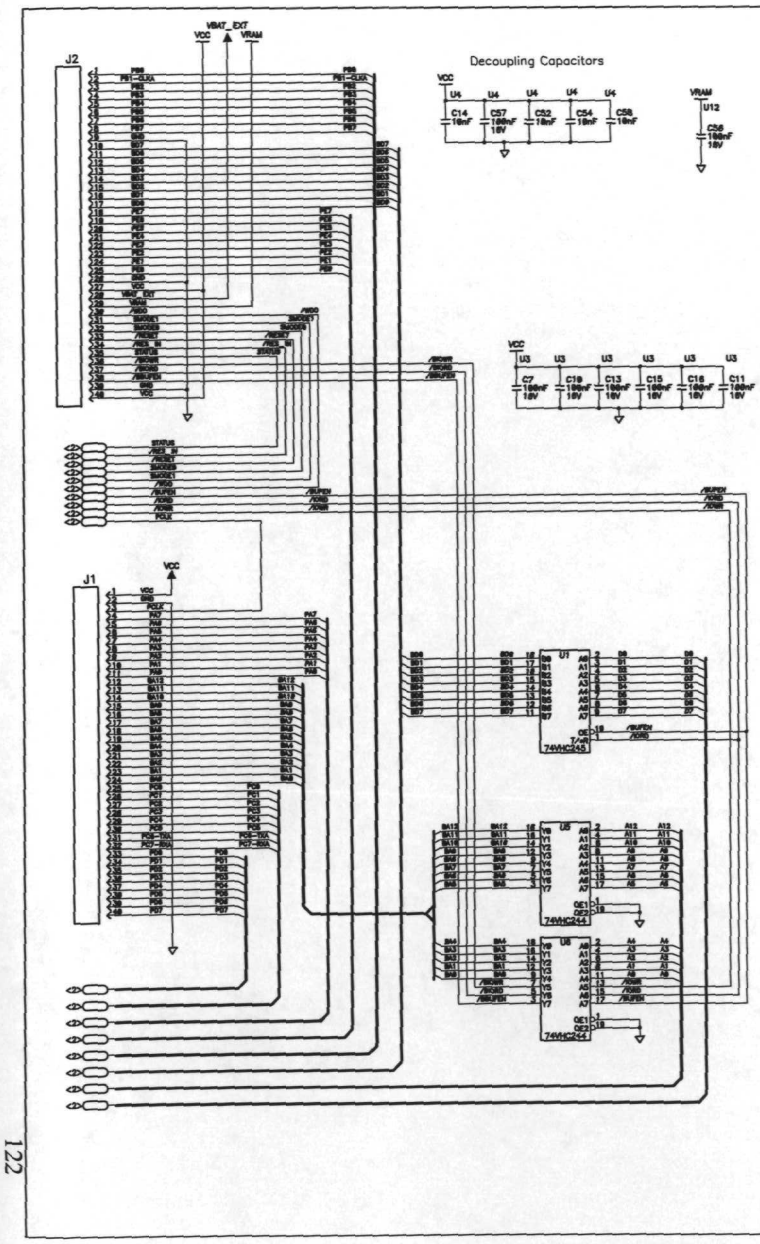


Foto de la tarjeta RCM2100.



Conectores J1 y J2 de la parte posterior de la tarjeta con sus respectivas funciones asignadas por default.

APENDICE B. Diagrama Eléctrico de la Tarjeta RCM2100



REVISION HISTORY			REVISION APPROVAL			
REV	ECO	DESCRIPTION OF CHANGE	PROJECT ENGINEER	APPROVAL DATE	DOCUMENT CONTROL	APPROVAL DATE
A	E11313	INITIAL RELEASE	DM	19Feb01	KIS	3/24/01
B	E11491	CORRECTED FLASH 2 SELECT TO JP2	DM	24Apr01	KIS	4/25/01
C	E11522	CHANGED C8, C9 TO 10pF	DM	14May01	KIS	5/14/01
D	E11588	MODIFIED VBAT CIRCUITRY, ADDED EXTERNAL 32KHz OSC	DM	17AUG01		

NOTES: UNLESS OTHERWISE SPECIFIED;
1. ALL RESISTOR VALUES ARE IN OHMS, 1/16W, 5%
2. ALL CAPACITORS ARE 50VDC OR HIGHER.
3. THE ORIGINATION SOURCE OF A VOLTAGE IS REPRESENTED BY ($\overset{VCC}{\uparrow}$), AND ALL REFERENCES TO THAT VOLTAGE ARE REPRESENTED BY ($\overset{VCC}{\top}$).

⚠ OUTLINED CIRCUIT MAY NOT BE STUFFED DEPENDING ON MODEL. SEE STUFFING CHART FOR CLARIFICATION.

5. COMPONENT VALUES SHOWN WITH AN ASTERISK (*) FOLLOWING THE VALUE, MAY HAVE DIFFERENT VALUES, OR MAY NOT BE STUFFED DEPENDING ON MODEL. SEE STUFFING CHART FOR CLARIFICATION.

TABLE A

REF DES	DEVICE	DEVICE VOLTAGE INFORMATION						DEVICE: FILTER CAP REF DES(s)
		AGND	GND	VCC	VRAM	uPBAT		
U1	74WHC245		10	20				C5
U2	EIC811L		2	4				C12
U3	RTL8019AS		14,38,44 52,63,86	6,17,47 57,76,89				C7,10,12,13,15,16
U4	RABBIT 2000		2,27,38 52,77,89	3,26,53, 78,82		42		C14, 57, 52, 54, 58 C25 – PIN42 (uPBAT)
U5	74WHC244		10	20				C21
U6	74WHC244		10	20				C22
U10	FLASH		24	8				C59
U11	FLASH		24	8				C55
U12	SRAM 512K X 8		15		32			C56
	SRAM 128K X 8		15		32			
	SRAM 32K X 8		14		28			

STUFFING TABLE

CIRCUIT	PART	RCM2100	RCM2110	*RCM2115	RCM2120	RCM2130
SRAM SRAM select	U12 JP3	512K 2–3	128K 1–2	128K 1–2	512K 2–3	128K 1–2
FLASH 1 FLASH 1 select	U10 JP1	256K 1–2	256K 1–2	256K 1–2	256K 1–2	256K 1–2
FLASH 2 FLASH 2 select	U11 JP2	256K 1–2	NOT INSTALLED	NOT INSTALLED	256K 1–2	NOT INSTALLED
ETHERNET OPTION	U3	INSTALLED	INSTALLED	INSTALLED	NOT INSTALLED	NOT INSTALLED
	J3	NOT INSTALLED	NOT INSTALLED	INSTALLED	NOT INSTALLED	NOT INSTALLED
	J4	INSTALLED	INSTALLED	NOT INSTALLED	NOT INSTALLED	NOT INSTALLED
	R1, R2 DS1, DS2	INSTALLED	INSTALLED	NOT INSTALLED	NOT INSTALLED	NOT INSTALLED
	R5, C1, C2, C3, C4	INSTALLED	INSTALLED	NOT INSTALLED	NOT INSTALLED	NOT INSTALLED
	R10, R11, R19 C1, C2, C3 C4, C7, C8 C9, C11, C12 C13, C15, C16	INSTALLED	INSTALLED	INSTALLED	NOT INSTALLED	NOT INSTALLED
	Y1	INSTALLED	INSTALLED	INSTALLED	NOT INSTALLED	NOT INSTALLED
PORTS D-E OPTION	R21, R24, R35 R36, R37, R38	NOT INSTALLED	NOT INSTALLED	NOT INSTALLED	INSTALLED	INSTALLED
INTERRUPT/JUMPER OPTIONS	R12, R18, R25 R58, R59	NOT INSTALLED	NOT INSTALLED	NOT INSTALLED	NOT INSTALLED	NOT INSTALLED

*Note: RCM2115 only availible for custom orders

COPYRIGHT 2001, Z-WORLD, INC. 08/17/01

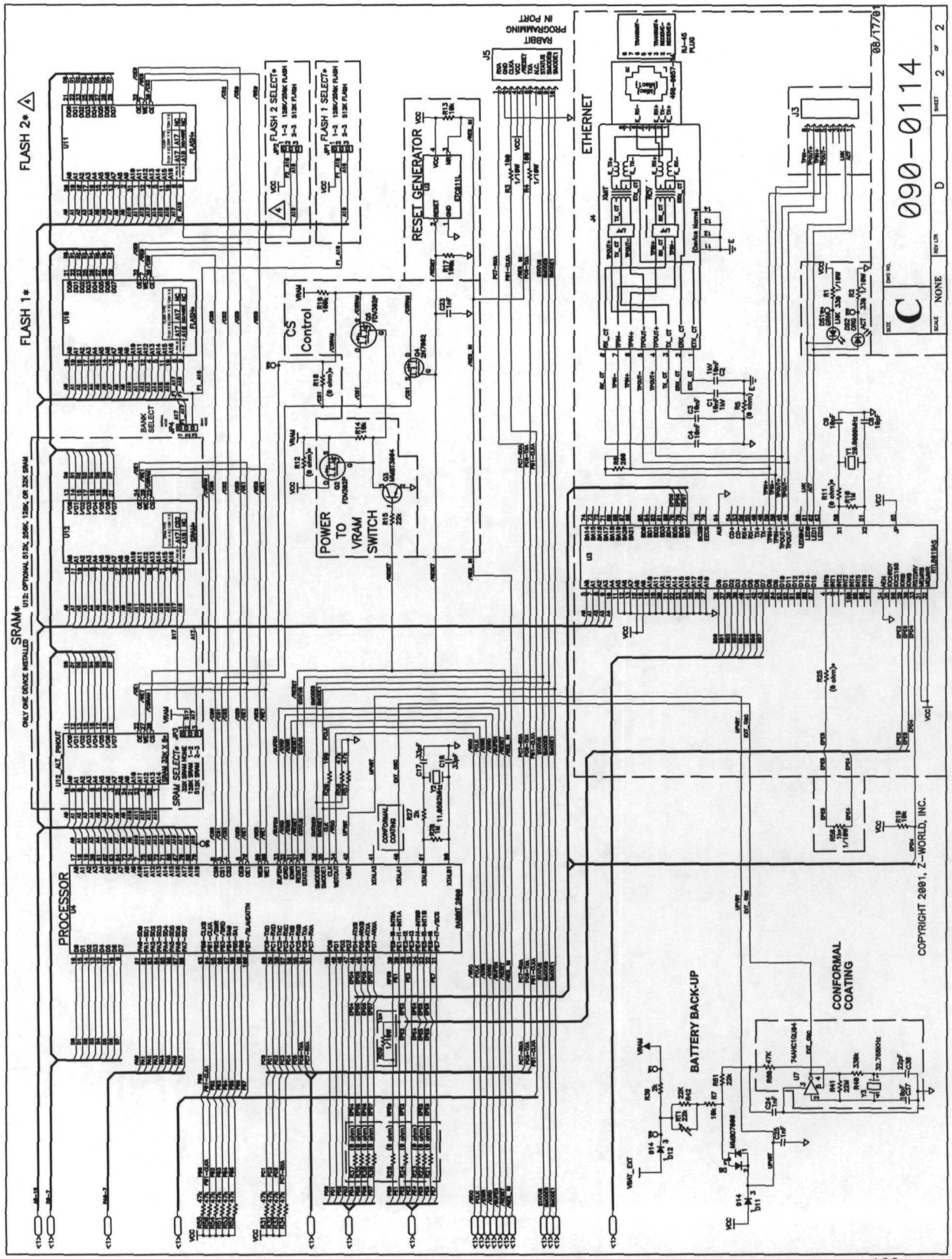
APPEND THE FOLLOWING DOCUMENTS WHEN CHANGING THIS DOCUMENT:	DRAWING CONTENT:	
	DRAWN BY: (INITIAL RELEASE)	25JUL00
	KAH	
	REVISED BY:	08/17/01
	DM	
	APPROVALS: INITIAL RELEASE	
	PROJECT ENGINEER:	
	ENGINEERING MANAGER:	
	SIGNATURES	DATE

TITLE: SCHEMATIC DIAGRAM RCM2100 SERIES

Z-WORLD
2900 SPAFFORD ST.
DAVIS, CA 95616
530 - 757-4616

SIZE: C | DWG NO.: 090–0114

SCALE: NONE | RELEASE DATE: 3/24/01 | SHEET 1 OF 2


FLASH 2*
FLASH 1*
SRAM*
PROCESSOR
POWER TO VRAM SWITCH
RESET GENERATOR
ETHERNET
RABBIT PROGRAMMING IN PORT
BATTERY BACK-UP
CONFORMAL COATING
090-0114
COPYRIGHT 2001, Z-WORLD, INC.

BIBLIOGRAFIA.

"32-megabit 2.7-volt Only Serial DataFlash® AT45DB321", ATMEL, EUA 2001.

"An Introduction to TCP/IP (rev. C)", Rabbit Semiconductor Inc., EUA 2000.

Comer Douglas E. "Redes Globales de Información con Internet y TCP/IP, Principios básicos, protocolos y arquitectura". Tercera Edición. Pearson, Prentice Hall, México 1999.

Dieck, Graciano. "Instrumentación, Acondicionamiento Electrónico y Adquisición de Datos". Preedición. Editorial Trillas, México 2000.

"Dynamic C TCP/IP User's Manual", Z-World Inc., EUA 2001.

"Dynamic C User's Manual", Z-World Inc., EUA 2001.

"Freedom16 mite v2 Advanced 16 bit Controller User's Manual", Intec Automation Inc., Canadá 1997.

"HC16 User's Manual", Motorola, EUA 1997.

Jamsa Kris, Ken Cope. "Programación en Internet". McGraw-Hill, México 1996.

"LabView Advanced I Course Manual", National Instruments, EUA 1998.

López Angel, Alejandro Novo. "Protocolos de Internet, Diseño e Implementación en sistemas UNIX". Alfaomega, Colombia 2000.

"RabbitCore RCM2100 User's Manual", Z-World Inc., EUA 2001.

"Rabbit 2000™ Microprocessor User's Manual", Z-World Inc., EUA 2000.

"SF1000 Serial Flash Expansion Card User's Manual", Z-World Inc., EUA 2001.

"TN301 - Rabbit 2000™ Microprocessor Interrupt Issue", Rabbit Semiconductor Inc., EUA.

Villanueva Rafael, Manuel E. Macías . "Procesamiento Digital con Compensación de Error para el Cálculo de Parámetros Eléctricos en Sistemas de Potencia". XXXII Congreso de Investigación y Extensión del Sistema Tecnológico de Monterrey, México 2002.

Centro de Información-Biblioteca
30002006243521