PROTOCOLO DE COMUNICACIÓN:

Modelo:

En la red de PLCs CAIPE se usa el modelo maestro-esclavo. Esto significa que un único dispositivo inicia la comunicación (maestro, que se halla en la cabecera de la red). De los PLCs esclavos en la red solo uno contestará (aquel cuyo identificador coincida).

Enlace:

En el enlace, la PC inicia la comunicación con el bloque de petición, y el PLC cuyo identificador coincide con el solicitado responde con su bloque de acknowlege. Los bloques intercambiados son de una longitud de 20 bytes, con un xor de los datos trasmitidos en el ultimo byte.
El PLC tolera un tiempo máximo de 50 mseg. entre dato y dato que trasmita la PC, pasado ese tiempo se descarta el bloque.
El PLC solo responde cuando la petición es correcta.

-FORMATO DE LA RS232:

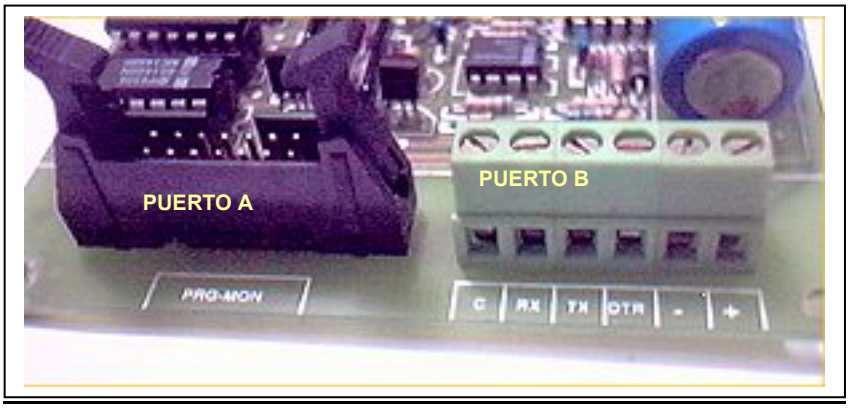
Puerto A:

Baud rate:	**4800**
Longitud de la palabra:	**8 bits**
Numero de stop bits:	**2**
Paridad:	**par (even)**
DTR:	**(Deshabilitado o desconectado)**

Puerto B:

Baud rate:	**4800**
Longitud de la palabra:	**8 bits**
Numero de stop bits:	**1**
Paridad:	**n (none)**
DTR:	**(Deshabilitado o desconectado)**

NOTA: el dato siempre es de 8 bits y modelos anteriores de PLC solo tienen puerto A a menor velocidad.



Comandos:

El protocolo de comunicaciones soporta dos tipos de comandos: de acceso por bloques y de acceso puntual.

Los comandos de acceso por bloques direccionan toda la memoria RAM del PLC (es decir las direcciones absolutas desde 0 hasta 8191), con bloques de 16 bytes consecutivos tomadas desde 0.Por ejemplo para direccionar 16 byte a partir de la dirección de la memoria usuario 288 (120 eh hex) se pide una comunicación con el bloque nro. 18 (12 en hex), o sea 288/16.

La memoria del PLC se divide en baja (de 0 a 575) y alta (de 2528 a 8191).La memoria baja a ser direccionada por los comandos de acceso por bloques tiene la siguiente disposición:

Direcciones de la memoria usuario de 0 a 501	0 **0x1F5**	**AREA DE I/0 Y DE VARIABLES INTERNAS DEL PLC**
Direcciones de la memoria usuario de 501 a 511 (área reservada)	**0x1F6** **0x1FF**	**AREA DE CONSTANTES Y ACUMULADORES**
Direcciones de la memoria usuario de 512 a 575	**0x200** **0x23F**	**AREA DE ESTADO DE TEMPORIZADORES Y CONTADORES**
	0x240 **0x2BF**	**AREA DE ESTADO DE CUENTA DE TEMPORIZADORES Y CONTADORES**
	0x300 **0xFFF**	**VARIABLES INTERNAS DEL BIOS (área reservada)**

El área de estado de cuenta de timers y contadores, son 128 bytes que forman 8 bloques de 16 bytes cada uno, donde cada contador o timer ocupa 2 bytes para su valor de cuenta con un numero que va desde 0 a 9999.Los bits mas altos deben enmascararse en la recepción con ceros pues no tienen significado. Por ejemplo:

Cuenta de Timer o Contador 512: **0x240** bbbbbbbb (parte baja)
0x241 **xx**bbbbbb (parte alta)

La memoria alta del PLC comienza en 2528 y continúa hasta 8191. El área del teclado display alfanumerico comienza en 2528, y el final de la memoria ocupada por éste se especifica en los bytes 0x80Dy 0x80E, en parte alta y parte baja.

NOTA: al leer estos datos se notará que las direcciones son del tipo 4xxx o 5xxx. Esto se debe a que corresponden a direcciones absolutas en la memoria del PLC. Por lo tanto deducimos que la dirección 0 del usuario corresponde a la dirección 0x4000 en la memoria del PLC.

El final XXXX se especifica en 0x80D y 0x80E (2061 y 2062)	**2528** **XXXX**	**ZONA RESERVADA TECLADO DISPLAY ALFANUMERICO**
Desde XXXX (lo ocupado por el teclado-display alfanumérico) hasta la dirección 0x1FFF	 **8192**	**AREA DE MEMORIA ALTA USUARIO**

Los comandos de acceso puntual solo direccionan sobre el área de I/O y variables internas de la memoria usuario del PLC,(entre 0 y 501).El valor que se le transfiere como dirección es el de la parte baja y la parte alta

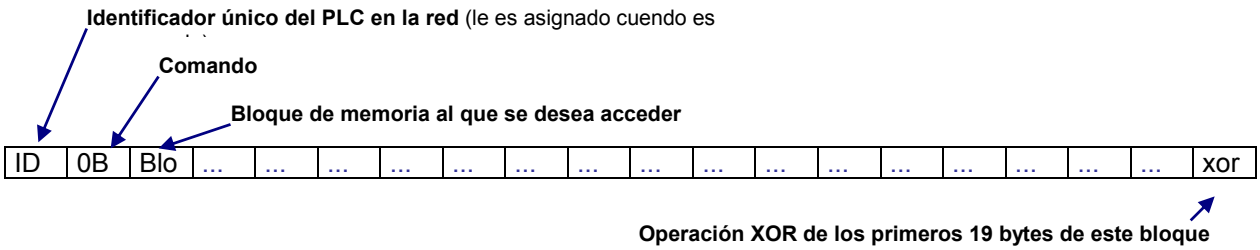
del numero de variable. Por ejemplo si se desea acceder a la dirección 288 (120 en hex) se le pasa un 1 en el byte de parte alta y un 20 en el de la parte baja.

Comando de acceso por bloque:

LECTURA DE BLOQUE EN LA MEMORIA BAJA (comando 0B hex):

Permite leer un bloque de 16 bytes de la memoria baja del PLC(de 0 a 0x2BF) y parte de la alta (0x9E0 a 0xFFF). Se le debe especificar para ello el , identificador del PLC, el comando y el numero de bloque, con el siguiente formato:

Bloque de petición :



Bloque de acknowlege (respuesta):

ID	0B	Blo	Dat 1	Dat 2	Dat 3	Dat 4	Dat 5	Dat 6	Dat 7	Dat 8	Dat 9	Dat 10	Dat 11	Dat 12	Dat 13	Dat 14	Dat 15	Dat 16	xor

El PLC en su bloque de acknowlege responde con los 16 bytes del bloque. El numero de bloque se calcula como se especifico antes. Por ejemplos: para leer la dirección de la memoria usuario 342, se debe pedir comunicación con el bloque 21 (342/16).

ESCRITURA DE BLOQUE EN LA MEMORIA BAJA (comando 0A hex):

Permite escribir un bloque de 16 bytes de la memoria baja del PLC(de 0 a 0x2BF) y parte de la alta (0x9E0 a 0xFFF). Se le debe especificar para ello el , identificador del PLC, el comando ,el numero de bloque, y seguido los 16 bytes a escribir con el siguiente formato:

Bloque de petición :

Bloque de petición :

ID	0A	Blo	Dat 1	Dat 2	Dat 3	Dat 4	Dat 5	Dat 6	Dat 7	Dat 8	Dat 9	Dat 10	Dat 11	Dat 12	Dat 13	Dat 14	Dat 15	Dat 16	xor

Bloque de acknowlege:

ID	0A	Blo	AA	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	xor

El PLC en su bloque de acknowlege responde con un 0xAA indicando transferencia efectuada o con un 0xEE si no se pudo realizar.

LECTURA DE BLOQUE EN LA MEMORIA ALTA (comando 1B hex):

Permite leer un bloque de 16 bytes de la memoria alta del PLC(de 0x1000 a 0x1FFF), de la misma manera que como se especifico para leer bloques en la parte baja.

Bloque de petición :

ID	1B	Blo	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	xor

Bloque de acknowlege:

ID	1B	Blo	Dat 1	Dat 2	Dat 3	Dat 4	Dat 5	Dat 6	Dat 7	Dat 8	Dat 9	Dat 10	Dat 11	Dat 12	Dat 13	Dat 14	Dat 15	Dat 16	xor

ESCRITURA DE BLOQUE EN LA MEMORIA ALTA (comando 1A hex):

Permite escribir un bloque de 16 bytes de la memoria alta del PLC(de 0x1000 a 0x1FFF). Se le debe especificar para ello todo tal , cual se definió para el comando 0A(hex) con la diferencia de que los bloques se toman desde 0x1000.

Bloque de petición :

ID	1 A	Blo	Dat 1	Dat 2	Dat 3	Dat 4	Dat 5	Dat 6	Dat 7	Dat 8	Dat 9	Dat 10	Dat 11	Dat 12	Dat 13	Dat 14	Dat 15	Dat 16	xor

Bloque de acknowlege:

ID	1 A	Blo	AA	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	xor

Comandos de acceso puntual:

LECTURA BINARIA (comando 6):

Lee a lo sumo 8 valores de un byte en direcciones de la memoria usuario del PLC. Estas direcciones se le especifican al PLC en el bloque de interrogación, como parte baja y parte alta respectivamente. El PLC en su bloque de acknowlege responde, con los 8 valores de esos lugares memoria ubicados en los 2 bytes de los lugares correspondientes a donde se le indico su dirección.

Bloque de petición :

Bloque de petición :

ID	6	Dir 1L	Dir 1H	Dir 2L	Dir 2H	Dir 3L	Dir 3H	Dir 4L	Dir 4H	Dir 5L	Dir 5H	Dir 6L	Dir 6H	Dir 7L	Dir 7H	Dir 8L	Dir 8H	0	xor

Bloque de acknowlege:

ID	6	dat 1	0	dat 2	0	dat 3	0	dat 4	0	dat 5	0	dat 6	0	dat 7	0	dat 8	0	0	xor

Cuando en una comunicación se desea leer menos de 8 bytes se deben especificar las direcciones a las que se desea acceder y seguida de la ultima, enviar un 255 (ff en hex) en la parte alta de la dirección.

ESCRITURA BINARIA (comando 7):

Escribe a lo sumo 4 bytes en la memoria usuario del PLC. Se especifica dirección (parte alta y baja) y valor de cada byte. Por cada byte a escribir el PLC devuelve en su bloque de acknowlege un 170 (AA en Hex) en el lugar del dato si se realizo la transferencia o un 238 (EE en hex) si hubo un error.

Bloque de petición :

ID	7	Dir 1L	Dir 1H	dat	x	Dir 2L	Dir 2H	dat	x	Dir 3L	Dir 3H	dat	x	Dir 4L	Dir 4H	dat	x	0	xor

Bloque de acknowlege:

ID	7	Dir 1L	Dir 1H	AA o EE	x	Dir 2L	Dir 2H	AA o EE	x	Dir 3L	Dir 3H	AA o EE	x	Dir 4L	Dir 4H	AA o EE	Dir 8H	0	xor

Cuando se desee escribir menos de 4 byte lo que se debe hacer es especificar en la parte alta de la próxima dirección a la ultima un 255 (ff en hex).

LECTURA ANALÓGICA (comando 8):

Su uso es similar a como se describió en el comando 6, con la diferencia de que los datos son de 2 bytes.

Bloque de petición :

ID	8	Dir 1L	Dir 1H	Dir 2L	Dir 2H	Dir 3L	Dir 3H	Dir 4L	Dir 4H	Dir 5L	Dir 5H	Dir 6L	Dir 6H	Dir 7L	Dir 7H	Dir 8L	Dir 8H	0	xor

Bloque de acknowlege:

ID	6	dat 1l	dat 1h	dat 2l	dat 2h	dat 3l	dat 3h	dat 4l	dat 4h	dat 5l	dat 5h	dat 6l	dat 6h	dat 7l	dat 7h	dat 8l	dat 8h	0	xor

ESCRITURA ANALÓGICA (comando 9):

Su uso es similar a lo que se especifico para el comando 7, con la diferencia de que los datos son de 2 bytes.
Bloque de petición :

ID	9	Dir 1L	Dir 1H	dat 1L	dat 1H	Dir 2L	Dir 2H	dat 2l	dat 2h	Dir 3L	Dir 3H	dat 3l	dat 3h	Dir 4L	Dir 4H	dat 4l	dat 4h	0	xor

Bloque de acknowlege:

ID	9	Dir 1L	Dir 1H	AA o EE	x	Dir 2L	Dir 2H	AA o EE	x	Dir 3L	Dir 3H	AA o EE	x	Dir 4L	Dir 4H	AA o EE	Dir 8H	0	xor

Ejemplo en VisualBasic

Transmisión:

```
    If Bloque > 255 Then           'Detecta a que área pertenece
        Tx(1) = &H1B
        Tx(2) = Bloque - 256
    Else
        Tx(1) = &HB
        Tx(2) = Bloque
    End If

    Tx(0) = Id
    Tx(19) = 0
    Tdx = ""

    For i = 0 To 18                'XOR de los primeros 19 bytes
        Tx(19) = Tx(19) Xor Tx(i)
        Tdx = Tdx & Chr(Tx(i))
    Next
    Tdx = Tdx & Chr(Tx(19))

    If MSComm1.PortOpen = False Then     'Si el Puerto de la PC no estaba
        If s.Puerto = A Then             'inicializado, lo inicializo ya
            Aux1 = ",e,8,2"              'acorde a los parámetros
        Else
            Aux1 = ",n,8,1"
        End If
        Aux1 = CStr(s.Baudios) & Aux1
        MSComm1.Settings = Aux1
        MSComm1.DTREnable = False
        MSComm1.PortOpen = True
    End If

    MSComm1.Output = Tdx           'Los datos serán transmitidos
```

Recepción:

La recepción es manejada por evento.

```
Private Sub MSComm1_OnComm()
    Dim Aux1 As String, i As Integer, Ch As Integer, B As Byte

    Select Case MSComm1.CommEvent
        Case comEventFrame, comEventOverrun, comEventRxOver, comEventRxParity,
comEventDCB
        ' Errores de comm
            Aux1 = MSComm1.Input
            MSComm1.InBufferCount = 0   'Abort
            RaiseEvent ErrorComm(17, "Corrupción en datos recibidos")
        Case comEvReceive
        ' Cada dato recibido
            tmrTOut.Enabled = False      'timer para detectar falla
            tmrTOut.Interval = 0         'de comunicación
```

```
            tmrTOut.Interval = 50
            tmrTOut.Enabled = True

            If MSComm1.InBufferCount >= 20 Then
                  tmrTOut.Enabled = False
                  tmrTOut.Interval = 0
                  ' Proceso Rx
                  GetFromMSComm
            End If
    End Select
End Sub


Private Sub GetFromMSComm()
    Dim Aux1 As String, i As Integer, Ch As Integer
    Dim L As Integer

    BufferRx = BufferRx & MSComm1.Input
    Aux1 = MidB(BufferRx, 1, 20)
    L = LenB(BufferRx)
    BufferRx = MidB(BufferRx, 20, L - 20)      'Guardo el resto
    Ch = 0
    For i = 1 To 20
        If MidB$(Aux1, i, 1) = "" Then         ‘Recupero datos
            Rx(i - 1) = 0
        Else
            Rx(i - 1) = AscB(MidB$(Aux1, i, 1))
        End If
        Ch = Ch Xor Rx(i - 1)                  ‘XOR de verificación
    Next

    If Ch <> 0 Then
        MSComm1.InBufferCount = 0   'Abort
        Aux1 = MSComm1.Input        'Clear
        RaiseEvent ErrorComm(93, "Los datos recibidos no son validos (no da
checksum)")
        BufferRx = ""               'Reset
        Exit Sub
    End If
    tmrWatchDog.Interval = 0
    tmrWatchDog.Enabled = False
    RaiseEvent RespuestaPLC(Rx(0), Rx(2), Rx(1))
End Sub
```