1771-DMC Control Coprocessor and Development Software

What This Document Contains

These release notes apply to all catalog numbers of the 1771 Control Coprocessor (1771-DMC, -DMC1, and -DMC4), 1771 Control Coprocessor Serial Expander (1771-DXPS), and Program Development Software (1771-PCB). This document contains information that is not found in the 1771 Control Coprocessor User Manual, publication number 1771-6.5.95.

Identifying Your Revision and Release Numbers

To identify your revision and release number:

To identify the:	**Do this:**	
Revision level of your control-coprocessor firmware	See the first screen at power-up or look at the side nameplate label: [1]	
	Screen	Side Label
	Rev 1.00	Rev A
	Rev 1.01	Rev B
	Rev 1.10	Rev C
	Rev 1.20	Rev D
	Series/Revision A/E (1.30)	Rev E
	Series/Revision A/G (1.41)	Rev G
	Series/Revision A/H (1.50)	Rev H
Release level of your Program Development Software	See the disk label.	

[1] **Important:** If you have received firmware updates, the screen identification will be the most accurate and it might not match the side nameplate label.

If you need to update your firmware, software, or hardware, contact Allen-Bradley Technical Support at (440) 646-6800.

Publication 1771-RN951C-EN-P - July 2001

Solving an Incompatibility

Note the following solution to an incompatibility:

If you want to:	You must:
use block transfers on a coprocessor in a remote rack with a 1771-ASB module having firmware revision G or later	use a coprocessor with firmware revision E or later.

Restrictions

The following describes how you can avoid current restrictions:

If you:	Then:	To avoid this problem:
have multiple block transfers occurring at a high rate on a coprocessor—or other module that can do both block and single transfers-in a remote rack with a 1771-ASB module	single transfers may seldom or never occur	eliminate any single transfers that could occur while block transfers are occurring
use a DT_DIAGNOSTIC_COUNTERS command with the INTERD INTERCHANGE™ server for the coprocessor	the following error is received: `PCCCSTS50-Addressing problem or memory protect rungs`	do not use a DTL_DIAGNOSTIC_COUNTERS command with the INTERD INTERCHANGE server for the coprocessor.

Updates for the Year 2000

Revision G (1.41) and Later

This release includes updates to the Control Coprocessor OS-9 operating system for the year 2000. Changes include:

- modifications to the real time clock driver rtc1284.a
- updated OS-9 modules from Microware

Module:	Description:
tsmon setime pr login fsave frestore dir pipeman.a time.h basic runb	supports the years after 1999

COMM Port Changes

Changes in Setting Up Communication Parameters

IMPORTANT A change was made to the configuration of the:

- COMM port T1 on the Control Coprocessor
- T2 and T3 COMM ports on the Serial Expander

In earlier revisions (revision 1.20 and earlier), all of the COMM ports were configured as terminals. In this revision, and all future revisions, the T2, T3, and T4 COMM ports are configured to default to raw binary configuration.

For more details, see the "Setting Up Communication Parameters" section in the 1771 Control Coprocessor User Manual, publication number 1771-6.5.95.

If you have an application that was designed prior to release 1.30, and you require the original settings of the serial port COMM1, for example, include `xmode` commands in the start-up file with the following argument:

xmode /t1 normal

This returns the serial-port settings to the settings used before firmware release 1.30. See the 1771 Control Coprocessor User Manual, publication number 1771-6.5.95, for a description of CC_VALCOMM, a utility that saves the changes made after an `xmode` operation and retains the changes over power cycles.

Corrected Anomalies

Revision G (1.41) and Later

This release corrects the following anomalies:

Serial Port Changes for 1771-DXPS

The serial port driver now responds to XOFF commands from attached devices; it ignores "abort" and "quit" if the settings are set to zero, and it does not echo the XON command when set for noecho.

API Function Calls

The BASIC and C versions of the CC_PLC_BTW and CC_PLC_BTR functions no longer scramble the rack, module, and group data.

New Functionality

Revision H (1.50) and Later

The new functionality of this release of the firmware, revision 1.50 or later - requires the latest version of the INTERD module - Revision 8003. The new INTERD module is included in the 1771-DCMU Update Kit. The new functionality includes:

INTERD Interchange Daemon

With this revision, the INTERD daemon allows:

- word range read and word range write messages (maximum transfer of up to 1000 words of data per call) routed to the coprocessor.
- typed read and write messages (maximum 2000 bytes of undefined typed data per call) routed to the coprocessor

The messages are routed to the coprocessor in one of two ways:

- the coprocessor station address (set by the CC_CFG utility) matches the destination address of the received INTERD PCCC packet, or
- the symbolic (TAG) address within the INTERD PCCC packet begins with an “x_” or an “X_”

The daemon has been changed to remove the screening of PCCC commands. For more information, refer to the following section.

TAG_DEFINE Function

With this revision, the maximum value for the tag_size parameter is increased to 2000 bytes. This increase allows up to 1000-word transfers.

Using the INTERD INTERCHANGE Daemon

INTERD is a PCCC INTERCHANGE server daemon that provides communication between the Control Coprocessor, its attached PLC-5 processor, and a host computer running INTERCHANGE software via the Ethernet connection of the coprocessor.

INTERD is included in revision 1.20 or later of the PCBridge software (1771-PCB) and requires Series A Revision E (1.30) or later firmware in the Control Coprocessor.

INTERCHANGE is an Allen-Bradley application-programming interface (API) that allows easy, consistent programmatic access to control-system information.

By installing the INTERD daemon on the Control Coprocessor, you have the ability to do the following over the Ethernet network:

- run RSLogix5 programming software on a host computer to program or monitor a PLC-5 processor connected to the coprocessor.
- run an INTERCHANGE program on a host computer to access the data table of the PLC-5 processor connected to the coprocessor.
- run an INTERCHANGE program on a host computer to access the TAG table of the coprocessor.
- access the data table of the PLC-5 processor connected to the coprocessor from a remote Ethernet PLC-5 processor by using the message instructions.
- access the TAG table of the coprocessor from a remote Ethernet PLC-5 processor by using the message instruction.

Introduction to INTERCHANGE Access to Coprocessor-Tagged Memory

The INTERD daemon (Version 8003) along with coprocessor release Revision H (1.50) allows word range read and word range write message access to the TAG memory of the coprocessor. The maximum transfer limit is 1000 words per call. Typed read and typed write message access to the TAG memory is also available. The maximum transfer limit is 2000 bytes of data per call.

There are two ways you can route INTERCHANGE messages to the Control Coprocessor:

1. Assign the Control Coprocessor its own station number. Do this via the CC_CFG utility. Any INTERCHANGE message received by the INTERD daemon with its own station address will be processed by the coprocessor and not transferred to the PLC-5.

2. Assign INTERCHANGE messages with the symbolic (TAG) address within the PCCC packet with an “X_” or “x_”.

Using Typed Read/Write instructions give you the ability to read/write down to the BYTE granularity of the TAG data. Typed accesses are done by issuing a PCCC typed read or write command of undefined type (0x22) with an element size of 1 byte. Word Range Read/Write limits the granularity to the WORD level, however this allows a PLC-5 processor on Ethernet to read/write the TAG data.

All external access to the Control Coprocessor's user memory is through the TAG table of the coprocessor. The TAG functions provide a way for you to specify access to control-coprocessor memory. The memory of the tagged area can be of any data type - e.g., char, short, float, etc. - or combination of data types.

It is your responsibility to understand the layout of the tagged memory. Transmission or reception of tagged memory data is done as a "byte stream." External devices can have different memory structures - i.e., byte order, data sizes, etc. When reading tagged data from the coprocessor, the external process must accommodate the difference when interpreting the byte stream. Similarly, when writing to the tagged area, the external process must generate a byte stream to match that of the coprocessor tagged memory.

EXAMPLE Program of INTERCHANGE Access to Coprocessor-Tagged Memory

The following example illustrates how to set up the tagged memory of the coprocessor and access that memory over Ethernet using INTERCHANGE software on the host computer.

In this example, we set up a tagged area defined by the structure CAR. The TAG name “X_Car” points to the start of the CAR structure. The memory allocated by the OS9 compiler for the CAR structure is:

Structure	Offset	Allocation
CAR ->	00	make (bits 31-24)
	01	make (bits 23-16)
	02	make (bits 15-8)
	03	make (bits 7-0)
	04	model
	05	type
	06	color
	07	“pad” byte
	08	serial (bits 31-24)
	09	serial (bits 23-16)
	10	serial (bits 15-8)
	11	serial (bits 7-0)

Note the inclusion of a “pad” byte generated by compiler. The pad byte is necessary to make serial start on an even addressed boundary. This illustrates how imperative it is that you know the exact memory layout of the tagged area.

The following example of a coprocessor program creates the X_Car TAG and periodically increments the make, model and type elements of the structure. In this example, the coprocessor is set up to be station 22 octal (12h). To increase readability of the example, no error checking is done.

This is a coprocessor example program (INTERTAG.C) used in conjunction with the host INTERCHANGE program itaghost.c to demonstrate the use of the INTERD daemon. The file is started on the coprocessor prior to running itaghost.

Coprocessor Program Example:

```
#include <copro.h>
typedef struct
     {
     unsigned    make;  /* Define CAR structure */
     char        model;
     char        type;
     char        color;
     unsigned    serial;
     }CAR;

  main()
  {
  unsigned tag_id1; /* TAG id for CAR Tag */
  CAR car; /* The CAR Structure */

 CC_INIT();      /* Initialize the Coprocessor Library */

  TAG_DEFINE (&tag_id1,&car,"X_Car",sizeof(car),TG_MODIFY);  /* define the TAG */
  car.make = car.model = car.type = car.color = car.serial = 0;/* init the data
*/
  while (1)/* loop forever */
    {
    TAG_LOCK (tag_id1,CC_FOREVER);/* prevent concurrent access to TAG */
    car.make += 1;/* modify the make */
    car.model += 2;/* and model */
    car.type += 3;/* and type */
    TAG_UNLOCK (tag_id1,CC_FOREVER);
    sleep (1);
   }
  }
```

The INTERCHANGE host program does the following:

1. reads and displays the entire Car TAG using typed read
2. writes a 0x99 to only the color element of the Car TAG using typed write
3. reads and displays the entire Car TAG using typed read
4. writes a 0x88 to the color element and increments the year element of the Car TAG using typed write
5. reads and displays the entire Car TAG using typed read
6. reads and displays the entire Car TAG using word range read

Note: the display routine takes the four bytes of the unsigned variable and places them in a temporary union variable before storing them. This - or another similar method - is necessary when the host requires that data larger than a byte be on even-address boundaries but the data for those variables in the byte stream are on odd-address boundaries.

The following is an explanation of the PCCC Typed Write packet to write only the color and serial elements to the CAR TAG.

0x12	DST - copro station address, since the TAG starts with "X_" this is don't care
0x05	CTRL - packet type must be 5 for Interchange
0x00	SRC - Source station filled in by Network Interface
0x00	LSAP - Set to 0 for local network
0x0F	CMD - command for typed write
0x00	STS - status byte
0x02 0x00	TNSW - L/H Transaction status word (used for command/reply matching)
0x67	FNC - typed write function
0x06 0x00	OFF - Offset L/H to requested data 6 bytes, 6 is the offset to the color element
0x06 0x00	TT - Total transaction L/H 6 items, 1 byte color, 1 byte pad, 4 bytes serial
0x00 X_Car 0x00	Symbolic address (TAG)
0x99	1st 9 - Type in next 1 byte, 2nd 9 - size in following 1 byte
0x09	09 - Type is array
0x08	Array is 8 bytes (2 bytes for typing information 6 bytes of data)
0x91	9 - Type in next 1 byte, size is 1 byte
0x22	Type is Undefined type
0x88 0xFF 0 0 0 0	Data to be transmitted

The following is an explanation of the PCCC Typed Read packet and the response from the coprocessor.

0x12	DST - copro station address, since the TAG starts with "X_" this is don't care
0x05	CTRL - packet type must be 5 for Interchange
0x00	SRC - Source station filled in by Network Interface
0x00	LSAP - Set to 0 for local network
0x0F	CMD - command for typed read
0x00	STS - status byte
0x03 0x00	TNSW - L/H Transaction status word (used for command/reply matching)
0x68	FNC - typed read function
0x00 0x00	OFF - Offset L/H to requested data 0 bytes, read from start of TAG
0x00 X_Car 0x00	Symbolic address (TAG)
0x0C, 0x00	Size L/H same as TT 12 items

The response from the coprocessor might be the following.

0x00	DST - DST/SRC Swapped
0x05	CTRL - packet type must be 5 for Interchange
0x12	SRC - DST/SRC Swapped
0x00	LSAP - Set to 0 for local network
0x4F	CMD - reply to command for typed read
0x00	STS - status byte - no error
0x03 0x00	TNSW - L/H Transaction status word
0x9A	9 - Type in next 1 byte, A - size in following 2 bytes
0x09	09 Type is array
0x0E 0x00	Size is 14 bytes (L/H) (2 bytes for typing information 12 bytes of data)
0x91	Type is next 1 byte, size is 1 byte
0x22	Type is Undefined type
0x00 0x00 0x03 0xA6	Make H/L (934)
0x4C	Model (76)
0xF2	Type (242)
0x88	Color (0x88)
0xFF	PAD
0x00 0x00 0x00 0x07	Serial (7)

The following is an explanation of the PCCC Word Range Read packet and the response from the coprocessor.

0x12	DST - copro station address, since the TAG starts with "X_" this is don't care
0x05	CTRL - packet type must be 5 for Interchange
0x00	SRC - Source station filled in by Network Interface
0x00	LSAP - Set to 0 for local network
0x0F	CMD - command for typed read
0x00	STS - status byte
0x04 0x00	TNSW - L/H Transaction status word
0x01	FNC - word range read function
0x00 0x00	OFF - Offset L/H to requested data 0 bytes, read from start of TAG
0x06 0x00	TT - Total transaction L/H 6 items (words)
0x00 X_Car 0x00	Symbolic address (TAG)
0x0C	Size in bytes (12)[1]

[1]The size parameter can be expanded for large reads by preceding the Size parameter with an 0xFF followed by the size L/H. To read 842 bytes (0x34A) 0xFF 0x4A 0x03

The response from the coprocessor might be the following.

0x00	DST - DST/SRC Swapped
0x05	CTRL - packet type must be 5 for Interchange
0x12	SRC - DST/SRC Swapped
0x00	LSAP - Set to 0 for local network
0x4F	CMD - reply to command for word range read
0x00	STS - status byte - no error
0x04 0x00	TNSW - L/H Transaction status word
0x00 0x00 0xA6 0x03	Make High word swapped, Low word swapped (934)
0xF2	Type (242) Type and Model swapped
0x4C	Model (76)
0xFF	PAD - Pad and Color swapped
0x88	Color (0x88)
0x00 0x00 0x07 0x00	Serial High word swapped, Low word swapped

```
/* This is an INTERCHANGE host example program (INTAGHOST.C) used in
   conjunction with the coprocessor example program intertag.c to
   demonstrate the use of the INTERD daemon.  This file is run on a host
   computer using INTERCHANGE after the file intertag is started on the
  Coprocessor */

#include "dtl.h"

#define HOSTNAME "your copro host name"
#define NI_ID 1

unsigned char pccc_color[] = {
  0x12,       /* DST  - copro station address */
  0x05,       /* CTRL - packet type must be 5 for Interchange */
  0x00,       /* SRC  - Source station filled in by NI */
  0x00,       /* LSAP - Set to 0 for local network */
  0x0F,       /* CMD  - command for typed write */
  0x00,       /* STS  - status byte */
  0x01, 0x00, /* TNSW - L/H Transaction status word */
  0x67,       /* FNC  - typed write function */
  0x06, 0x00, /* OFF  - Offset L/H to requested data 6 bytes */
  0x01, 0x00, /* TT   - Total transaction L/H 1 item */
  0x00, 'X','_','C','a','r',0x00, /* Symbolic address (TAG) */
  0x91,       /* Type in next byte, size of 1 byte */
  0x22,       /* Undefined type */
  0x99};      /* Data to be transmitted */

unsigned char pccc_col_ser[] = {
  0x12,       /* DST  - copro station address */
  0x05,       /* CTRL - packet type must be 5 for Interchange */
  0x00,       /* SRC  - Source station filled in by NI */
  0x00,       /* LSAP - Set to 0 for local network */
  0x0F,       /* CMD  - command for typed write */
  0x00,       /* STS  - status byte */
  0x02, 0x00, /* TNSW - L/H Transaction status word */
  0x67,       /* FNC  - typed write function */
  0x06, 0x00, /* OFF  - Offset L/H to requested data 6 bytes */
  0x06, 0x00, /* TT   - Total transaction L/H 6 items */
  0x00, 'X','_','C','a','r',0x00, /* Symbolic address (TAG) */
  0x99,       /* Type in next byte, size in following byte */
  0x09,       /* Type is array */
  0x08,       /* of 8 bytes */
  0x91,       /* Type in next byte, size of 1 byte */
  0x22,       /* Undefined type */
  0x88, 0xFF, 0x0, 0x0, 0x0, 0x0};  /* Data to be transmitted */

unsigned char pccc_tyread[] = {
  0x12,       /* DST  - copro station address */
```

```
  0x05,       /* CTRL - packet type must be 5 for Interchange */
  0x00,       /* SRC  - Source station filled in by NI */
  0x00,       /* LSAP - Set to 0 for local network */
  0x0F,       /* CMD  - command for typed read */
  0x00,       /* STS  - status byte */
  0x03, 0x00, /* TNSW - L/H Transaction status word */
  0x68,       /* FNC  - typed read function */
  0x00, 0x00, /* OFF  - Offset L/H to requested data 0 bytes */
  0x0C, 0x00, /* TT   - Total transaction L/H 12 items */
  0x00, 'X','_','C','a','r',0x00, /* Symbolic address (TAG) */
  0x0C, 0x00};/* SIZ  - Size L/H same as TT 12 items */

unsigned char pccc_wwread[] = {
  0x00,       /* DST  - copro station address */
  0x05,       /* CTRL - packet type must be 5 for Interchange */
  0x00,       /* SRC  - Source station filled in by NI */
  0x00,       /* LSAP - Set to 0 for local network */
  0x0f,       /* CMD  - command for word range read */
  0x00,       /* STS  - status byte */
  0x03, 0x00, /* TNSW - L/H Transaction status word */
  0x01,       /* FNC  - word range read function */
  0x00, 0x00, /* OFF  - Offset L/H to requested data 0 bytes */
  0x06, 0x00, /* TT   - Total transaction L/H 6 items (words) */
  0x00, 'X','_','C','a','r',0x00, /* Symbolic address (TAG) */
  0x0C}       ;/* SIZ  - Size in bytes 12 */

 union
  {
   unsigned tmp;
   unsigned char c[4];
  }u;

 unsigned make, serial;
 unsigned char model, type, color;
 unsigned char pccc_rpl[50];
 unsigned long rpl_siz;        /* size of pccc reply */

 int main( int argc, char** argv )

{
  unsigned long iostat,status;      /* function completion value */
  DTSA_BKPLN addr;              /* structured address */

  status = DTL_INIT( 1 );  /* Initialize the Data Table Library */

  addr.atype = DTSA_TYP_BKPLN;/* set up plc address */
  addr.ni_id = NI_ID;
  addr.module = 0;
  addr.pushwheel = 0;
  addr.channel = 0;

  status = DTL_C_CONNECT( NI_ID, HOSTNAME, 0);  /* Connect */

  rpl_siz = sizeof (pccc_rpl);  /* size of response buffer */

  status = DTL_PCCC_DIRECT_W ((DTSA_TYPE *) &addr, pccc_tyread, sizeof
     (pccc_tyread),
    pccc_rpl, &rpl_siz, DTL_TYP_RAW, DTL_TYP_RAW, &iostat, 60000);  /* do
     typed read */

  display_ty();  /* show the result of the read */

   rpl_siz = sizeof (pccc_rpl);  /* size of response buffer */

  status = DTL_PCCC_DIRECT_W ((DTSA_TYPE *) &addr, pccc_color, sizeof
    (pccc_color),
     pccc_rpl, &rpl_siz, DTL_TYP_RAW, DTL_TYP_RAW, &iostat, 60000);  /* do
    typed write to color 0x99 */

  rpl_siz = sizeof (pccc_rpl);  /* size of response buffer */

  status = DTL_PCCC_DIRECT_W ((DTSA_TYPE *) &addr, pccc_tyread, sizeof
    (pccc_tyread),
     pccc_rpl, &rpl_siz, DTL_TYP_RAW, DTL_TYP_RAW, &iostat, 60000);  /*
    do typed read */

  display_ty();  /* show the result of the read - note color = 0x99*/

  rpl_siz = sizeof (pccc_rpl);  /* size of response buffer */

  u.tmp = serial + 1;           /* increment serial in tmp */
```

```
 pccc_col_ser[25] = u.c[0];  /* transfer to PCCC packet */
 pccc_col_ser[26] = u.c[1];
 pccc_col_ser[27] = u.c[2];
 pccc_col_ser[28] = u.c[3];

 status = DTL_PCCC_DIRECT_W ((DTSA_TYPE *) &addr, pccc_col_ser, sizeof
(pccc_col_ser),
 pccc_rpl, &rpl_siz, DTL_TYP_RAW, DTL_TYP_RAW, &iostat, 60000); /*
write to color and serial */

 rpl_siz = sizeof (pccc_rpl);  /* size of response buffer */

 status = DTL_PCCC_DIRECT_W ((DTSA_TYPE *) &addr, pccc_tyread, sizeof
(pccc_tyread),
 pccc_rpl, &rpl_siz, DTL_TYP_RAW, DTL_TYP_RAW, &iostat, 60000);  /* do
typed read */

 display_ty();  /* show the result of the read - note color = 0x99 serial
incremented*/

 rpl_siz = sizeof (pccc_rpl);  /* size of response buffer */

 status = DTL_PCCC_DIRECT_W ((DTSA_TYPE *) &addr, pccc_wwread, sizeof
(pccc_wwread),
 pccc_rpl, &rpl_siz, DTL_TYP_RAW, DTL_TYP_RAW, &iostat, 60000);  /* do
typed read */

 display_ww();  /* show the result of the read - note color = 0x99 serial
incremented*/}

int display_ty ()
  {

/* since the pccc "byte stream" from the coprocessor might put unsigned
   variables on uneven address boundries we move them to a temporary union
  variable before storing them */

  u.c[0] = pccc_rpl[14];      /* get the make from the reply buffer */
  u.c[1] = pccc_rpl[15];      /* and put it in the temp buffer */
  u.c[2] = pccc_rpl[16];
  u.c[3] = pccc_rpl[17];
  make   = u.tmp;             /* store make in make variable */

  model  = pccc_rpl[18];      /* get model from reply buffer */

  type   = pccc_rpl[19];      /* also type */

  color  = pccc_rpl[20];      /* as well as color */

  u.c[0] = pccc_rpl[22];      /* get the serial from the reply buffer
(skip */
  u.c[1] = pccc_rpl[23];      /* over pad byte at offset [21]) and put it
*/
  u.c[2] = pccc_rpl[24];      /* in the temp buffer */
  u.c[3] = pccc_rpl[25];
  serial = u.tmp;             /* store serial in serial variable */

  printf ("make = %d model = %d type = %d color = %xH serial = %d\n",
           make,model,type,color,serial);  /* display the "Car" tag */ }

int display_ww ()
  {
/* since the words returned in the pccc packet from the coprocessor is in
   little endian format we must byte swap the return values */

  u.c[0] = pccc_rpl[9];      /* get the make from the reply buffer */
  u.c[1] = pccc_rpl[8];      /* and put it in the temp buffer */
  u.c[2] = pccc_rpl[11];
  u.c[3] = pccc_rpl[10];
  make   = u.tmp;             /* store make in make variable */

  model  = pccc_rpl[13];      /* get model from reply buffer */

  type   = pccc_rpl[12];      /* also type */

  color  = pccc_rpl[15];      /* as well as color */

  u.c[0] = pccc_rpl[17];      /* get the serial from the reply buffer (skip */
  u.c[1] = pccc_rpl[16];      /* over pad byte at offset [14]) and put it */
  u.c[2] = pccc_rpl[19];      /* in the temp buffer */
```

```
u.c[3] = pccc_rpl[18];
serial = u.tmp;             /* store serial in serial variable */
printf ("make = %d model = %d type = %d color = %xH serial = %d\n",
         make,model,type,color,serial);  /* display the "Car" tag */ }
```

Using the SNMPD Daemon

SNMPD is a daemon that provides Simple Network Management Protocol (SNMP) services between the Control Coprocessor and a host computer. This daemon supports MIB-1 variables. After installing the SNMPD daemon on the Control Coprocessor, you have the ability to:

- allow 6200 Series PLC-5 Programming Software to identify the coprocessor on the Ethernet network using the “WHO” function
- monitor MIB-1 variables from a host computer running the SNMP-monitoring software

Reach us now at www.rockwellautomation.com

Wherever you need us, Rockwell Automation brings together leading brands in industrial automation including Allen-Bradley controls, Reliance Electric power transmission products, Dodge mechanical power transmission components, and Rockwell Software. Rockwell Automation's unique, flexible approach to helping customers achieve a competitive advantage is supported by thousands of authorized partners, distributors and system integrators around the world.

Americas Headquarters, 1201 South Second Street, Milwaukee, WI 53204, USA, Tel: (1) 414 382-2000, Fax: (1) 414 382-4444
European Headquarters SA/NV, avenue Herrmann Debroux, 46, 1160 Brussels, Belgium, Tel: (32) 2 663 06 00, Fax: (32) 2 663 06 40
Asia Pacific Headquarters, 27/F Citicorp Centre, 18 Whitfield Road, Causeway Bay, Hong Kong, Tel: (852) 2887 4788, Fax: (852) 2508 1846

Publication 1771-RN951C-EN-P - July 2001
Supersedes Publication 1771-6.5.95-RN1 September 1999

PN 957555-94