Classic I/O Interface for CRISP/32

User's Guide

CRISP *Software Products*

SQUARE D
GROUPE SCHNEIDER

Classic I/O Interface for CRISP/32
User's Guide
Document number: 500 075 - 001, Rev. 1

Document History

Revision	Date	Pages affected/Description of change
1	2/94	Initial Release. ECN 4375

Table of Contents

Notes:

Introduction

General

Beginning with CRISP®/32 Version 2.8, limited support for CRISP® Classic I/O is provided. This functionality allows CRISP Automation to offer a computer upgrade to customers having this traditional equipment without requiring the customer to replace the entire I/O system.

The original systems using Classic I/O are based on one of several models of the PDP-11 computer.

Supported Classic I/O Functions

The Classic I/O system for CRISP/32 supports the following.

- Digital inputs using CR486 panels
- Digital outputs using CR194 or CR587 panels
- Analog inputs using CR786 panels
- Analog outputs using CR587 panels
- Pulse inputs using CR486 panels in the 8-bit or 4-bit counter mode.

Unsupported Classic I/O Functions

- Analog inputs using CR188 panels
- Pulse outputs (CR272 cards)
- Momentary pushbutton latching including the normally-closed option.
- Motor start staggering.
- Smoothing of analog input values
- Systems without the ICC panel (CR774).
- Absolute guarantee of the 100 ms scanning rate in the presence of general data processing.
- The device CSR and Vectors are fixed, not configurable.

NOTE

In order to define the Classic I/O data the CRISP system must be configured for Classic I/O. In addition the application logics require the CIO_READ and CIO_WRITE logic statements.

Example

```
$ @crisp_config

      +----------------------------------------+
      | CRISP Software Configuration Procedure |
      +----------------------------------------+
      | Copyright 1987-93, Square D Company    |
      | All rights reserved                    |
      +----------------------------------------+

%C32CONFIG-I-CPUTYP, Configuring CRISP on a VAXstation 4000-60

Is the target system a dual CPU . . . . . [NO]?
Enter the system output device  . . . . [OPA0:]: TTA2:
Enter the secondary output device . . . [NLA0:]:
Does this system have I/Onyx  . . . . . [NO]? NO

            Classic I/O was used with CRISP/16 systems running on PDP-11
            computers.  This consists of a set of open-rack panels connected
            by three 40-wire ribbon cables terminated in the computer.  These
            CRISP/16 I/O systems may be used on CRISP systems with restrictions.
            Hardware requirements are (2) DRV11 parallel interfaces, a KWV11
            clock, and a CRISP ICC panel.

Does this system have Classic I/O . . . . [NO]? YES
What should the CIO bus delay in microseconds be {20}?

            The IEEE 802 SAP for the Database Access Server (DBASRV) must be
            the same for all nodes in a network.  How the SAP is specified has
            changed from earlier versions of CRISP.  The following answers
            produce identical SAPs across different versions of CRISP:

               SAP  4   V2.3
               SAP  8   V2.4, V2.4A, V2.4B
               SAP 12   V2.5 and later

What should the SAP be on this node [12]?

            The number of Database Access Server (DBASRV) read-ahead buffers
            should be one greater than the number of clients (i.e., workstations
            or trend processes) that will communicate with this system via
            Ethernet simultaneously.  Note that a workstation with an annunciator
            panel counts as two clients.

How many read-ahead buffers should be allocated [4]?

            The CRISP Access Server (CASRV) SAP must be the same for all CRISP
            nodes on the network.

            Note:  The SAP used must not be the same as that specified for
                   the Database Access Server (DBASRV).

What should the SAP be on this node [4]?
How many requests can be pending simultaneously [32]?
How many requests can be processed simultaneously [5]?
At what priority should requests be executed [4]?
```

Example (cont)

```
            The CRISP Access Server (CASRV) may be used by CRISP PC Workstations
            (PCWS) to access a CWS display file stored on the VAX (remote screen
            mode).  The next question configures the file specification for that
            display file.  It has no effect on the Color Workstation (CWS).

What is the PCWS display filespec . . [CRISP$CWS:USER.CWS]?

            The "CRISPconnect Server for NetDDE" may be used to access CRISP
            Real-Time data by Microsoft Windows applications running on PCs
            that are DDE aware.

            To successfully run, this product requires its license to be
            registered and loaded and also requires the installation of
            "NetDDE for VMS" from Wonderware Software Development Corporation.

            If not yet installed, the product license and the Wonderware
            product can be installed after the CRISP configuration procedure
            is complete.

Do you want the CRISPconnect Server for NetDDE . . . [NO]? YES
Enter directory of "NetDDE for VMS" product [SYS$SYSDEVICE:[NETDDE]]:
At what priority should NetDDE Server requests be executed [20]?

            The message throughput of NetDDE can be boosted by configuring the
            NetDDE Server to send a burst of messages that do not require client
            acknowledgement.  After a burst of "n" messages, the "nth" message
            sent requires client acknowledgement, thereby ensuring that the
            message processsing in the PC client catches up with the NetDDE Server.

Enter max messages to send before requiring client acknowledgement [20]?

            The "CRISPconnect Server for @aGlance/IT" may be used to access
            CRISP Real-Time data and CRISP Historical data by client applications
            running on other nodes and platforms that are @aGlance/IT compliant.

            To successfully run, this product requires its license to be
            registered and loaded and also requires the installation of two
            other products, which are 1) @aGlance/IT and 2) DEC ACA Services.

            If not yet installed, the product license and the other products can
            be installed after the CRISP configuration procedure is complete.

Do you want the CRISPconnect Server for @aGlance/IT . . . [NO]? YES

            The @aGlance Server (CSRAAG) to access "real-time" data must be
            configured for the maximum number of @aGlance client applications
            that will concurrently establish sessions to access "real-time" data.

Enter the maximum number of CSRAAG concurrent client sessions [32]?
At what priority should CSRAAG requests be executed [20]?

            The @aGlance Server (CSHAAG) to access "historical" data must be
            configured for the maximum number of @aGlance client applications
            that will concurrently establish sessions to access "historical" data.
```

Example (cont)

```
Enter the maximum number of CSHAAG concurrent client sessions [32]?
At what priority should CSHAAG requests be executed [15]?
How many requests will CSHAAG clients issue simultaneously [1]?

How many arbiters do you have . . . . . [1]? 0

            Two methods of performing database transfers are available -- The CRISP
            Logic Executive (CLE) and the Inter-Database Communications facilities.
            The older CLE method is limited to transfers within the same CPU.  IDC
            is new with V2.7 and can perform transfers to both local and remote
            databases.  See the V2.7 release notes for more information.  In the
            next question you can select to run the IDC process or not.  You should
            answer no if you are not performing database transfers or if you are
            only performing transfers between local databases.

Do you want the IDC process . . . . . [NO]? YES
At what priority should IDC execute . [19]?

            The CRISP Historian System consists of 3 processes for collecting
            and storing historical process data.

Do you want to enable the Historian System [NO]? YES
    Enter historian disk name [CRISP$DEVICE:]:
    Enter historian initialization filespec [CRISP$CFG:USER.HST]:
    Enter historian logger qualifiers  . . . []: /INTERVAL=60
    Enter historian separator qualifiers . . []: /WAKE=HOURS=1
```

Example (cont)

```
            To reduce file fragmentation, each point file will be created at a
            specific size.  Whenever a new version of a point file is created,
            the old file will be truncated to actual size.  Users with limited
            disk space may want to specifiy a smaller value, while users who
            are collecting large amounts of data may want a larger value.

    Enter historian allocation size . . . . [100]:

Creating CRISP$CFG:CONFIG_RESULTS.COM

            The CRISP Basic Workstation displays a CRISP prompt on startup and
            has no pixel graphics capability.  The CRISP Color Workstation
            (CWS) displays a menu on startup and has pixel graphics support.

CRISP Basic Workstation files were not installed

Writing CRT portion of CONFIG_RESULTS.COM

            No CRISP/32 Workstations have been configured.  If you are using
            networked CRISP Workstations, you may request that the trend process
            be executed to collect trend data.

Do you want the trend process . . [NO]? YES
Enter trend control file spec [CRISP$CFG:USER.TRC]:

Writing CWS portion of CONFIG_RESULTS.COM
Writing CRISP$CFG:EXECUTE_CWSTND.COM
Completing CONFIG_RESULTS.COM

%C32CONFIG-I-PATCH, Patching configuration into CRISP system database

            Patching the CRISP configuration
  PATCH  Version 5-05    20-June-1991
%PATCH-I-NOLCL, image does not contain local symbols
old:    CONFIG_B_DUALCPU:  00
new:    CONFIG_B_DUALCPU:  00
old:    CONFIG_B_CLE_DBT:  00
new:    CONFIG_B_CLE_DBT:  00
old:    DBASRV_W_BASE_SAP:  12
new:    DBASRV_W_BASE_SAP:  12
old:    DBASRV_W_READAHEAD:  4
new:    DBASRV_W_READAHEAD:  4
old:    CASRV_W_SAP:  4
new:    CASRV_W_SAP:  4
old:    CASRV_W_HISTORIAN_PRIORITY:  4
new:    CASRV_W_HISTORIAN_PRIORITY:  4
old:    CASRV_W_QSIZE:  32
new:    CASRV_W_QSIZE:  32
old:    IONYX_B_IN_SYSTEM:  00
new:    IONYX_B_IN_SYSTEM:  01
old:    CIO_B_IN_SYSTEM:  00
new:    CIO_B_IN_SYSTEM:  00
old:    ICC_B_ARB_PATH_ENABLED:  00
new:    ICC_B_ARB_PATH_ENABLED:  00
old:    ICC_B_DBASRV_PATH_ENABLED:  00
new:    ICC_B_DBASRV_PATH_ENABLED:  00
%PATCH-I-WRTFIL, updating image file DISK$USER:[CRISP.EXE]CRISP.EXE;1
```

Example (cont)

```
%C32CONFIG-S-DONE, The CRISP Configuration procedure is finished

$
```

CPU

Any VAX computer having a Q-BUS interface in the 4000-class or above. Typically a 4000-100 or 4000-400. The Classic I/O will run in the single or dual-computer configuration. If the system is dual, then CRISP Arbitor hardware is required to control the switchovers.

Q-BUS Interfaces

The Classic I/O requires two DRV11, 16-bit parallel interface cards and one KWV11 real-time clock card. The DRV11s are connected to the Classic I/O hardware in the same manner that the original PDP-11s were connected. The real-time clock is used to control the Classic I/O bus timing.

The DRV11 connected to ABUS must be addressed at 167770 (octal) and the DRV11 connected to DBUS must be addressed at 167760 (octal). The KWV11 must have the CSR at 770420 (octal) and vector at 440 (octal).

Classic I/O Hardware

The hardware beyond the ICC panel must conform to the rules for cable length and card loading that have always applied to Classic I/O.

Any of the Classic I/O sumulator panels may be in the configuration.

Notes:

VMS Requirements

Classic I/O requires the use of the VMS service "connect to vector". In order to use this facility the VMS sysgen parameter REALTIME_SPTS must be set non-zero. This value is the number of memory pages that will be used for the interrupt code and buffers. A value of 10 will be satisfactory for almost every system.

The best way to change this value is to edit SYS$SYSTEM:MODPARAMS.DAT and add the line `REALTIME_SPTS = 10`. Then perform an autogen (ex., `SYS$UPDATE:AUTOGEN.COM`). Refer to the appropriate installation / operations guide for more information on the autogen command procedure.

Configure CRISP

Configure CRISP/32 to include Classic I/O support by answering the Classic I/O question in CRISP_CONFIG with "YES". The default bus delay value of 20 microseconds will be satisfactory on almost every system. Now when CRISP/32 starts, the Classic I/O program CIO will be started and monitored with CRMON. Also CRISP_SETUP will now install the VMS "connect to interrupt" driver CONINTERR for use with the Classic I/O program.

If the DRV11 interfaces or the KWV11 real-time clock are not present at the specified addresses, the CIO program will complain and exit.

Logic Calls

The following CIO_READ logic call must be inserted at the beginning of the CRISP/32 source code in order to read inputs from the Classic I/O system after the RESTART; statement.

```
RECALL; CIO_READ, TRUE
```

This call has no arguments and causes the input lists defined in file "CRISP$CFG:dbname_CIO_DEF.DAT" to be filled from the physical input values.

The following CIO_WRITE logic call must be inserted at the end of the CRISP/32 source code in order to write outputs to the Classic I/O system. The statement should be located just above the END; statement.

```
RECALL; CIO_WRITE, ,DO_OUTPUT
```

This call causes the output lists defined in the file "CRISP$CFG: dbname_CIO_DEF.DAT" to be copied to the tables for physical output. The argument DO_OUTPUT is a CRISP LOGICAL variable and when FALSE, inhibits all physical output to the Classic I/O hardware.

Internal Operation

The first time that either of the Classic I/O calls is executed, the Classic I/O definition file CRISP$CFG:dbname_CIO_DEF.DAT is opened and read in. The specified symbols are looked up and a global section named "dbname_CIO.GBL" is created and filled with values necessary to perform the physical I/O and buffer the results. This section is then submitted to the CIO program for actual physical I/O.

On subsequent calls to CIO_READ, buffered input data in the section from the most recent physical I/O scan is copied to the CRISP database.

Internal Operation (cont)

On subsequent calls to CIO_WRITE, output data from the CRISP database is copied to the buffers in the section for physical output on the next scan.

If the Logic is stopped, a stop flag is set in the section and the CIO program discontinues I/O for that section and drops it.

The CIO program executes each 100 Ms and scans a list of sections upon which to perform physical I/O. Each section represents a CRISP/32 logic which has made calls to CIO_READ and CIO_WRITE. For each of these sections, an I/O scan for input is performed and if the machine is ACTIVE and the DO_OUTPUT bit is on, an I/O scan for output is performed.

The CIO program and the logic calls are interlocked by a VMS lock such that both routines will not be accessing the section at the same time.

File Name

Each logic that requests Classic I/O requires a separate definition file. The Classic I/O definition file is named dbname_CIO_DEF.DAT and is assumed to be located in CRISP$CFG:. "dbname" is the name of the database and associated logic requiring Classic I/O.

The user enters a text editor to create the Classic I/O definition file.

Comment Lines

On any line of the file all text following a " (double quote) or ! (exclamation point) is ignored. Comments cannot be embedded within the body of a single line as everything past the first comment delimiter will be ignored.

Continuation Lines

There is no need for continuation lines since the definition context is broken by another keyword or the end of file.

Keyword Lines

Keyword lines are denoted by starting with a legal keyword terminated in a ; (semicolon). Additional arguments then follow the semicolon separated by commas. Missing arguments are denoted by commas separated by space characters or no characters.

Symbol Definitions

CRISP/32 symbols are specified as string of up to 31 characters and may be subscripted with a constant subscript. The database associated with any symbols is the one from the calling logic.

Logical Variables

CRISP/32 logical variables are used with Classic I/O digital input and output points. Note that in CRISP/32, LOGICAL-type variables are declared modulo 16, that is each block of logical variables between LOGICAL; keywords will be rounded out to a multiple of 16 variables with 0 to 15 unnamed variables. These unnamed variables will be used as Classic I/O points if present in a Classic I/O block.

I/O Blocks

Each type of Classic I/O is represented by an I/O block. There may be only one instance of each type of I/O block in a definition file. If more than one is specified, only the last one will be built. The types of I/O implemented are as follows.

- DIGITAL INPUT
- DIGITAL OUTPUT
- ANALOG INPUT (CR774 version)
- ANALOG OUTPUT
- PULSE INPUT (8-bit or 4-bit counters)
- WORD INPUT

I/O Block Keywords

The following keyword lines define the CRISP/32 to CRISP Classic I/O blocks.

- `INPUT; mux_addr, start_symbol, num_points`

  This defines a digital input point block, beginning at the specified multiplexor address and specified symbol for the specified number of points. The points specified in this context are assumed to be contiguous bits from the Classic I/O input cards. Each successive point from the I/O system is placed into the CRISP/32 variable area starting at the specified start_symbol for num_points variables. The first point is bit 15 of the first input card. The last point is determined by the number of points.

  The multiplexor address is in OCTAL to be compatible with Classic I/O definitions and setups. If the multiplexor address is missing the default is 1000 (octal).

  Symbols must be CRISP/32 LOGICAL type variables. 16 contiguous variables represent a single Classic I/O digital input card. If the specified number of points causes the CRISP/32 LOGICAL table to be overflowed, the number of points is truncated and a message is sent to CRISP$TT:.

  Note that there is no checking that other databases are performing I/O functions in the same multiplexor address space or that the end of the available multiplexor addresses has been reached.

- `OUTPUT; mux_addr, start_symbol, num_points`

  This defines a digital output point block, beginning at the specified multiplexor address and specified symbol for the specified number of points. The points specified in this context are assumed to be contiguous bits to the Classic I/O output cards. Each successive variable value from the specified start_symbol is placed into the I/O digital output cards starting with bit 15 of the first output card. The last point is determined by the number of points.

  The multiplexor address is in OCTAL to be compatible with Classic I/O definitions and setups. If the multiplexor address is missing the default is 40 (octal).

  Symbols must be CRISP/32 LOGICAL type variables. 16 contiguous variables represent a single Classic I/O output card. If the specified number of points causes the CRISP/32 LOGICAL table to be overflowed, the number of points is truncated and a message is sent to CRISP$TT:.

  Note that there is no checking that other databases are performing I/O functions in the same multiplexor address space or that the end of the available multiplexor addresses has been reached.

I/O Block Keywords (cont)

- `ANAIN; mux_addr, start_symbol, num_points`

  This defines an analog input point block, beginning at the specified multiplexor address and specified symbol for the specified number of points. The points specified in this context are assumed to be contiguous words from the Classic I/O analog input cards. Each successive point from the I/O system is placed into the CRISP/32 variable area starting at the specified start_symbol for num_points variables. The first point is the first analog input card. The last point is determined by the number of points.

  The 12-bit analog value from the I/O system is placed into the low 12 bits of the corresponding CRISP/32 variable, the remaining bits are set to zero.

  The multiplexor address is in OCTAL to be compatible with Classic I/O definitions and setups. If the multiplexor address is missing the default is 5000 (octal).

  Symbols must be CRISP/32 NUMERIC or LONGWORD type variables. Each variable represents a single Classic I/O analog input point. If the specified number of points causes the CRISP/32 NUMERIC or LONGWORD table to be overflowed, the number of points is truncated and a message is sent to CRISP$TT:.

  Note that there is no checking that other databases are performing I/O functions in the same multiplexor address space or that the end of the available multiplexor addresses has been reached.

- `ANAOUT; mux_addr, start_symbol, num_points`

  This defines an analog output point block, beginning at the specified multiplexor address and specified symbol for the specified number of points. The points specified in this context are assumed to be contiguous words to the Classic I/O analog output cards. Each successive variable from the CRISP/32 tables is placed into the I/O system starting at the specified start_symbol for num_points variables. The first point goes to the first analog output card. The last point is determined by the number of points.

  The low 12-bits from the CRISP/32 variable is placed into the I/O system analog output cards. The remaining bits in the CRISP/32 variable are ignored.

  The multiplexor address is in OCTAL to be compatible with Classic I/O definitions and setups. If the multiplexor address is missing the default is 4000 (octal).

  Symbols must be CRISP/32 NUMERIC or LONGWORD type variables. Each variable represents a single Classic I/O analog input point. If the specified number of points causes the NUMERIC or LONGWORD table to be overflowed, the number of points is truncated and a message is sent to CRISP$TT:.

I/O Block Keywords (cont)

Note that there is no checking that other databases are performing I/O functions in the same multiplexor address space or that the end of the available multiplexor addresses has been reached.

- `PLSIN; mux_addr, start_symbol, num_points`

  This defines a pulse input point block where the pulse input hardware consists of the 8-bit counter cards, beginning at the specified multiplexor address and specified symbol for the specified number of points. The points specified in this context are assumed to be contiguous words from the Classic I/O pulse input cards. Each successive point from the I/O system is placed into the CRISP/32 variable area starting at the specified start_symbol for num_points variables. The first point is the first counter of the first pulse input input card. The last point is determined by the number of points.

  The unsigned 8-bit counter value is read from the I/O system and added to the proper CRISP/32 variable. The counter is then reset. This occurs in such a way that any counts received during this process are held in abeyance until the counter is reset.

  The multiplexor address is in OCTAL to be compatible with Classic I/O definitions and setups. If the multiplexor address is missing the default is 6000 (octal).

  Symbols must be CRISP/32 NUMERIC or LONGWORD type variables. Each variable represents a one half of a single Classic I/O 8-bit pulse input point. If the specified number of points causes the NUMERIC or LONGWORD table to be overflowed, the number of points is truncated and a message is sent to CRISP$TT:.

  Note that there is no checking that other databases are performing I/O functions in the same multiplexor address space or that the end of the available multiplexor addresses has been reached.

- `PLSIN4; mux_addr, start_symbol, num_points`

  This defines a pulse input point block where the pulse input hardware consists of the 4-bit counter cards, beginning at the specified multiplexor address and specified symbol for the specified number of points. The points specified in this context are assumed to be contiguous words from the Classic I/O pulse input cards. Each successive point from the I/O system is placed into the CRISP/32 variable area starting at the specified start_symbol for num_points variables. The first point is the first counter of the first pulse input input card. The last point is determined by the number of points.

  The unsigned 4-bit counter value is read from the I/O system and added to the proper CRISP/32 variable. The counter is then reset. This occurs in such a way that any counts received during this process are held in abeyance until the counter is reset.

I/O Block Keywords (cont)

The multiplexor address is in OCTAL to be compatible with Classic I/O definitions and setups. If the multiplexor address is missing the default is 6000 (octal).

Symbols must be CRISP/32 NUMERIC or LONGWORD type variables. Each variable represents a one quarter of a single Classic I/O 4-bit pulse input point. If the specified number of points causes the NUMERIC or LONGWORD table to be overflowed, the number of points is truncated and a message is sent to CRISP$TT:.

Note that there is no checking that other databases are performing I/O functions in the same multiplexor address space or that the end of the available multiplexor addresses has been reached.

Notes:

General

The program CIOCRM provides limited functionality to perform testing of the Classic I/O system. This program will construct a global section suitable for submission to CIO from typed-in specifications.

⚠ CAUTION

The maintenance program produces output patterns that will almost surely result in serious consequences in a real plant situation. It is primarily intended to use during checkout with simulator panels. All field wiring output power should be removed in a real plant before exercising CIOCRM.

CIOCRM is run with CRISP/32 configured with Classic I/O and with no logics running with calls to the CIO logic calls. Start the program with the command:

```
RUN CRISP$EXE:CIOCRM
```

The user is asked for parameters for the classic I/O including number of points and starting Multiplexor address for each type of I/O.

Then one of 3 types of tests are available for selection:

0 - Run a "rotating worm" pattern on the digital outputs and log to CRISP$TT any changes in the digital inputs.

1 - Copy the digital inputs to the digital outputs.

2 - Copy the pulse inputs to the digital outputs.

NOTE: This program is preliminary and will be expended upon at a later time.

Notes:

Special Input Type

General

A special customer requirement is provided that reads 16-bits words from the Classic I/O bus and converts them to CRISP floating point values. The raw input consists of 16-bit quantities where the low 14 bits represents an integer value and the high two bits represent a scale factor as follows.

Bit Value	Decimal Multiplier
00	1
01	.1
10	.01
11	.001

The Classic I/O system takes the low 14 bits of value, converts it to floating point and multiplies the result times the proper multiplier based on the high 2 bits. The final destination is the next CRISP FLOAT variable in the specified array.

The following input specification in the Classic I/O Definition file connects the input hardware to the CRISP FLOAT variables.

- `BRKTMP; mux_addr, start_symbol, num_points`

  This defines a special 16-bit word input point block, beginning at the specified multiplexor address and specified symbol for the specified number of points. The words specified in this context are assumed to be contiguous words from the Classic I/O word input cards. Each successive word from the I/O system is converted to the correct floating-point value and placed into the CRISP/32 variable area starting at the specified start_symbol for num_points variables. The first word is the first word of the first input card. The last word is determined by the number of points.

  Each unsigned 16-bit word value is read from the I/O system, converted to floating-point and written to the proper CRISP/32 variable.

  The multiplexor address is in OCTAL to be compatible with Classic I/O definitions and setups. If the multiplexor address is missing the default is 2000 (octal).

  Symbols must be CRISP/32 FLOAT type variables. Each variable represents a single word of Classic I/O input. If the specified number of points causes the FLOAT table to be overflowed, the number of points is truncated and a message is sent to CRISP$TT:.

  Note that there is no checking that other databases are performing I/O functions in the same multiplexor address space or that the end of the available multiplexor addresses has been reached.

Notes: