CONTROL DATA CORPORATION
Energy Management Systems Division

DESIGN SPECIFICATION

NO.: ART D980
REV.: 0
DATE: 7/25/85

SUBJECT: ARTECS ADVANCED PRODUCTS Display Controller Interface

REFERENCES: See Section 2.0

PURPOSE:

ISSUED BY: D. McMullen

INTERNAL
APPROVED

TABLE OF CONTENTS

TABLE OF CONTENTS (Cont.)

1.0 INTRODUCTION

This design specification describes the Driver for the Display Controller Interface/40-815 CRT Controller. Many times, this Driver will be called simply the CRT Driver.

This Driver allows the Man-Machine subsystem to control a 40-815 CRT Controller connected to a Display Controller Interface (DCI) Card. Data input/output to a CRT unit on the 40-815 CRT Controller is controlled by the Driver. Other functional features of the CRT controller can also be controlled by the Driver.

The Driver processes status input, function output, and data read/write type requests for a specified 40-815 CRT Controller connected to the DCI card.

Unsocilited function key and lite pen interrupts are handled by the Driver. These interrupts occur as the result of some operator action at the keyboard associated with the 40-815 CRT Controller. The Man-Machine subsystem is notified of these interrupts when they occur so that it can respond to the operator request.

2.0 APPLICABLE DOCUMENTS

Control Data Network Architecture (CDNA) Executive External Reference Specification

Device Interface (DI) General Design Specification

Display Controller Interface (DCI) Engineering Specification

40-815 CRT Controller Engineering Specification

Motorolla M68000 Reference Manual

3.0 ENVIRONMENT

3.1 HARDWARE

The following hardware is required to run the DCI/40-815 CRT Controller Driver software:

a. A Device Interface (DI) Unit with its Internal System Bus (ISB)
b. A Master Processor Board (MPB) Card
c. A System Main Memory Module (SMM) Card
d. A Display Controller Interface (DCI) Card
e. A 40-815 CRT Controller Unit
f. A CRT Monitor/Keyboard Unit

Cards b through d must of course be installed in the DI and Units e and r must be connected up properly to the DCI Card.

3.1.1 MPB Features

The MPB is built around a Motorolla M68000 Microprocessor. Interrupt processing by the M68000 is fairly timeconsuming because 16 32-bit Data/Address Registers values must be saved in addition to the interrupted user's return address and status register whenever an interrupt is processed.

While it is true that the M68000 automatically saves the interrupted user's return address and status register values when an interrupt occurs and that the M68000 allows all the interrupted user's registers to be saved by the execution of a single instruction, these operations are still time consuming because of the number of memory accesses required. Ideally, the peripheral equipment used in the DI should be buffered I/O type devices because interrupt processing is not one of the M68000's strong points. The DCI/CRT Controller however isn't a buffered I/O type device.

The M68000 Microprocessor utilizes a Memory Mapped I/O scheme to communicate with its peripheral equipment. This This means the M68000 can read/write its peripheral equipment's internal working registers in the same way that it reads/writes memory. This can lead to problems because the M68000 can change such a device's registers when that device doesn't expect them to be changed. A write to a peripheral device's working registers is never rejected. This means the Driver must always ensure that the DCI is ready for a function command before it transfers a command to the DCI card. The Driver must also utilize status input requests to determine when and if command output to the DCI have been successfully processed.

3.1.2 ISB Features

The scheme devised by the Arden Hill's developemnt group to allow the MPB to talk to its peripheral cards must be used to communicate with the DCI. The details of this scheme are included in Appendix A of this design specification.

3.1.3 DCI/CRT Controller Features

The DCI/CRT Controller devices aren't buffered devices. This means the Driver must handle each eight-bit byte of data that is input/output to the CRT Controller. There are 256 different, distinct bytes (characters) that can be output to the 40-815 CRT Controller. A subset ($80 - $AF) of this 256 character set is treated by the CRT Controller as Function Control Characters. Most of the remaining characters in the set are treated as displayable data characters. Only displayable data characters can be read from the CRT Controller.

The CRT Monitor associated with the controller has a fixed number of fixed length lines (normally 48 lines of 72 character positions). The 40-815 Controller utilizes an internal memory to

keep track of the displayable data characters that have been transferred to it. Displayable data characters can be transferred to the CRT Controller by either the Driver utilizing the DCI or by an operator using the keyboard associated with the CRT Controller.

Each CRT Monitor (Screen) position has associated with it video display characteristics including color, inverse video mode, and blink mode. The CRT Controller's internal memory keeps track of the video display characteristics associated with each screen position. The video display characteristics associated with a screen position are the video display characteristics that were selected the last time that a displayable data character was transferred to that screen position. The currently selected video display characteristics can be changed either by outputting the appropriate function control characters to the CRT controller or by depressing the appropriate keys on the keyboard.

To keep track of where data characters transferred to the CRT controller will be displayed on the screen, a current position marker, called a CURSOR, is used. When a data character is transferred to the CRT, it is moved to the screen position indicated by the CURSOR and the CURSOR is moved one position ahead. Wrap-around occurs to the top of the screen when a character is output to the last screen position. The CRT Controller maintains X,Y coordinates to keep track of the CURSOR's current position. The Y ordinate indicates the line on which the CURSOR is currently located. The X ordinate indicates the column position where the CURSOR is located in that line. The CURSOR can be moved to a new position in a number of different ways. Function commands can be output to the controller to change either the X or Y ordinates. Control Control characters can be used to position the CURSOR relative to its current position. When the keyboard is enabled, the operator can position the CURSOR using the keyboard's CURSOR positioning keys and devices.

All things considered, the DCI/CRT Controllers are fairly complex devices to control. This has an adverse effect on the amount of error retry logic that can be included in the Driver.

The details of the procedures needed to control the DCI//CRT Controller units are included in Appendix A.

3.2 SOFTWARE

a. CDNA Executive

b. System's TIMER Task

c. Configuration Control Management Entity

d. Application tasks designed to process function key and litepen interrupts.

3.3 TERMINOLOGY AND NOTATION CONVENTIONS

3.3.1 Acronyms

CDNA -- Control Data Network Architecture
DI -- Device Interface Unit
ICB -- Internal Control Bus (Part of ISB)
ISB -- Internal System Bus
ITB -- Internal Transfer Bus (Part of ISB)
MPB -- Master Processor Board
SMM -- System Main Memory

3.3.2 Notation Conventions

Hexidecimal number are indicated by an "H" following the the number. For example, 6000H and 0BA4H are hexidecimal numbers.

4.0 SUBSYSTEM OVERVIEW

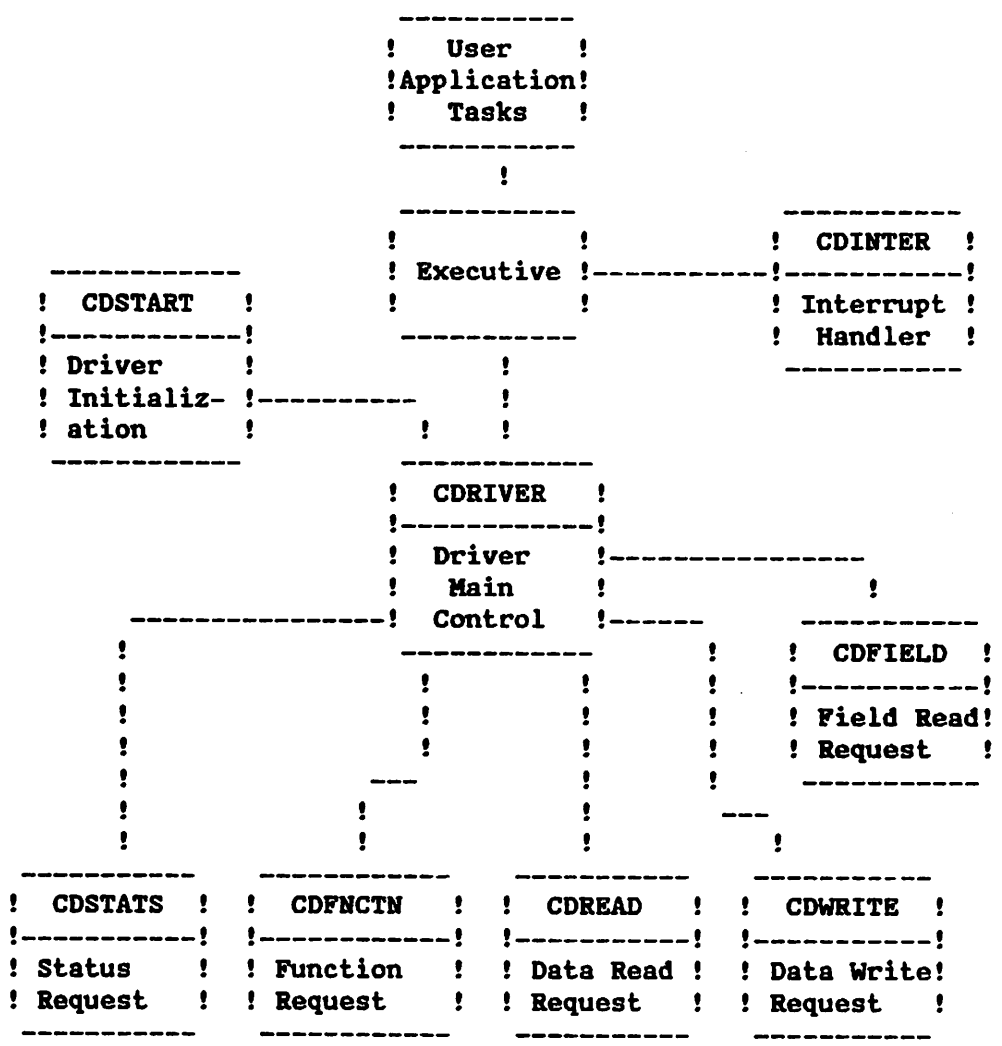
This Driver allows the Man-Machine subsystem to control a 40-815 CRT Controller connected to a Display Controller Interface (DCI) Card. Data input/output to a CRT unit on the 40-815 CRT Controller is controlled by the Driver. Other functional features of the CRT controller can also be controlled by the Driver.

The Driver processes status input, function output, and data read/write type requests for a specified 40-815 CRT Controller connected to the DCI card.

Unsocilited function key and lite pen interrupts are handled by the Driver. These interrupts occur as the result of some operator action (the depressing of a function key) at the keyboard associated with a 40-815 CRT Controller. The driver notifies the Man-Machine subsystem of these function key interrupts by sending it intertask messages whenever one occurs. Lite Pen interrupts indicate that the operator wants his cursor moved to a new position. The Driver processes Lite Pen interrupts by determining the X,Y co-ordinates for the screen position touched with the lite pen and then moves the cursor to that position.

The basic software components of the DCI Driver and its software environment are shown on the next page.

DCI DRIVER SOFTWARE COMPONENTS/ENVIRONMENT



DCI Driver Utility Function Subroutines

The Utility Subroutines are used by the major modules.

PRE_PROC - Pre-Read/Write Options.
POST_OPT - Post Read/Write Options.
GET_CUR -- Get Cursor Position.
MOVE_CUR - Position Cursor.
GET_VID -- Get Video Characteristics.
SET_VID -- Select Video Characteristics.
READCHAR - Read Single Character.
WRT_CHAR - Write Single Character.
DCI_OK --- Wait Until DCI Okay.
CRT_OK --- Wait Until CRT Controller Okay.
CDFOUT --- Select CRT Functions.
CDVMODE -- Select CRT Operational Modes.

4.1 PROBLEM DEFINITION

4.1.1 Status Driven Data Input/Output

In the hardware environment section, it was mentioned that the DCI card is not a buffered type device and that processing interrupts is not one of the 68000's strong points. These two factors as well as the speed of the DCI/CRT controller lead to a decision not to use interrupts to control data input/output operations.

Since the DCI is not a buffered type device, the 68000 must handle each individual character to be input/output. The speed of the DCI/CRT controller hardware is such that the 68000 can't keep it busy in most cases. For most of the normal data characters, the hardware can complete its processing in 4-6 microseconds. Since it usually takes the software 30 - 60 microseconds to handle a character, it is easy to see that the software can't keep up with the hardware when handling normal data character.

There is a certain subset of the character set however that are treated as control characters by the 40-815 CRT controller. Whenever one of these control characters is output, the CRT controller treats it as a command to perform some special hardware function such as a CLEAR SCREEN operation. These special control characters can take several milliseconds to process. These control characters are not very frequently used however so they do not really make a good argument for the use of interrupts.

Since it takes far longer to process an interrupt than it takes for the hardware to normally finish an input/output operation, it would slow up data input/output operations tremendously. Thus, all the data input/output are driven on a status basis rather than on an interrupt basis. These means of course that all DCI Driver requests are processed serially.

4.1.2 Video Characteristics Preservation

If the requester wants the video display characteristics of the data characters read from a CRT to be preserved, the Driver compares the video display characterisics of each character input with those of the previous character input. If there is a change in video display characteristics between the two characters, the appropriate "video change" control characters are stored between the two characters in the read buffer.

Since the NUMBER OF BYTES parameter in a data read request specifies the number of displayable data characters to be input, the requester has to be able to accept more data than what was asked for when this feature is selected.

4.1.3 Screen Tabs Processing

When a read requester specifies that it wants the video display characteristics of the data to be preserved, this Driver also perserves any TABS associated with the data.

The CRT controller's STATUS ONE byte contains a flag that specifies if the screen position, from which the last character was read, was either tabbed or protected. When this PROTECTED/TABBED bit is set for an input character, the following scheme determines if this character position is a tabbed position:

a. If preceeding character position also had the PROTECTED/TABBED bit set, the current character's position is assumed to be protected, not tabbed.

b. If current character is either a "START PROTECTED FIELD" character or a "END PROTECTED FIELD" character, the current character's position is assumed to be protected, not tabbed.

c. For all other cases, the current character's position is assumed to be tabbed.

When a character is read from a tabbed position, a SET TAB character is stored before the character in the data read buffer if the PRESERVE VIDEO DISPLAY CHARACTERISTICS option has been specified on the read request.

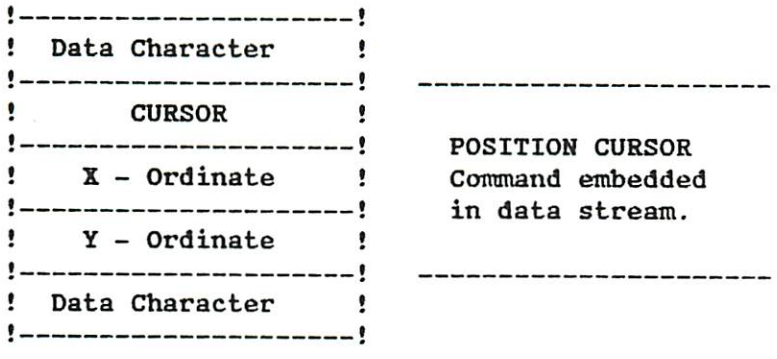
4.1.4 Repeated Space String Suppression

This feature is implemented in a following way:

a. A prosign character (CURSOR) is selected.

b. All concerned parties agree that when the Driver encounter this prosign character in a stream of data being output to a CRT, the Driver is to interprete this prosign character and the following two bytes as being a POSITION CURSOR command. The two bytes following the prosign character are to be interpreted as being the X and Y ordinates for the new screen position of the CURSOR.

The following diagram illustrates this feature:



This feature can be utilized for two different purposes:

a. First of all, it can be used to suppress strings of repeated, non-inverse space characters from data being read. It is understood that the entire screen needs to be cleared immediately before this data is rewritten to the screen.

Because of the fact that the system must maintain large numbers of pre-formatted display skeletons which usually contain a fair number of repeated, non-inverse space character strings, this feature is utilized to cut down upon both amount of mass storage needed to store these display skeletons and the amount of I/O channel time needed when these skeletons have to moved.

b. This feature can also be used to allow dynamic data associated with displays to be output to multiple fields on the CRT screen in a random order without forcing the requester to issue multiple requests.

Each pre-defined display has associated dynamic data definition packets that define which data values are to be displayed, how they are to be represented on the screen, and where they are to be written on the screen. These dynamic value definition packets aren't usually stored in the display definition record in the same order that their associated values are found on the screen. Many of these dynamic data values also have associated attributes that are represented by special character strings which are output relative to data values themselves.

The Driver allows a data read requester to specify if it wants repeated space character strings to be suppressed using this scheme. On CRT Write requests, the Driver scans the output data for the prosign character associated with this feature and performs the appropriate CURSOR positioning operation whenever it encounters one.

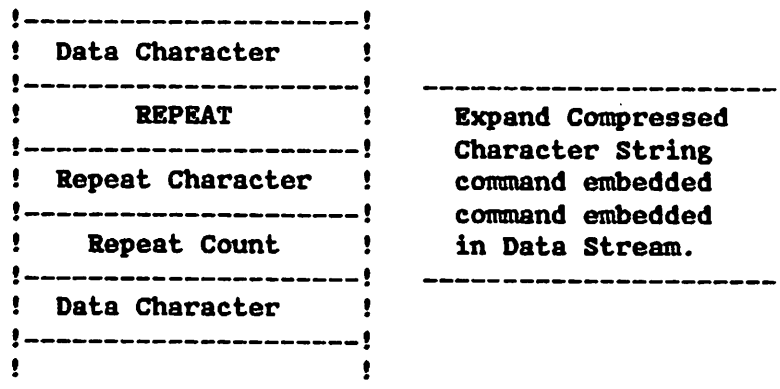
4.1.5 Repeated Character String Compression

This feature is implemented in a following way:

a. A prosign character (REPEAT) is selected.

b. All concerned parties agree that when the Driver encounters this prosign character in a stream of data being output to a CRT Controller, the Driver is to interprete this prosign character as a command to output the next character to the CRT Controller the number of times specified by the second character following the prosign character.

The following diagram illustrates this feature:



The Driver allows a data read requester to specify if it wants its data compressed using this scheme. On CRT Write requests, the Driver scans the output data for the prosign character associated with this feature and expands compressed strings whenever they are encountered.

4.1.6 Reading Multiple Data Fields

Most system displays contain enterable data fields that allow the operator to enter new data values into the system data base. When a display is called to a CRT, the current values of the data items associated with the enterable fields are displayed and in most cases these values are updated periodically as long the display remains up on the screen. The enterable data fields on a display and the data items associated with these enterable fields have to be identified by the static background portion of the display.

To enter new values into the data base for the data items associated with the enterable fields, the operator enters one or more new values in the enterable fields on the screen and presses the ENTER function key.

When the Man-Machine subsystem processes such a request, it must first determine which of the enterable fields associated with the display contain new operator entered values that it needs to process. There are a number of schemes that can be used to identify operator entered values:

a. Sometimes color is used to identify operator entries. When entering a value in an enterable field, the operator uses a different color than the Man-Machine subsystem uses when it outputs the current value of the data item associated with the enterable field. Many systems in the past have reserved White as the color to be used for data entry operations.

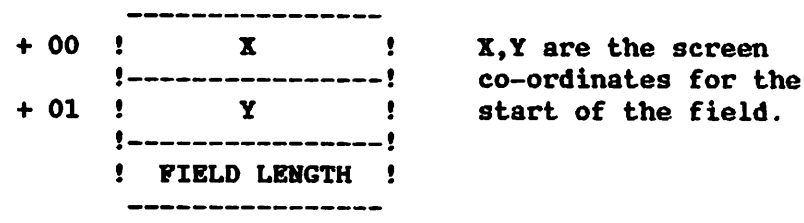
b. Sometimes the Man-Machine subsystem keeps track of the last value that it displayed in an enterable field and compares it to the current value in the field. If they don't match, it is assumed that the operator has enter a new value here. This scheme result in inadvertant data entries if the operator alters a field by mistake.

Regardless of the scheme actually used, the Man-Machine has to read all the enterable fields associated with the display to determine which fields contain new data values entered by the operator. The regular CRT Read request could be used for this function. Each enterable field could read individually and the data would not be very hard to process, but this would be very inefficient because there is a certain amount of overhead involved with every request processed. On the other hand, the entire screen could read and the enterable fields could be extracted from the data read, but this would be a very difficult procedure to implement.

To get around these problems, the Multiple Field Read request was designed so that all the enterable fields associated with a display could be read with a single request and that the data read would be formatted in such a fashion as to make it easy to process.

4.1.6.1 Data Field Definitions Data Buffer Chain

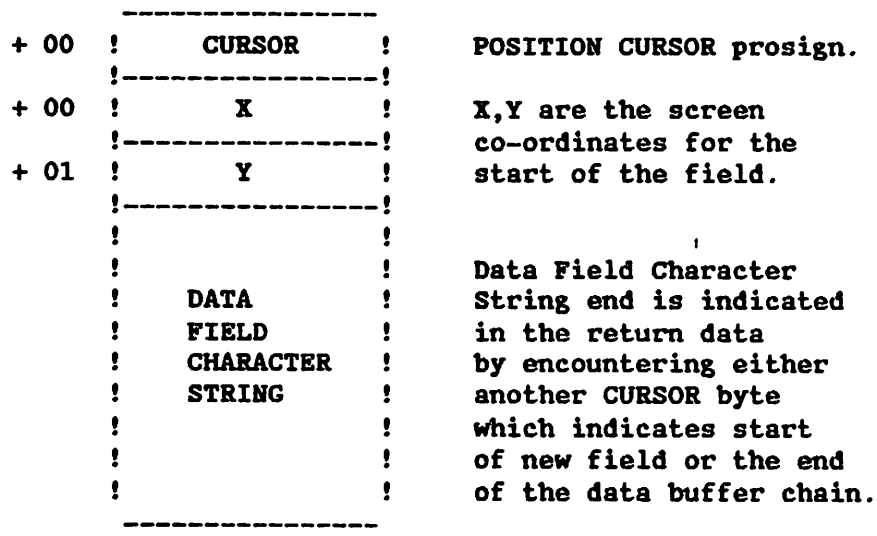
The requesting task provides a chain of data buffers that contain Data Field Definition Packets that define the data fields to be read. For each field, the task has to identify the starting screen position for the field and the length of the field. This information is packed into three byte packets that are formatted as follows:



The Data Field Definitions data buffer chain contains one or more Data Field Definition packets. The Data buffer control fields will indicate how many Data Field Definition packets are contained in the chain of Data buffers.

4.1.6.2 Multiple Field Read Request Return Data

The data read from each of the fields defined by the requesting task is formatted as follows and stored in chain of Data buffers:



It should be noted that this data format allows the Man-Machine subsystem to write the data read back out to the CRT.

4.1.6.3 Video Display Characteristics Change Command

There is still one more feature associated with the Multiple Field Read request return data that must be explained. As was mentioned earlier, the Man-Machine subsystem is interested in the video display characteristics (color, blink, and inverse) of the characters read from the enterable data fields.

The appropriate Video Display Mode Change control characters could be inserted in the Field Data Character String to indicate the video display characteristics of the actual data character in the same way as the normal CRT Read request. But this scheme would produce a data stream that would be complicated for the Man-Machine subsystem to process.

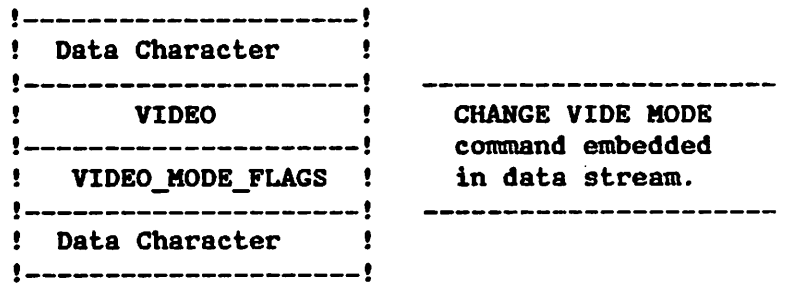
Instead, CHANGE VIDEO MODE command sequences are embedded in the Data Field Character String in front of the first data character and in front of any other data character whose video display characteristics differ from those of the preceding data character.

The CHANGE VIDEO MODE command is implemented in the following manner:

a. A prosign character (VIDEO) is selected.

b. All concerned parties agree that when the Driver encounter this prosign character in a stream of data being output to a CRT, the Driver is to interprete this prosign character the following character as being a CHANGE VIDEO MODE command. The character following the VIDEO prosign character is really a set of Video Display Characteristic flags. The flags are described below.

The following diagram illustrates this feature:



Where: VIDEO_MODE_FLAGS

Bits 7-6 -- Not Used

Bits 5-3 -- Color Code:

7 - White

6 - Cyan

5 - Magenta

4 - Blue

3 - Yellow

2 - Green

1 - Red

0 - Black (Illegal)

Bit 2 -- Inverse Video Mode

Bit 1 -- Not Used

Bit 0 -- Blink Mode

4.1.7 Mode Change Character Usage Conventions

The 40-815 CRT Controller has several operation modes which are enabled/disabled by outputting the appropriate control characters to the controllers.

These operational modes are:

- o Protected mode
- o Scroll Up mode
- o Scroll Down mode
- o Insert Character in Field/Line mode
- o Insert Character in Field/Page mode

The 40-815 CRT Controller doesn't however provide a status function that can be used to determine if the controller currently has these operational modes selected. This means that a record must be maintained to indicate if the controller is supposed to have any of these operational modes selected. This record can be accessed when the operational modes for the controller are required. Since these operational modes get cleared during the processing of data read/write requests, a OPERATIONAL MODES record is needed to restore the controller to the proper state after the processing of these requests. This record also allows this information to be provided when a CRT Status request is processed.

This OPERATIONAL MODES record must be kept up-to-date. Since these optional modes are selected/cleared by outputting control characters to the CRT Controller, one way to ensure that this table is up-to-date would be to scan all data being output to the CRT Controller and check for the six Mode Change control characters and then adjust the OPERATIONAL MODES record whenever such a control character is encountered. This would involve a lot of overhead.

To avoid this overhead, it was decided that the Driver would be the only one allowed to output these control characters. By convention, these control characters are not to be embedded in user output data. But a user can request the Driver to output these control characters by using a CRT Function Output request. This means that the Driver simply keeps track of the Mode Change characters that it outputs.

4.2 GENERAL SOLUTIONS

4.2.1 CRT Status Request Processing

The DCI Driver allows an applications task to input the status of a specified CRT unit with a simple status input request.

This request allows the applications task to input the following information about a CRT unit:

a. CRT Unit Ready status.

b. CRT Unit Busy status.

c. Cursor Position.

d. Character located at current cursor position and its associated video characteristics: color, blink, and inverse video status.

e. Current video characteristics selected for CRT.

f. Keyboard Enabled status.

g. Lite Pen Enabled status.

4.2.2 CRT Function Request Processing

The DCI Driver allows an applications task to output control functions to a specified CRT with a simple CRT Function request.

This request allows the application task to perform the following functions:

a. Enable Keyboard/Function Keys.

b. Disable Keyboard/Function Keys.

c. Enable Lite Pen.

d. Disable Lite Pen.

e. Master Clear CRT.

f. Set CRT's alarm relay.

g. Clear CRT's alarm relay.

h. Position CRT cursor.

i. Initiate a copy operation to the CRT's slave print device if it has one.

j. Select/De-select the following operational modes for the CRT controller:

   1. Protected Mode.
   2. Scroll-Up Mode.
   3. Scroll-Down Mode.
   4. Insert Character in Field/Page Mode.
   5. Insert Character in Field/Line Mode.

4.2.3 CRT Read Request Processing

The DCI Driver allows an applications task to read data from a specified CRT with a simple read request.

The requester can specify the following parameters for a data read operation from a CRT screen:

a. CRT unit identifier.

b. Either address of a chain of data buffers provided by the requester for the data or an ALLOCATE DATA BUFFERS indicator.

c. Number of characters to be read.

d. Beginning screen position of data to be read:

   1. Current cursor position.
   2. Home position on screen.
   3. Specified Cursor Position.
   4. Beginning of Current Line.

e. Data Compression Desired indicators:

   1. Repeated spaces suppression.
   2. Repeated characters compression.

f. A Data Characters/Data Characters Plus Video Control Characters indicator.

g. A Pre-Read Protect Mode Alteration indicator.

h. Perform read operation without changing unit's protect mode.

i. Perform read operation with unit's PROTECT mode disabled and then restore unit's PROTECT mode to pre-read operation state.

j. A Post Read Cursor Position Option indicator:

1. Leave cursor after last character read.
2. Move cursor to HOME position on screen.
3. Return cursor to pre-read position.

k. A Post Read Keyboard Status Option indicator:

1. Enable keyboard.
2. Disable keyboard.
3. Return keyboard to pre-read state.

l. A Post Read Lite Pen Status Option indicator:

1. Enable Lite Pen.
2. Disable Lite Pen.
3. Return Lite Pen to pre-read state.

After the Driver attempts to process a read request, it returns a request processing completion code to the requesting task indicating whether or not it was able to process the request successfully.

Device failures are reported to Configuration Control.

4.2.4 CRT Write Request Processing

The DCI Driver allows an application task to write data to a specified CRT with a simple data write request.

The requester can specify the following parameters for a CRT Write operation:

a. CRT Identifier.

b. Address of chain of data buffers.

c. Number of bytes to output.

d. Pre-Write Clear Screen/Home Cursor flag.

e. Beginning screen position for data to be written:

   1. Current cursor position.
   2. Home position on screen.
   3. Specified Cursor Position.

f. A Pre-Write Protect Mode Alteration indicator.

g. Perform write operation without changing unit's PROTECT mode.

h. Perform write operation with unit's PROTECT mode disabled and then restore unit's PROTECT mode to pre-write state.

i. A Post Write Cursor Position Option indicator:

1. Leave Cursor after last character written.
2. Move cursor to HOME position on screen.
3. Return cursor to pre-write position.

j. A Post Write Keyboard Status Option indicator:

1. Enable keyboard.
2. Disable keyboard.
3. Return keyboard to pre-write state.

k. A Post Write Lite Pen Status Option indicator:

1. Enable Lite Pen.
2. Disable Lite Pen.
3. Return Lite Pen to pre-write state.

l. A Release Data Buffers flag.

After the Driver attempts to process a write request, it returns a request processing completion code to the requesting task indicating whether or not it was able to process the request successfully.

Device failures are reported to Configuration Control.

4.2.5 Multiple Field Read Request Processing

The DCI Driver allows an applications task to read multiple data fields from a specified CRT with a simple read request. The Multiple Data Field Read request is designed mainly for the data entry function. It allows all the enterable fields on the CRT to be read with a single read request. The data read is packed up in a format that is easy for the Man-Machine subsystem to process.

The requester can specify the following parameters for a Multiple Field Read operation from a CRT screen:

a. CRT unit identifier.

b. Either address of a chain of data buffers provided by the requester for the data or an ALLOCATE DATA BUFFERS indicator.

c. Field Definitions for the data fields to be read. A Data Field Definition consists of the X and Y co-ordinates for the field's beginning screen position and the length of the field.

d. A Post Read Cursor Position Option indicator:

   1. Leave cursor after last character read.
   2. Move cursor to HOME position on screen.
   3. Return cursor to pre-read position.

e. A Post Read Keyboard Status Option indicator:

   1. Enable keyboard.
   2. Disable keyboard.
   3. Return keyboard to pre-read state.

f. A Post Read Lite Pen Status Option indicator:

   1. Enable Lite Pen.
   2. Disable Lite Pen.
   3. Return Lite Pen to Pre-Read State.

After the Driver attempts to process a read request, it returns a request processing completion code to the requesting task indicating whether or not it was able to process the request successfully. Device failures are reported to Configuration Control.

4.2.6 Function Key Interrupt Processing

The Driver preprocess function key interrupts from a CRT unit, formats a FUNCTION KEY INTERRUPT message and sends this message to an application task designed to process function key interrupts.

The preprocessing of a function key interrupt consists of the following actions:

a. Saving the information necessary to restore CPU control back to the interrupted user later.

b. Determining identity of the CRT Controller from which the function key interrupt was received.

c. Determining number of the function key depressed.

d. Disabling interrupting CRT Controller's keyboard/interrupts.

The FUNCTION KEY INTERRUPT message is formatted as shown here:

```
        !---------------------------------!
+ 00    !   REQUEST TYPE CODE  (     )    !
        !---------------------------------!
+ 02    !   CRT LOGICAL UNIT NUMBER       !
        !---------------------------------!
+ 04    !   FUNCTION KEY NUMBER.          !
        !---------------------------------!
```

NOTE: The Driver assumes the task identifier, for the application task designed to handle the CRT function key interrupts, is located in an external cell that can be referenced by the name "DDTASK3".

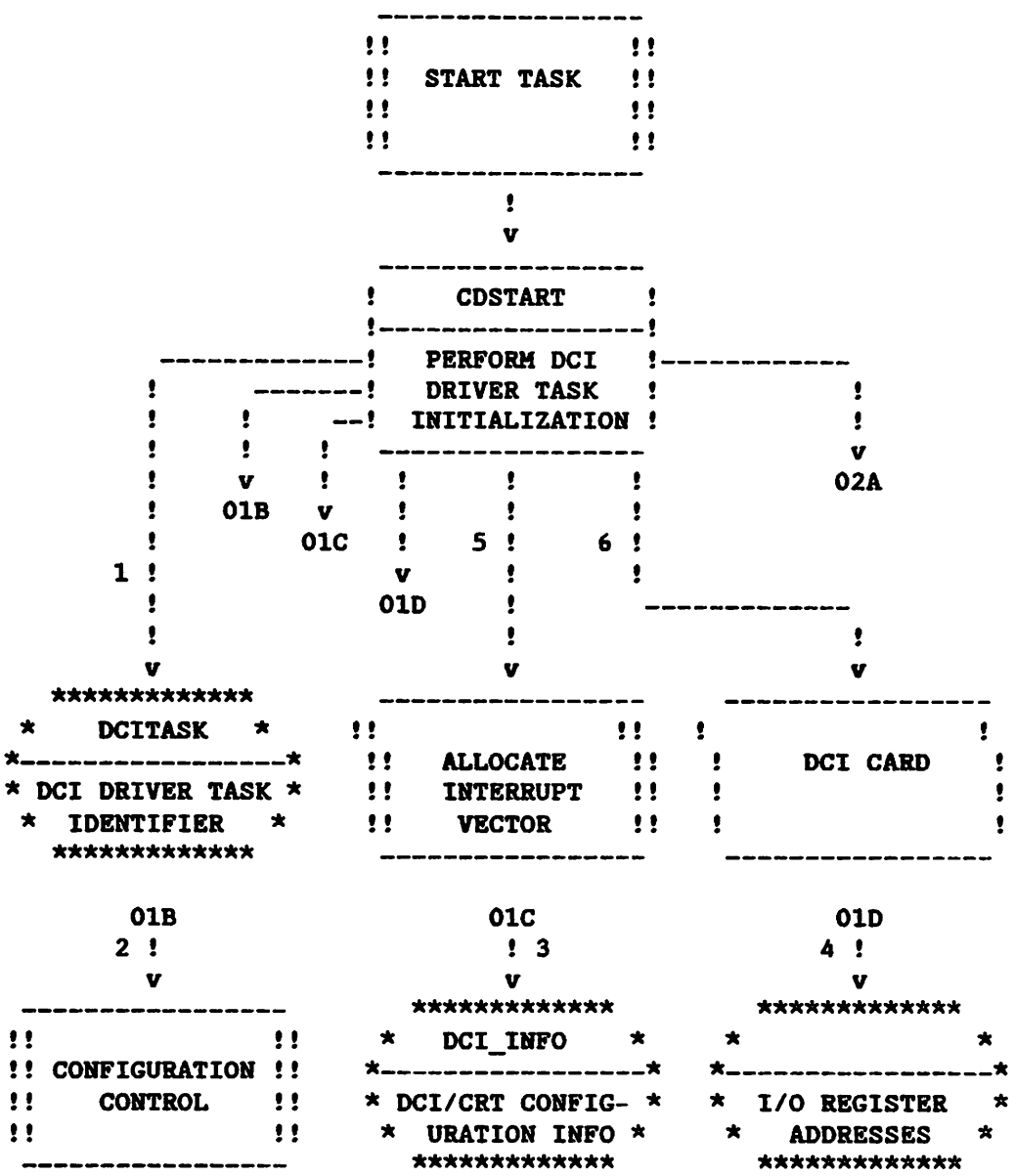
4.2.7 Lite Pen Interrupt Processing

The Lite Pen is a hardware device that can be attached to the keyboard unit associated with a CRT Controller. The Lite Pen is used by the operator to position the CRT screen's cursor. When the Lite Pen is enabled, the operator can cause a Lite Pen interrupt by touching the tip of the Lite Pen to the position on the screen to which he wants the cursor moved.

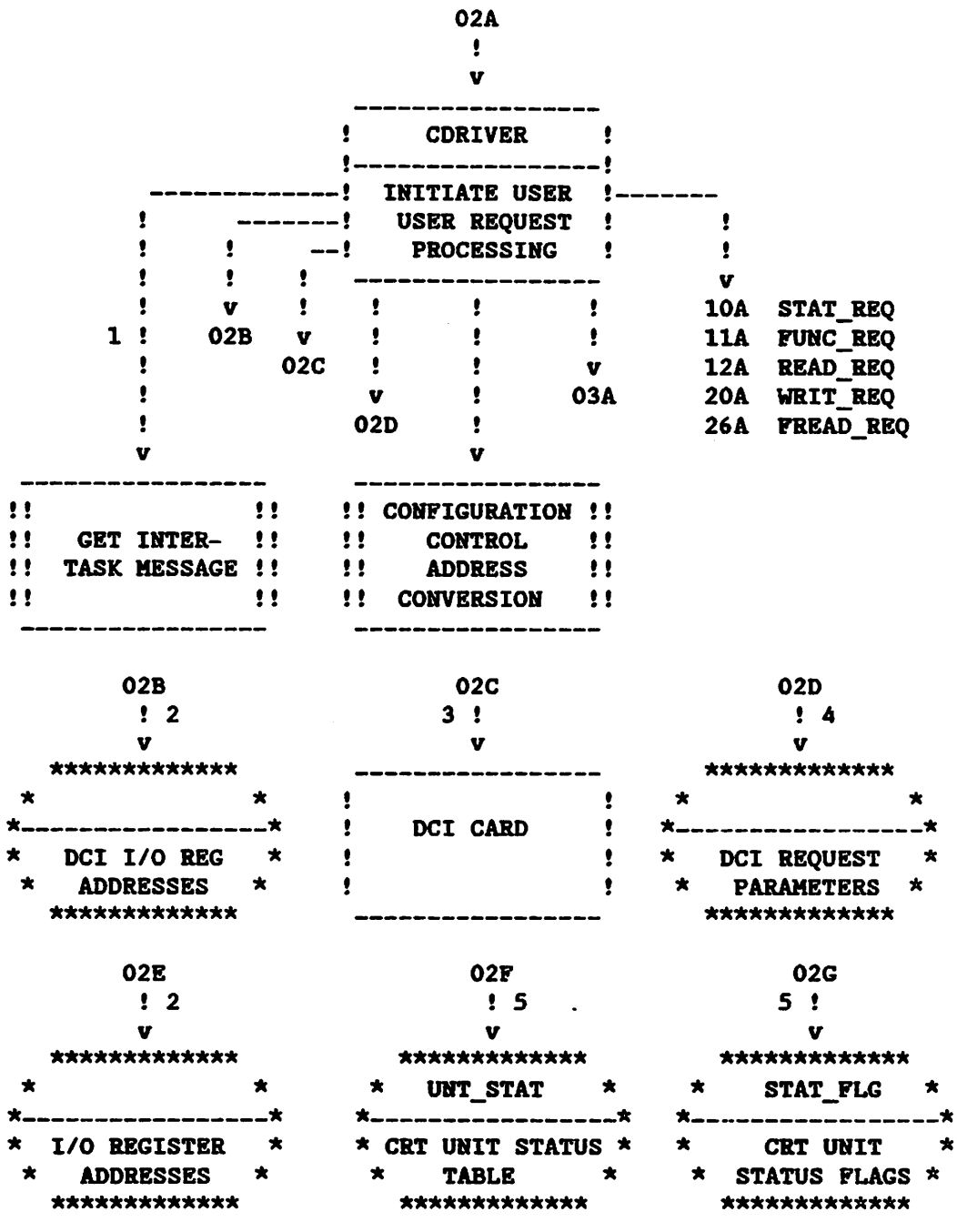
The CRT Driver process Lite Pen interrupts from a CRT controller by simply determining where the requester wants the cursor to be positioned and moving the cursor to that position. It should be noted that the CRT Controller hardware is designed in such a fashion that the first CURSOR POSITION INPUT operation performed after a lite pen interrupt will provide the X,Y coordinates for the point on the screen touched by the Lite Pen rather the cursor's current co-ordinates.

The operator can also cause an interrupt by grounding the Lite Pen unit, but this interrupt is equated by the hardware to a selected function key interrupt. This selected function key is normally either the DISPLAY or the EXECUTE key. This interrupt is processed as a function key interrupt.

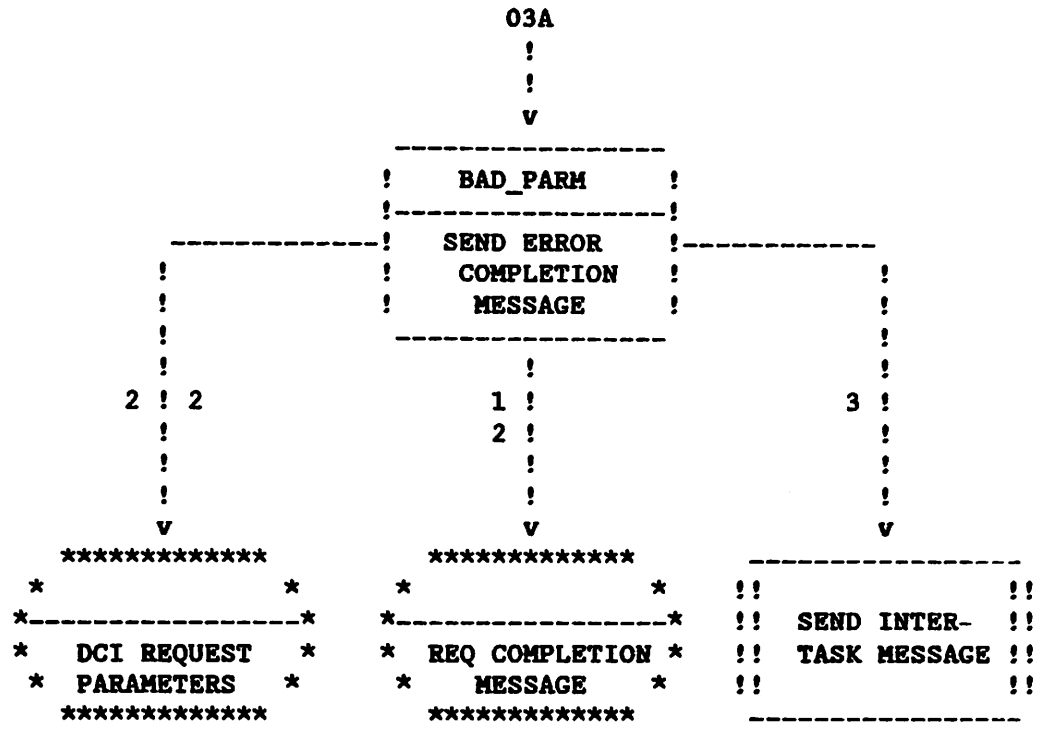
DESIGN STRUCTURE CHARTS -- PAGE ONE



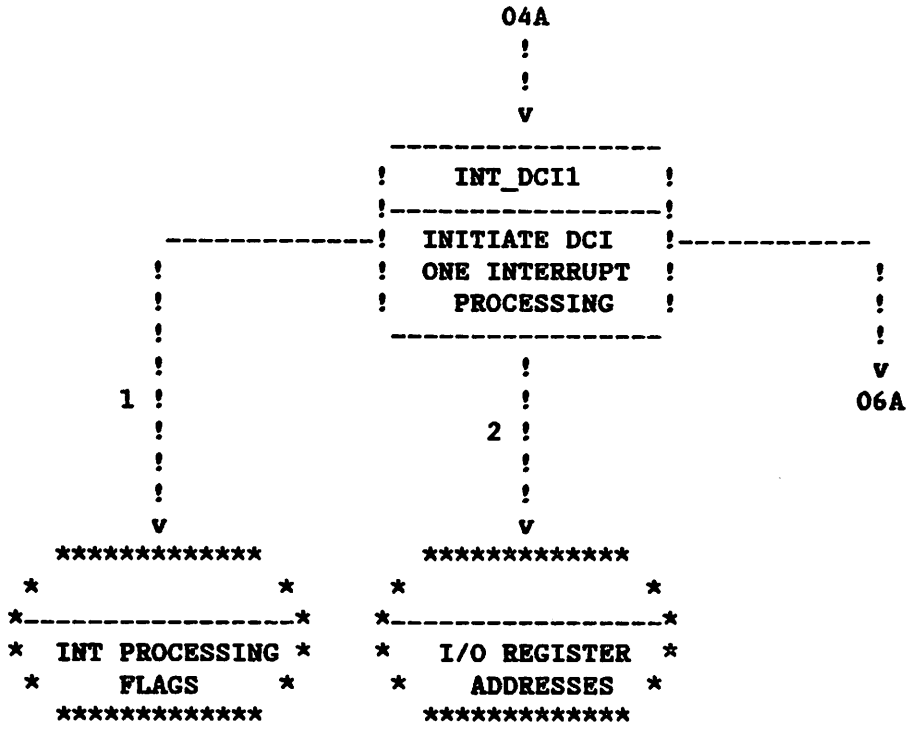
1. Task ID.
2. DCI_INFO Addr.
3. DCI Config. Info.
4. I/O Reg Addr.
5. Vector #, vector Subroutine Addr.
6. Vector #



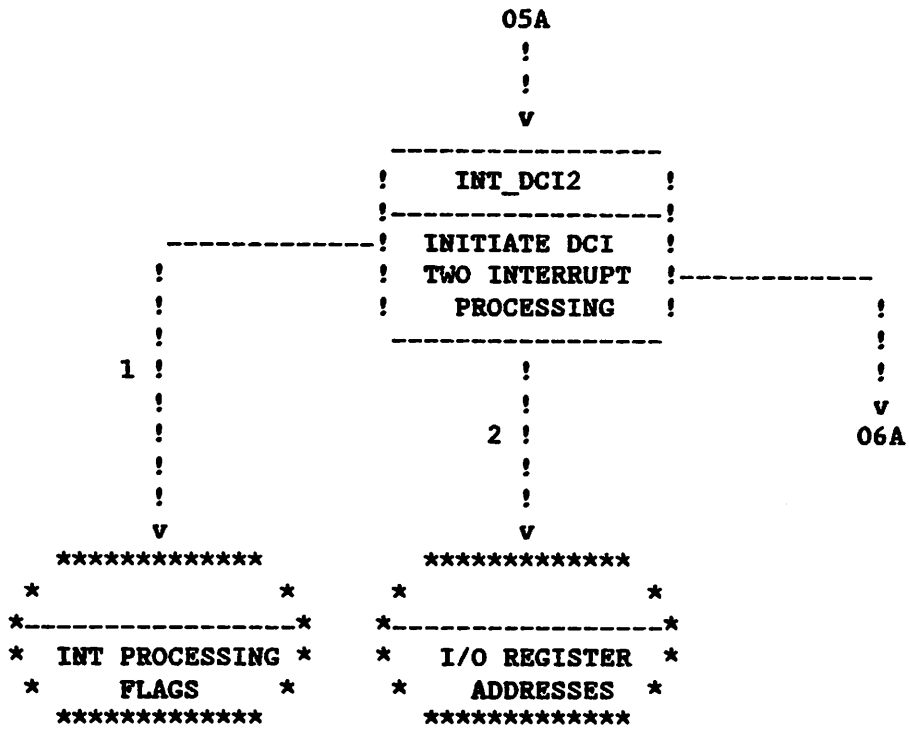
1. Msg Receive Buf Addr. 2. I/O Reg Addr. 3. Function Commands
4. DCI Req Parameters 5. CRT Status flags.



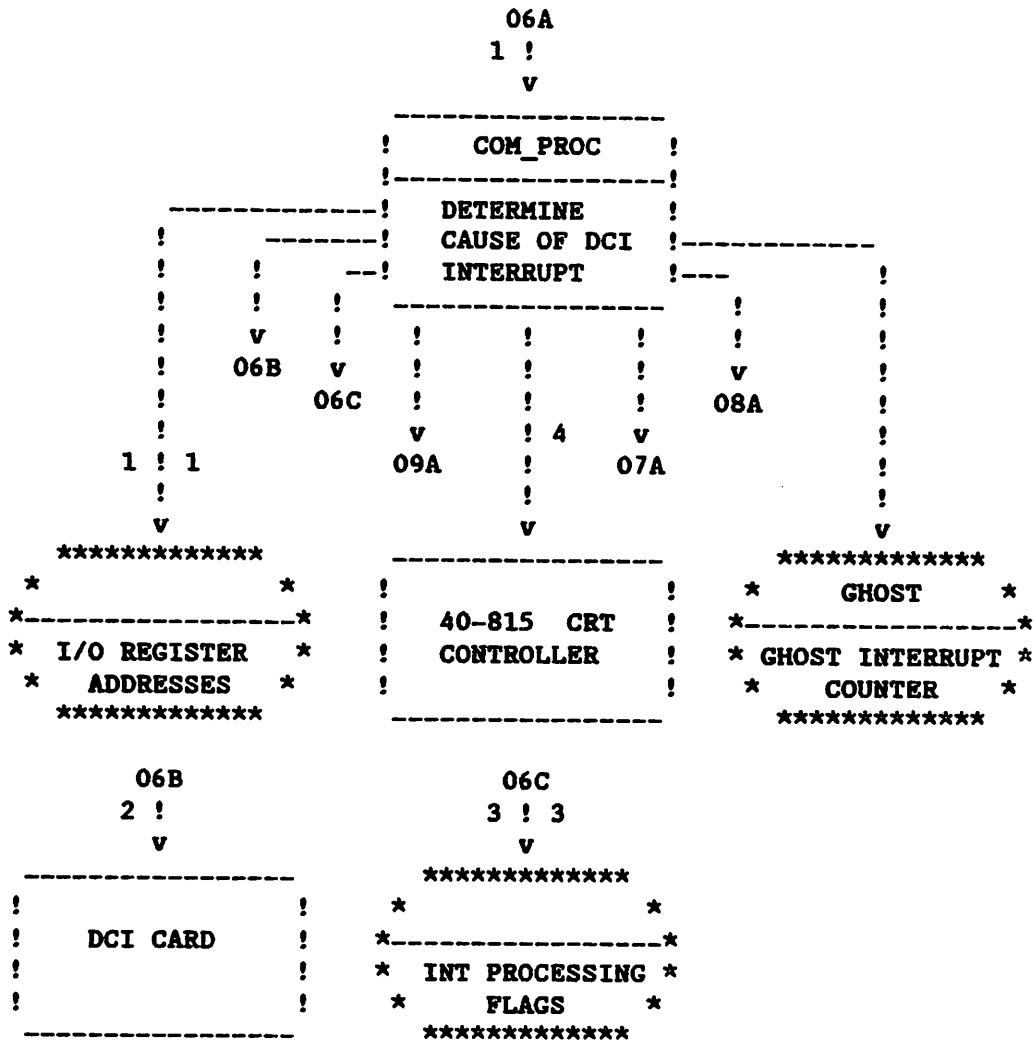
1. Error Code.
2. Request Parameters.
3. Message Address.



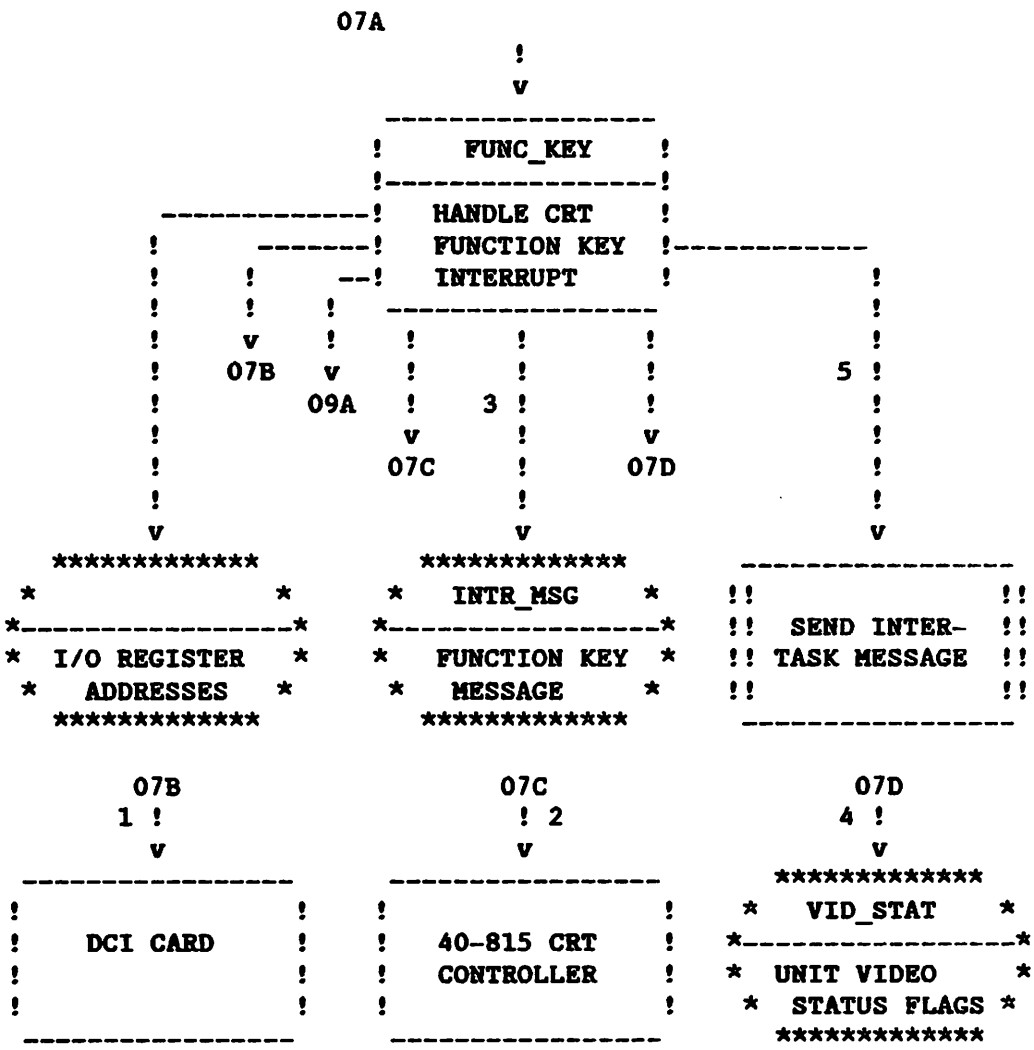
1. DCI Index, Number of CRTs, Base CEN.
2. I/O Register Address.



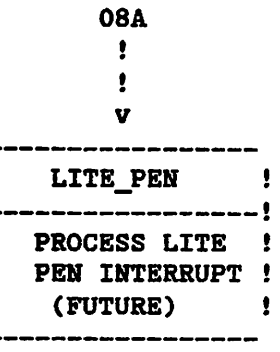
1. DCI Index, Number of CRTs, Base CEN.
2. I/O Register Address.

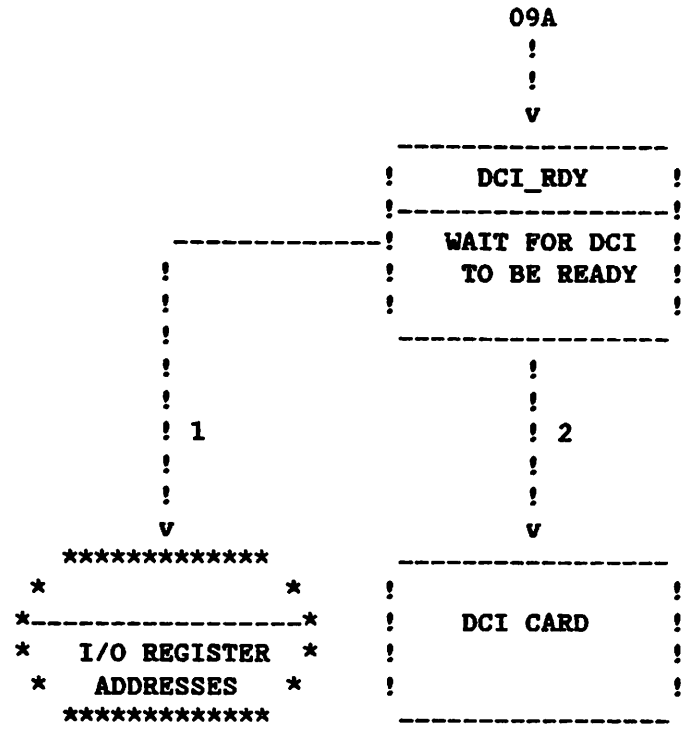


1. DCI_CMND0 Address. 2. Function Command. 3. Current CRT #
4. CRT STATUS ONE.

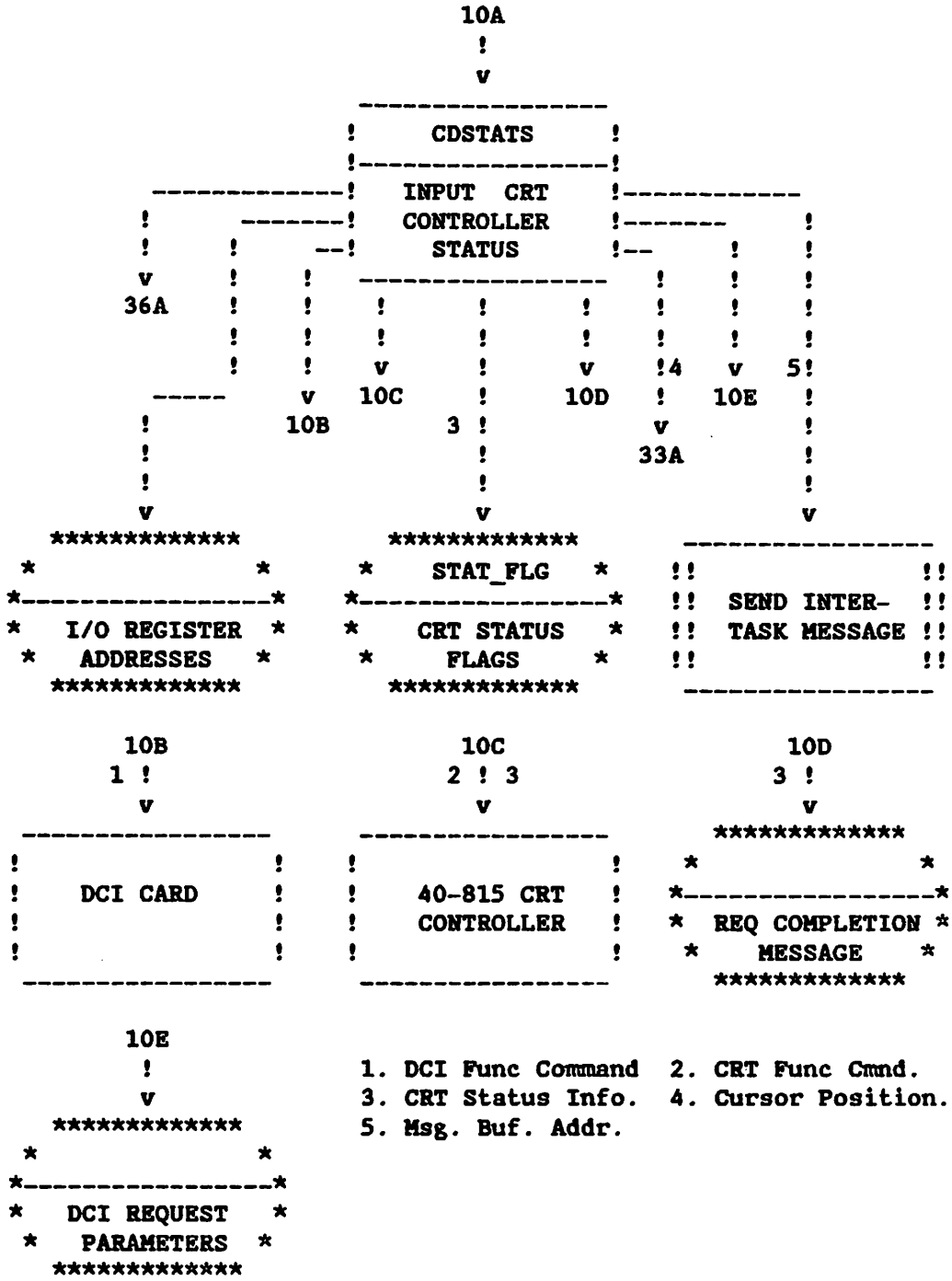


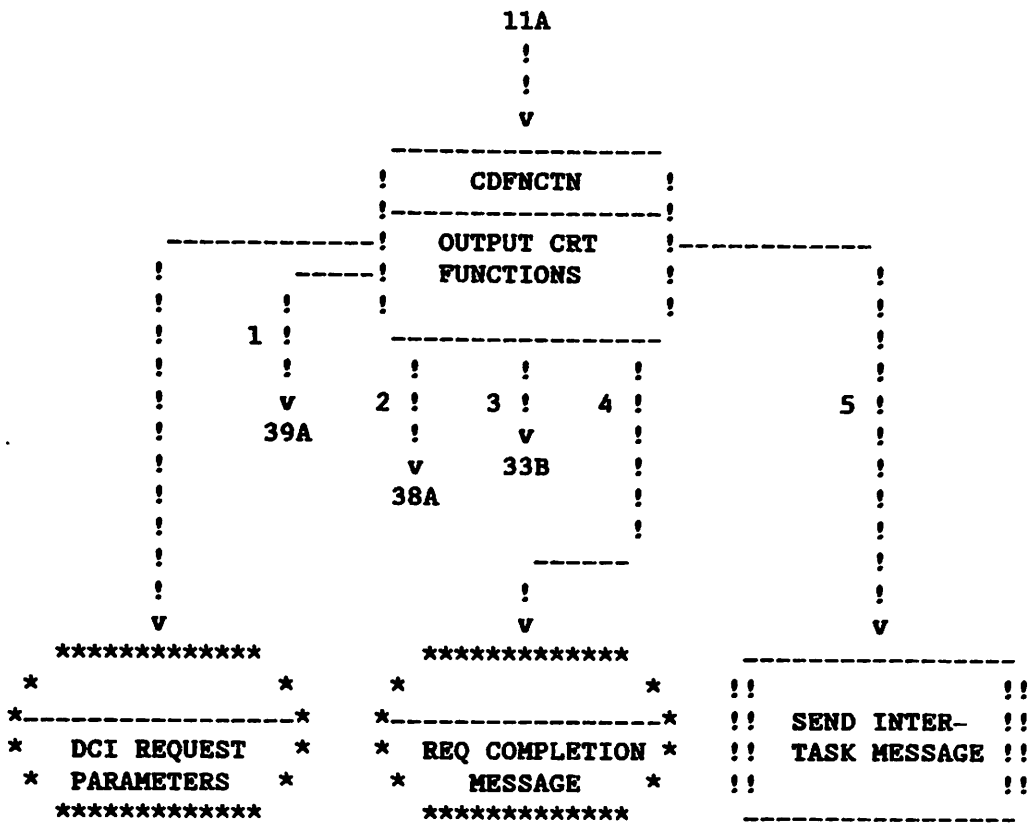
1. CRT STATUS THREE input command. 2. CRT STATUS THREE
3. Function Key # 4. Keyboard Disabled flag. 5. MSG Buf Addr.


08A
LITE_PEN
PROCESS LITE PEN INTERRUPT (FUTURE)

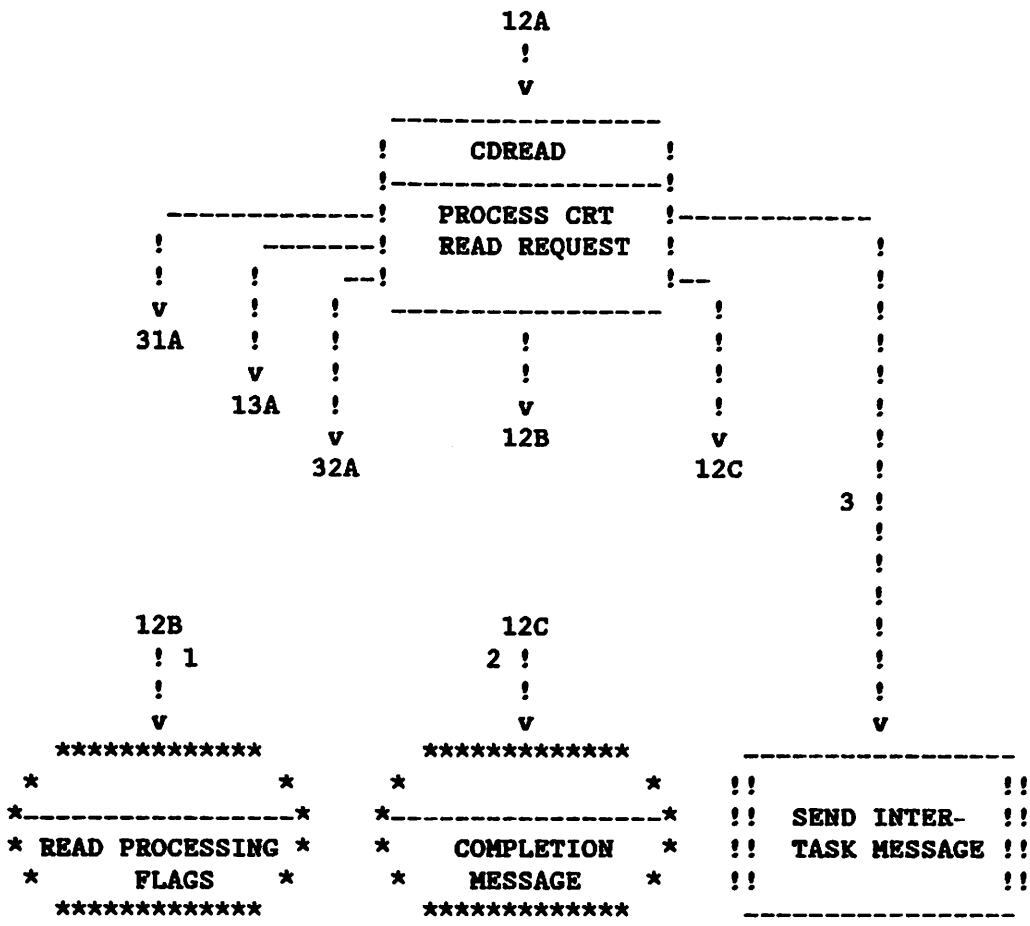


1. DCI_CMND2 Register Addr. 2. DCI STATUS TWO.

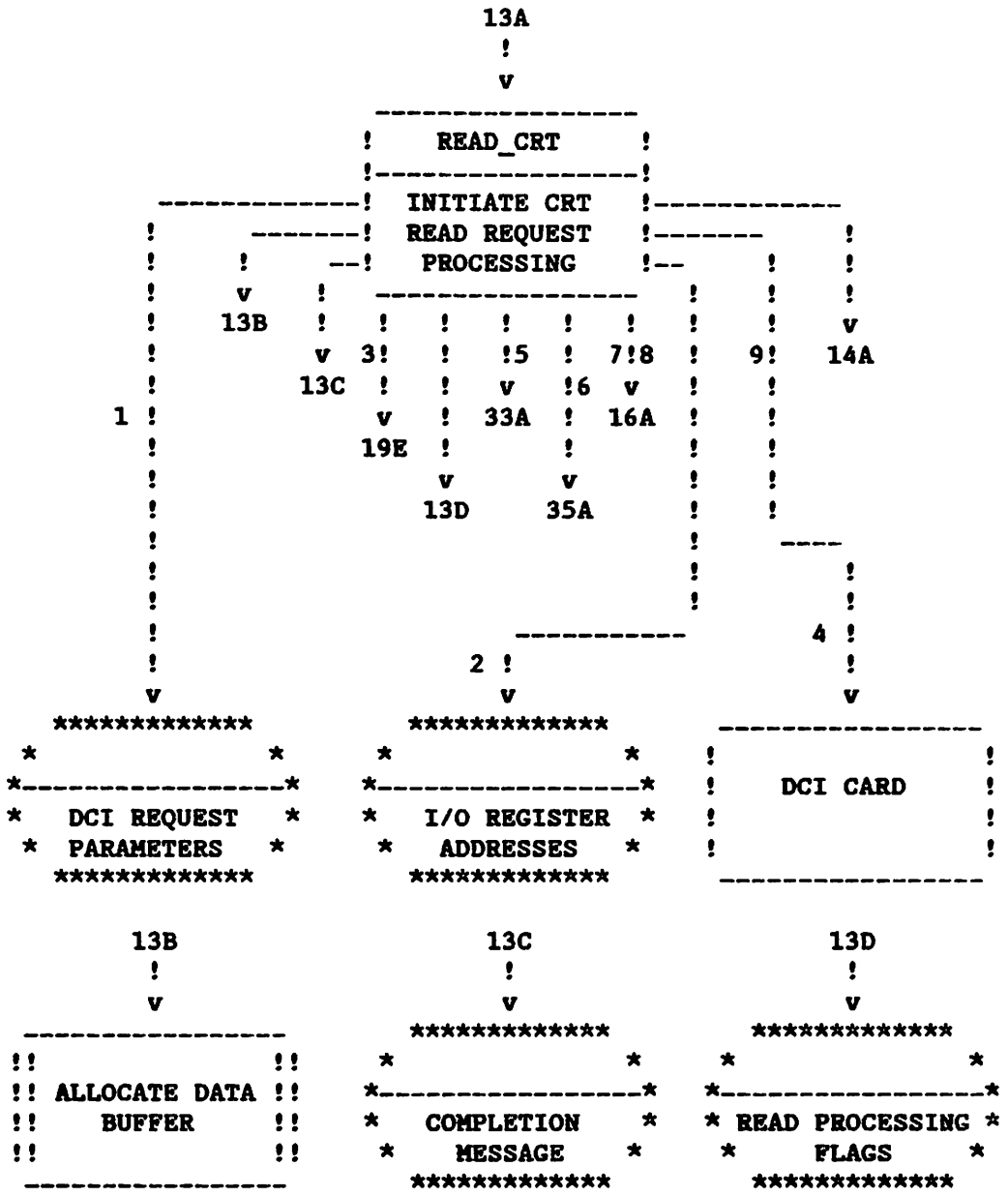

10A
CDSTATS
INPUT CRT CONTROLLER STATUS
36A
10B
10C
3
10D
4
10E
5
I/O REGISTER ADDRESSES
STAT_FLG
CRT STATUS FLAGS
SEND INTER-TASK MESSAGE
10B
1
DCI CARD
10C
2 3
40-815 CRT CONTROLLER
10D
3
REQ COMPLETION MESSAGE
10E
DCI REQUEST PARAMETERS
1. DCI Func Command 2. CRT Func Cmnd.
3. CRT Status Info. 4. Cursor Position.
5. Msg. Buf. Addr.



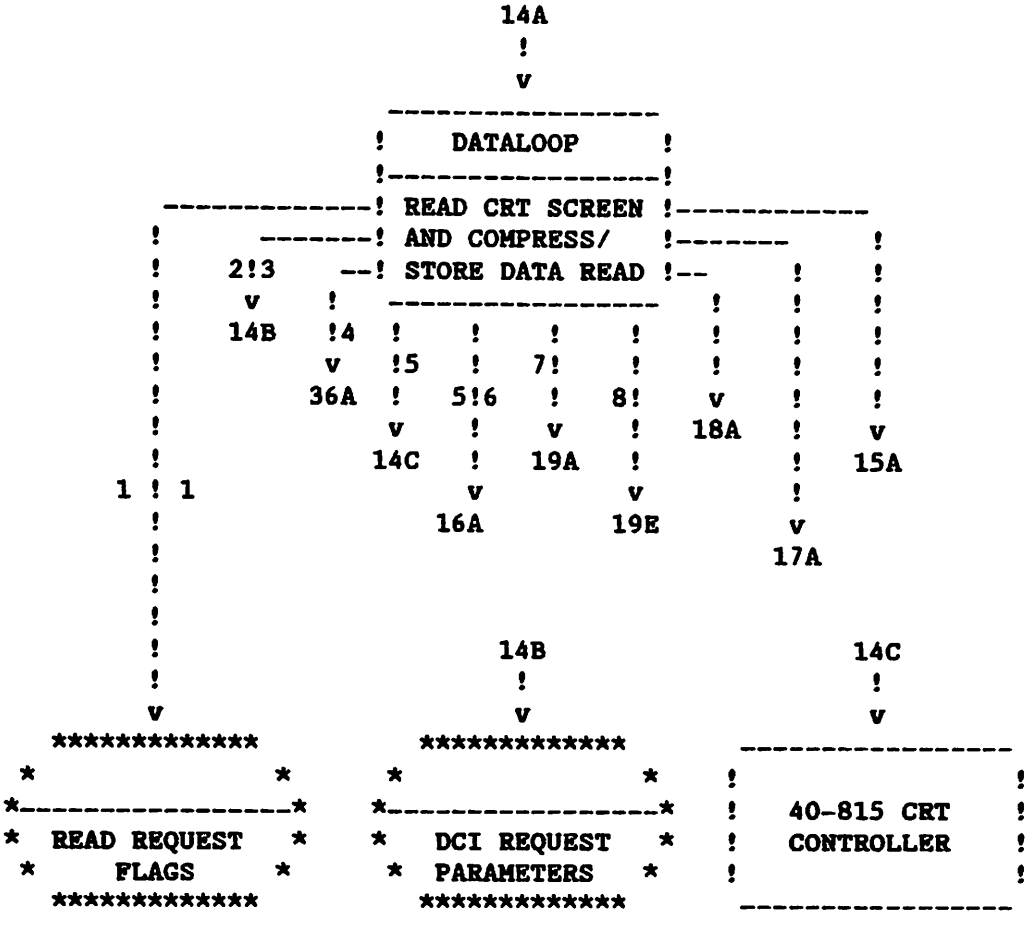
1. Mode Select flgs 2. Function Select flgs 3. Cursor Position
4. Completion Info. 5. Msg Buf Addr.



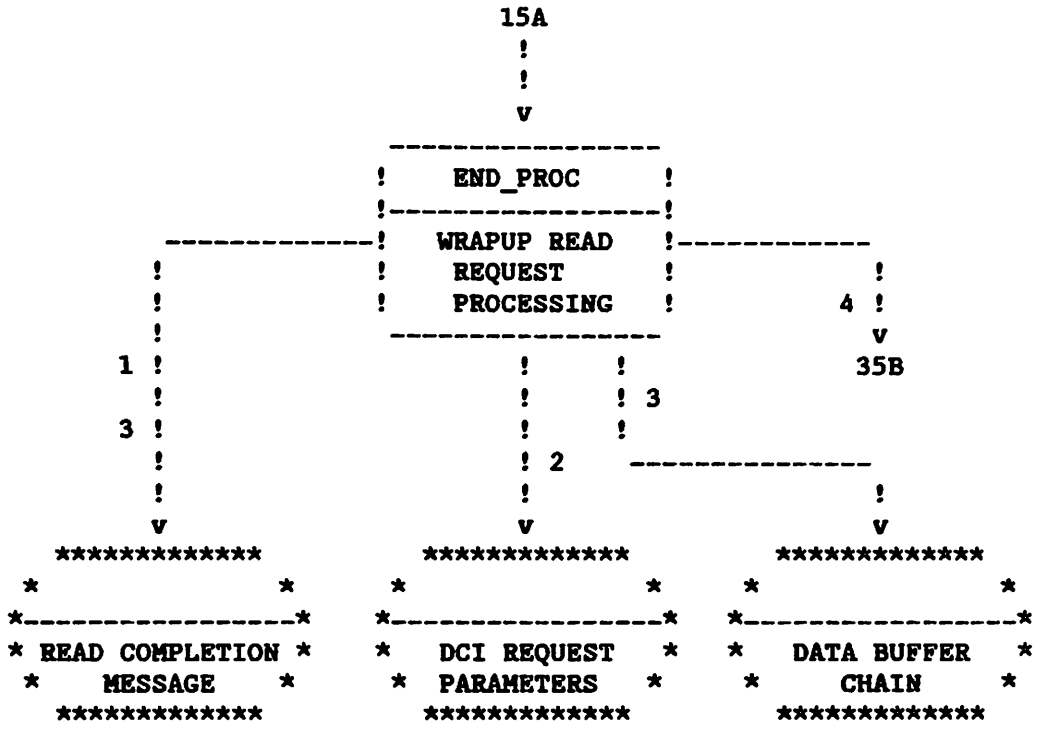
1. Data Buf Addr., Req Type, CRT Identifier.
2. Req Type, CRT Identifier, Completion Info.
3. Message Buffer Address.



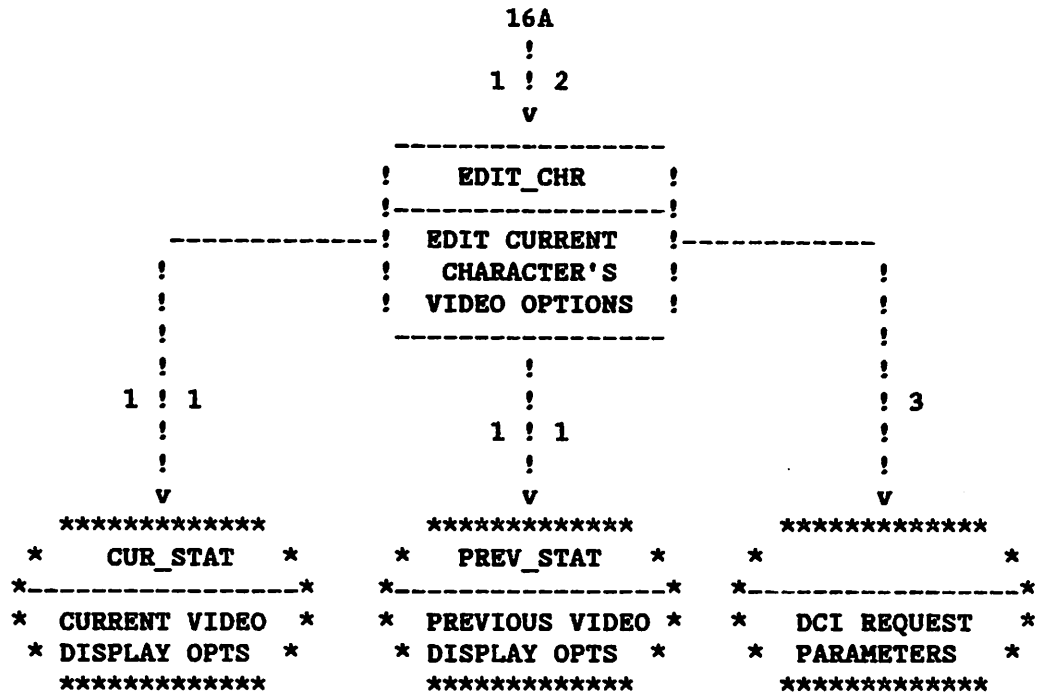
1. Data Buf Addr, Number of Bytes, Flags 2. First Data Buf Addr.
3. Data/Control Bytes 4. Processing Control Info.
5. Cursor Position 6. Data character 7. Character/Video Modes
8. Cha /Edited video Modes 9. Read Character Command



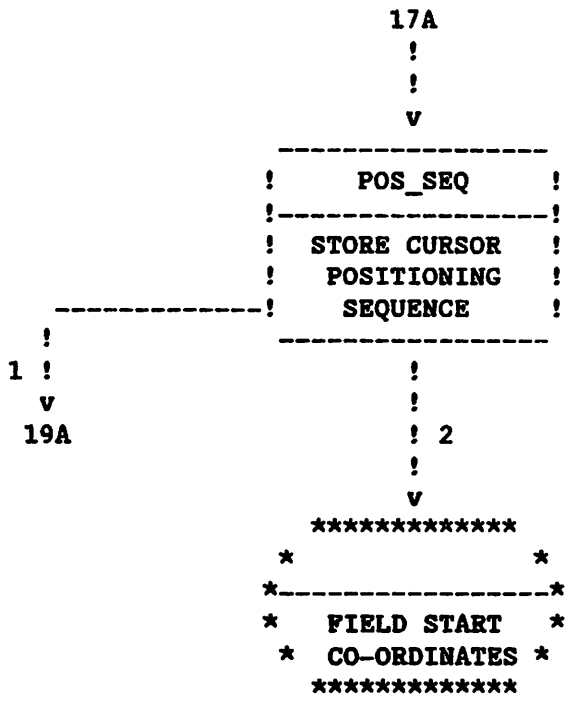
1. Read Request Processing flags.
2. First Data Buffer Address.
3. Request flags, Data Buffer Address.
4. DCI Ready Status.
5. Character/Associated Video Display Characteristics.
6. Character/Editted Video Display Characteristics.
7. Data/Control Character. 8. Next Data Buffer Address.



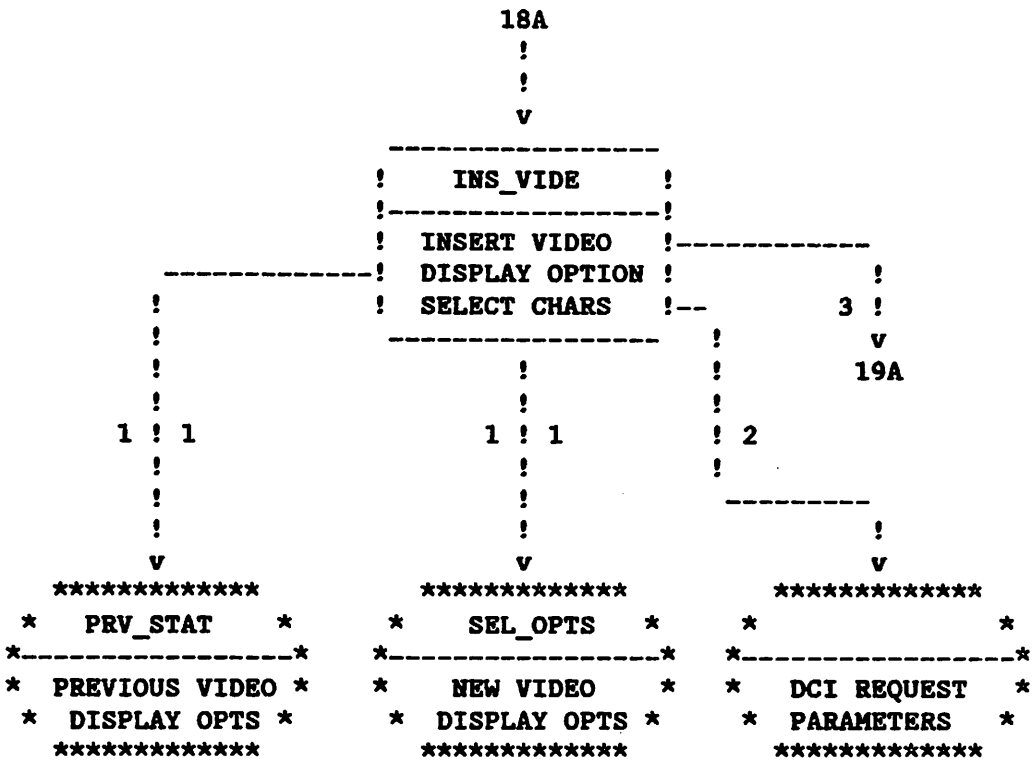
1. Request Completion flags.
2. Request flags (Request Parameters).
3. Data Access flags.
4. Backspace character.



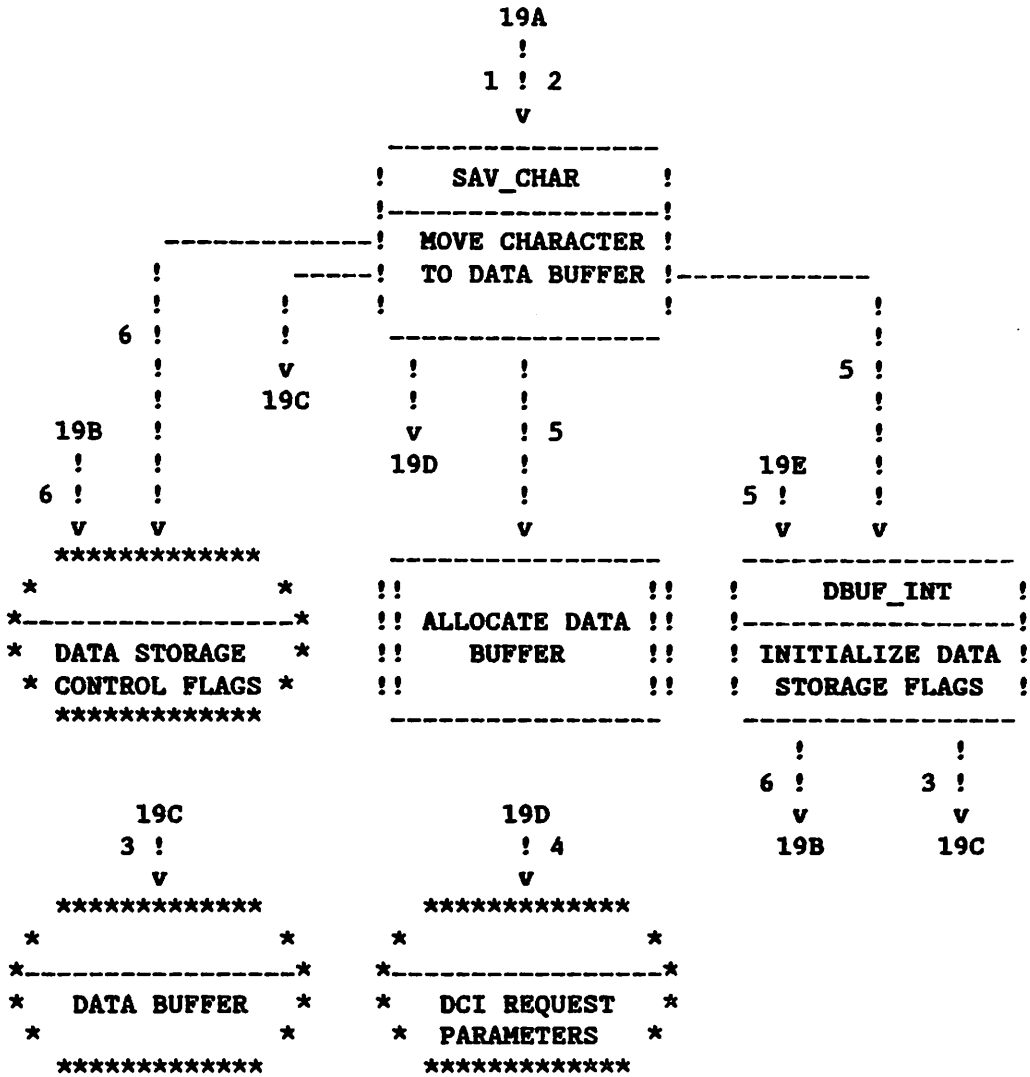
1. Character/Associate Video Display Characteristics
2. Character/Editted Video Display Characteristics
3. Request Processing Flags (Parameter List)



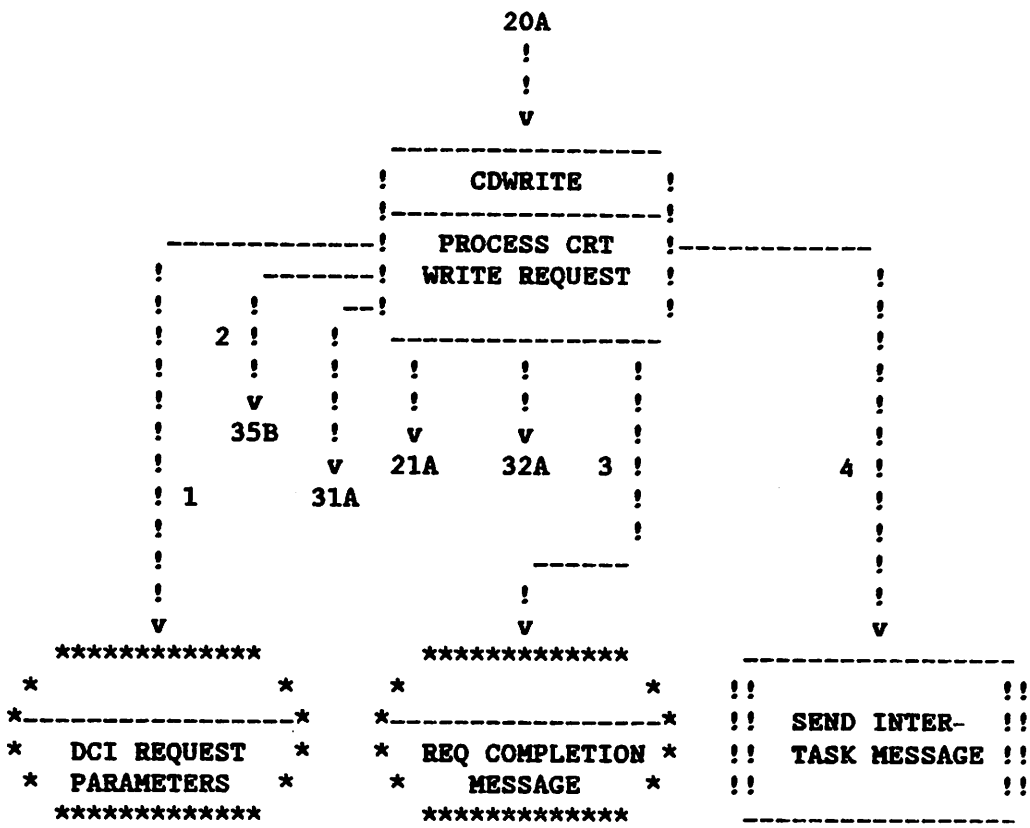
1. Prosign character, Cursor Co-ordinates.
2. Cursor Coordinates.



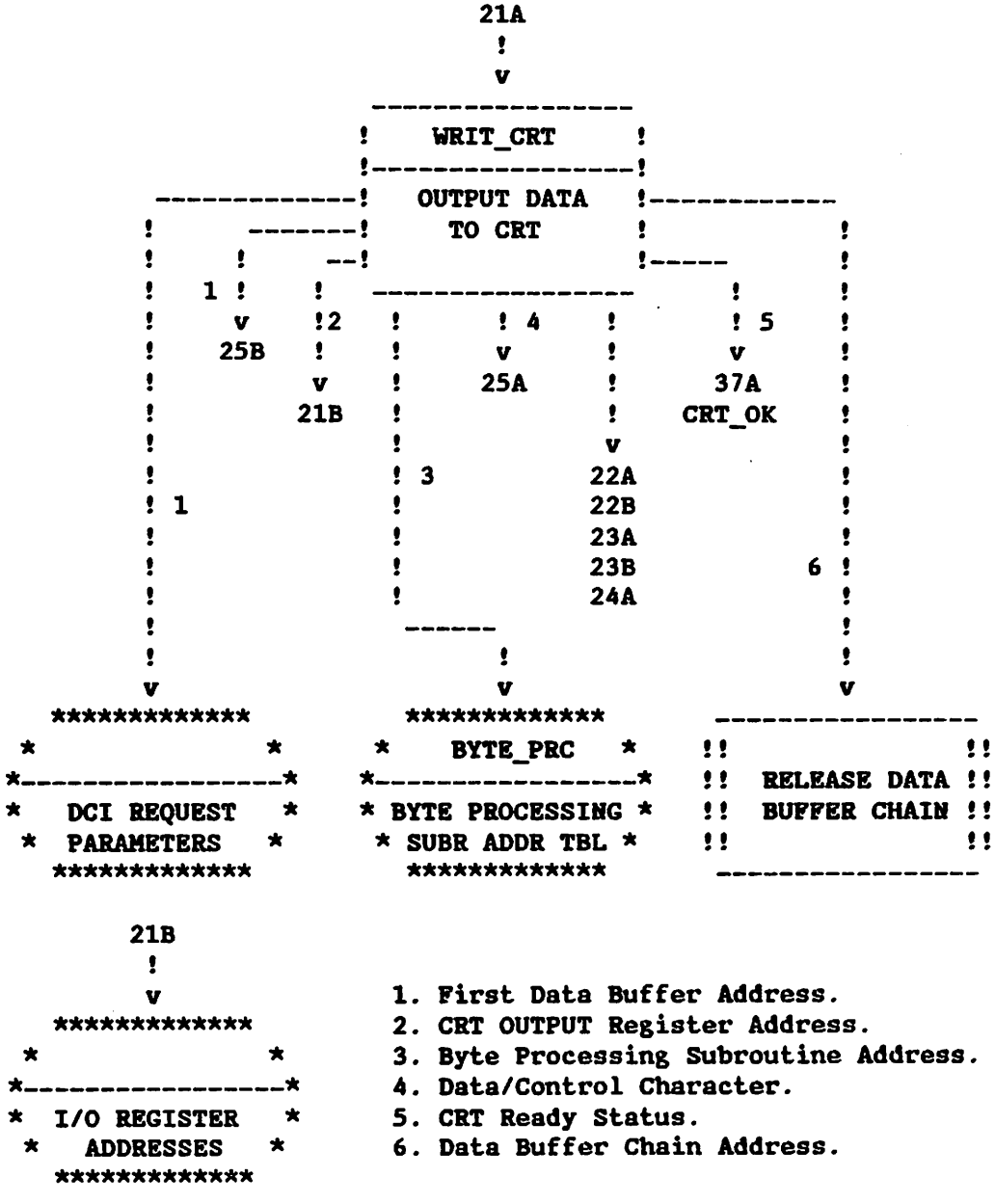
1. Character/Associated Video Display Characteristics.
2. Request flags (Request Parameters).
3. Video Mode Select characters.



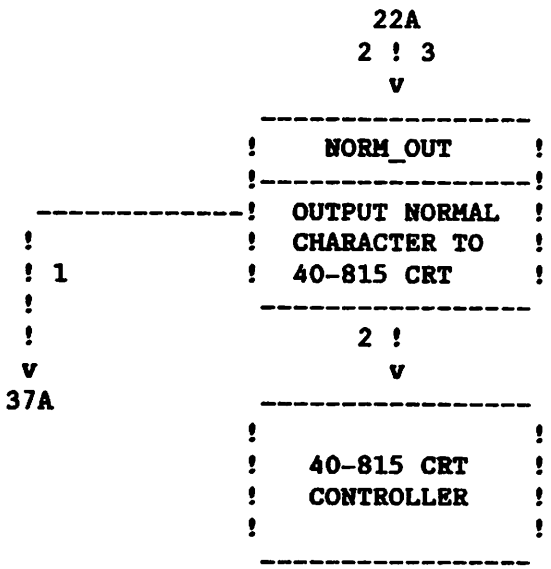
1. Data/Control character. 2. Completion code.
3. Data/Control Information. 4. Request flags.
5. Data Buffer Address. 6. Data Storage Control Info.

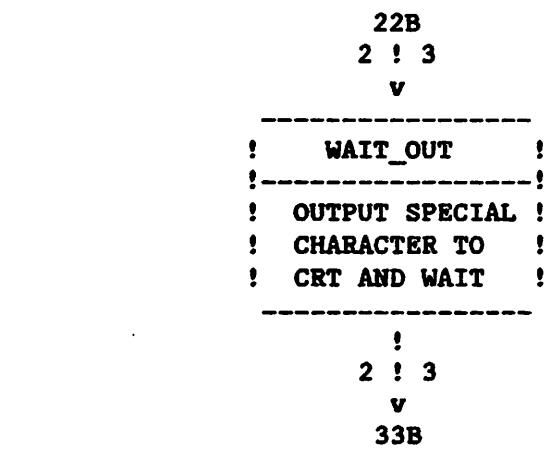


1. Request flags (Parameter List).
2. CLEAR MODES control character.
3. Request Type, CRT Identifier, Completion Code.
4. Message Buffer Address.

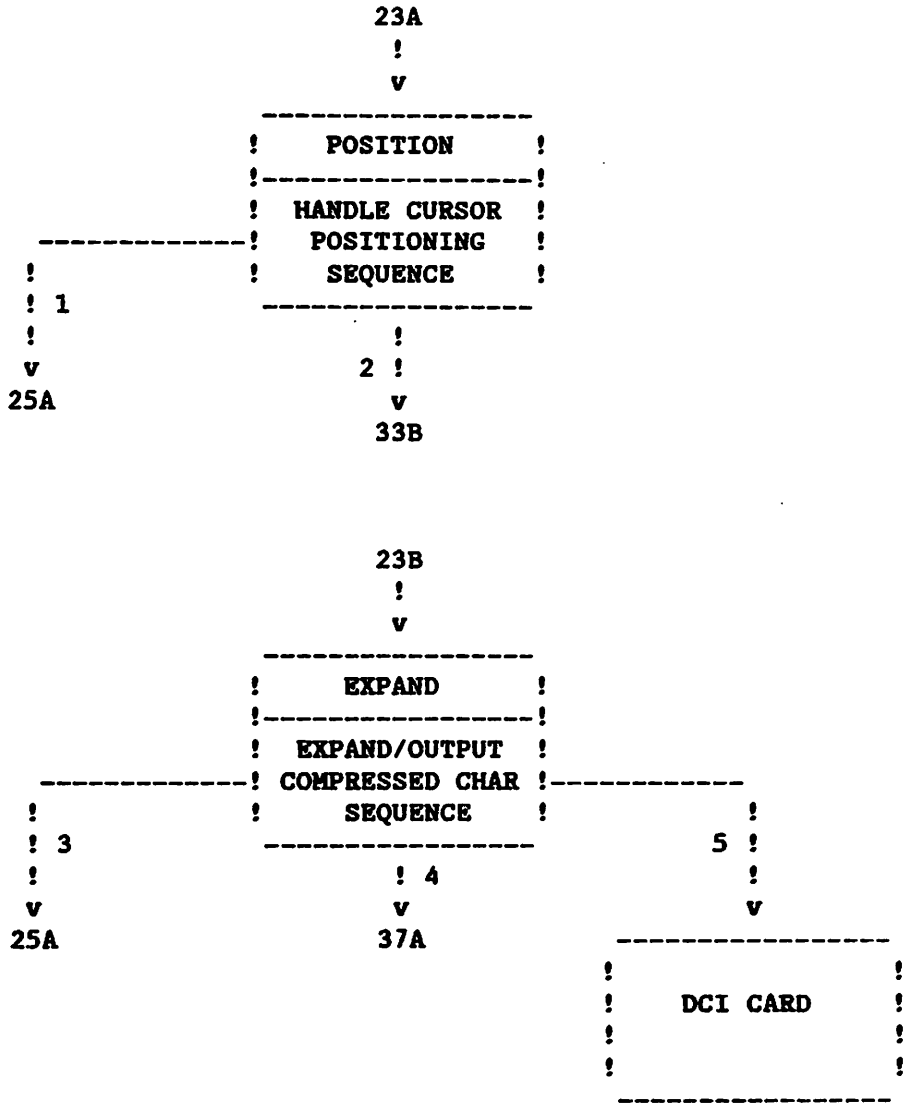


1. First Data Buffer Address.
2. CRT OUTPUT Register Address.
3. Byte Processing Subroutine Address.
4. Data/Control Character.
5. CRT Ready Status.
6. Data Buffer Chain Address.

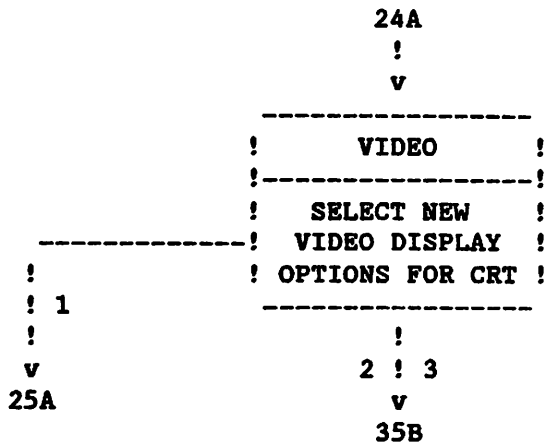




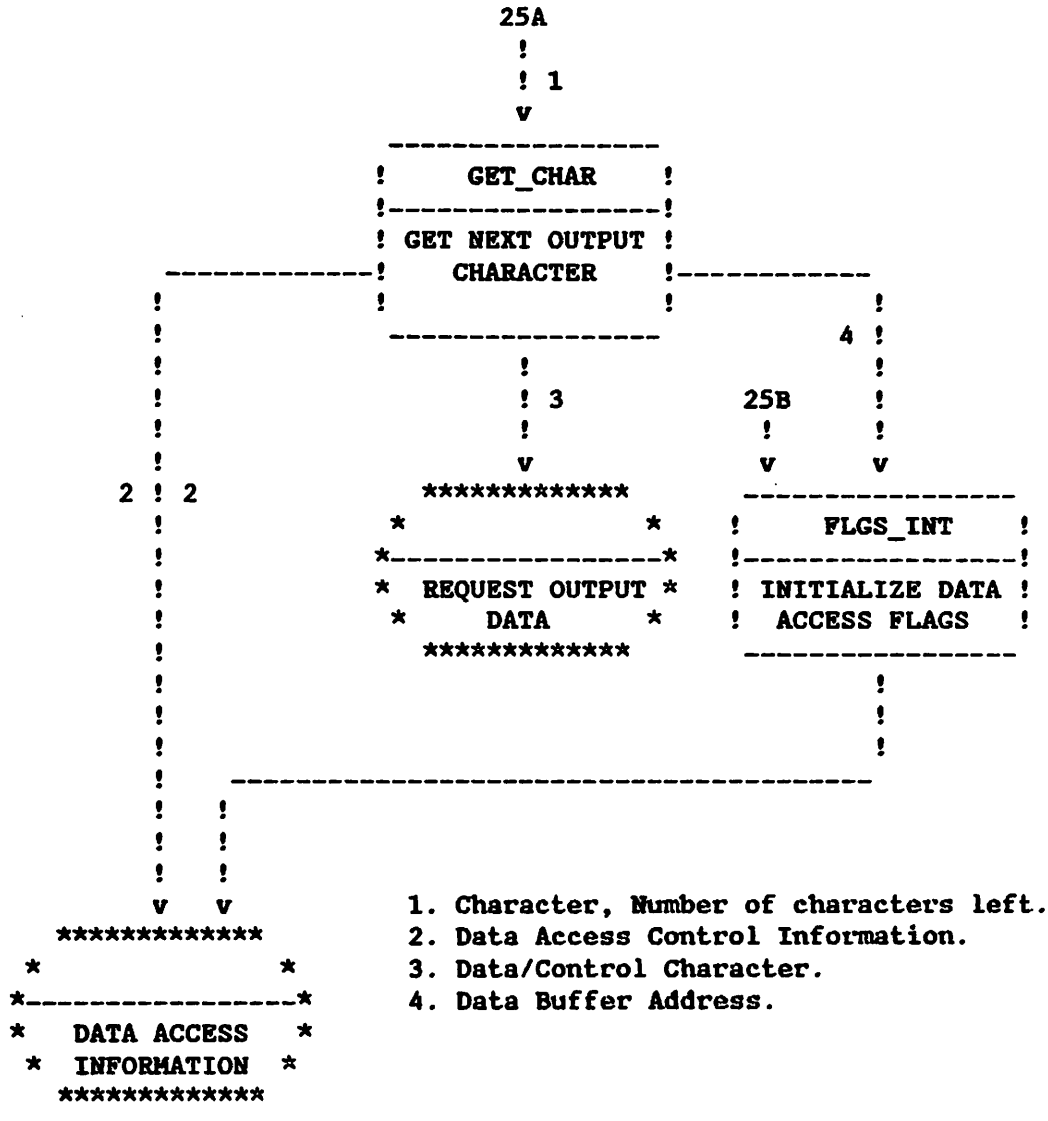
1. CRT Ready Status. 2. Data/Control character.
3. Completion Indicator.



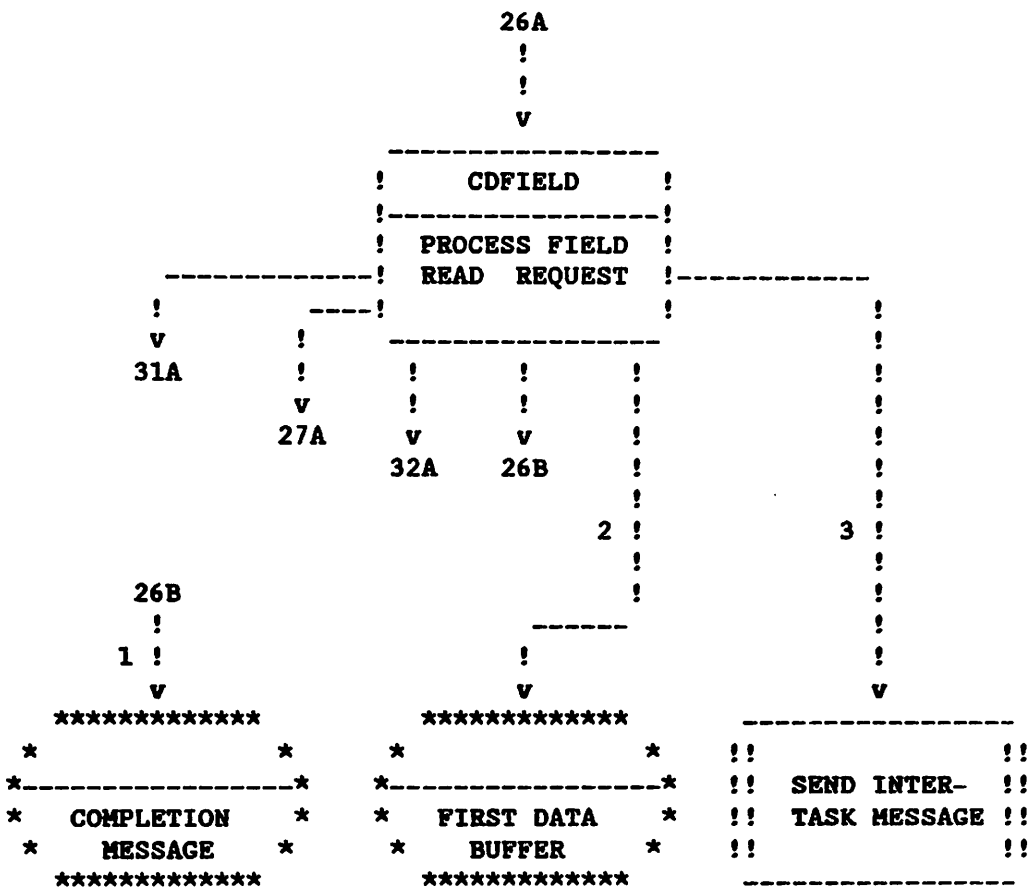
1. Cursor Co-ordinates.
2. Prosign character, Cursor Co-ordinates.
3. Repeated character, repeat count.
4. CRT Ready Status.
5. Prosign Character, Repeated character.



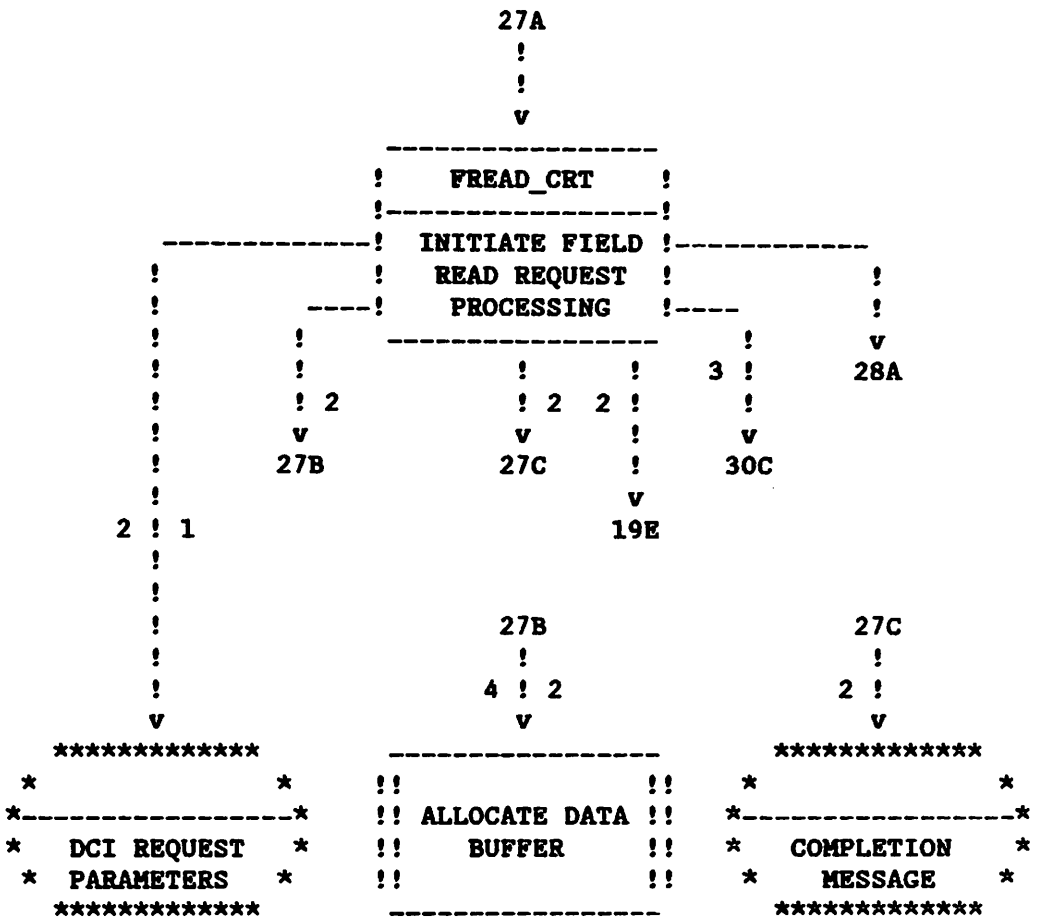
1. Video Display flags.
2. Video Mode Select Characters.
3. Completion Indicator.



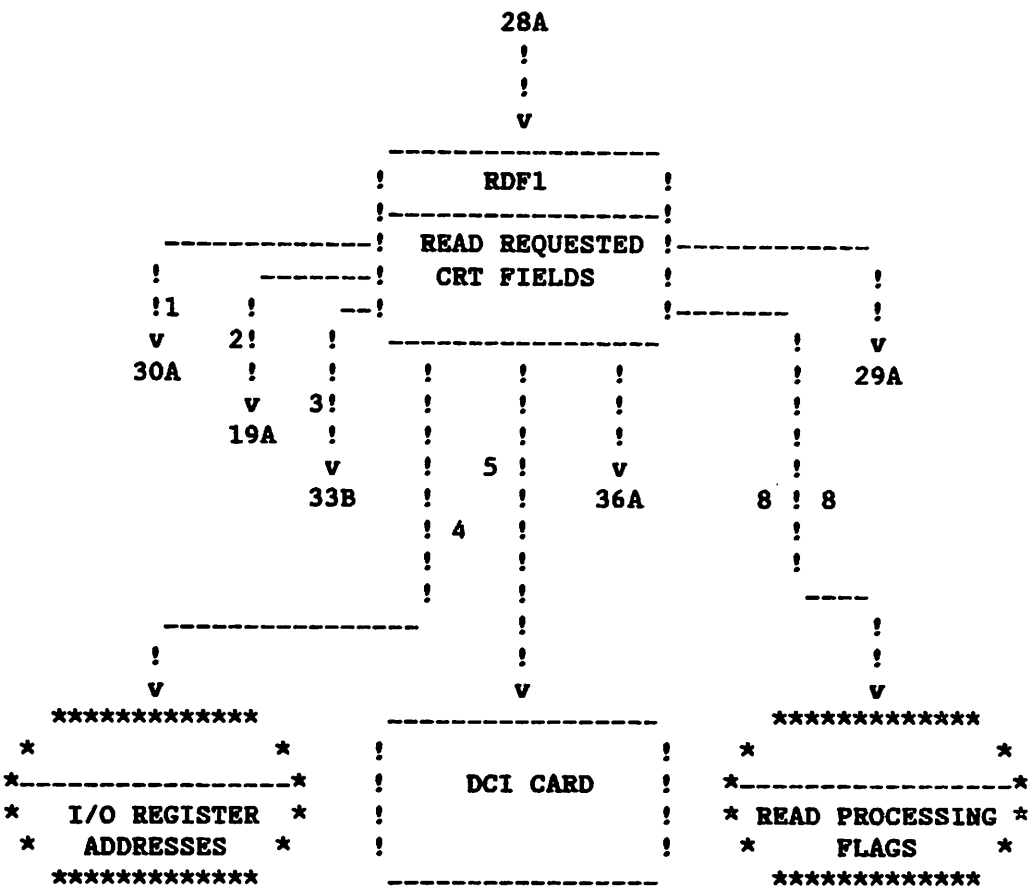
1. Character, Number of characters left.
2. Data Access Control Information.
3. Data/Control Character.
4. Data Buffer Address.



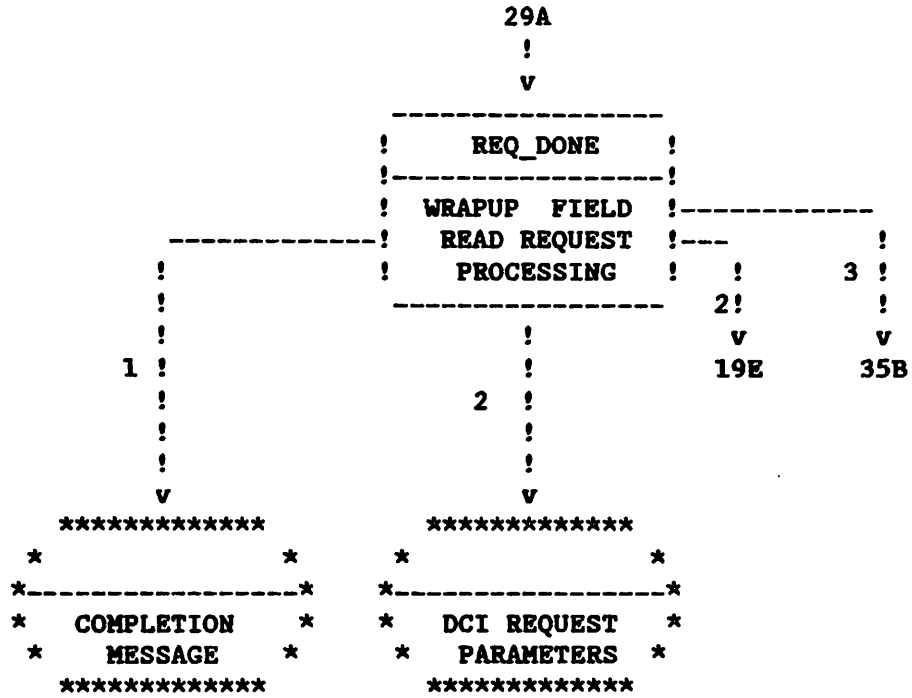
1. Request Type, CRT Identifier, Completion Code.
2. Number of Bytes of Data.
3. Completion Message Buffer Address.



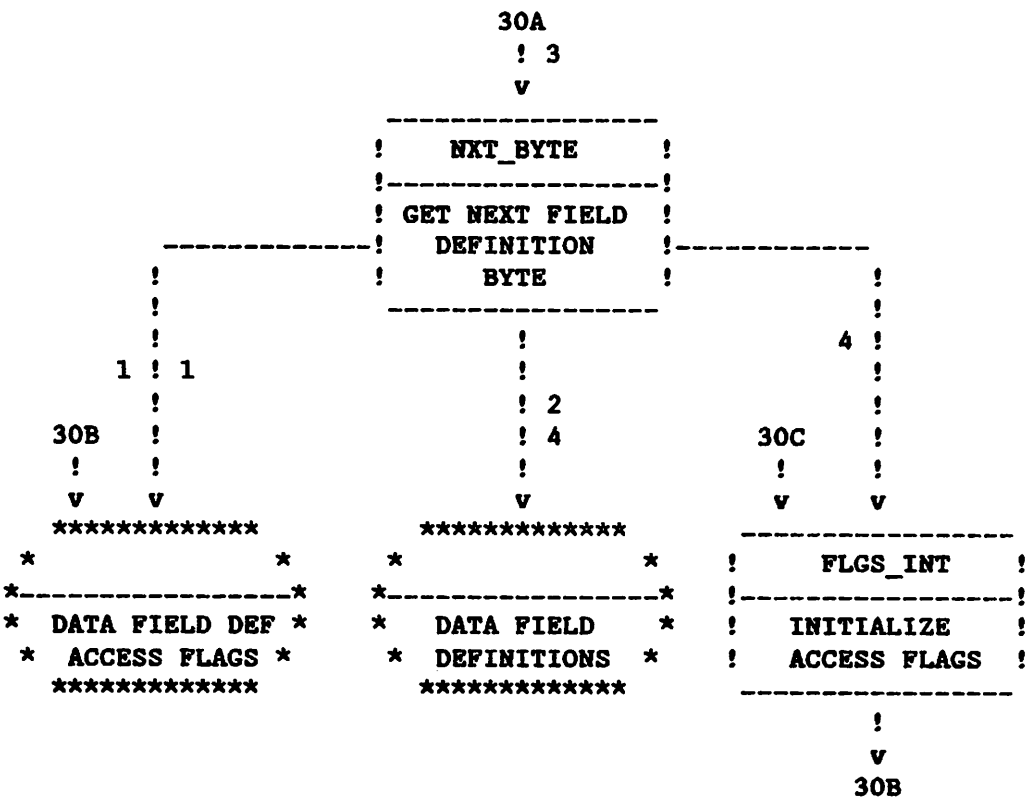
1. First Data Buffer Address, Field Definitions Data Buffer Chain Address.
2. Data Buffer Address.
3. Field Definitions Data Buffer Chain Address.
4. Number of Data Buffers Wanted.



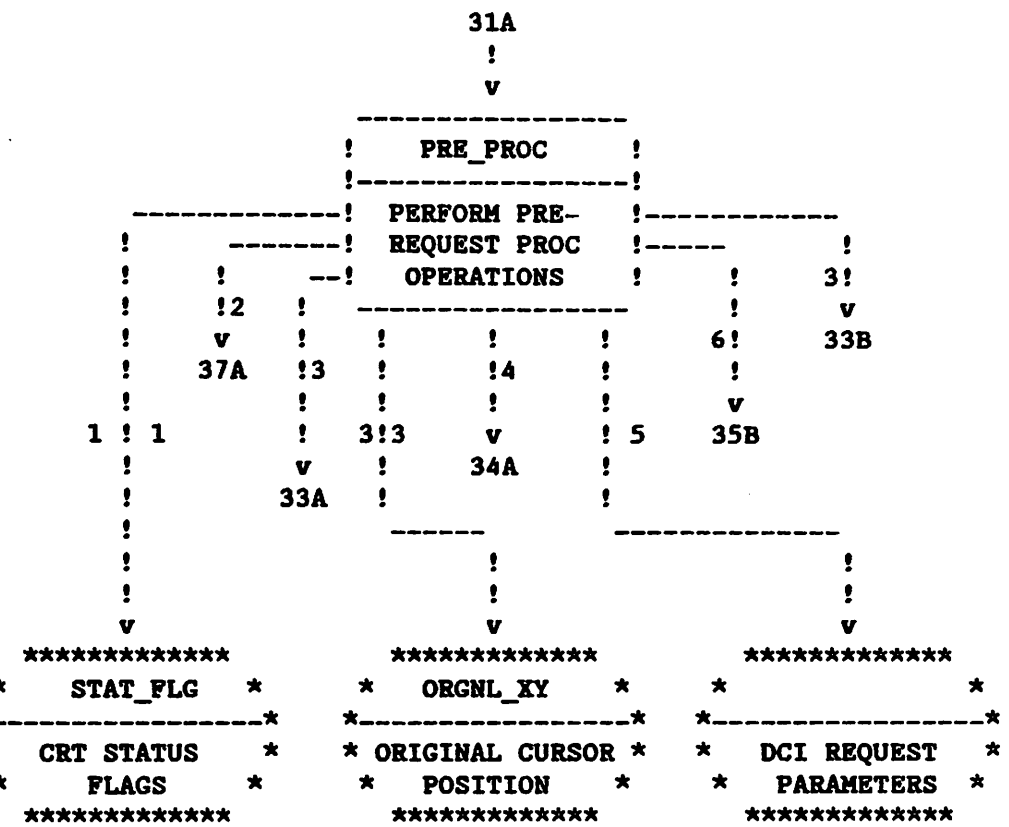
1. DATA DEFINITION FIELD character.
2. Data/Control Character.
3. Cursor Co-ordinates.
4. DCI COMMAND TWO Register Addr, CRT INPUT Register Addr.
5. Input Character and Associated Video Characteristics.
6. Character/Associated Video Display Characteristics.
7. DCI Ready Status.
8. Request Processing Control Information.



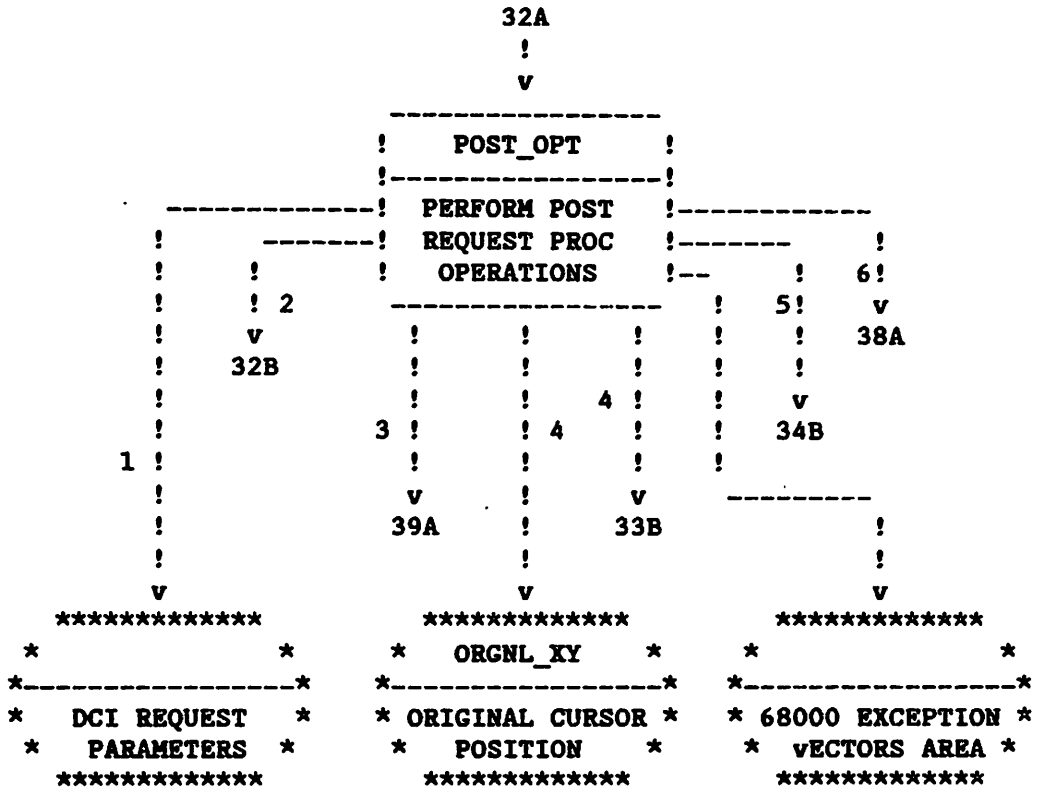
1. Completion Code.
2. Data Buffer Address.
3. Backspace character.

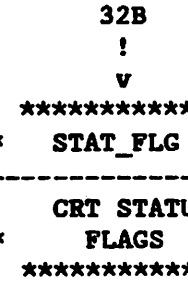


1. Data Field Definitions Access Flags.
2. Data Field Definitions.
3. Data character.
4. Next Data Buffer Address.

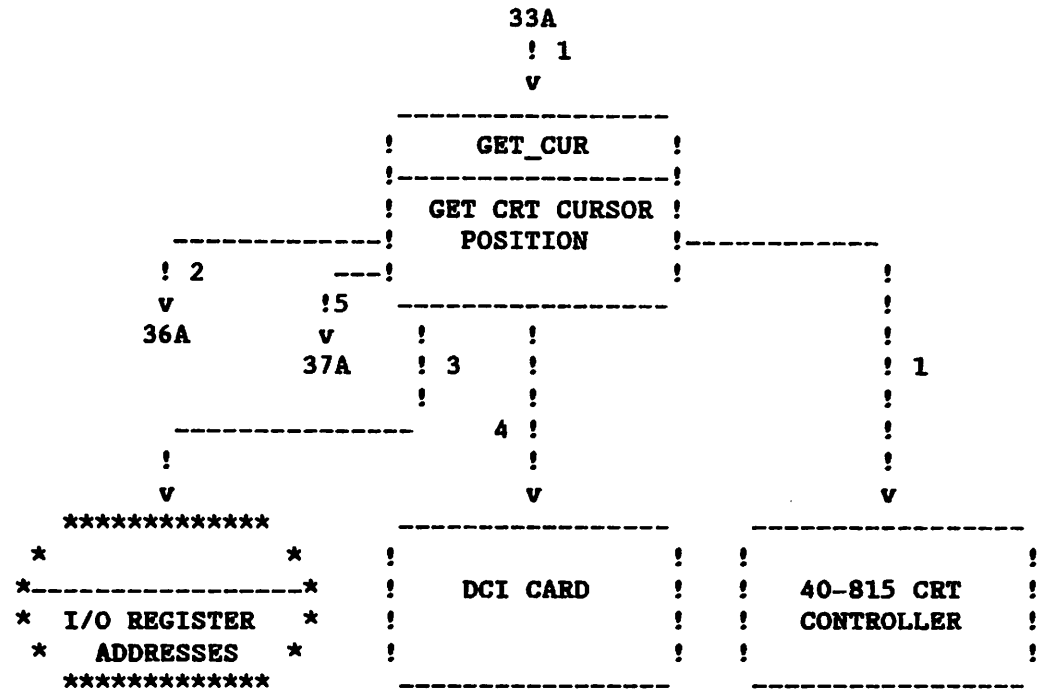


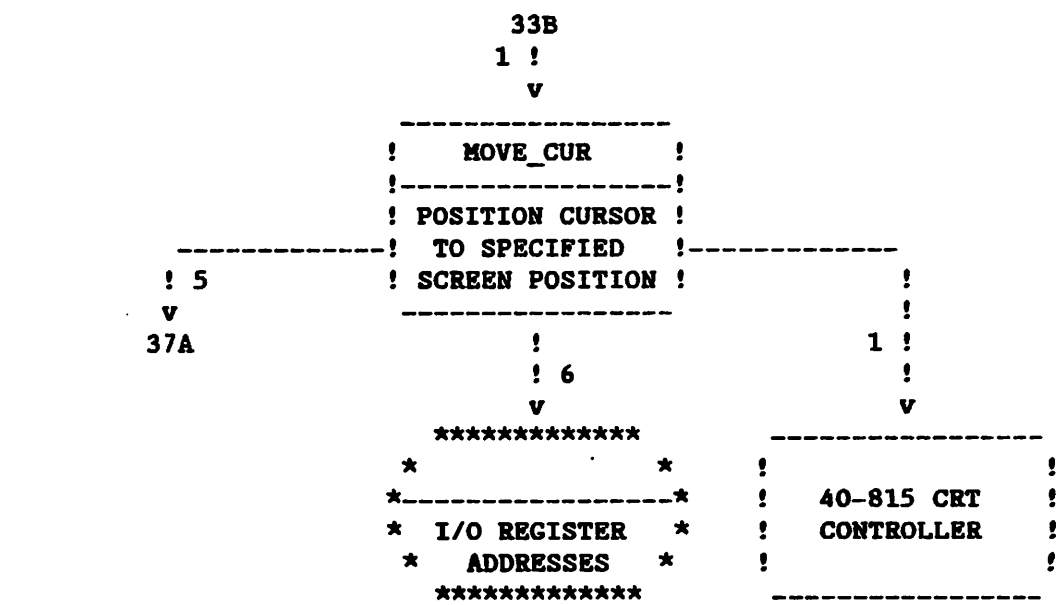
1. Original CRT Status flags.
2. CRT Ready Status.
3. Cursor Co-ordinates.
4. Selected Video Display Modes.
5. Request Type, Request Flags.
6. Clear Screen control character.



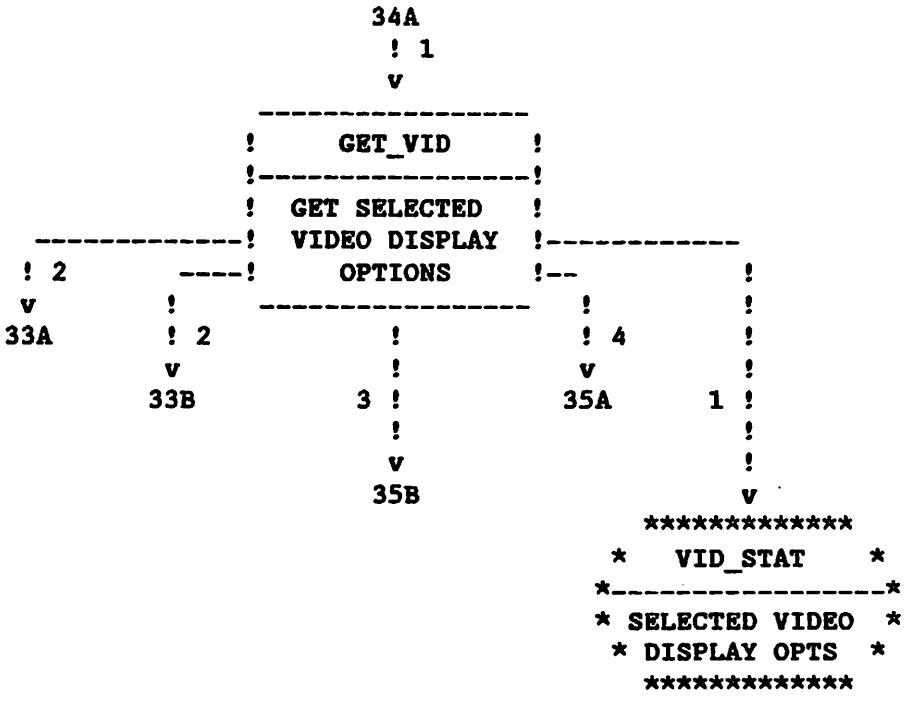


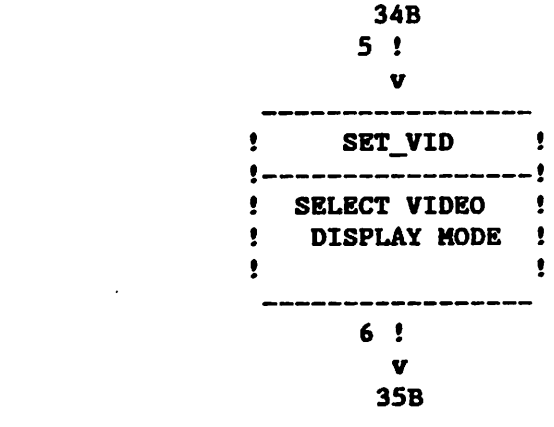
1. Request flags.
2. CRT Unit Status flags.
3. Video Mode flags.
4. Cursor Co-ordinates.
5. Video Mode Select code.
6. CRT Function Command code.



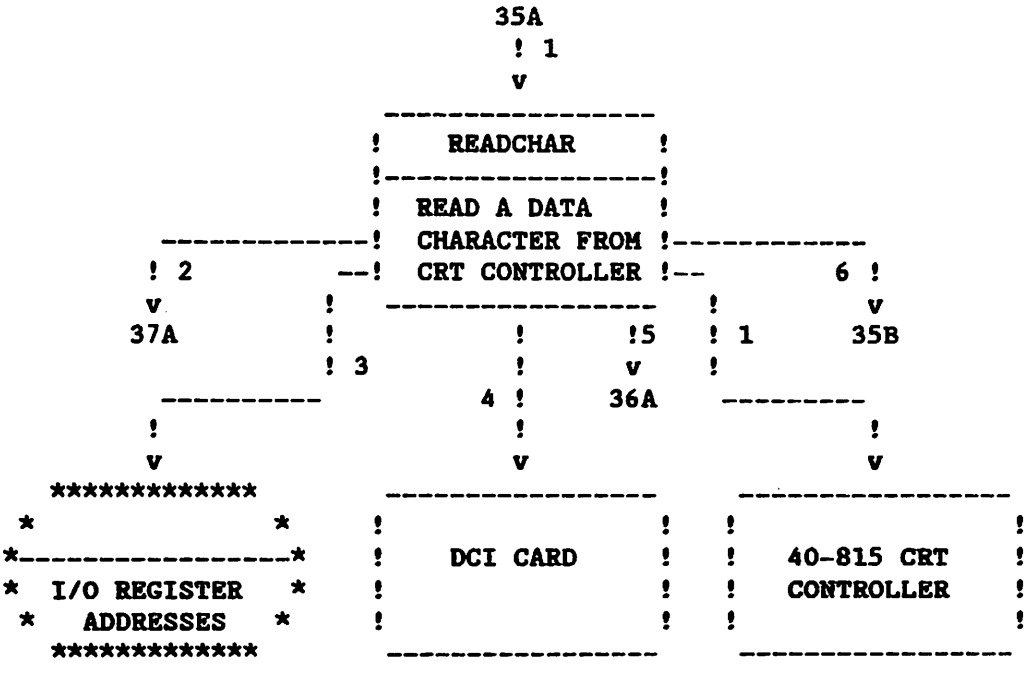


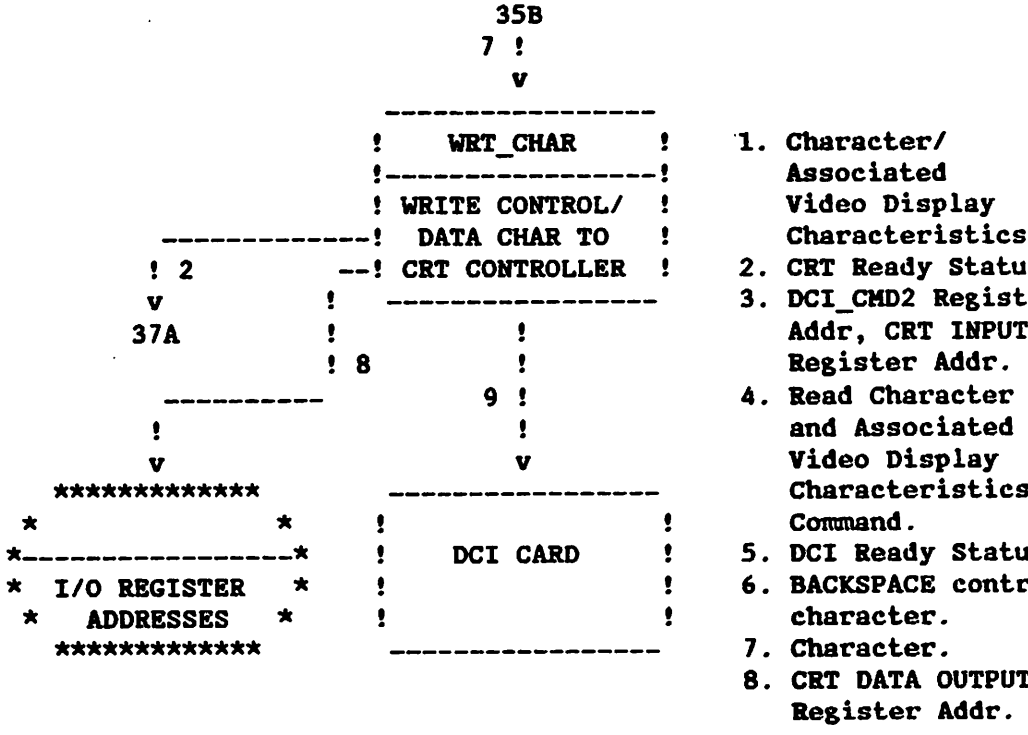
1. Cursor Co-ordinates. 2. DCI Ready Status.
3. DCI_CMD2 Register Addr., CRT INPUT Register Addr.
4. INPUT CURSOR POSITION Command. 5. CRT Ready Status.
6. CRT CONTROL FOUR Register Address.



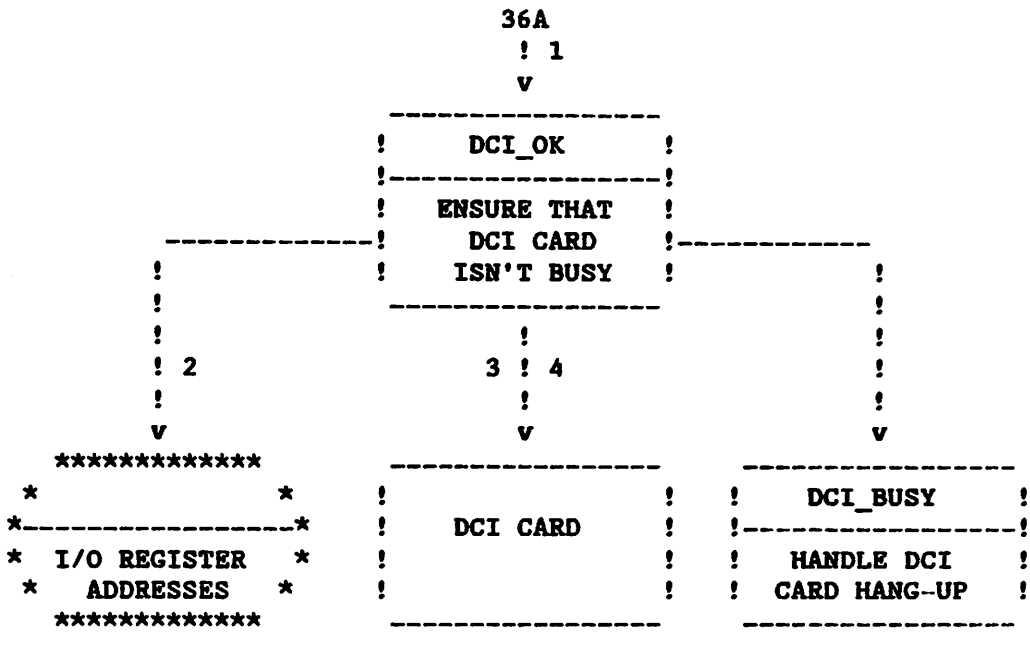


1. Video Display Mode flags. 2. Cursor Position.
3. Space, Backspace characters.
4. Space character/Associatied Video Display Characteristics.
5. Video Mode Select code. 6. Video Mode Select characters.

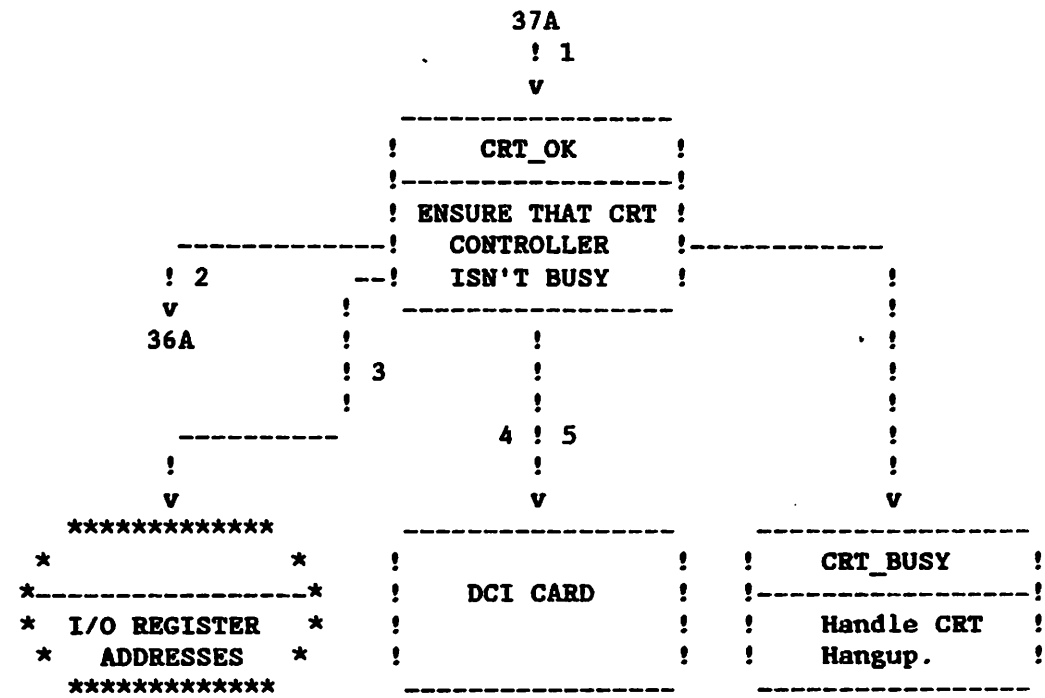




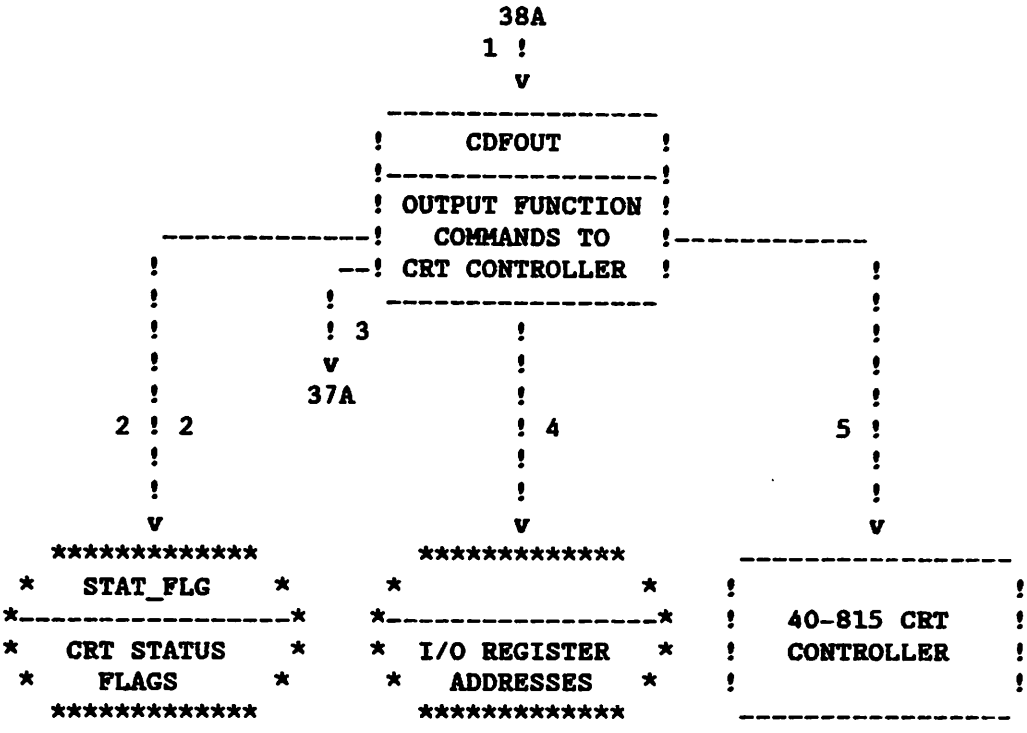
1. Character/ Associated Video Display Characteristics.
2. CRT Ready Status.
3. DCI_CMD2 Register Addr, CRT INPUT Register Addr.
4. Read Character and Associated Video Display Characteristics Command.
5. DCI Ready Status.
6. BACKSPACE control character.
7. Character.
8. CRT DATA OUTPUT Register Addr.



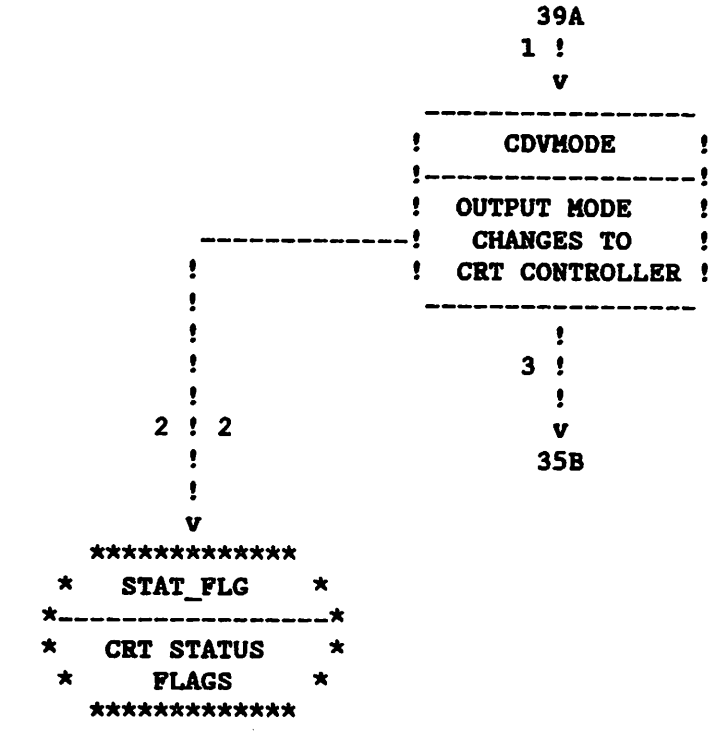
1. DCI Ready Status.
2. DCI STATUS TWO Register Address.
3. INPUT DCI STATUS TWO Command.
4. DCI STATUS TWO.



1. CRT Ready Status.
2. DCI Ready Status.
3. DCI COMMAND TWO Register Address, CRT INPUT Register Address.
4. Input CRT Status One Command.
5. CRT STATUS ONE.



1. CRT Function Command code.
2. CRT Unit Status flags.
3. CRT Ready Status.
4. CRT CONTROL ONE and CRT CONTROL THREE Register Address.
5. CRT CONTROL ONE and CRT CONTROL THREE bytes.



1. Video Display Mode Select code.
2. CRT Unit Status flags.
3. Video Mode Select control characters.

4.3.1 DCI Driver Initialization (CDSTART)

This module runs once when the DCI/CRT driver task is created by the Configuration Control Subsystem.

The driver initialization module performs the following functions:

a. It moves its Task Identifier from the RUNNING cell in the Executive's Configuration Information table to the DCI_TASK cell.

   The DCI_TASK cell is a globally defined location that can be referenced by the other MPB resident application tasks that want to communicate with the DCI Driver.

b. It uses the CC_DCI_INFO_MAC to get Configuration Control to move the DCI/CRT Hardware Configuration Information to the DCI_INFO table which the driver references for all the hardware configuration information that it needs.

c. It calculates the address(es) for the DCI COMMAND ZERO Memory Mapped I/O register(s) associated with the DCI card(s) configured in the MCU. These Memory Mapped I/O register address(es) are saved in the DCI1CMD0 and DCI2CMD0 cells.

   These register addresses are needed to enable and disable interrupts from the DCI during normal request processing. They are also needed by the DCI interrupt processing subroutines.

c. It calculates the address(es) for the CRT DATA/STATUS INPUT Memory Mapped I/O register(s) that are associated with the DCI card(s) configured in the MCU. These Memory Mapped I/O register address(es) are saved in the DCI1INPT and the DCI2INPT cells.

The address ot the CRT DATA/STATUS INPUT register associated with a DCI card is required by any section of the driver that needs to obtain data/information from a CRT controller connected to that DCI card.

d. It takes control of the Exception vector(s) that are associated with the ISB slot(s) occupied by the DCI card(s) and supplies the Interrupt Processing subroutine to be used for these vectors.

There is an Executive directive that is used to perform these functions. Since the vectors table is located in MPB RAM and the DCI Driver is an application task, the DCI Driver can not change the vector table itself because it is running in USER mode.

e. It tells each DCI card which Exception vector it is to use.

f. It enables interrupts from each DCI card.

g. It transfers control to Driver's Main Control Section.

4.3.2 Main Control (CDRIVER)

The Main Control section interfaces with the Executive using a RECEIVE NORMAL MESSAGE directive to get the next application request to be processed. This user interface ensures that requests are processed on a First In/First Out basis. The request parameter list is received into an area that can be accessed by all the other parts of the driver.

When a request is received, the DCI Driver first gets the CRT's CYBER Equipment Number (CEN) from the request parameter message and then uses the CC_CRT_STAT_MAC macro to find out from Configuration Control if the specified CRT unit is available for use (UP). If the CRT is marked DOWN, an error completion message is sent back to the requesting task. Otherwise, request processing continues.

The next step that the DCI Driver performs is to disable interrupts from all the DCI cards configured in the MCU. This is necessary because we can't allow two users to communicate with a DCI card at the same time because of the serial nature of the hardware. Many logical interface operations with the DCI/CRT hardware involve a series of physical operations that can't be interrupted. Thus, request processing is never allowed to be interrupted. Function Key interrupts aren't lost, but are simply held out until the current request completes.

Then the DCI Driver uses the CC_CRT_INFO_MAC macro to get the Configuration Control subsystem to convert the CRT CEN parameter to the corresponding physical address and relative CRT number. The physical address for a CRT consists of the following two components:

a. ISB slot number for the DCI card to which the CRT is connected.

b. CRT Controller Address (0-15).

The relative CRT number identifies a CRT relative to the MCU to which the CRT is connected. Since an MCU can be configured with either one or two DCI cards and each DCI card can have up to 16 CRTs connected to it, this relative CRT number can range from zero to thirty-one. The relative CRT number is used to access the Driver's tables used to keep track of the CRT Unit Status Information.

Once the physical address of CRT has been obtained, the addresses for the Memory Mapped I/O Registers required to communicate with the DCI card and the CRT controller are calculated. The Memory Mapped I/O Registers for which addresses are calculated and the cells where these address are saved are listed here:

- o DCI COMMAND/STATUS ZERO register -- (DCI_ZERO).
- o DCI COMMAND/STATUS ONE register -- (DCI_ONE).
- o DCI COMMAND/STATUS TWO register -- (DCI_TWO).
- o DCI COMMAND/STATUS THREE register -- (DCI_THREE).
- o CRT DATA/STATUS INPUT register -- (CRT_IN).
- o CRT DATA OUTPUT register -- (CRT_OUT).
- o CRT CONTROL ONE OUTPUT register -- (CONTRL_1).
- o CRT CONTROL THREE OUTPUT register -- (CONTRL_3).
- o CRT CONTROL FOUR OUTPUT register -- (CONTRL_4).
- o CRT CONTROL FIVE OUTPUT register -- (CONTRL_5).

The cells where these Memory Mapped I/O Registers are saved can be accessed globally by other driver sections.

A DCI COMMAND ONE skeleton is set up in the DCI_CMD2 cell to make it easy to generate DCI COMMAND TWO bytes. DCI COMMAND TWO bytes are to used to tell the DCI card to obtain status informationand data from the CRT.

A copy of the CRT's Unit Flags is obtained from the UNT_STAT table and saved in the STAT_FLG cell to make these flags easier to access. After the processing of the request has been completed, the local copy of the CRT's Unit Status flags is used to update the CRT's entry in the UNT_STAT table.

Next the request type code is validated to ensure that it is a request type that this driver can handle. If the request code is bad, a CHK Instruction failure is caused to occur. And then, the driver subroutine designed to process this type of request is called. No parameters are passed to the called subroutine because the request parameters and any other required processing information is stored in globally accessable areas. It is the called subroutine's responsibility to process the request and send a request processing completion message back to the requesting task.

After the called subroutine has completed its processing, it returns control to this Main Control section. When control is received back, DCI interrupts are re-enabled first and then control is transferred to the beginning portion of this module to get the next user request to be processed.

4.3.3 DCI Interrupt Processing (CDINTER)

The driver contains an interrupt processing subroutine for each DCI card. If there are two DCI cards, the two DCI interrupt processing subroutines share a majority of their code. A DCI Interrupt Processing subroutine performs the following functions:

a. Disables interrupts from interrupting DCI card.

b. Inputs status from CRT Controllers connected to DCI to determine which one caused the interrupt.

c. Determines cause of the CRT controller interrupt:

   1. FUNCTION KEY.
   2. LITE PEN.
   3. END-OF-OPERATION condition.

d. Processes interrupt that occurred.

   1. FUNCTION KEY interrupt -- The number of the FUNCTION KEY depressed is determined by inputting the STATUS THREE byte from the CRT controller first. Next, the CRT controller's keyboard and lite pen functions are disabled by outputting the appropriate CONTROL ONE and CONTROL THREE function bytes to the CRT controller. The CRT's UNIT STATUS table entry is then updated to reflect the fact that its keyboard and Lite Pen have been disabled. And finally, a FUNCTION KEY message is sent to the Function Key Interrupt Processing task to let it know about the function key depressed. The format of this message was described in this document earlier.
   2. LITE PEN interrupt -- It is assumed here that the reader has read the beginning portion of this document and knows that the Lite Pen is a hardware device that can be used to move a CRT's cursor to any position on the CRT screen. This is accomplished by touching the desired

screen position with the Lite Pen. When a LITE PEN interrupt occurs, the CRT's STATUS FOUR and STATUS FIVE bytes are input to determine the X,Y co-ordinates for the screen position touched by the lite pen. The CRT cursor is then moved to this position by outputting the appropriate CONTROL FOUR and CONTROL FIVE bytes to the CRT controller.

3. END-OF-OPERATION interrupt -- The driver never enables the END-OF-OPERATION interrupt and this interrupt is simply ignored if it should occur.

e. Restore interrupted CPU user's registers and exit the interrupt processing state.

NOTE: Since no significant processing information is contained in the DCI card status bytes, no time is wasted inputting and saving this information.

4.3.4 CRT Status Processing (CDSTATS)

This section processes 40-815 CRT Status requests and returns the CRT's status information to the requesting task in the request completion message.

A CRT Status request is proceesed in the following fashion:

a. The specified CRT is statused to get its READY and NOT BUSY status bits and this information is moved to the request completion message.

b. The CRT's Unit Status flags are obtained from the Unit Status Flags table and moved to the request completion message.

c. The character located at the CRT's current cursor postion is read and the character's associated video display characteristics are input from the CRT. This information is moved to the request completion message.

d. The CURSOR is restored to its orginal screen position (Reading the character in previous step caused cursor to move forward) by outputting a BACKSPACE control character to the CRT controller.

e. The X,Y co-ordinates for the current position of the CRT's CURSOR are input and moved to request completion message.

f. The Completion message is sent to requesting task.

g. Control is returned to Main Control section.

4.3.5 CRT Function Processing (CDFNCTN)

This section processes 40-815 CRT Function requests and returns a completion message to the requesting task to indicate whether or not the request was successfully processed.

There are two different types of function commands:

a. The Control Funtion Output commands

   1. Set/Clear Alarm Output Contact.
   2. Initiate Copier Operation.
   3. Enable/Disable Lite Pen.
   4. Enable/Disable Keyboard/Function Keys.
   5. Master Clear CRT Controller.

b. The Operational Mode Change commands

   1. Clear All Currently Selected Operational Modes.
   2. Select Insert Character in Field/Page Mode.
   3. Select Insert Character in Field/Line Mode.
   4. Select Scroll-Up Mode.
   5. Select Scroll-Down Mode.
   6. Select Scroll-Up Mode.

The first type of function commands are output to the CRT Controller using function outputs that the Controller is designed to process. The second type of function commands are output to the CRT Controller by outputting the appropriate control characters.

A CRT Function request is processed in the following manner.

a. If requester wants any Operational Mode Change type function commands output, the MODE_SEL subroutine is called.

b. If requester wants any Normal Function type commands output, the FUNC_SEL subroutine is called.

c. If requester wants CURSOR moved to a specified screen position, the MOVE_CUR subroutine is called.

d. A Request Completion message is generated and sent to the requesting task.

e. Control is returned to the Main Control section.

4.3.6 CRT Data Read Processing (CDREAD)

The CDREAD subroutine handles the processing of a CRT Read request. A CRT Read request is processed as follows:

a. The PRE_PROC subroutine is called to perform all necessary pre-read operations. These operations include:

   1. Disabling the CRT controller's keyboard and lite pen.
   2. Inputting and saving the X,Y co-ordinates for the position of the CRT's cursor.
   3. Inputting and saving CRT's selected video display options.
   4. Moving cursor to a screen position that the requester has specified (optional).

b. The READ_CRT subroutine is called to actually input the data. The READ_CRT subroutine should be referred to for a discussion of the various options that may be selected during the data input operation.

c. The POST_OPT subroutine is called to perform all necessary post-read operations. These operations include:

   1. Returning CRT controller to the appropriate protected/ unprotected mode of operation.
   2. Positioning the CRT controller's to the position specified by the requester. The requester can specify that the cursor be moved to the HOME position, its pre-read position, or its current position.
   3. Optionally reselecting the video display options that were selected for the CRT controller prior to the read operation.
   4. Optionally enabling/disabling the CRT controller's keyboard/function keys.
   5. Optionally enabling/disabling the CRT controller's lite pen unit interrupt.

6. Optionally re-storing the CRT controller's keyboard/ function keys to their pre-read operation enabled/disabled state.
7. Optionally re-storing the CRT controller's Lite Pen unit interrupt to its pre-read enabled/disabled state.

d. A request completion message is send back to the task that requested the read operation. This message contains both read completion flags and the address of the chain of Data buffers used for the data.

e. Control is returned to the Main Control section.

4.3.6.1 CRT Data Read (READ_CRT)

This subroutine reads a specified number of characters beginning at a specified point on the CRT screen. The data read is saved in a chain of Data buffers which is either provided by the requester or allocated by this subroutine depending upon the option specified.

If the requester wants the video display characteristics (color, blink mode, and inverse video mode) associated with the data characters read to be preserved, this subroutine inserts the necessary video Display Option Select control characters into the read data. At the requester's option, this subroutine also suppresses strings of repeated, non-inverse space characters; these these character strings are replaced by POSITION CURSOR commands. At the requester's option, this subroutine also compresses repeated character strings; these strings are replaced by EXPAND COMPRESSED STRING commands. The CRT Write Request Processing subroutine scans for these commands and processes the ones found.

4.3.6.2 Edit Character/Video Status (EDIT_CHR)

This subroutine performs the following two basic functions:

a. If the current character is a non-inverse mode space character and the requester wants the video display characteristics preserved, the previous character's blink and color video display options are substituted for the current character's blink and color. (This has effect of not allowing a space character to cause a non-seeable change in the selected video display options).

b. Replace the END-OF-LINE flag in the character's video status flags with a TABBED POSITION flag. This step simplifies the procedure required to preserve tabbed position information during a read operation.

4.3.6.3 Move Position Cursor Command to Buffer (POS_SEQ)

This subroutine simply moves the POSITION CURSOR command prosign character and the X and Y co-ordinates for the current position of the CURSOR to the read data buffer. The SAV_CHAR subroutine is used to move these characters to the read buffer. The X and Y co-ordinates for the current position of the CURSOR are obtained from the X_CURSOR and Y_CURSOR cells.

4.3.6.4 Move Video Control Characters to Buffer (INS_VIDE)

This subroutine preserves the video display characteristics of the data being read. Prior to moving a data data character to the data read buffer, this subroutine is called to determine which video Display Option control characters are necessary to preserve the character's visual display characteristics and move these control characters to the data read buffer.

When called the first time for a request, this subroutine moves control characters for all the selectable video display characteristics to the data read buffer. For the remaining calls, video Display Option control characters are moved to the read buffer only when necessary because of a change in video display characteristics from the previous character moved to the read buffer.

This subroutine also determines if the character was read from a tabbed screen position and moves a SET TAB control character to the data read buffer when this is the case.

4.3.6.5 Move Character to Buffer (SAV_CHAR)

This subroutine saves a character in a Data buffer and updates the buffer accounting information flags to reflect the data moved to the buffer. Prior to the first call to this subroutine, the DBUF INT subroutine must have been called to set up the Data Storage Control flags used by this subroutine. The requesting task may have provided the Data buffers to be used or it may have requested that the DCI Driver allocate the Data buffers needed.

If this subroutine is called when the current Data buffer is full, it saves the character temporially and then determines how to proceed. If the requesting task has provided the Data buffers to be used, this subroutine either stores the character in next buffer if there is one or ignores the character and returns an error indicator. Otherwise if the driver is supposed to allocate the Data buffers needed, this subroutine allocates a Data buffer, threads it to the current one, calls the DBUF_INT subroutine to initialize this subroutine's control flags for the new Data buffer and finally saves the character in the new buffer.

The NUM_SAVD flag is incremented every time a character is saved by this subroutine.

4.3.6.6 Initialize Data Storage Flags (DBUF_INT)

This subroutine sets up a Data buffer's header and data storage areas to indicate that the Data buffer is empty and it also sets up the Data Storage Control flags used by the SAV_CHAR subroutine when it stores characters into this Data buffer.

The Data buffer is initialized to indicate that it is empty because the requester may have provided the Data buffers being used and we want the requester to be able to re-use Data buffers without having to zero out the buffer character counts.

The Data Storage Control flags set up for the SAV_CHAR subroutine include:

a. CHR_ADDR -- Address of first storage byte in Data buffer's data storage area.

b. BUF_ROOM -- Number of bytes available in the Data buffer's data storage area.

4.3.7 CRT Write Request Processing (CDWRITE)

The CDWRITE subroutine handles the processing of write requests. A CRT Write request is processed in the following fashion:

a. The PRE_PROC subroutine is called to perform all the necessary pre-write operations. These operations include:

   1. Disabling the CRT controller's keyboard and lite pen.
   2. Inputting and saving the X,Y co-ordinates for the position of the CRT controller's cursor.
   3. Inputting and saving the CRT controller's selected video display options.
   4. Optionally clearing the CRT controller's screen.
   5. Optionally positioning the cursor to a screen position that the requester has specified.

b. The WRIT_CRT subroutine actually outputs the data provided. The WRIT_CRT subroutine should be checked for a discussion of how embedded command sequences are processed.

c. The POST_OPT subroutine is called to perform all necessary post-write operations. These operations include:

   1. Returning CRT controller to appropriate protected/ unprotected mode of operation.
   2. Positioning the CRT controller's to the position specified by the requester. The requester can specify that the cursor be moved to the HOME position, its pre-write position, or its current position.
   3. Optionally reselecting the video display options that were selected for the CRT controller prior to the write operation.
   4. Optionally enabling/disabling the CRT controller's keyboard/function keys.
   5. Optionally enabling/disabling the CRT controller's lite pen unit interrupt.

6. Optionally re-storing the CRT controller's keyboard/ function keys to their pre-write state.
7. Optionally re-storing the CRT controller's Lite Pen unit interrupt to its pre-write state.

c. A completion message is send back to the task that requested the write operation. This message contains both write completion flags and the address the chain of data buffers (zero if driver was directed to release the data buffers).

d. Control is returned to the Main Control section.

4.3.7.1 CRT Data Write (WRIT_CRT)

This subroutine outputs the data to the specified CRT. The data is located in a chain of Data buffers. The requester passes the address of the Data buffer chain in the write request. The internal buffer character count fields in the Data buffers control the access of the output data. When all the data has been output, this subroutine releases the chain of Data buffers if this option has been selected.

As the data is being output, it is scanned for embedded POSITION CURSOR, EXPAND COMPRESSED STRING, and VIDEO SELECT commands. These embedded commands are recognized by their prosign characters. The CURSOR symbol (11H) is prosign character for the POSITION CURSOR command. The REPEAT symbol (12H) is the prosign character for the EXPAND COMPRESSED STRING command. The VIDEO symbol (13H) is the prosign character for the VIDEO SELECT command.

When a CURSOR character is found, the next two output characters are expected to be the X,Y co-ordinates for the new position to which the cursor is to be moved. The cursor is moved there by calling the MOVE_CUR subroutine. (Note: X,Y coordinates are not validated).

When a REPEAT character is encountered, the next output byte is the character which is be repeated and following byte is the number of times that it is be output. The character is simply output to the CRT the number of times specified.

When a REPEAT character is encountered, the next output byte is a set VIDEO MODES flags. The appropriate control character are output to the CRT to ensure that the VIDEO DISPLAY OPTIONS indicated the flags are selected at the CRT.

Normal data characters are of course simply output to the CRT.

Output Data SCanning Scheme

To simplify the output data scanning operation, a Character Classification table is used to classify each character as to its processing type. The current character is used as an index to this table to obtain the character's Processing Type code. A character's processing type code is then used to determine which section of this subroutine should process the current character and then control is transferred there.

4.3.7.2 Get Character from Data Buffer (GET_CHAR)

This subroutine obtains the next character from the requester's chain of data buffers. The data character is returned to the caller in bits 7-0 of the D7 register and the number of characters left in the current data buffer is returned to caller in bits 15-0 of the D6 register. (NOTE: A negative value returned in the D6 register indicates that all the output data has already been processed.)

A GET_CHAR subroutine call is processed in the following manner:

a. A check is made to determine if the current data buffer contains another unprocessed data character.

1. If current data buffer still contains unprocessed data characters, decrement the Number of Characters Remaining in Current Data Buffer flag (NUM_LEFT) and get the address of next character to be processed from the NXT ADDR flag.
2. If current data buffer is empty, a check is made to determine if there is another data buffer threaded to the current data buffer.
   a) If there is another data buffer threaded to the current one that needs to be processed yet, the FLGS_INT subroutine is called to re-initilize this subroutine's control flags so that the data in the next data buffer can be accessed and then Step A is repeated again.
   b) If there is no more output data, an error indicator is returned to the caller.

b. A character is obtained from current Data buffer and the Next Character Address flag (NXT_ADDR) is updated.

c. Control is returned to caller. The character and the number of characters remaining in current Data buffer are passed to the requester in the D7 and D6 registers.

4.3.7.3 Initialize Data Access Flags (FLGS_INT)

This subroutine initializes the control flags used by the GET_CHAR subroutine to control the access of data contained in a Data buffer.

An Initialize Data Buffer Access Flags subroutine call is processed as follows:

a. The address of the data buffer is saved in the Current Data Buffer Address flag (OUT_DBUF).

b. The address of the first byte is moved to the Next Output Character Address flag (NXT_ADDR).

c. The data buffer character count is used to initialize the Number of Characters Left in Current Data Buffer flag (NUM_LEFT).

d. Control is returned to the caller.

4.3.8 Multiple Field Read Processing (CDFIELD)

The CDFIELD subroutine handles the processing of a Multiple Field Read request. A Multiple Field Read request is processed as follows:

a. The PRE_PROC subroutine is called to perform all necessary pre-read operations. These operations include:

   1. Disabling the CRT controller's keyboard and lite pen.
   2. Inputting and saving the X,Y co-ordinates for the position of the CRT's cursor.
   3. Inputting and saving CRT's selected video display options.
   4. Moving cursor to a screen position that the requester has specified (optional).

b. The FREAD_CRT subroutine is called to actually read the screen fields specified.

c. The POST_OPT subroutine is called to perform all necessary post-read operations. These operations include:

   1. Returning CRT controller to the appropriate protected/ unprotected mode of operation.
   2. Positioning the CRT controller's to the position specified by the requester. The requester can specify that the cursor be moved to the HOME position, its pre read position, or its current position.
   3. Optionally reselecting the video display options that were selected for the CRT controller prior to the read operation.
   4. Optionally enabling/disabling the CRT controller's keyboard/function keys.
   5. Optionally enabling/disabling the CRT controller's lite pen unit interrupt.

6. Optionally re-storing the CRT controller's keyboard/ function keys to their pre-read operation enabled/ disabled state.
7. Optionally re-storing the CRT controller's Lite Pen unit interrupt to its pre-read enabled/disabled state.

d. A request completion message is send back to the task that requested the read operation. This message contains both read completion flags and the address of the chain of Data buffers used for the data.

e. Control is returned to the Main Control section.

4.3.8.1 Multiple Field Read (FREAD_CRT)

This subroutine reads the specified fields from the CRT screen. The data read is saved in a chain of Data buffers which is either provided by the requester or allocated by this driver depending upon the option specified.

The requester's data field definition packets are used to determine which fields to read. The data fields are of course read in the same order that the data field definition packets are ordered. Each field definition packet defines the starting position of a field and the length of the field.

Each field read is saved in the read data buffer as follows:

a. A POSITION CURSOR command, that will move the cursor to the beginning position of the field, is moved to the data read buffer.

b. The actual field data is moved to the data read buffer. The field data will be represented as follows:

1. A VIDEO SELECT command is located in front of the first field character and in front of any field character that has different video display characteristics than those of the preceeding character.
2. The actual field data characters are stored in the order that they appear in the screen field.

This format allows the field data read to be processed easily and it also allows this data to be rewritten to the screen.

The FLGS_INT and NXT_BYTE subroutines are used to handle the access of information from the data field definition packets. The DBUF INT and SAV_CHAR subroutines are used to store the data/control information needed to satify this type of request.

4.3.8.2 Get Byte from Data Buffer (GET_CHAR)

This subroutine obtains next byte from the requester's chain of data buffers which contain the data field definition packets. The control byte is returned to the caller in bits 7-0 of the D7 register and the number of bytes left in the current Data buffer is returned to caller in bits 15-0 of the D6 register.

NOTE: A negative value returned in D6 register indicates that all control bytes have already been obtained.

A NXT_BYTE subroutine call is processed in the following manner:

a. A check is made to determine if the current data buffer contains another unprocessed byte.

    1. If current data buffer still contains unprocessed control bytes, decrement the Number of Bytes Remaining in Current Data Buffer flag (B_LEFT) and get the address of next byte to be obtained from the BYTE_ADR flag.

2. If current data buffer is empty, a check is made to determine if there is another data buffer threaded to the current data buffer.
   a) If there is another data buffer threaded to the current one that needs to be processed yet, the FLGS INT subroutine is called to re-initilize this subroutine's control flags so that the bytes in the next data buffer can be accessed and then Step A is repeated again.
   b) If there is no more control bytes, an error indicator is returned to the caller.

b. A control byte is obtained from the current Data buffer and the Next Byte Address flag (BYTE_ADR) is updated.

c. Control is returned to caller. The control byte and the number of bytes remaining in current Data buffer are passed to requester in the D7 and D6 registers.

4.3.9 Read/Write Request Preprocessing (PRE PROC)

The PRE_PROC subroutine performs those common operations associated with data read/write requests that must be done before the actual data input/output operations. These operations include:

a. Disabling the CRT controller's keyboard and lite pen.

b. Inputting and saving the X,Y co-ordinates for the current position of the CRT controller's cursor.

c. Inputting and saving the CRT controller's selected video display options.

d. Optionally clearing the CRT controller's screen (Applicable only for write requests).

e. Positioning the cursor to a position that the requester has specified (optional).

This subroutine accesses the request parameters located in the DCI Card driver main control module (DCI_DRIV).

4.3.10 Read/Write Request Post-processing (POST OPT)

The POST_OPT subroutine performs those common operations associated with data read/write requests that must be done after the actual data input/output operations. These operations include:

a. Returning the CRT controller to the appropriate protected/unprotected mode of operation.

b. Positioning the CRT controller's to the position specified by the requester. The requester can specify that the cursor be moved either to the HOME position or its orginal position. The requester can also choose to let the cursor remain in its current position.

c. Optionally reselecting the video display options that were selected for the CRT controller prior to the data read/write request.

d. Optionally enabling/disabling the CRT controller's keyboard/function keys.

e. Optionally enabling/disabling the CRT controller's lite pen unit interrupt.

f. Optionally re-storing the CRT controller's keyboard/function keys to their orginal state.

g. Optionally re-storing the CRT controller's lite pen unit interrupt to its orginal state.

This subroutine accesses the request parameters located in the DCI Card driver main control module (DCI_DRIV).

4.3.11 Get Video Display Characteristics (GET VID)

This subroutine determines which video display options are currently selected for the requester's CRT and returns this information to the caller.

The CRT controller doesn't provide a status that tells which video display options are selected unfortunately; It does provide a status function that gives the video display characteristics associated with last character read. The video display characteristics associated with a character are determined by the video display options selected at the time the character was written. The CRT controller also allows its cursor to moved to a position off the end of the screen which isn't visable to the operator.

These hardware features are used in the following fashion to determine the video display options currently selected for the CRT controller:

a. The X,Y co-ordinates for current position of the cursor are input and saved.

b. The cursor is moved to a position off the end of the CRT screen.

c. A Space character is written there.

d. A Backspace Cursor control character is output.

e. The Space and its associated video display characteristics are input. The associated video display characteristics of the Space character reflect the selected video display options.

f. The cursor is returned to its orginal position.

g. Control is returned to caller with the Selected Video Display Options information stored in the D7 register.

Bits 5-3 -- Color Code.
Bit 2 -- Inverse Video Mode Enabled.
Bit 1 -- Protected/Tabbed.
Bit 0 -- Blink Mode Enabled.

NOTE: The Selected video Display Options information is saved in the VID_STAT flag which is a global variable located in the Main Control module.

4.3.12 Select Video Characteristics (SET_VID)

The subroutine outputs the control characters necessary to ensure that the specified set of video display is selected at the requester's CRT controller.

The caller passes flags in D7 indicating the set of video display options that he wants selected. These flags are defined as follows:

Bits 5-3 -- Color Code.
Bit 2 -- Inverse Video Mode Enabled.
Bit 1 -- Protected/Tabbed.
Bit 0 -- Blink Mode Enabled.

A SET_VID subroutine call is processed in the following manner:

a. The caller's Video Display Option flags are saved in the VID PARM flag.

b. The appropriate Color Select control character is output to the CRT controller using the WRT_CHAR subroutine.

c. Either a Clear Inverse Video Mode or a Select Inverse Video Mode control character is output to the CRT controller depending upon the mode desired by the caller. The WRT_CHAR subroutine is used to output the control character to the CRT controller.

d. Either a Blink Mode Off or a Blink Mode On control character is output to the CRT controller depending upon the blink mode desired by the caller. Again, the WRT_CHAR subroutine is used to output this control character to the CRT controller.

e. Control is returned to the caller.

4.3.13 Get Cursor Position (GET CUR)

This subroutine inputs the CRT controller's STATUS FOUR and STATUS FIVE bytes (X,Y co-ordinates for the current position of the cursor). The cursor's X,Y co-ordinates are returned to caller in the D7 register (X ordinate in bits 15-08 and Y ordinate in bits 07-00).

This subroutine utilizes the memory mapped I/O register addresses set up by the DCI Card driver's main control module (DCI_DRIV) to communicate with the appropriate CRT controller.

4.3.14 Move Cursor Subroutine (MOVE CUR)

This subroutine outputs the CRT Controller CONTROL FOUR and CONTROL FIVE commands required to move the cursor to the screen position specified. The caller passes the X,Y co-ordinates for the new cursor position in the D7 register (X ordinate in bits 15--08 and Y ordinate in bits 07--00).

This subroutine utilizes the memory mapped I/O register addresses set up by the DCI Card driver's main control module (DCI_DRIV) to communicate with the appropriate CRT controller.

4.3.15 Read Character (READCHAR)

This subroutine inputs a single character from the CRT controller. The current position of the CRT's cursor determines the screen position read. Because of the hardware's read ahead feature, it is really impossible to just read a single character. As soon as a character is input from a DCI card, the DCI automatically begins obtaining the next character from the CRT. The hardware access of the next character occurs concurrently with the processing of the current character. This means that the cursor must be moved back one space to leave it at the right place on the screen.

This subroutine is designed for those cases where only one character needs to be input.

This subroutine follows the procedure listed here:

a. It tells DCI to get a character from the CRT.

b. It waits until DCI has obtained the character.

c. It inputs the character from the DCI.

d. It waits for DCI to finish obtaining the next character from the CRT.

e. It tells DCI to output a Backspace Cursor control character to the CRT. This returns the cursor to the position following the first character read. The WRT_CHAR subroutine is used to perform this step.

f. It returns control to caller with the character in lower byte of the D7 register.

This subroutine utilizes the memory mapped I/O register addresses set up by the DCI Card driver's main control module (DCI_DRIV) to communicate with the appropriate CRT controller.

4.3.16 Write Character (WRT CHAR)

This subroutine outputs a single data/control character to a CRT. The character to output is passed in the lower byte of the D7 register. Control isn't returned to the caller until CRT has finished processing the character.

This subroutine follows the procedure listed here:

a. It outputs character to DCI and tells DCI that it wants character output to the specified CRT.

b. It waits until DCI has transferred character to the CRT and the CRT has processed the character. The CRT_OK subroutine is used to perform this step.

c. It returns control to the caller.

This subroutine utilizes the memory mapped I/O register addresses set up by the DCI Card driver's main control module (DCI_DRIV) to communicate with the appropriate CRT controller.

4.3.17 Wait until DCI Ready (DCI OK)

This subroutine delays further processing until a DCI card is no longer busy. This subroutine is called before any functions are output to either a DCI itself or any CRT controllers connected to the DCI. This is necessary because the Memory Mapped I/O Register scheme used to communicate with the DCI doesn't allow the DCI to reject a function if it is still busy processing a previous one. If a command is output to the DCI while it is still busy processing a previous command, the response of the DCI is not predicatable.

A counter is used to ensure that we don't wait for longer than a reasonable amount of time for a DCI to become NOT BUSY. If DCI get hung up BUSY, this subroutine should detect this error condition and report it to Configuration Control task.

4.3.18 Wait until CRT Ready (CRT OK)

This subroutine delays further processing until a CRT CRT controller is no longer BUSY. This subroutine is called to determine if the CRT controller is ready for another data/control character during output operations. Data characters are handled by the CRT Controller as fast as they can be output, but there are special edit control function characters that can take up to 2.5 milliseconds to be processed by the CRT controller. Using the End-of-Operation interrupt to control the output of data to the CRT Controller would slow the transfer rate tremendously; therefore this subroutine is used to control the operation when special control characters must be output to a CRT.

A counter is used to ensure that we don't wait for longer than a reasonable amount of time for a CRT controller to become NOT BUSY. If a CRT controller get hung up BUSY, this subroutine should detect this error condition and report it to the Configuration Control task.

4.3.19 Output Function to CRT (CDFOUT)

This subroutine is used to output CRT controller function commands that can not be output using special control characters. These include function commands to accomplish the following actions:

a. Master clear the CRT controller.

b. Enable controller's keyboard/function keys.

c. Disable controller's keyboard/function keys.

d. Enable the controller's lite pen.

e. Disable the controller's lite pen.

f. Initiate a copy operation to CRT Controller's slave print device (never used).

g. Set CRT controller's "ALARM OUTPUT" contact.

h. Clear CRT controller's "ALARM OUTPUT" contact.

This subroutine updates the CRT's CURRENT UNIT STATUS flags to reflect the function command(s) and then it outputs the function commands required to ensure that the controller is operating in the manner indicated by its CURRENT UNIT STATUS flags.

NOTE: A Master Clear function request must be issued alone because this function request overrides any other function requests accompanying it.

Caller passes request command flags in the D7 register.

Bit 7 -- Clear the Alarm Output Contact.
Bit 6 -- Set the Alarm Output Contact.
Bit 5 -- Initiate Copy Operation.

Bit 4 -- Disable Lite Pen.
Bit 3 -- Enable Lite Pen.
Bit 2 -- Disable Keyboard/Function Keys.
Bit 1 -- Enable keyboard/Function Keys.
Bit 0 -- Master Clear Controller.

4.3.20 Output Mode Change to CRT (CDVMODE)

This subroutine outputs the control characters necessary to communicate operation mode changes to a CRT controller. These operational mode changes include the following:

- o Clearing all selected special operational modes
- o Selecting insert character/page mode
- o Selecting insert character/line mode
- o Selecting scroll down mode
- o Selecting scroll up mode
- o Selecting protected mode

This subroutine updates the CRT's Current Unit Status flags to reflect the operational mode change commands and then outputs the control characters required to ensure that the CRT controller is operating in manner indicated by its Current Unit Status flags.

The request command flags passed by caller in the D7 register are defined as follows:

Bit 5 -- Clear Selected Modes.
Bit 4 -- Select Insert Character/Page Mode.
Bit 3 -- Select Insert Character/Line Mode.
Bit 2 -- Select Scroll Down Mode.
Bit 1 -- Select Scroll Up Mode.
Bit 0 -- Select Protected Mode.

4.4 DATA ORGANIZATION

4.4.1 DCI Driver Task Identifier (DCI TASK)

Application task that want to send requests to the DCI Driver task need the DCI Driver's TASK IDENTIFIER in order to be able to send an intertask message to the DCI Driver.

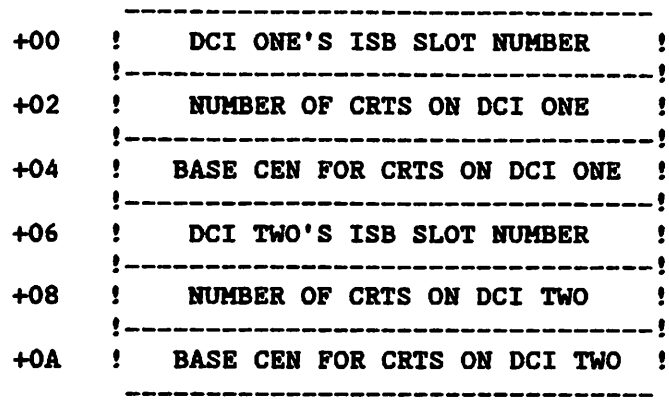
When the DCI Driver task is created and runs the first time, it obtains its TASK IDENTIFIER from the RUNNING cell in the Executive's Configuration Information table and stores it in the DCI_TASK cell which is globally defined.

4.4.2 DCI/CRT Configuration Information Table (DCI INFO)

When the DCI Driver is processing requests and interrupts, it must know both how many DCI cards there are configured in the MCU and how many CRTs are connected to each DCI card. When the DCI Driver is processing interrupts, it also needs to know the CYBER Equipment Number (CEN) that is associated with the first CRT connected to each DCI card. It is understood that all the CRTs connected to a DCI card will have consecutive CENs.

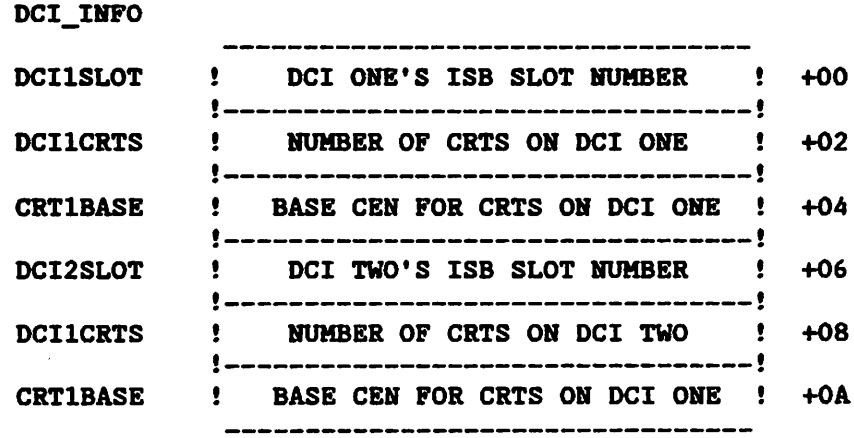
Configuration Control maintains data base tables that hold this type of hardware configuration information. A macro has been provided by the Configuration Control subsystem which will move this information to a user provided area.

The hardware configuration information provided by the CC_DCI_INFO_MAC macro is formatted as shown here:



This makes it unnecessary for the DCI Driver to know anything about the structure of the Equipment Configuration Information tables that Configuration Control maintains.

When the DCI Driver task is created and runs the first time, it calls the Configuration Control subsystem to have it move this information to the DCI_INFO table.



Each DCI_INFO table entry is globally defined so that its contents can be referenced directly by the DCI Driver sections that require this information.

It is assumed that a MCU will be configured with either one or two DCI Cards. The DCI1SLOT entry indicates which ISB slot is supposed to contain the first DCI card. If the MCU contains two DCI cards, the DCI2SLOT entry indicates which ISB slot is supposed to contain the second DCI card; otherwise if the MCU has only one DCI, the DCI2SLOT entry contains a negative value to indicate this fact.

4.4.3 Command Zero I/O Register Addresses

The DCI_CMD0 table contains a long word entry for each possible DCI card that may be configured in an MCU. It is assumed that an MCU will be configured with one or two DCI cards. Thus this table has two entries and each entry holds the address of the Memory Mapped I/O register to be used to transfer COMMAND ZERO bytes to the corresponding DCI card. The DCI Driver needs to know these addresses in order to be able to talk to the hardware.

The two entries are labeled DCI1CMD0 and DCI2CMD0 for easier reference.

When the DCI Driver task is created and runs the first time, it uses the DCI hardware configuration information that it obtains from the Configuration Control subsystem to generate the entries in this table.

4.4.4 CRT Input I/O Register Address (DCI INPT)

The DCI_CMD0 table contains a long word entry for each possible DCI card that may be configured in an MCU. It is assumed that an MCU will be configured with one or two DCI cards. Thus this table has two entries and each entry holds the address of the Memory Mapped I/O register to be used to input the data and status information from a CRT Controller connected to the corresponding DCI card. The DCI Driver needs to know the address of these I/O registers in order to get the data/status information that a DCI card has retrieved from one of the CRTs that is connected up to it.

The two entries are labeled DCI1INPT and DCI2INPT for easier reference.

When the DCI Driver task is created and runs the first time, it uses the DCI hardware configuration information that it obtains from the Configuration Control subsystem to generate the entries in this table.

4.4.5 CRT CEN to Physical Address Conversion Table

The Configuration Control subsystem maintains data base tables that can be used to convert a CRT CYBER Equipment Number (CEN) to its corresponding physical address. The CYBER Equipment Numbers are logical addresses that have been assigned to the hardware components by the Data Base Generation subsystem.

A CRT physical address has two components:

a. The number of the ISB slot holding the DCI card to which the CRT is connected.

b. The CRT unit number. This is a number that is set in the CRT Controller unit.

The Configuration Control subsystem has provided a macro CC_CRT INFO_MAC that can be used to convert a CRT CEN to its corresponding physical address. This makes it unnecessary for the DCI Driver to know anything about the structure of the Equipment Address Conversion tables that the Configuration Control subsystem maintains.

4.4.6 Functional Features/Operational Modes Table

The 40-815 CRT Controller has several operational modes that can be enabled/disabled by outputting appropriate control characters to the CRT controller.

These operational modes are:

- o Protected mode
- o Scroll up mode
- o Scroll down mode
- o Insert character in field/line mode
- o Insert character in field/page mode

The 40-815 CRT Controller doesn't however provide a status function that can be used to determine if the controller has these operational modes selected.

This means that a record must be maintained to indicate if the controller is supposed to have any of these operational modes selected. This record can be accessed when the proper operation modes for the controller are required. Since these operational modes get cleared during the processing of data read/write requests, this Operational Modes record is needed to restore the CRT controller to the proper state after the processing of such requests. This record also allows Operational Modes information to be provided when a CRT Status request is processed.

The 40-815 CRT Controller also has an Alarm Output Contact that can be both set and cleared by function outputs to the controller. The CRT Controller does not however provide a status to determine whether the Alarm Output Contact is set or cleared. This Alarm Output Contact may also get cleared during the normal processing of requests. This means that a record must be maintained of the correct state of the Alarm Output Contact so that it can be restored after the processing of a request. Again, this record can also be used to provide information on this feature when a CRT Status request is processed.

The 40-815 CRT Controller can support a Lite Pen unit. If a Lite Pen is connected to the CRT controller, its operation can be enabled and disabled using function outputs to the controller. Again, however there is no status function that can be used to determine the current enabled/disabled state of the Lite Pen unit. A record must be maintained of the appropriate state for the Lite Pen unit for the same reasons mentioned above.

This information must be maintained for each CRT and the CRT Current Status table is used to maintain this information. This table contains an eight bit entryfor each CRT and it is sized for the maximum hardware configuration (two DCI cards with sixteen CRTs connected to each DCI card).

Entries in the CRT Current Status table are formatted as follows:

Bit 7 -- Console Alarm Set
Bit 6 -- Lite Pen I/O Enabled
Bit 5 -- Keyboard/Function Keys Enabled
Bit 4 -- Insert Page Mode Selected
Bit 3 -- Insert Line Mode Selected
Bit 2 -- Scroll Down Selected
Bit 1 -- Scroll Up Selected
Bit 0 -- Protected Mode Selected

4.4.7 Unit Up/Down Operational Status Table

The Configuration Control subsystem maintains data base tables that keep track of the UP/DOWN status of the DCI cards and the CRTs connected to these DCI cards.

Again, Configuration Control has provided a macro that can be used to obtain the operational status of the system's hardware components. When a request is processed, the CC_CRT_STATUS_MAC macro is used to determine if the request unit is available. If the requested unit is down, the request is of course completed with a CRT Down indication.

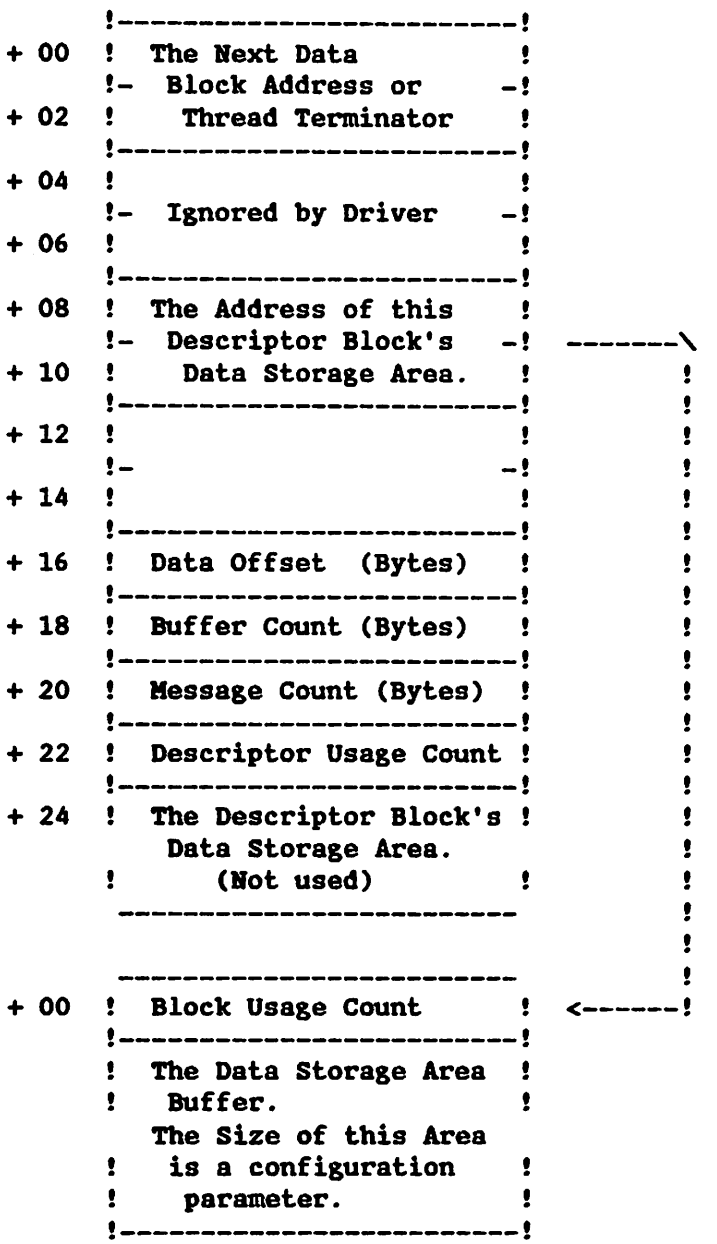
4.4.8 Utilization of Chains of Data Buffers

Chains of CDNA Executive Data buffers are utilized to hold the data that the DCI Driver reads from and writes to the CRTs.

A chain of Data buffers consists of one or more Data buffers that are threaded together. A Data buffer chain is identified by the address of the first Data buffer in the chain. The first location of each chain member contains either the address of the next member or a CHAIN TERMINATOR indicator (zero). The Executive maintains a pool of available Data buffers and provides directives that can be used to allocate release chains of data buffers. On allocate requests, the requester specifies the length of the buffer chain desired, (i.e. the number of data buffers wanted).

Although a Data buffer is considered to be a single logical entity, it really consists of two physical parts. The first physical part of a Data buffer is really just a Descriptor buffer. The Data Storage Area Pointer in this Descriptor points to a separate Data Storage Area buffer rather than to its own internal data storage area. It is important to note that more than one Data buffer can share the same Data Storage Area buffer (see Usage Count field in the Data Storage Area buffer).

SYSTEM DATA BUFFER



4.5 SUBSYSTEM CHECKPOINT REQUIREMENTS

This Driver has no data that needs to be checkpointed.

4.6 INITIALIZATION REQUIREMENTS

Configuration Control ensures that the DCI Driver software and control tables have been downloaded to MCU first. Then, Configuration Control creates the DCI Driver task using one of the Executive's task creation directives. When the DCI Driver task is created, the DCI Driver task initialization module (DCI_STRT) gets control and completes any other needed initialization functions.

4.7 SUBSYSTEM INTERFACE

The DCI/40-815 CRT Controller Driver is set up as an applications software task and the other application tasks utilize the CDNA Executive's SEND NORMAL MESSAGE directive to send requests to the DCI Driver. To send a request to the DCI Driver task, the requesting task must supply the DCI Driver's Task Identifier. The DCI_TASK cell holds the Driver's Task Identifier for those application programs that need it.

The Driver uses a RECEIVE NORMAL MESSAGE directive to obtain requests for processing. The CDNA Executive's intertask message scheme is utilized by the DCI Driver to serialize its processing of the requests sent to it. Since the Executive treats all messages the same, DCI Driver request processing occurs on a FIRST IN/FIRST OUT basis.

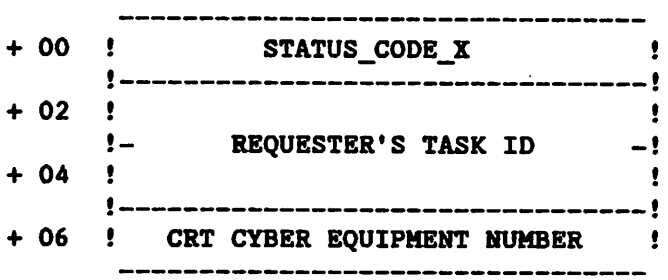
Every request sent to the DCI Driver must contain the requesting task' Task Identifier so that the Driver can sent a Request Processing Completion message to the requesting task after it has completed processing the request. This Request Completion message contains a request completion code and any other relevant request processing information such as the number of characters actually read for a CRT Read request or the CRT status for a CRT Status request.

The request type code are defined in the DCKEQUS comdeck which is contained in the DIKPL program library under userid APD.

4.7.1 CRT Status Request/Response Formats

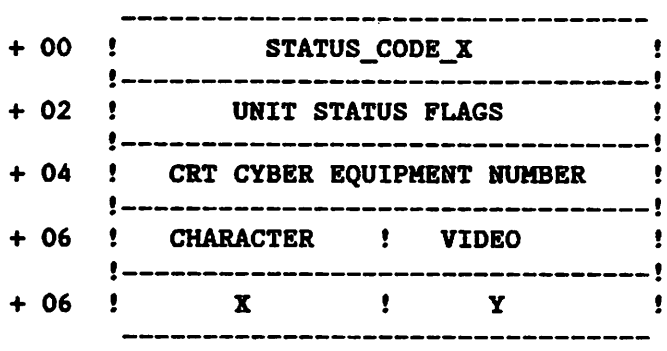
4.7.1.1 CRT Status Request Format

The CRT Status request message is formatted as shown here:



4.7.1.2 CRT Status Request Response Format

The CRT Status Response message is formatted as shown here:



Where: UNIT STATUS FLAGS

Bit 00 -- CRT Controller Not Busy Flag.

Bit 01 -- CRT Controller Ready.

Bit 02 -- Protected Mode Selected.

Bit 03 -- Scroll Up Mode Selected.

Bit 04 -- Scroll Down Mode Selected.

Bit 05 -- Insert Line Mode Selected.

Bit 06 -- Insert Page Mode Selected.

Bit 07 -- Keyboard/Function Keys Enabled.

Bit 08 -- Lite Pen I/O Enabled.

Bit 09 -- Console Alarm Output Contact Set.

CHARACTER is character located at cursor position.

VIDEO is currently selected video display characteristics.

Bits 5-3 ----- Color Code.

Bit 2 ----- Inverse Video Mode Enabled.

Bit 1 ----- Protected/Tabbed.

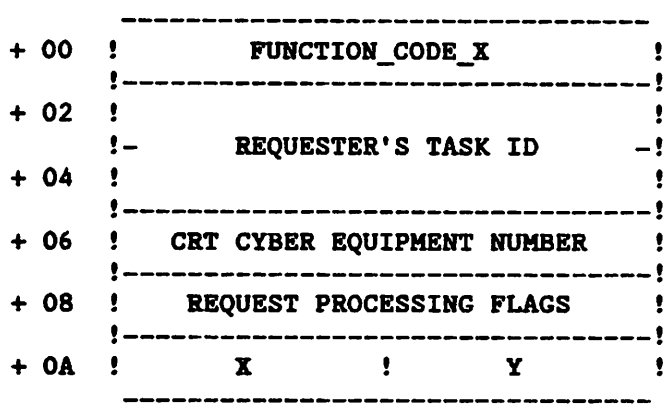
Bit 0 ----- Blink Mode Enabled.

X,Y are co-ordinates for the screen position where the cursor is located.

4.7.2 CRT Function Request/Response Formats

4.7.2.1 CRT Function Request Format

The CRT Function Request is formatted as shown here:



WHERE: X,Y are the co-ordinates for the screen position where the requester wants to cursor moved.

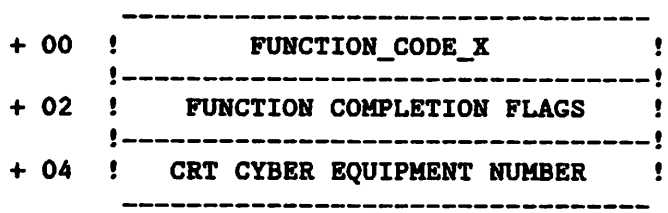
REQUEST PROCESSING FLAGS are defined as follows.

Bit 15 -- Not Used.
Bit 14 -- Not Used.
Bit 13 -- Clear Selected Modes.
Bit 12 -- Select Insert Character/Page Mode.
Bit 11 -- Select Insert Character/Line Mode.
Bit 10 -- Select Scroll Down Mode.
Bit 09 -- Select Scroll Up Mode.
Bit 08 -- Select Protected Mode.

Bit 07 -- Clear Alarm Output Contact.
Bit 06 -- SET Alarm Output Contact.
Bit 05 -- Initiate Copy Operation.
Bit 04 -- Disable Lite Pen.
Bit 03 -- Enable Lite Pen.
Bit 02 -- Disable Keyboard/Function Keys.
Bit 01 -- Enable Keyboard/Function Keys.
Bit 00 -- Master Clear Controller.

4.7.2.2 CRT Function Response Format

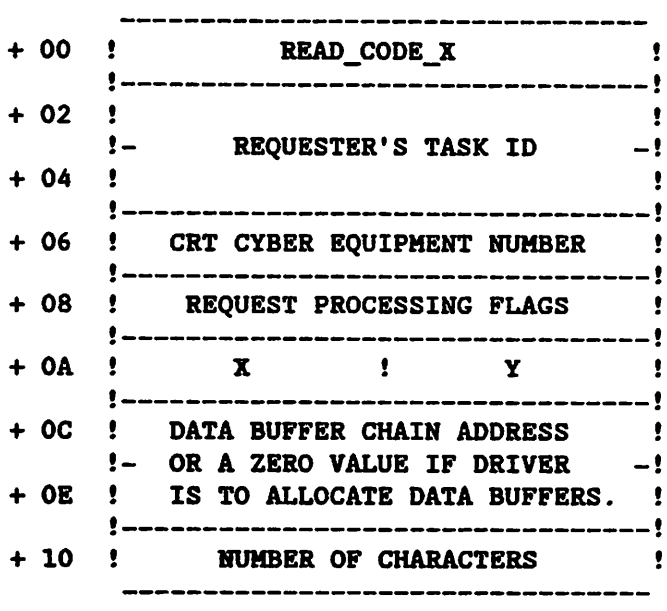
The CRT Function Response is formatted as shown here:



4.7.3 CRT Read Request/Response Formats

4.7.3.1 CRT Read Request Format

The CRT Read Request is formatted as shown here:



WHERE: X,Y are the co-ordinates for a screen position.

REQUEST PROCESSING FLAGS are defined as follows.

Bits 15-14 -- Not Defined.

Bits 13-12 -- Post Read Lite Pen Options.

00 = Enabled.

01 = Disabled.

10 = Pre-Read Status.

11 = Enabled.

Bits 11-10 -- Post Read Keyboard/Function Key Options.

00 = Enabled.

01 = Disabled.

10 = Pre-Read Status.

11 = Enabled.

Bits 09-08 -- Post Read Cursor Options.

00 = Leave cursor where it is located after read.

01 = Home cursor.

10 = Return cursor to shere it was located before read.

11 = Leave cursor where it is located after Read.

Bit 07 -- CRT's Selected Video Display Characteristics Preservation flag.

Bit 06 -- Data Character's Video Display Characteristics Preservation flag.

Bit 05 -- Repeated Character String Compression flag. Note, if repeated space character string compression option is selected, repeated non-inverse space character strings will be processed using that scheme.

Bit 04 -- Repeated Space Character String Compression flag. The X,Y coordinates of next non-(non-inverse space) character are saved rather than the spaces.

Bit 03 -- Not Used.

Bit 02 -- Protect Mode Options.

0 = Don't change protect mode.

1 = Disable protect mode, read data, and restore to original protect enabled/disabled state.

Bits 01-00 -- Pre-Read Cursor Options.

00 = Do nothing.

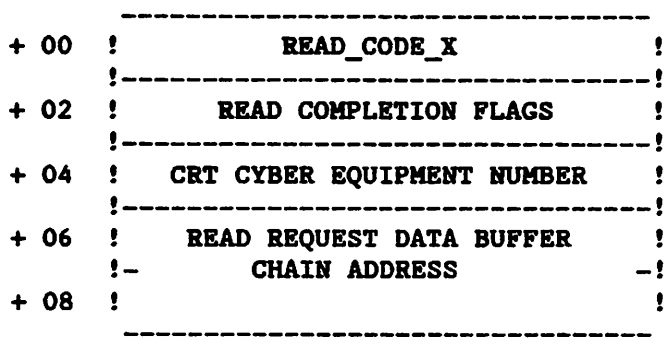
01 = Home cursor.

10 = Move cursor to specified position.

11 = Move cursor to beginning of current line.

4.7.3.2 CRT Read Response Format

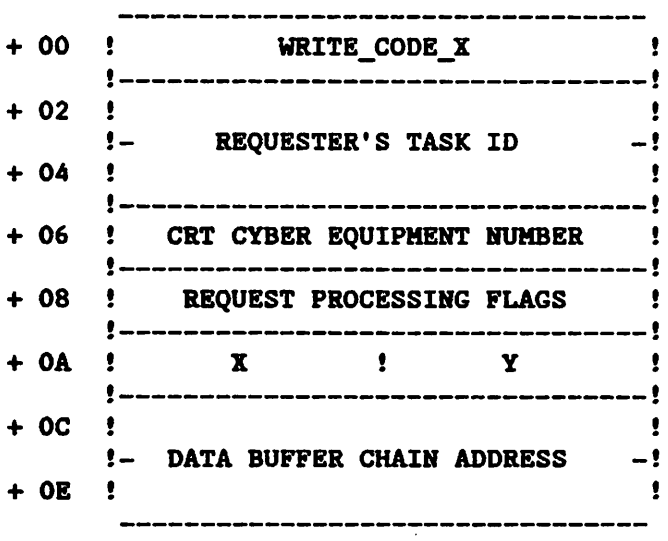
The CRT Read Request Response is formatted as shown here:



4.7.4 CRT Write Request/Response Formats

4.7.4.1 CRT Write Request Format

The CRT Write request is formatted as shown here:



Where: X,Y are the co-ordinates for a screen position.

REQUEST PROCESSING FLAGS are defined as follows.

Bits 15-14 -- Not Defined.

Bits 13-12 -- Post Write Lite Pen Options.

00 = Enabled.

01 = Disabled.

10 = Pre-Write Status.

11 = Enabled.

Bits 11-10 -- Post Write Keyboard/Function Key Options.

00 = Enabled.

01 = Disabled.

10 = Pre-Write Status.

11 = Enabled.

Bits 09-08 -- Post Write Cursor Options.

00 = Leave cursor where it is located after write.

01 = Home cursor.

10 = Return cursor to where it was located before write.

11 = Leave cursor where it is located after write.

Bit 07 -- CRT'S Selected Video Display Characteristics Preservation flag.

Bit 06 -- Not used.

Bit 05 -- Release chain of data buffers after write flag.

Bit 04 -- Pre-write clear screen/home cursor flag.

Bit 03 -- Not used.

Bit 02 -- Protect Mode Options.

0 = Don't change protect mode.

1 = Disable protect mode, write data, and restore to original protect enabled/disabled state.

Bits 01-00 -- Pre-Write Cursor Options.

00 = Do nothing.

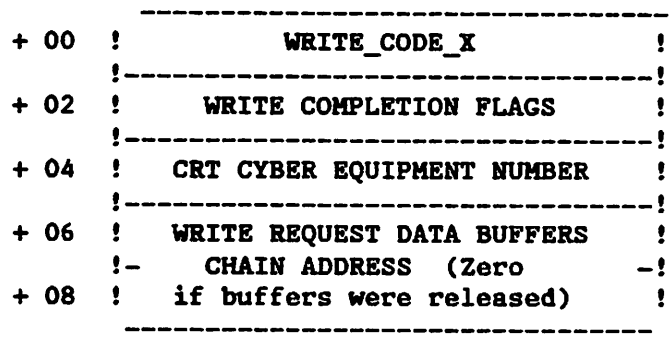
01 = Home cursor.

10 = Move cursor to specified position.

11 = Move cursor to beginning of current line.

4.7.4.2 CRT Write Response Format

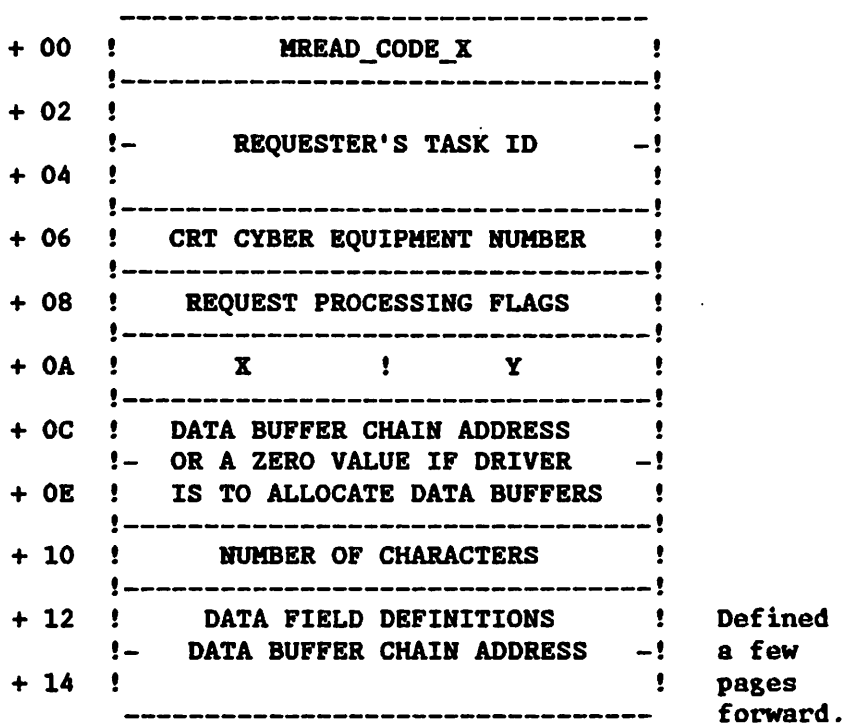
The CRT Write Response is formatted as shown here:



4.7.5 CRT Multiple Field Read Request/Response Formats

4.7.3.1 CRT Multiple Field Read Request Format

The CRT Multiple Field Read Request is formatted as shown here:



WHERE: X,Y are the co-ordinates for a screen position.

REQUEST PROCESSING FLAGS are defined as follows.

Bits 15-14 -- Not Defined.

Bits 13-12 -- Post Read Lite Pen Options.

00 = Enabled.

01 = Disabled.

10 = Pre-Read Status.

11 = Enabled.

Bits 11-10 -- Post Read Keyboard/Function Key Options.

00 = Enabled.

01 = Disabled.

10 = Pre-Read Status.

11 = Enabled.

Bits 09-08 -- Post Read Cursor Options.

00 = Leave cursor where it is located after read.

01 = Home cursor.

10 = Return cursor to shere it was located before read.

11 = Leave cursor where it is located after Read.

Bit 07 -- Not Used.

Bit 06 -- Not Used.

Bit 05 -- Not Used.

Bit 04 -- Not Used.

Bit 03 -- Not Used.

Bit 02 -- Protect Mode Options.

0 = Don't change protect mode.

1 = Disable protect mode, read data, and restore to original protect enabled/disabled state.

Bits 01-00 -- Pre-Read Cursor Options.

00 = Do nothing.

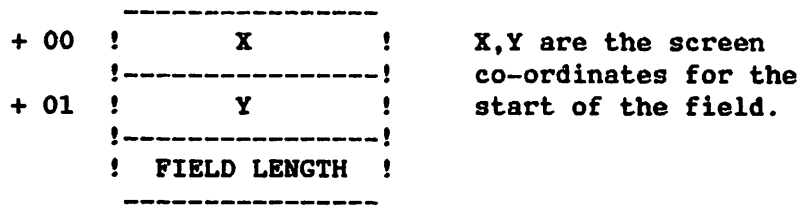
01 = Home cursor.

10 = Move cursor to specified position.

11 = Move cursor to beginning of current line.

DATA FIELD DEFINITIONS DATA BUFFER CHAIN

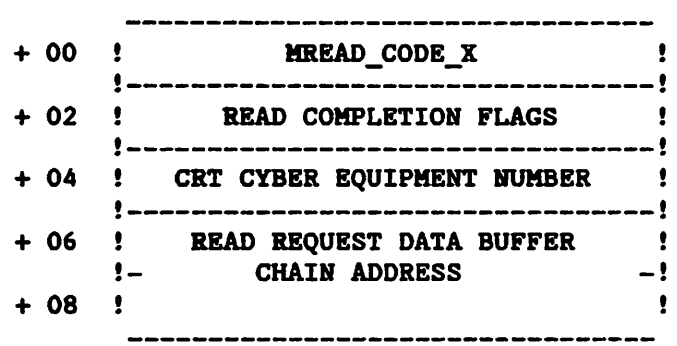
This is a chain of Data buffers that the requesting task provides to identify the CRT Screen Fields that that it wants to be read. For each field, the task has to identify the starting screen position for the field and the length of the field. This information is packed into three byte packets that are formatted as follows:



The Data Field Definitions data buffer chain contains one or more Data Field Definition packets. The Data buffer control fields will indicate how many Data Field Definition packets are contained in the chain of Data buffers.

4.7.5.2 CRT Multiple Field Read Response Format

The CRT Multiple Field Read Response is formatted as shown here:



APPENDIX A

DCI HARDWARE INTERFACE INFORMATION

TALKING TO THE DCI CARD

Since the DCI card connects up to the Internal System Bus of the Device Interface, there are four command bytes that can be output to the DCI and there are four status bytes that can be input from the DCI card. These command and status bytes are described on the following pages.

OUTPUTTING DCI COMMAND ZERO

A DCI COMMAND ZERO can be output by doing a byte write to the Memory Mapped I/O register whose address is defined as follows:

Bits 23-07 -- Base address for the Internal Control Bus Memory Mapped I/O registers.
Bits 06-03 -- DCI's ISB slot number.
Bits 02-01 -- Internal Control Bus operation code (zero).
Bit 00 -- Zero.

DCI COMMAND ZERO DEFINTION

Bit 7 -- Enable card.
Bit 6 -- Enable DCI's Internal Transfer Bus (ITB) interface.
Bit 5 -- Turn fault indicator off.
Bit 4 -- Turn attention indicator off.
Bit 3 -- Enable DCI's Internal Control Bus interrupt.
Bit 2 -- Disable diagnostic mode.
Bit 1 -- Disable NMI restart.
Bit 0 -- Disable maskable interrupt.

OUTPUTTING DCI COMMAND ONE

A DCI COMMAND ONE can be output by doing a byte write to the Memory Mapped I/O register whose address is defined as follows:

Bits 23-07 -- Base address for Internal Control Bus Memory Mapped I/O registers.
Bits 06-03 -- DCI's ISB Slot Number.
Bits 02-01 -- Internal Control Bus Operation Code (1).
Bit 00 -- Zero.

The DCI COMMAND ONE is used to tell DCI to input either data or status information from a specified 40-815 CRT Controller and store it in an internal memory mapped I/O register. The DCI COMMAND ONE byte identifies the 40-815 CRT Controller and the type of information wanted.

DCI COMMAND ONE DEFINITION

Bits 7-4 -- 40-815 CRT Controller Number.
Bits 3-0 -- Input Operation Code.

0 = Data Character Input.
1 = 40-815 CRT Controller Status Two/Data Character Input.
2 = 40-815 CRT Controller Status One Input.
3 = Undefined.
4 = 40-815 CRT Controller Status Two Input.
5 = Undefined.
6 = 40-815 CRT Controller Status Three Input (number of last CRT function key pressed).
7 = Undefined.
8 = 40-815 CRT Controller Status Four Input (CRT's cursor position x co-ordinate).
9 = 40-815 CRT Controller Status Four/Status Five Input (CRT's cursor position x,y co-ordinates).
A = 40-815 CRT Controller Status Five Input (CRT's cursor postion y co-ordinate).

B = Undefined.

C = 40-815 CRT Controller Status Six Input (contents of the 40-815 controller's data echo register).

D = Undefined.

E = 40-815 CRT Controller Status Seven Input (Settings of the selectable switches).

F = Undefined.

When a DCI COMMAND ONE is output, the DCI goes busy until it has retrieved the needed information from the 40-815 CRT Controller and saved it in its CRT CONTROLLER INPUT register. After a DCI COMMAND ONE has been output, it is necessary to wait until the DCI BUSY bit clears before attempting to input the requested information from the CRT CONTROLLER INPUT Memory Mapped I/O register.

The address of the DCI's CRT CONTROLLER INPUT Memory Mapped I/O register is defined as follows:

```
0 W 6 0 X X
  -     ---
  !      !
  !      !
  !       ---   X = Any hexidecimal digit.
  !
   ---   W = DCI's ISB Slot Number (4-8).
```

OUTPUTTING DCI COMMAND TWO

A DCI COMMAND TWO can be output by doing a byte write to the Memory Mapped I/O register whose address is defined as follows:

Bits 23-07 -- Base address for Internal Control Bus Memory Mapped I/O registers.

Bits 06-03 -- DCI's ISB Slot Number.

Bits 02-01 -- Internal Control Bus Operation Code (two).

Bit 00 -- Zero.

The DCI COMMAND TWO is used to tell the DCI which 68000 exception vector it is to use to inform MPB of conditions that require its attention. This request is normally just used once at system start up time.

OUTPUTTING DCI COMMAND THREE

A DCI COMMAND THREE can be output by doing a byte write to the Memory Mapped I/O register whose address is defined as follows:

Bits 23-07 -- Base address for Internal Control Bus Memory Mapped I/O registers.
Bits 06-03 -- DCI's ISB slot number.
Bits 02-01 -- Internal Control Bus operation code (three).
Bit 00 -- Zero.

The DCI COMMAND THREE output request is used to tell the DCI to perform a PROGRAMMED RESET operation for the 40-815 CRT Controller. The data output has no significance.

INPUTTING DCI STATUS ZERO

DCI STATUS ZERO can be input by doing a byte read from the Memory Mapped I/O register whose address is defined as follows:

Bits 23-07 -- Base address for Internal Control Bus Memory Mapped I/O registers.
Bits 06-03 -- DCI's ISB Slot Number.
Bits 02-01 -- Internal Control Bus operation code (zero).
Bit 00 -- Zero.

DCI STATUS ZERO DEFINITION

Bit 7 -- DCI Card Okay.
Bit 6 -- Device Available.
Bit 5 -- Attention Switch On.
Bit 4 -- Bootstrap Operation Allowed (zero for DCI card).
Bit 3-0 -- Card Type Code (fourteen).

It should be possible to perform a DCI STATUS ZERO input at any time.

INPUTTING DCI STATUS ONE

DCI STATUS ONE can be input by doing a byte read from the Memory Mapped I/O register whose address is defined as follows:

Bits 23-07 -- Base address for Internal Control Bus Memory Mapped I/O registers.
Bits 06-03 -- DCI's ISB Slot Number.
Bits 02-01 -- Internal Control Bus operation code (one).
Bit 00 -- Zero.

DCI STATUS ONE DEFINITION

Bit 7 -- DCI Busy flag.
Bit 6 -- I/O Timeout.
Bit 5 -- I/O Reject.
Bit 4 -- Attention Switch Set.
Bit 3 -- Interrupt.
Bit 2 -- Undefined.
Bit 1 -- Parity Error - Upper.
Bit 0 -- Parity Error - Lower.

It is possible to perform a DCI STATUS ONE input at any time.

INPUTTING DCI STATUS TWO

DCI STATUS TWO can be input by doing a byte read from the Memory Mapped I/O register whose address is defined as follows:

Bits 23-07 -- Base address for Internal Control Bus Memory Mapped I/O registers.
Bits 06-03 -- DCI's ISB Slot Number.
Bits 02-01 -- Internal Control Bus operation code (two).
Bit 00 -- Zero.

DCI STATUS TWO is number of system interrupt vector that the DCI is using. The number of the interrupt vector that the DCI uses is output to the DCI during startup. A DCI COMMAND TWO output is used to perform this function.

INPUTTING DCI STATUS THREE

DCI STATUS THREE can be input by doing a byte read from the Memory Mapped I/O register whose address is defined as follows:

Bits 23-07 -- Base address for Internal Control Bus Memory Mapped I/O registers.
Bits 06-03 -- DCI's ISB Slot Number.
Bits 02-01 -- Internal Control Bus operation code (three).
Bit 00 -- Zero.

DCI STATUS THREE is undefined.

OUTPUTTING DATA TO CRT VIA DCI

An eight bit character can be output to a 40-815 CRT connected to a DCI by writting it to the memory mapped I/O register whose address is defined as follows:

```
0 W 6 0 Y 0
  -     -
  !     !
  !     !
  !      ---- 40-815 CRT Controller Number  (0-15).
  !
   ---------- DCI's ISB Slot Number (4-f).
```

Before such an operation is attempted however, DCI STATUS ONE should be input and checked to ensure that the DCI isn't still busy processing a previous operation.

OUTPUTTING CRT CONTROL ONE

A CONTROL ONE byte can be output to a 40-815 CRT Controller connected to a DCI by doing a byte write to the Memory Mapped I/O register whose address is as follows:

```
0 W 6 0 Y 2
  -     -
  !     !
  !     !
  !      --- 40-815 CRT Controller Number (0-15).
  !
   --------- DCI's ISB Slot Number.
```

CONTROL ONE DEFINITION

Bit 7 -- Unused.
Bit 6 -- Select keyboard.
Bit 5 -- Enable display function key interrupt.
Bit 4 -- Enable display copier interrupt.
Bit 3 -- Enable end-of-operation interrupt.
Bit 2 -- Enable lite pen interrupt.
Bit 1 -- Unused.
Bit 0 -- Programmed Master Clear.

Before such an operation is attempted however, DCI STATUS ONE should be input and checked to ensure that the DCI isn't still busy processing a previous operation.

OUTPUTTING CRT CONTROL TWO

A CONTROL ONE byte can be output to a 40-815 CRT Controller connected to a DCI by doing a byte write to the Memory Mapped I/O register whose address is as follows:

```
0 W 6 0 Y 4
  -     -
  !     !
  !     !
  !      --- 40-815 CRT Controller Number (0-15).
  !
   --------- DCI's ISB Slot Number.
```

The CONTROL TWO command is not defined.

OUTPUTTING CRT CONTROL THREE

A CONTROL ONE byte can be output to a 40-815 CRT Controller connected to a DCI by doing a byte write to the memory mapped I/O register whose address is as follows:

```
O W 6 0 Y 6
  -     -
  !     !
  !     !
  !      --- 40-815 CRT Controller Number (0-15).
  !
   --------- DCI's ISB Slot Number.
```

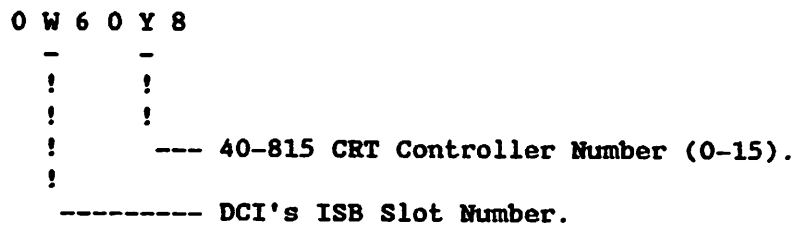
CONTROL THREE DEFINITION

Bit 7 -- Enable alarm output.
Bit 6 -- Etart copier.
Bit 5 -- Not used.
Bit 4 -- Clear copier interrupt.
Bit 3 -- Not used.
Bit 2 -- Not used.
Bit 1 -- Enable lite pen I/O.
Bit 0 -- Disable keyboard I/O.

Before such an operation is attempted however, DCI STATUS ONE should be input and checked to ensure that the DCI isn't still busy processing a previous operation.

OUTPUTTING CRT CONTROL FIVE (SET CURSOR X-ORDINATE)

A CONTROL ONE byte can be output to a 40-815 CRT Controller connected to a DCI by doing a byte write to the memory mapped I/O register whose address is as follows:

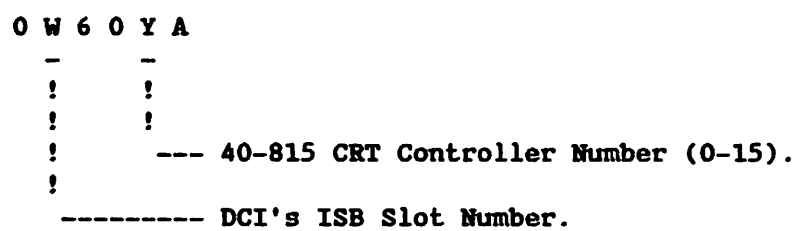


Bits 6-0 of CONTROL FOUR byte give the X co-ordinate (column number) for new position of display's cursor. Bit 7 is not used.

Before such an operation is attempted however, DCI STATUS ONE should be input and checked to ensure that the DCI isn't still busy processing a previous operation.

OUTPUTTING CRT CONTROL FIVE (SET CURSOR Y-ORDINATE)

A CONTROL FIVE COMMAND can be output to a 40-815 CRT Controller connected to a DCI by doing a byte write to the memory mapped I/O register whose address is as follows:

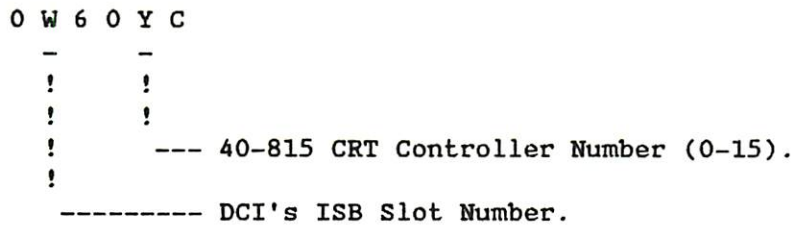


Bits 5-0 of CONTROL FIVE byte give the Y co-ordinate (line number) for new position of display's cursor. Bits 7-6 are not used.

Before such an operation is attempted however, DCI STATUS ONE should be input and checked to ensure that the DCI isn't still busy processing a previous operation.

OUTPUTTING CRT CONTROL SIX (WRITE TO ECHO REGISTER)

A CONTROL SIX COMMAND can be output to a 40-815 CRT Controller connected to a DCI by doing a byte write to the memory mapped I/O register whose address is as follows:

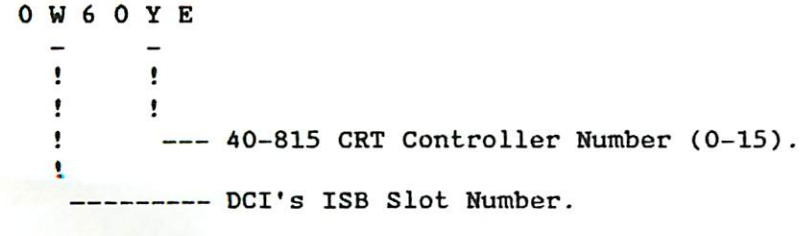


Control SIX bits don't have any meaning associated with them. The byte is simply a data pattern which is stored in the 40-815 CRT Controller's echo register. This data value can then input from the CRT Controller's echo register and verified; a CRT CONTROLLER STATUS SIX input is used to read the value from the CRT Controller's echo register. These two operations are used to test the data input/output paths between the MPB and the 40-815 CRT Controllers.

Before such operations are attempted however, DCI STATUS ONE should be input and checked to ensure that the DCI isn't still busy processing a previous operation.

OUTPUTTING CRT CONTROL SEVEN

A CONTROL SEVEN COMMAND can be output to a 40-815 CRT Controller connected to a DCI by doing a byte write to the memory mapped I/O register whose address is as follows:



CONTROL SEVEN is not currently defined.

INPUTTING DATA FROM CRT CONTROLLER USING DCI

The method used to input data from a 40-815 CRT controller is as follows:

a. A DCI COMMAND ONE is output to the DCI card to tell it to either retrieve a data character or a data character and its associated video display characteristics.

The DCI COMMAND ONE byte is formatted as shown here:

```
 07 06 05 04 03 02 01 00
 -----------------------
!    CRT    !    OPER   !
 -----------------------
```

Where: CRT -- 40-815 CRT Controller Number.
OPER -- Input Operation Code.
0 = Data Character Input.
1 = 40-815 CRT Controller Status Two/Data Character Input.

The DCI Driver alway uses an Input Operation Code of one to input a character and its associated video display characteristics with a one operation.

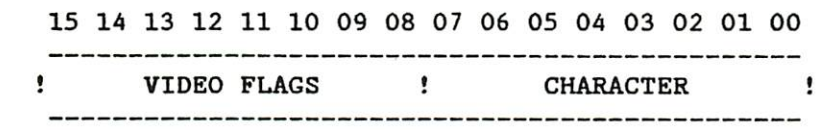
b. The DCI card goes busy until it has retrieved the requested information from the CRT Controller and saved it in its CRT CONTROLLER INPUT register.

Therefore after the DCI COMMAND ONE is output, a program loop is entered that inputs DCI STATUS ONE from the DCI card until the DCI BUSY bit clears.

c. Once a DCI STATUS ONE has been input that indicates the DCI card is no longer busy, this same DCI STATUS ONE byte is checked to ensure that no errors occurs.

d. The requested data/video display characteristics information is input from the CRT CONTROLLER INPUT Memory Mapped I/O register.

The information come back formatted in the following fashion:



Where: VIDEO FLAGS Video display characteristics associated with the character input. See CRT STATUS TWO definition which is located a couple of pages forward.

This completes the inputting of a single character from a CRT. The DCI card is designed to read the next character automatically so that the next character can be obtained by going to step B. This eliminates the overhead that is associated with step A when multiple characters are being input, but it does mean that one extra character always get read and that we must output a BACKSPACE control character after a read operation if we want the cursor to be located after the last character read.

INPUTTING CRT STATUS ONE

The method used to input CRT STATUS ONE is described in DCI COMMAND ONE output section.

CRT STATUS ONE DEFINITION

Bit 7 -- Interrupt condition exists.
Bit 6 -- CRT's keyboad selected.
Bit 5 -- Function key has been depressed.
Bit 4 -- Copier operation done.
Bit 3 -- End-Of-Operation.
Bit 2 -- Lite pen has been used.
Bit 1 -- Not busy.
Bit 0 -- Ready.

INPUTTING CRT STATUS TWO

The method used to input CRT STATUS TWO is described in DCI COMMAND ONE output section.

CRT STATUS TWO DEFINITION

Bit 7 -- Cursor located at end-of-screen.

Bit 6 -- Cursor located at end-of-line.

Bit 5-3 -- Color of last character read.

- 000 -- Black (illegal).
- 001 -- Red.
- 010 -- Green.
- 011 -- Yellow.
- 100 -- Blue.
- 101 -- Magenta.
- 110 -- Cyan.
- 111 -- White.

Bit 2 -- Inverse video display mode of last character read.

Bit 1 -- Protected/Tabbed status of screen position of last character read.

Bit 0 -- Blink status of last character read.

INPUTTING CRT STATUS THREE (FUNCTION KEY NUMBER)

The method used to input CRT STATUS THREE is described in DCI COMMAND ONE output section.

INPUTTING CRT STATUS FOUR (CURSOR POSITION X ORDINATE)

The method used to input CRT STATUS FOUR is described in DCI COMMAND ONE output section.

INPUTTING CRT STATUS FIVE (CURSOR Y ORDINATE)

The method used to input CRT STATUS FIVE is described in DCI COMMAND ONE output section.

INPUTTING CRT STATUS SIX (CONTENTS OF ECHO REGISTER)

The method used to input CRT STATUS SIX is described in DCI COMMAND ONE output section.

INPUTTING CRT STATUS SEVEN (SWITCH SELECTABLE CONFIGURATION)

The method used to input CRT STATUS SEVEN is described in DCI COMMAND ONE output section.

APPENDIX B

40-815 CRT CONTROLLER CONTROL CHARACTERS

80 -- NO OPERATION

The Display Controller terminates its current operation.

81 -- CLEAR SCREEN

The screen is cleared of all data, protected fields, tab marks and color bits. The CURSOR is returned home.

82 -- CLEAR NON-PROTECT

When the CRT controller is in Protect mode, the screen is cleared of all data not within a protected field. The cursor is returned to the HOME position or to the first subsequent unprotected location. When the CRT controllwr isn't in Protect mode, the entire screen is cleared and the cursor is homed.

83 -- CLEAR TO END OF FIELD/LINE

All characters are cleared from the cursor position to the first START PROTECT FIELD character (Protect mode) or the end of line. The position of cursor isn't changed.

84 -- CLEAR TO END OF FIELD/PAGE

All characters are cleared from the cursor position to the first START PROTECT FIELD character (Protect mode) or the end of line. The position of cursor isn't changed.

85 -- DELETE CHARACTER IN FIELD/LINE

86 -- DELETE CHARACTER IN FIELD/PAGE

87 -- DELETE LINE

The cursor moves to beginning of the line that it is located on and all character on that line are cleared. All lines below the line containing cursor move up one line position and the last line is left empty.

88 -- INSERT LINE

The cursor moves to beginning of the line it is located on currently. Current line and all lines below it shift down one line. The current line is cleared. The original last line is lost.

89 -- CURSOR HOME

The cursor moves to first column of first line (HOME position). If CRT controller is in Protect mode, cursor moves to either the HOME position or the first subsequent unprotected position.

8A -- CURSOR UP AND RIGHT

The cursor moves up one line and to the right one character position. In Protect mode, cursor moves to first unprotected character by successively moving up and right until an unprotected location is found. If cursor is on the top line, it moves to the last line. If cursor is in the last column of a line, it moves to first column of line above.

8B -- CURSOR RIGHT

The cursor moves right one character position. In Protect mode, the cursor moves to first unprotected character position to the right. If the cursor is positioned at the end of a line, it moves to first position of next line.

8C -- CURSOR LEFT

The cursor moves left one character position. In Protect mode, the cursor moves to first unprotected character position to the left. If the cursor is positioned at start of a line, it moves to last position of line above.

8D -- CURSOR UP

The cursor moves up a line position. In Protect mode, the cursor moves up to first unprotected character position. If positioned on first line, cursor moves to the last line.

8E -- CURSOR DOWN

The cursor moves down a line position. In Protect mode, the cursor moves down to first unprotected character position. If positioned on last line, cursor moves to the first line.

8F -- LINE SKIP

The cursor moves to first column of next line down. In Protect mode, cursor moves to first unprotected position of next line or lines down.

90 -- CURSOR UP AND LEFT

The cursor moves up one line and to the left one character position. In Protect mode, cursor moves to first unprotected character by successively moving up and left until an unprotect location is found. If cursor is on top line, it moves to last last line. If cursor is in first column of a line, it moves to last column of the line above.

91 -- TAB FORWARD

The cursor moves right to first character position containing a tab mark. In Protect mode, the cursor moves right to first unprotected character position after the next protected field. If no tab mark or protected field is encountered, cursor moves to last screen character position.

92 -- TAB REVERSE

The cursor moves left to first character position containing a tab mark. In Protect mode, the cursor moves left to first unprotected character position after the last protected field. If no tab mark or protected field is encountered, cursor moves to the HOME position.

93 -- TAB SET

A tab mark is stored in CRT SCREEN IMAGE memory at position of cursor. No symbol is displayed and the cursor position is not affected. This character is not allowed when CRT controller is in Protect mode.

94 -- TAB CLEAR

A tab mark, stored in memory at position of cursor, will be cleared. No symbol is displayed and cursor position isn't affected. This character isn't allowed when CRT controller is in Protect mode.

95 -- CURSOR DOWN AND LEFT

The cursor moves down one line and to the left one character position. In Protect mode, the cursor is moved to first unprotected character by successively moving down and left until an unprotect location is found. If cursor is on last line, it moves to top line. If cursor is in first column of a line, it moves to last column of line below.

96 -- CURSOR DOWN AND RIGHT

The cursor moves down one line and to the right one character position. In Protect mode, cursor moves to the first unprotected character by successively moving down and right until an unprotected location is found. If cursor is on last line, it moves to top line. If cursor is in last column of a line, it moves to first column of line below.

97 -- START PROTECTED FIELD

A Start Protect Field symbol is displayed at cursor position. In Protect mode, the protect bit is set for all character positions from the orginal cursor position to either the end of page or the next End Protect Field symbol.

98 -- END PROTECTED FIELD

An End Protect Field symbol is displayed at cursor position. In Protect mode, protect bit is cleared from one position to the right of the orginal cursor position to either the end of page or the next Start Protect Field symbol.

99 -- ENABLE PROTECT MODE (CPU ONLY)

The Protect mode feature is enabled. All character positions having a protect bit or tab bit set will be interpreted as protected data. The CURSOR is moved right to the first unprotected character position if it is residing within a protected data field. A read while the protected mode feature is enabled results in only unprotected data being read.

9A -- READ MEMORY

9B -- WRITE MEMORY

9C -- SCROLL-UP

9D -- SCROLL-DOWN

9E -- INSERT CHARACTER IN LINE

9F -- INSERT CHARACTER IN PAGE

A0 -- CLEAR INVERSE VIDEO MODE

The inverse video feature is disabled and characters subsequently written are displayed in normal video mode.

A1 -- WRITE GREEN

All displayable data characters subsequently output to CRT Controller are displayed in green.

A2 -- WRITE BLUE

All displayable data characters subsequently output to CRT Controller are displayed in blue.

A3 -- WRITE CYAN

All displayable data characters subsequently output to CRT Controller are displayed in cyan.

A4 -- WRITE RED

All displayable data characters subsequently output to CRT Controller are displayed in red.

A5 -- WRITE YELLOW

All displayable data characters subsequently output to CRT Controller are displayed in yellow.

A6 -- WRITE MAGENTA

All displayable data characters subsequently output to CRT Controller are displayed in magenta.

A7 -- WRITE WHITE

All displayable data characters subsequently output to CRT Controller are displayed in white.

A8 -- SET INVERSE VIDEO

The inverse video feature is enabled and characters subsequently written are displayed in inverse video mode.

A9 -- BLINK OFF

All characters subsequently output to CRT Controller are displayed in non-blinking video.

AA -- INSERT CHARACTER IN FIELD/LINE

AB -- INSERT CHARACTER IN FIELD/PAGE

The Display is set to insert character in page mode. The insert character in page feature is disabled upon receipt of a second "AB" code.

AC -- BLINK ON

All characters subsequently output to CRT Controller are displayed in blinking video.

AD -- ENABLE SCROLL-UP MODE

The Display is set to Scroll-Up Mode. Not allowed in Protect Mode.

AE -- ENABLE SCROLL-DOWN MODE

The Display is set to Scroll-Down Mode. Not allowed in Protect Mode.

AF -- CLEAR MODE CONTROL

The Display Controller is cleared from Protect, Scroll-Up, Scroll-Down, and Insert Modes.

REVISION RECORD

REVISION	DESCRIPTION
0 7/25/85	ARTECS ADVANCED APPLICATIONS Design Specification #D980 -- Display Controller Interface