/

NAME:

/ - Division of Floating Point Numbers

FUNCTION:

The / function is used for the division of two floating point numbers.

FORMAT:

X=n/p

RETURNS:

The result is a floating point number.

EXAMPLE:

```
VAR1=2.0/4              ; Value of variable VAR1 will be 0.5.
VAR1=VAR2/VAR3          ; Value of variable VAR1 will be the quotient of VAR2 (dividend)
                          divided by VAR3 (divisor) or 0.5.

VAR2=H,4                ; Value of variable VAR2 will be 4.


VAR1=2.0/BTF(VAR2)      ; Value of variable VAR1 will be 0.5.
```

NOTES:

Variables may also be utilized.

RELATED COMMANDS:

DVAR

( )

NAME:

( ) - Mathematical Phrases of Floating Point Numbers

FUNCTION:

The ( ) function is used to establish mathematical phrases of floating point numbers that are to be treated as a single term.

FORMAT:

RETURNS:

The result is a floating point number.

EXAMPLE:

```
VAR1=2*(2+4)              ; Value of variable VAR1 will be 12.
VAR1=VAR2*(VAR2+VAR3)     ; Value of variable VAR1 will the product of the quantity VAR2
                            plus VAR3 multiplied by VAR2 or 12.
```

NOTES:

Variables may also be utilized.

RELATED COMMANDS:

DVAR

NAME:

! - Exponents of Floating Point Numbers

FUNCTION:

The ! function is used to express exponents of floating point numbers.

FORMAT:

X = n!p

RETURNS:

The result is a floating point number.

EXAMPLE:

```
VAR1=2!4                ; Value of variable VAR1 will be 2x2x2x2 or 16.
VAR1=VAR2!VAR3          ; Value of variable VAR1, in this case, will be evaluated by
                          VAR2xVAR2xVAR2xVAR2 or 16.
```

Exponents can be some fixed number as well as the value of some other variable as shown in the example above.

NOTES:

Variables may also be utilized.

RELATED COMMANDS:

DVAR

SIN

NAME:

SIN - Sine Value of a Floating Point Angle

FUNCTION:

The **SIN** function is used to derive the Sine value of a floating point angle.

FORMAT:

SIN(X)

RETURNS:

The result is a floating point number.

EXAMPLE:

```
VAR1=SIN(30.0)      ; Value of variable VAR1 will be the Sine of 30° or 0.5.
VAR1=SIN(VAR2)      ; Value of variable VAR1 will be the Sine of VAR2 or 0.5.
```

NOTES:

1) Variables may also be utilized.

2) All angles are expressed in decimal degrees.

RELATED COMMANDS:

DVAR

COS

NAME:

COS - Cosine Value of a Floating Point Angle

FUNCTION:

The **COS** function is used to derive the Cosine value of a floating point angle.

FORMAT:

COS(X)

RETURNS:

The result is a floating point number.

EXAMPLE:

```
VAR1=COS(60.0)      ; Value of variable VAR1 will be the Cosine of 60° or 0.5.
VAR1=COS(VAR2)      ; Value of variable VAR1 will be the Cosine of VAR2 or 0.5.
```

NOTES:

1) Variables may also be utilized.

2) All angles are expressed in decimal degrees.

RELATED COMMANDS:

DVAR

TAN

NAME:

TAN - Tangent Value of a Floating Point Angle

FUNCTION:

The **TAN** function is used to derive the Tangent value of a floating point angle.

FORMAT:

TAN(X)

RETURNS:

The result is a floating point number.

EXAMPLE:

```
VAR1=TAN(30.0)      ; Value of variable VAR1 will be the Tangent of 30° or 0.5773.
VAR1=TAN(VAR2)      ; Value of variable VAR1 will be the Tangent of VAR2 or 0.5773.
```

NOTES:

1) All angles are expressed in decimal degrees.

2) Variables may also be utilized.

RELATED COMMANDS:

DVAR

ATN

NAME:

ATN - Arctangent value of a Floating Point Number

FUNCTION:

The **ATN** function is used to derive the Arctangent (Inverse function of the Tangent) value of a floating point number.

FORMAT:

ATN(X)

RETURNS:

The result is a floating point number in decimal degrees.

EXAMPLE:

```
VAR1=ATN(1.732)      ; Value of variable VAR1 will be the Arctangent of 1.732 or 60°.
VAR1=ATN(VAR2)       ; Value of variable VAR1 will be the Arctangent of VAR2 or 60°.
```

NOTES:

Variables may also be utilized.

RELATED COMMANDS:

DVAR

DEG

NAME:

DEG - Radian to Decimal Degree Conversion of Floating Point Numbers

FUNCTION:

The **DEG** function is used to convert radians to decimal degrees (both are floating point numbers).

FORMAT:

DEG(X)

RETURNS:

The result is a floating point number in decimal degrees.

EXAMPLE:

```
VAR1=DEG(0.5236 )      ; Value of variable VAR1 will be the decimal degree equivalent
                         of 0.5236 radians or 30°.
VAR1=DEG(VAR2)         ; Value of variable VAR1 will be the decimal degree equivalent
                         of VAR2 or 30°.
```

NOTES:

Variables may also be utilized.

RELATED COMMANDS:

DVAR

RAD

NAME:

RAD - Decimal Degree to Radian Conversion of Floating Point Numbers

FUNCTION:

The **RAD** function is used to convert decimal degrees to radians (both are floating point numbers).

FORMAT:

RAD(X)

RETURNS:

The result is a floating point number in radians.

EXAMPLE:

```
VAR1=RAD(45.0)      ; Value of variable VAR1 will be the radian equivalent of 45° or
                      0.7854 radians.
VAR1=RAD(VAR2)      ; Value of variable VAR1 will be the radian equivalent of VAR2
                      or 0.7854 radians.
```

NOTES:

Variables may also be utilized.

RELATED COMMANDS:

DVAR

ABS

NAME:

ABS - Absolute Value of a Floating Point Number

FUNCTION:

The **ABS** function is used to express the Absolute value of a floating point number.

FORMAT:

ABS(X)

RETURNS:

The result is a positive floating point number.

EXAMPLE:

```
VAR1=ABS(-30)          ; Value of variable VAR1 will be 30.0.
VAR1=ABS(VAR2)         ; Value of variable VAR1 will be the Absolute value of VAR2 or
                         30.0.
```

NOTES:

Variables may also be utilized.

RELATED COMMANDS:

DVAR

SQR

NAME:

SQR - Square Root of a Floating Point Number

FUNCTION:

The **SQR** function is used to derive the Square Root value of a floating point number.

FORMAT:

SQR(X)

RETURNS:

The result is a positive floating point number.

EXAMPLE:

```
VAR1=SQR(25)        ; Value of variable VAR1 will be the Square Root of 25 or 5.0.
VAR1=SQR(VAR2)      ; Value of variable VAR1 will be the Square Root of VAR2 or
                      5.0.
```

NOTES:

1) Variables may also be utilized.

2) The Unidex 21 recognizes only positive floating point numbers.

RELATED COMMANDS:

DVAR

INT

NAME:

INT - Rounding of Floating Point Numbers

FUNCTION:

The **INT** function is used to round-off the fractional part of any floating point number.

FORMAT:

INT(X)

RETURNS:

The result is a floating point integer.

EXAMPLE:

```
VAR1=INT(123.05)      ; Value of variable VAR1 will be 123.0.
VAR1=INT(VAR2)        ; Value of variable VAR1 will round-off the fractional part of
                        VAR2 resulting in a value of VAR1 equals 123.0.

VAR3=INT(123.5)       ; Value of variable VAR1 will be 124.0.
VAR3=INT(VAR4)        ; Value of variable VAR3 will round-off the fractional part of
                        VAR4 resulting in a value of VAR4 equals 124.0.
```

NOTES:

Variables may also be utilized.

RELATED COMMANDS:

DVAR

BTF

NAME:

BTF - Convert a Binary Number to a Floating Point Number

FUNCTION:

The **BTF** function is used to convert a binary number to a floating point number. Floating point numbers and binary numbers cannot be used within the same equation, unless one or the other is converted.

FORMAT:

BTF(X)

RETURNS:

The result is a floating point number.

EXAMPLE:

```
(DVAR,VAR1,VAR2)          ; Define variables VAR1 and VAR2.

VAR1=H,1E                 ; Value of variable VAR1 will be 1E Hex (30.0).
VAR2=BTF(VAR1)            ; Value of variable VAR2 will be 30.0.
```

NOTES:

Variables may also be utilized.

RELATED COMMANDS:

DVAR

FTB

NAME:

FTB - Convert a Floating Point Number to a Binary Number

FUNCTION:

The **FTB** function is used to convert a floating point number to a binary number. Floating point numbers and binary numbers cannot be used within the same equation, unless one or the other is converted.

FORMAT:

FTB(X)

RETURNS:

The result is a binary number.

EXAMPLE:

```
VAR1=40.0                ; Value of variable VAR1 will be H,28.
VAR2=FTB(VAR1)           ; Value of variable VAR2 will be converted to H,28.
```

NOTES:

1) Variables may also be utilized.

2) The fractional portion of the floating point number is rounded to the nearest integer.

RELATED COMMANDS:

DVAR

.EQ.

NAME:

.EQ. - Condition compares two Floating Point Numbers

FUNCTION:

The **.EQ.** function is used to compare two floating point numbers. If the numbers **are equal,** the result is "True" (1). If the numbers **are not equal,** the result is "False" (0).

FORMAT:

floating point number1**.EQ.**floating point number2

RETURNS:

The result is a floating point number, "1" for True, "0" for False.

EXAMPLE:

```
(JUMP,ENT1,VAR1.EQ.SIN<30>)   ; Program flow will go to Entry Point ENT1 if VAR1 is equal to
                                SIN <30>.

VAR2=VAR1.EQ.SIN<30>          ; VAR2 equals H,1 if VAR1 is equal to SIN<30>, otherwise
                                VAR2 equals H,0.
```

NOTES:

Variables may also be utilized.

RELATED COMMANDS:

DVAR

.NE.

NAME:

.NE. - Condition compares two Floating Point Numbers

FUNCTION:

The **.NE.** function compares two floating point numbers. If the numbers **are not equal** the result is "True" (1). If the numbers **are equal**, the result is "False" (0).

FORMAT:

floating point number1**.NE.**floating point number2

RETURNS:

The result is a floating point number, "1" for True, "0" for False.

EXAMPLE:

```
(JUMP,ENT1,VAR1.NE.SIN<30>)   ; Program flow will go to Entry Point ENT1 if VAR1 is not equal
                                to SIN <30>.

VAR2=VAR1.NE.SIN<30>          ; VAR2 equals H,1 if VAR1 is not equal to SIN<30>, otherwise
                                VAR2 equals H,0.
```

NOTES:

Variables may also be utilized.

RELATED COMMANDS:

DVAR

.GT.

NAME:

.GT. - Condition compares two Floating Point Numbers

FUNCTION:

The **.GT.** function compares two floating point numbers. If the value of the first number **is greater** than the value of the second number, the result is "True" (1). If the value of the first number **is not greater** than the value of the second number, the result will be "False" (0).

FORMAT:

floating point number1**.GT.**floating point number2

RETURNS:

The result is a floating point number, "1" for True, "0" for False.

EXAMPLE:

(JUMP,ENT1,VAR1**.GT.**SIN<30>)	; Program flow will go to Entry Point ENT1 if VAR1 **is greater** than SIN <30>.
VAR2=VAR1**.GT.**SIN<30>	; VAR2 equals H,1 if VAR1 **is greater** than SIN<30>, otherwise VAR2 equals H,0.

NOTES:

Variables may also be utilized.

RELATED COMMANDS:

DVAR

.GE.

NAME:

.GE. - Condition compares two Floating Point Numbers

FUNCTION:

The **.GT.** function compares two floating point numbers. If the value of the first number **is greater than or equal to** the value of the second number, the result is "True" (1). If the value of the first number **is not greater than or equal to** the value of the second number, the result will be "False" (0).

FORMAT:

floating point number1**.GE.**floating point number2

RETURNS:

The result is a floating point number, "1" for True, "0" for False.

EXAMPLE:

```
(JUMP,ENT1,VAR1.GE.SIN<30>)   ; Program flow will go to Entry Point ENT1 if VAR1 is greater
                                than or equal to SIN <30>.

VAR2=VAR1.GE.SIN<30>          ; VAR2 equals H,1 if VAR1 is greater than or equal to SIN
                                <30>, otherwise VAR2 equals H,0.
```

NOTES:

Variables may also be utilized.

RELATED COMMANDS:

DVAR

CHAPTER 5: IEEE-488 OPTION

The IEEE-488 option for the Unidex 21 provides control of the Unidex 21 from a host computer through the IEEE-488 Bus. Once communication is established, the command sequence and operating instructions are as described in the *Unidex 21 Programming Manual* and the *Unidex 21 User's Manual.*

SECTION 5-1 HARDWARE REQUIREMENTS

5-1-1 IEEE-488 INTERFACE

IEEE-488 has 8 data lines and 8 control lines. (Refer to the *Unidex 21 Hardware Manual* for connector details.) It can accommodate up to 14 devices and provides a Service Request line from all devices to the Bus Controller. These properties lead to a more rapid form of communication between Unidex 21 and the controller. Bus disciplines are not necessary if the controller has IEEE-488 interface and device driver software that interfaces with the language to be used.

5-1-2 SIGNAL LINES OF THE IEEE-488 BUS

The IEEE-488 transfers data and commands between devices through 16 signal wires.

Eight of the lines are for the transfer of data (DI01 to DI08). Data and message transfers are asynchronous and are coordinated by the three handshake lines.

The remaining five lines, for example "ATN" (Attention) and "SRQ" (Service Request), are used for bus management. Each line, when asserted Low (ground), represents a single line message sent on the bus. (Refer to the *Unidex 21 Hardware Manual* for a description of each of these lines.)

5-1-3 CABLE RESTRICTIONS OF THE IEEE-488 BUS

The devices in a system are connected by a 24-wire cable using 24-pin connectors as specified in the IEEE-488 standard.

Certain limitations exist concerning the length of the cables and the number of devices allowable on the bus. The maximum number of devices on the bus is 14. The total length of the cable is limited to 20 meters (65.6 feet) or 2 meters multiplied by the number of devices (whichever is shorter in length). A list of cable suppliers follows:

CABLE MANUFACTURERS

HEWLETT-PACKARD
Palo Alto, California 94304

PN	Length
HP 10833D	.5 Meter
HP 10833A	1 Meter
HP 10833B	2 Meters
HP 10833C	4 Meters
HP 10834A	Adapter

BELDEN CORPORATION
Richmond, Indiana 47374

PN	Length
9642	1 Meter
9643	2 Meters
9644	4 Meters
9645	8 Meters
9646	16 Meter

SECTION 5-2: SET UP

Connect the Controller to the IEEE-488 Connector (P11) on the Rear Panel of the Unidex 21.

NOTE: **In order for the Unidex 21 to recognize the IEEE-488 interface, connection must be made to the Unidex 21 before System power up or Reset.**

Power Up the Unidex 21.

5-2-1 INITIAL CONFIGURATION

Remote operation of the Unidex 21 may be initiated with several configurations. The following is a list of the possible configurations and the key combinations that must be entered to initiate them.

NOTE: The following keyboard entries may be made either from the Remote Controller or the Unidex 21's Front Panel.

Condition	Keyboard Entry
Remote Control OFF	Ctrl\0
Remote Control ON, Display OFF	
RS-232 Port A	Ctrl\1
IEEE-488	Ctrl\3
Remote ON, Display ON	
RS-232 Port A	Ctrl\2
IEEE-488	Ctrl\4

5-2-2 PARAMETER SETTINGS

Before using an IEEE-488 device to control the Unidex 21, certain Parameter settings must be established. The paragraphs that follow provide instructions for setting the applicable parameters. For a detailed description of all parameter settings refer to the *Unidex 21 User's Manual.*

The Initial Selection Screen shown below is displayed upon power-up of the Unidex 21:

```
UNIDEX 21    Version XX

EPROM OK   PARAMETER OK    RAM checksum



User's RAM (bytes) = xxxxxxx

Edit, File, Machine, Parameter, Test, System, Batch, Console, Debug
```

Press the "P" key to enter the Parameter Mode.

The following screen will be displayed:

```
0 : System password                    1 : Skip auto-boot function ?
2 : IDX buffer 1 block only ?          3 : IDX seg. calculate base (1/2/3)
4 : COMM input feedback ?              5 : System default at metric ?
6 : RS232 protocol port-A              7 : Additional RAM in 1024 bytes
8 : RS232 protocol port-B              9 : Debug display is at front panel ?
10 : RS232 time out (seconds)          11 : Parts program stack size in bytes
12 : Edit block buffer (1 to 40)       13 : Edit default Char-insert ?
14 : Edit default Line-insert ?        15 : Edit TAB space
16 : End of all file code CHR$(n)      17 : End of file code CHR$(n)
18 : Beeper duration (1 to 280) ms     19 : Double side floppy disk ?
20 : Beeper frequency (2 to 20K)       21 : Display blank-out (minutes)
22 : MFO inc./step (0 to 100)          23 : Tracking display program step ?
24 : Y pixel size reduce to (%)        25 : Print screen to port-A ?
26 : Joystick axis pair                27 : Digitize with joystick ?

200 : NEXT PAGE                        201 : Axes auto-tune

300 : Load/save parameter              301 : Front panel function keys
401 : 1st axis    402 : 2nd axis       403 : 3rd axis    404 : 4th axis
405 : 5th axis    406 : 6th axis       407 : 7th axis    408 : 8th axis

ctrl-Quit, number <cr> to each parameter =
```

5-2-2-1 COMM INPUT FEEDBACK

General Parameter 4, Communication Input Feedback, determines whether "echo" characters are required for RS-232 or IEEE-488 input. The default setting requires "echo" characters. Refer to the *Unidex 21 Programming Manual* for details concerning the (COMM command.

Enter "4" to change the Feedback status. Press the "N" key to toggle between Yes (feedback echoes are required) or No (no echo characters required).

5-2-2-2 TIME-OUT

The Unidex 21 contains a time-out feature when files are input or output through the RS-232 or IEEE-488 ports. When the IEEE-488 mode of file transmission is initiated, the Unidex 21 will "look" for the data for a predetermined amount of time before displaying an error message. The default time is 600 seconds.

Enter "10" to set the length of time the Unidex 21 will wait for a return signal following an IEEE-488 transmission. Enter the new time in seconds.

5-2-2-3 END OF ALL FILE CODE

General Parameter 16 provides the User the ability to establish a character that will signal to the Unidex 21 that data transmission is complete. The default character is 17. (See also Parameter 53, IEEE-488 Set Up)

Enter "16" to change the End of All File Character. Enter the new End of All File character(s).

5-2-2-4 END OF FILE CODE

A character may be established to signal the Unidex 21 that a file data transmission is complete. General Parameter 17 provides the User the ability to establish an End of File Code for each system. The default character is "9". (See also Parameter 53, IEEE-488 Set Up)

Enter "17" to change the End of File character. Enter the new End of File character.

5-2-2-5 POWER ON REMOTE CONTROL

General Parameter 31 may be configured such that the Unidex 21 will be in the desired Remote state following a power-up or reset. The default is for no Remote Control.

Enter "31"to change the Remote Power status.

Enter "0" for no Remote Control upon power-up or reset.

Enter a "1" to establish Remote Control of the Unidex 21 through RS-232 Port A following a power-up or reset. The Unidex 21 Front Panel display will not be active.

Enter a "2" to establish Remote Control of the Unidex 21 through RS-232 Port A following a power-up or reset. The Unidex 21 Front Panel display is active.

Enter a "3" to establish Remote Control of the Unidex 21 through the IEEE-488 Port following a power-up or reset. The Unidex 21 Front Panel display will not be active.

Enter a "4" to establish Remote Control of the Unidex 21 through the IEEE-488 Port following a power-up or reset. The Unidex 21 Front Panel Display will be active.

5-2-2-6 IEEE-488 SET-UP

General Parameter 53 provides the User with a variety of set-up parameters for use specifically with a IEEE-488 interface.

Enter "53" to establish IEEE-488 parameters. The display will be:

IEEE488 SET UP

0: address mode (0 talk only) (1 listen only) (2 major only) (3 major/minor)
(4 primary/secondary) (5 primary/primary) = 2
1: 1st address (0 to 31) = 2
2: 2nd address (0 to 31) = 3
3: PPR (0 no) (1 to 8 - in phase) (9 to 16 - reverse phase) = 1
4: EOS data (0 to FF) = 0A
5: EOS bits (0-7) (1-8) = 1
6: set EOI with last byte of write? (0 - yes) (1 - no) = 0
7: terminate read on EOS? (0 - yes) (1 - no) = 0
8: set EOI with last byte of write? (0 - yes) (1 - no) = 0

NOTE: EOS will not affect EOI during File mode Input/Output case
Input - Unidex 21 will wait for EOI or end-of-file code
Output - Unidex 21 will set EOI with end-of-file code

Ctrl-Quit, Ctrl Default , code/nnnnnnnn =

A description of each of the IEEE-488 parameters follows:

Code 0 - Establishes the mode to which the Unidex 21 is to be addressed. The default is setting "2", the Unidex 21 is addressed as Major (1 Address only)

Enter "0/0" to configure the Unidex 21 as a device that only sends data to receivers (Talker).

Enter "0/1" to configure the Unidex 21 as a device that only receives data messages from a Talker (Listener).

Enter "0/2" to configure the Unidex 21 for 1 Address Bit.

Enter "0/3" to configure the Unidex 21 for 2 Address Bits, one being Major and one being Minor. (Either may be assigned to be a Talker or a Listener.)

Enter "0/4" to configure the Unidex 21 such that the 1st address is Primary (Talker) and the 2nd address is Secondary (Listener).

Enter "0/5" to configure the Unidex 21 such that both the 1st address and the 2nd address are Primary (Talkers)

Code 1 - Establishes the address byte for the 1st address. Enter the desired address byte for the 1st address (1/1 thru 31). The default address byte is 2.

Code 2 - Establishes the address byte for the 2nd address. Enter the desired address byte for the 2nd address (2/1 thru 31). The default address byte is 3.

Code 3 - Configures Parallel Poll Response. The default configuration is In-Phase Parallel Polling at address "1".

Enter "3/0" for no Parallel Polling.

Enter"3/1 thru 8" to establish an address for In-Phase Parallel Polling of the Unidex 21. (Selected Bit will go "High" with Parallel Poll.)

Enter"3/9 thru 16" to establish an address for Reverse Phase Parallel Polling of the Unidex 21. (Selected bit will go "Low" with Parallel Poll.)

Code 4 - Establishes the character(s) (0 thru FF) used to terminate a read or an output operation. The default End Of String data is 0A.

Code 5 - The number of bits available for EOS data must be delineated by this parameter. ("0" establishes 7 bits, "1" establishes 8 bits) The default is "1" for an 8 bit system.

Code 6 - The End Of Identify signal may be sent with the End Of String signal to indicate to the Unidex 21 the last byte of the data string to be transmitted. The default is for the EOI signal to accompany the EOS signal.

Enter "6/0" for the EOI signal occur in conjunction with the EOS signal.

Enter "6/1" for no EOI signal.

Code 7 - The EOS signal may be used to indicate read data termination. This parameter may be used to configure the Unidex 21 such that the EOS signal will or will not terminate the reading of data. The default is "0", the reading of data will be terminated by the EOS signal.

Enter "7/0" if the EOS signal is to terminate the reading of data.

Enter "7/1" if the EOS signal is not to terminate the reading of data.

Code 8 - The EOI signal may be set to be used in conjunction with the last byte of the write signal to eliminate the need for an EOS character at the end of every data string.

Enter "8/0" to set the EOI signal with the last byte of the write signal.

Enter "8/1" if the EOI signal is not to be sent with the last byte of the write signal.

SECTION 5-3: SERVICE REQUEST AND POLLING

5-3-1 SERVICE REQUEST

A Service Request signal is necessary in remote operation where a host controller is a master and a controlled device is a slave. The purpose of the Service Request signal is for the slave device to catch the attention of the master controller.

The slave device has the capacity to send a request signal to the master controller whenever it requires the attention of the master. The reason for the request may be an error condition or the completion of a task.

NOTE: If the Service Request was initiated by an error condition, the Unidex 21 will not respond to any further system commands until it is serial polled by the master controller.

The Unidex 21 implements a Service Request by asserting the SRQ line on the IEEE-488 bus. The master controller may be programmed to be interupted by a SRQ and to take the necessary action.

5-3-1 PARALLEL POLLING

Parallel Polling is done to identify configured devices and indicate to the host controller when a device on the IEEE-488 bus is requesting service (SRQ). A composite poll response is sent to the host controller. The host controller then may Serial Poll the devices to determine the device number and the nature of the service request.

The Parallel Poll bit assigned to each Unidex 21 is selected in the Parameter Mode (See Section 5-2.)

5-3-2 SERIAL POLLING

Serial Polling is done on one device at a time to determine which device has made a Service Request and the reason for the request. Any device may be polled at any time, regardless of the number of devices on the line.

A Unidex 21 will Request Service (set SRQ) at specific times, such as when a program is completely executed. At such a time, further operations will be suspended until Unidex 21 is Serial Polled by the Controller. Upon being polled, the Unidex 21 will transmit its status.

The Unidex 21 sends following status codes as a result of a Serial Poll:

CODE	DESCRIPTION
40H	Remote status good, awaits instruction
C0H	Remote Error, read 1 or 2 byte error code. Unidex 21 awaits Serial Poll.
41H	Remote status good, awaits Input
42H	Remote status good, awaits Output

SECTION 5-4: OPERATION

Following initialization, the Unidex 21 is controlled by the Host Controller.

Communication from the Host Controller to the Unidex 21 is accomplished in the same manner as communication from the TeleVideo 905 Terminal. (See Chapter 2 of the *Unidex 21 User's Manual.*)

> **NOTE: Regardless of the keyboard configuration of the Host Controller, communication to the Unidex 21 must follow TeleVideo 905 Terminal input conventions**

The next Section provides a complete list of possible error codes and their corresponding messages.

5-4-1: ERROR CODES AND MESSAGES

During data transmission and/or performance of a function, if an error is detected the Unidex 21 will feed back an error code in the following format:

Master error Code (C0H) followed by the Secondary Error Code (1 or 2 bytes)

The following is a list of the Secondary Error Codes and Messages as well as the function from which they may occur.

5-4-1-1: EDIT MODE

The following Secondary Error Codes/Messages may appear while in the Edit Mode:

10H - Input key undefined
11H - Not enough User's RAM space
12H - File format error
13H - File not found
14H - File read only
15H - Block functions got range error
16H - Input key not ctrl-Q or ctrl-W
17H - Input key not Y or N

5-4-1-2: FILE MODE

The following Secondary Error Codes/Messages may appear while in the File Mode:

20H - Input key undefined
21H - Undefined I/O port
22H - File format error
23H - File not found
24H - File read only
25H - File currently active
26H - No disk
27H - Not enough User's RAM space
28H - File verify error
29H - RS-232/IEEE-488 time out, or transfer interface fail

2AH - Target file already exists
2BH - Not enough disk space
2CH - Disk write protected
2DH - Disk access fail
2EH - Disk up load fail

5-4-1-3: MACHINE MODE

The following Secondary Error Codes/Messages may appear while in the Machine Mode:

30H - Input key undefined
31H - File not found
32H - Illegal filename.type
33H - Sub-program not found
34H - Can't open read file
35H - Can't open write file
36H - Write file not closed

40H - Undefined symbol
41H - Format error
42H - Undefined Type 2 command
43H - Undefined G code
44H - Undefined M code
45H - Illegal BCD format
46H - Illegal system variable
47H - Undefined variable
48H - Illegal I/O format
49H - Illegal mathematics format
4AH - Undefined array
4BH - Miss CLS command
4CH - Undefined subroutine
4DH - Undefined entry
4EH - Undefined condition
4FH - Stack overflow

50H - Miss return address
51H - Undefined safe zone
52H - Illegal function in MDI
53H - Not enough memory space
54H - Circle miss center point
55H - No feed rate

56H - Move into safe zone
57H - Undefined data in read file
58H - In ICRC look ahead
59H - <no>
5AH - MALC format error
5BH - CPAG format error, or need (MALC, < 1, option
5CH - Undefined H code
5DH - Undefined axis plane
5EH - Axis can't be both master & slave, or more than 1 master
5FH - PLC Option not foun, or ladder program not exist
60H - Need (MALC to allocate memory
61H - No recorded position to play back, need (RECO
62H - PSO Option not found

5-4-1-4: PARAMETER MODE

The following Secondary Error Codes/Messages may appear while in the Parameter Mode:

70H - Input key undefined
71H - Input data error
72H - Not enough memory for p-meter save
73H - File exist already for p-meter save
74H - File not found for p-meter load

5-4-1-5: TEST MODE

The following Secondary Error Codes/Messages may appear while in the Test Mode:

80H - Input key undefined
81H - RAM fail at (0) case
82H - RAM fail at (F) case
83H - RAM fail at (5) case
84H - RAM fail at (A) case
85H - RAM checksum error
86H - EPROM checksum error
87H - PARAMETER checksum error

5-4-1-6: SYSTEM MODE

The following Secondary Error Codes/Messages may appear while in the System Mode:

90H - Input key undefined
91H - TIME input error
92H - DATE input error

5-4-1-7: MISC. ERRORS

The following Secondary Error Codes/Messages may also appear during Remote operation:

A0H - Input key undefined
A1H - No password privlege
A2H - Batch file not found or format error
A3H - RAM error during power on test
A4H - Indexing board error during power on test
A5H - Real time clock fail, set at default data

5-4-1-8: SPECIAL REMOTE SYSTEM-FAIL ERROR

During data transmission and/or performance of a function, a Special Remote System error having two bytes of Secondary Error may be detected, it will be displayed in the following format:

Master Error Code (C0H) followed by 0E0H and the Secondary Error Code

The following Secondary Error Codes/Messages may appear during data transmission and/or performance of a function:

80H - Indexer 68000 CPU Bus Error
81H - Indexer 68000 Address Error
82H - Indexer 68000 Illegal Instruction
83H - Indexer 68000 Zero Divide
84H - Indexer 68000 Line 1010 Emulation
85H - Indexer 68000 Line 1111 Emulation

86H - Indexer 68000 Uninitialized Interrupt Vector
87H - Indexer 68000 Spurious Interrupt
88H - Indexer Dual-Port Ram Group B Checksum
89H - Indexer Dual-Port Ram Group B Data Out of Boundary
8AH - Feedrate is 0 or Negative Value
8BH - Invalid Sin/Cos Combination
8CH - Invalid Contouring Plane

A0H - Axis in Limit (Software or Hardware)
A1H - Axis Trap (Velocity or Position or Integral)
A2H - M Function Output Fail to Detect the Acknowledge Signal
A3H - S Function Output Fail to Detect the Acknowledge Signal
A4H - T Function Output Fail to Detect the Acknowledge Signal
A5H - DSP Feedback Illegal Code

B0H - MFO = 0 or Feedhold is On
B1H - AC Fail
B2H - Joy-Stick/ Trackball/Handwheel Motion Hit Software or Hardware Limit

SECTION 5-5: SAMPLE PROGRAM

The following program is representative of a QuickBasic Program (Version 4.0+) that may be sent to the Unidex 21 from a Host Controller.

NOTE: This program is applicable only to controllers equipped with the GPIB Interface Board.(National Instruments, Austin, Texas)

```
' Common GPIB staus variables:
COMMON SHARED /NISTABLK/ IBSTA%, IBERR%, IBCNT%
' GPIB Subroutine Declarations:

  DECLARE SUB IBBNA (BD%, BDNAME$)
  DECLARE SUB IBCAC (BD%, V%)
  DECLARE SUB IBCLR (BD%)
  DECLARE SUB IBCMD (BD%, CMD$)
  DECLARE SUB IBCMDA (BD%, CMD$)
  DECLARE SUB IBDMA (BD%, V%)
  DECLARE SUB IBEOS (BD%, V%)
  DECLARE SUB IBEOT (BD%, V%)
  DECLARE SUB IBFIND (BDNAME$, BD%)
  DECLARE SUB IBGTS (BD%, V%)
  DECLARE SUB IBIST (BD%, V%)
  DECLARE SUB IBLOC (BD%)
  DECLARE SUB IBONL (BD%, V%)
  DECLARE SUB IBPAD (BD%, V%)
  DECLARE SUB IBPCT (BD%)
  DECLARE SUB IBPPC (BD%, V%)
  DECLARE SUB IBRD (BD%, RD$)
  DECLARE SUB IBRDA (BD%, RD$)
  DECLARE SUB IBRDF (BD%, FLNAME$)
  DECLARE SUB IBRDI (BD%, IARR%( ), CNT%)
  DECLARE SUB IBRDIA (BD%, IARR% ( ), CNT%)
  DECLARE SUB IBRPP (BD%, PPR%)
  DECLARE SUB IBRSC (BD%, V%)
```

```
DECLARE SUB  IBRSP (BD%, SPR%)
DECLARE SUB IBRSV (BD%, V%)
DECLARE SUB IBSAD (BD%, V%)
DECLARE SUB IBSIC (BD%)
DECLARE SUB IBSRE (BD%, V%)
DECLARE SUB IBSTOP (BD%)
DECLARE SUB IBTMO (BD%, V%)
DECLARE SUB IBTRAP (MASK%, MODE%)
DECLARE SUB IBTRG (BD%)
DECLARE SUB IBWAIT (BD%, MASK%)
DECLARE SUB IBWRT (BD%, WRT$)
DECLARE SUB IBWRTA (BD%,WRT$)
DECLARE SUB IBWRTF (BD%, FLNAME$)
DECLARE SUB IBWRTI (BD%, IARR%( ), CNT%)
DECLARE SUB IBWRTIA (BD%, IARR%( ), CNT%)

' GPIB Function Declarations

DECLARE FUNCTION ILBNA% (BD%, BDNAME$)
DECLARE FUNCTION ILCAC% (BD%, V%)
DECLARE FUNCTION ILCLR% (BD%)
DECLARE FUNCTION ILCMD% (BD%, CMD$, CNT%)
DECLARE FUNCTION ILCMDA% (BD%, CMD$, CNT%)
DECLARE FUNCTION ILDMA% (BD%, V%)
DECLARE FUNCTION ILEOS% (BD%, V%)
DECLARE FUNCTION ILEOT% (BD%, V%)
DECLARE FUNCTION ILFIND% (BDNAME$)
DECLARE FUNCTION ILGTS% (BD%, V%)
DECLARE FUNCTION ILIST% (BD%, V%)
DECLARE FUNCTION ILLOC% (BD%)
DECLARE FUNCTION ILONL% (BD%, V%)
DECLARE FUNCTION ILPAD% (BD% V%)
DECLARE FUNCTION ILPCT (BD%)
DECLARE FUNCTION ILPPC% (BD%, V%)
DECLARE FUNCTION ILRD% (BD%, RD$, CNT%)
DECLARE FUNCTION ILRDA% (BD%, RD$, CNT%)
DECLARE FUNCTION ILRDF% (BD%, FLNAME$)
```

```
DECLARE FUNCTION ILRDI% (BD%, IARR%,( ), CNT%)
DECLARE FUNCTION ILRDIA% (BD%, IARR%( ), CNT%)
DECLARE FUNCTION ILRPP% (BD%, PPR%)
DECLARE FUNCTION ILRSC% (BD%, V%)
DECLARE FUNCTION ILRSP% (BD%, SPR%)
DECLARE FUNCTION ILRSV% (BD%, V%)
DECLARE FUNCTION ILSAD% (BD%, V%)
DECLARE FUNCTION ILSIC% (BD%)
DECLARE FUNCTION ILSRE% (BD%, V%)
DECLARE FUNCTION ILSTOP% (BD%)
DECLARE FUNCTION ILTMO% (BD%, V%)
DECLARE FUNCTION ILTRAP% (MASK%, MODE%)
DECLARE FUNCTION ILTRG% (BD%)
DECLARE FUNCTION ILWAIT% (BD%, MASK%)
DECLARE FUNCTION ILWRT% (BD%, WRT$, CNT%)
DECLARE FUNCTION ILWRTA% (BD%, WRT$, CNT%)
DECLARE FUNCTION ILWRTF% (BD%, FLNAME$)
DECLARE FUNCTION ILWRTI% (BD%, IARR%( ), CNT%)
DECLARE FUNCTION ILWRTIA% (BD%, IARR%( ), CNT%)

CLS

DIM B$(255)


BOARD$ = "GPIB0"
CALL IBFIND(BOARD$, BOD%)
IF BOD% < 0 THEN 1000

PRINT "BOARD DESCRIPTOR = ", BOD%

BOARD$ = "U21"
CALL IBFIND(BOARD$, EQP%)
IF EQP% < 0 THEN 2000

PRINT "DEVICE DESCRIPTOR = ", EQP%
```

```
      CALL IBCLR (EQP%)

5     PRINT " "
      PRINT " 0 - REMOTE CONTROL  1 - INPUT FILES"
      PRINT " 2 - OUTPUT FILES        3 - INPUT THEN OUTPUT STRING"
6     PRINT " "
      INPUT " SELECT 0 TO 3 ONLY = ", SE

      IF SE = 0 THEN 10
      IF SE = 1 THEN 20
      IF SE = 2 THEN 30
      IF SE = 3 THEN 40
      GOTO 6

10    PRINT "KEY INPUT = ";
11    A$ = INKEY$
      IF A$ = " " THEN 11
      PRINT ASC(A$); A$
      GOSUB 200
      GOTO 10

20    NST% = 64 +2
      GOSUB 400

      ' PRINT "ST% = "; ST%

      MASK% = &H6000

      B$ = SPACE$(250)
      CALL IBRD(EQP%, B$)

21    CALL IBWAIT(EQP%, MASK%)
      IBS% = IBSTA% \ 256

      IF IBS% < 64 THEN 22
      IBS% = IBS% - 64
22    IF IBS% < 32 THEN 21
```

```
        PRINT B$

25      INPUT "FILE INPUT DONE, 0-QUIT, 1-SENT BACK "; SE
        IF SE = 0 THEN 5
        GOTO 31
30      INPUT "STRING TO OUTPUT = ", B$

31      NST% = 64 + 1
        GOSUB 400

        ' PRINT "ST% ="; ST%

        CALL IBWRT(EQP%, B$)
        INPUT "OUTPUT DONE, 0-QUIT, 1-MORE"; SE
        IF SE = 0 THEN 5
        GOTO 30

40      CAS = 0
41      NST% = 64 + 2
        GOSUB 400

        MASK% = &H6000
        B$ = SPACE$(30)
        CALL IBRD(EQP%, B$)
42      CALL IBWAIT(EQP%, MASK%)

        IBS% = IBSTA% \ 256
        IF IBS% < 64 THEN 43
        IBS% = IBS% - 64
43      IF IBS% < 32 THEN 42

        IF CAS < > 0 THEN 350
        PRINT B$

        INPUT "DATA TO OUTPUT = ", DA$
        DA$ = DA$ + CHR$(13)
```

```
        NST% = 64 + 1
        GOSUB 400

        CALL IBWRT(EQP%, DA$)

        GOTO 40

200     CALL IBWRT(EQP%, A$)
        MASK% = &HE800
210     CALL IBWAIT(EQP%, MASK%)

        IF IBSTA% / 256 > = 128 THEN 300
        IF IBSTA% / 256 > = 64 THEN 310
        IF IBSTA% / 256 > = 32 THEN 320
        IF IBSTA% / 256 > = 8 THEN 330

        GOTO 210

300     PRINT "GPIB ERROR"
        RETURN
310     PRINT "TIME OUT"
        RETURN
320     PRINT "DETECT END"
        RETURN
330     PRINT "SERIAL POLL = ";
        CALL IBRSP(EQP%, ST%)
        IF ST% = 192 THEN 340
        PRINT ST%
335     RETURN
340     PRINT ST%;
        CAS = 1
        GOTO 41
350     PRINT ASC(MID$(B$, 1, 1)); ASC(MID$(B$, 2, 1))
        RETURN
```

```
400   MASK% = &H4800
410   CALL IBWAIT(EQP%, MASK%)
      IBS% = (IBSTA% \ 256) AND 72

      IF IBS% < 72 THEN 420
      IBS% = IBS% - 64
420   IF IBS% < 8 THEN 410

      CALL IBRSP(EQP%, ST%)
      IF ST% = NST% THEN 335
      GOTO 410

1000  PRINT "BOARD NOT FOUND"
      STOP
2000  PRINT "DEVICE NOT FOUND"
3000  STOP

      END
```