



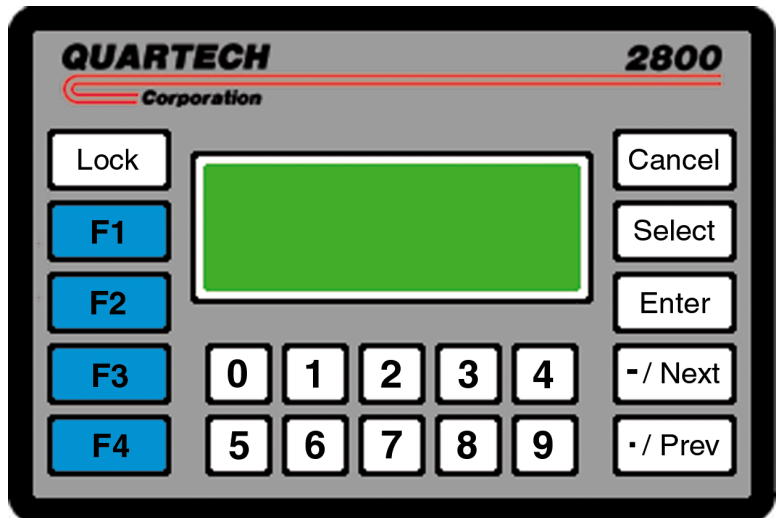
## ***Table Of Contents***

Section 1: Introduction .....	1
Additional Drivers .....	1
Section 2: Generic ASCII Driver .....	2
ASCII String Editor .....	3
Setup Parameters .....	5
Function Keys .....	7
Field Notes .....	9
Section 3: Slave ASCII Terminal Driver .....	9
Communication Setup .....	9
Displaying Text .....	10
Control Sequences .....	11
Read OIT Status .....	13
String Editor .....	14
Keypad Activation .....	14
CTS Handshaking .....	15
Power-up String .....	15
Networking OITs .....	15
Appendix A: Communication Cables .....	5
Appendix B: Standard ASCII Code Table .....	4

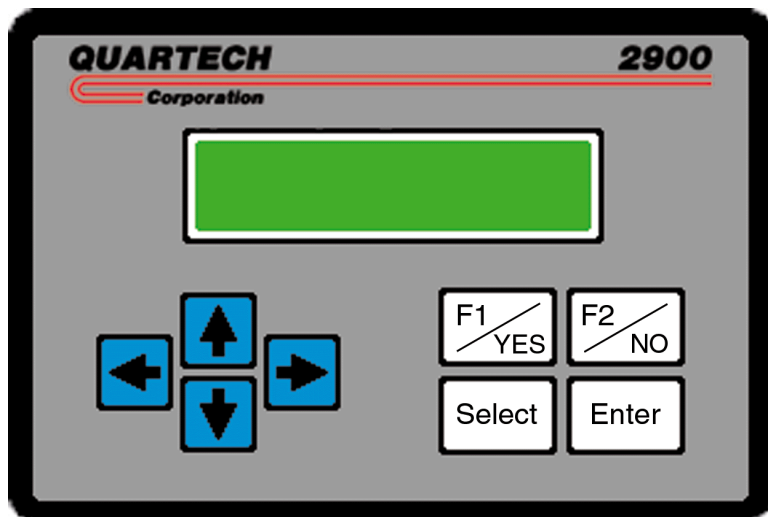
The 2800 and 2900 Operator Interface Terminals (OIT) share a common software package, a common panel cutout and have a near identical list of communication drivers. This manual provides important information relative to the general purpose ASCII drivers.

The 2800 has a twenty character by four line display which is available as a back lit LCD or vacuum fluorescent display.

Two ASCII drivers are available for the 2800 that allow it to function as a master device or as a slave device. When operated as a master device the 2800 can be configured to emulate the communication protocol of a host device.



Both the 2800 and 2900 can function as a slave device. In this mode the units operate much like a standard ASCII dumb terminal. Cursor and option control is performed through sequence codes that are completely user definable. In addition RS-232 and RS-485 point-to-point or network configurations are possible.



The 2900 has a twenty character by two line display which is available as a back lit LCD or vacuum fluorescent display.

At Quarteck we are continually working on new communication drivers for the 2800 and 2900. These new drivers and upgrades to existing drivers are available free of charge via our web site at [www.quartechcorp.com](http://www.quartechcorp.com).

### Other available Communication Drivers include:

Allen-Bradley SLC500, DH-485  
Allen-Bradley SLC500, DF1  
Allen-Bradley PLC-5, DF1  
Fuji Electric N Series  
Mitsubishi FX Series  
Omron Host Link  
Toshiba, EX100 & T-Series

Allen-Bradley PLC-2  
Allen-Bradley MicroLogix, DF1  
Allen-Bradley Ultra 100/200 Drives  
GE Fanuc Series 90.  
Modicon Modbus.  
Idec, Micro3 & Micro3C  
Yaskawa, MP930 Memobus

## Section 2: Generic ASCII Driver - 2800 only

The generic ASCII driver can be easily configured to send and receive ASCII strings used by many devices. This driver differs from the specific PLC drivers in three major areas.

- ! Only one screen can be displayed at any time. Only one Trigger Word is available.
- ! The Command Word and Screen Trigger are not required for normal operation.
- ! The Bit Status field is not available.

The OIT is configured using the ScreenMaker 2000 Configuration Editor for Microsoft Windows or NT. Support for the Generic ASCII Driver begins with version 1.12. Both the configuration file and driver file are downloaded using the Configuration Editor. As with the PLC drivers, all variables are referenced by tag names. A Data Reference Editor is part of ScreenMaker 2000 and allows the system designer to generate the tag name list. For the ASCII driver the Data Reference Editor is different than that used by the PLC drivers. With a PLC driver, a PLC specific address and data format are specified for each tag name. With the ASCII driver, up to four strings can be specified for each tag name. Here is a description of the four strings.

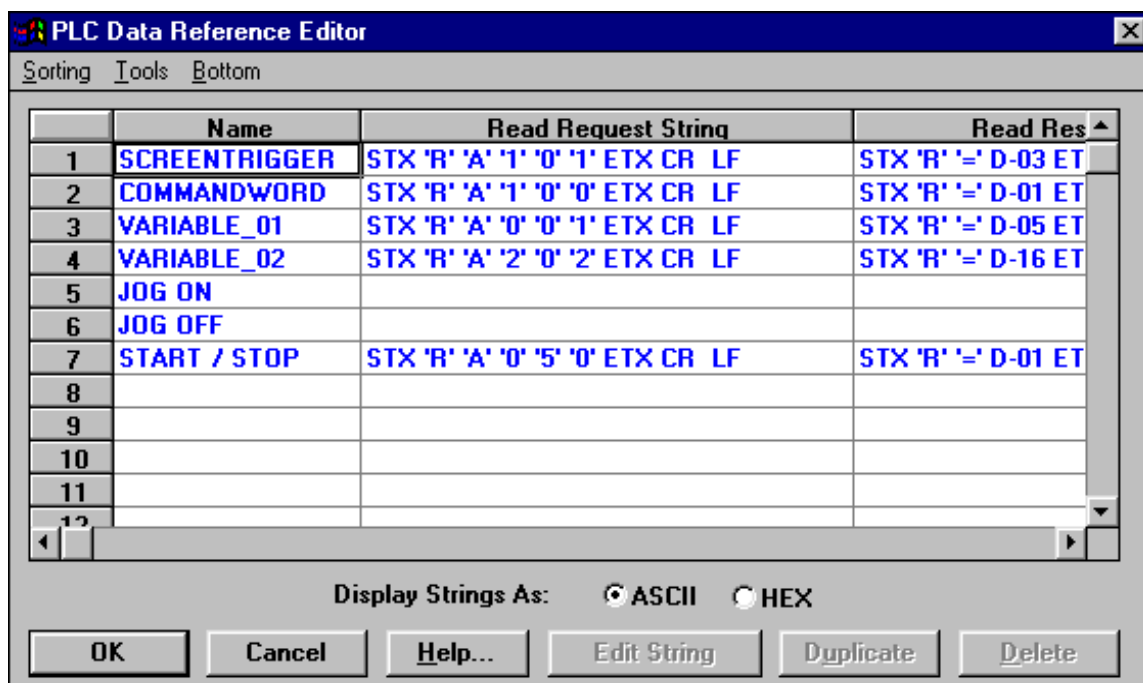
**Read Request String:** This defines a string the 2800 will send to the host device requesting data.

**Read Response String:** This defines the expected response from the host device to a 2800 read request. It specifies where the actual data is located within the returned string and how much data is included.

**Write Request String:** This defines the string the 2800 will send to the host device when a fields data is modified or activated. It is also used with function keys.

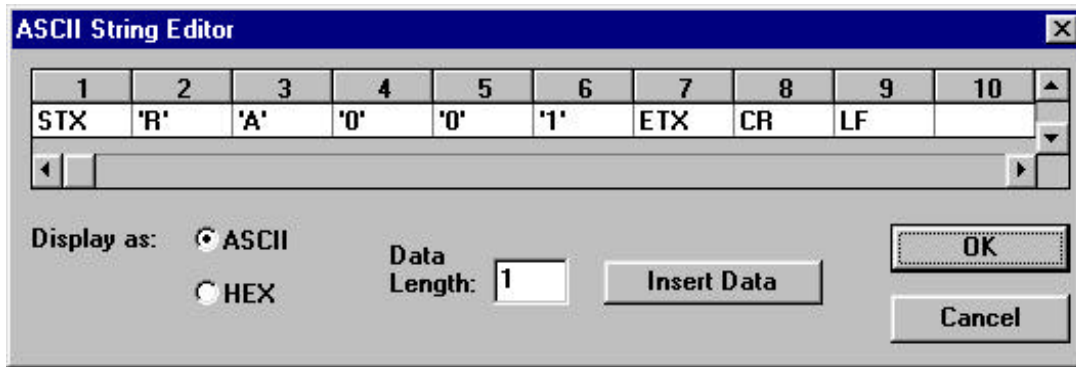
**Write Response String:** This defines the expected response from the host device to a 2800 write request. A Write Response string lets the 2800 know if its write was completed successfully. Although it is not required, it is a good practice to include the response.

This is what the tag name Data Reference Editor looks like.



### ASCII String Editor:

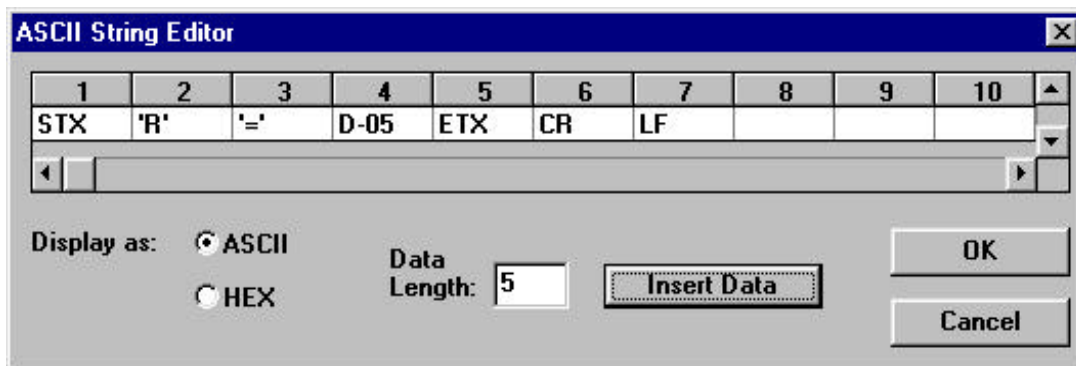
The ASCII String Editor is used to construct the individual strings that are associated with a tag name. The string Editor allows up to twelve characters to be specified. The graphic below shows the *Read Request* string for the tag named "Variable\_01" in the Data Reference Editor on the previous page. This string represents an arbitrary protocol developed to better explain string creation.



In this protocol, here are the definitions of the characters in the String Editor:

Character 1	STX	Start Of Text, marks the beginning of a packet.
Character 2	R	Indicates a Read function.
Character 3	A	Indicates an Address will follow.
Character 4, 5, & 6	001	Indicates the memory address in the host device.
Character 7	ETX	End Of Text, marks the end of packet data.
Character 8 & 9	CR LF	Carriage Return / Line Feed. Not required but makes it easy for a dumb terminal to monitor communications.

Above is the string the 2800 would send to the host device to get a value for "VARIABLE\_01". The 2800 must be able to interpret the response from the host device. The *Read Response* string provides the 2800 with the information it needs to interpret the response. Below is the *Read Response* string for "VARIABLE\_01".



Character 3	=	Indicates data will follow.
Character 4	D-05	This tells the 2800 that a maximum of five ASCII data characters will be inserted here by the host device. The maximum number of data characters allowed with the 2800 is sixteen. The host device may actually send less than the maximum number of characters specified by terminating with the "ESC" character (1Bhex). In this example, if the host sent "1" "2" "ESC" the 2800 would expect the next received character to be the "ETX" character.

## Section 2: Generic ASCII Driver - 2800 only

Lets suppose an operator enters a value of 12345 into a PLC Data field that uses "VARIABLE\_01" as its tag name. The 2800 would use the *Write Request* string associated with "VARIABLE\_01" to determine exactly what to send to the host device. Here is the *Write Request* string for "VARIABLE\_01".

The screenshot shows the 'ASCII String Editor' dialog box. It features a table with 10 columns and 1 row. The columns are numbered 1 through 10. The row contains the following characters: 'STX', 'W', 'A', '0', '0', '1', '=', 'D-05', 'ETX', and 'CR'. Below the table, there are radio buttons for 'Display as: ASCII' (selected) and 'HEX'. A 'Data Length' field contains the number '5'. There are 'Insert Data', 'OK', and 'Cancel' buttons.

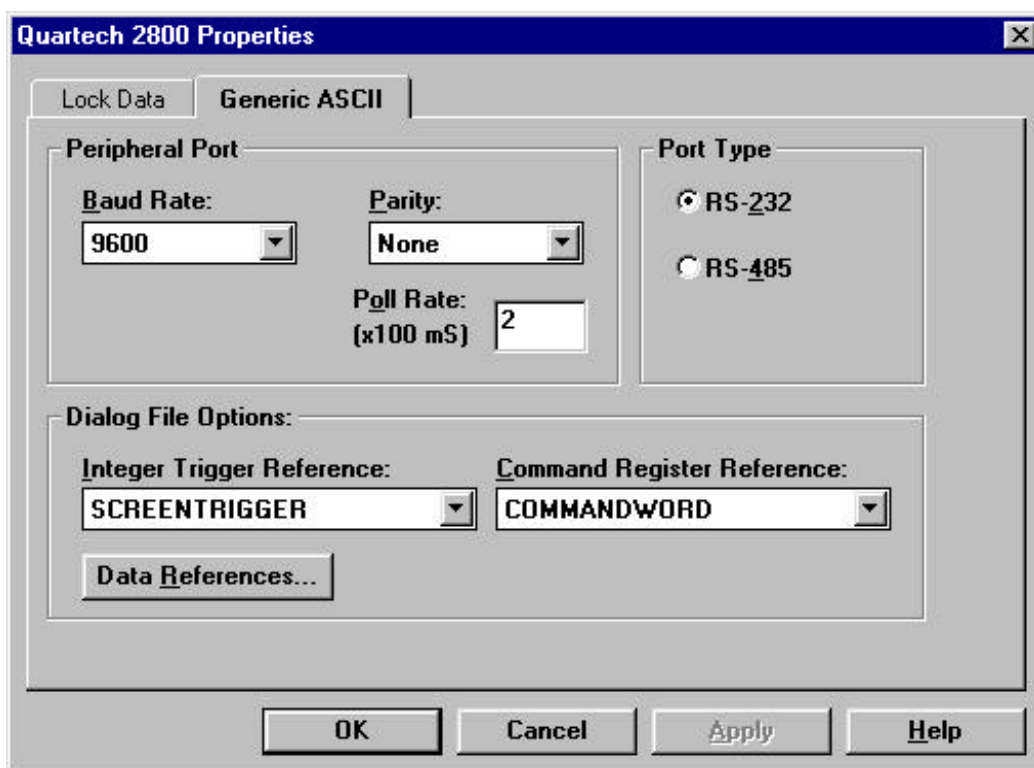
- Character 2        W        Indicates a Write function.
- Character 3        A        Indicates an Address will follow.
- Character 4, 5, & 6    001    Indicates the memory address in the host device.
- Character 7        =        Indicates data will follow.
- Character 8        D-05   Tells the 2800 to place a five character variable string here. In this case it will be the operator entered value 12345.

For this example the final action would be for the host device to indicate to the 2800 that the write was accepted. The *Write Response* string tells the 2800 what to expect. Although not a requirement, we have chosen for the host to respond with an acknowledge. In this protocol it would look like this:

The screenshot shows the 'ASCII String Editor' dialog box. It features a table with 10 columns and 1 row. The columns are numbered 1 through 10. The row contains the following characters: 'STX', 'ACK', 'ETX', 'CR', 'LF', and five empty cells. Below the table, there are radio buttons for 'Display as: ASCII' (selected) and 'HEX'. A 'Data Length' field contains the number '1'. There are 'Insert Data', 'OK', and 'Cancel' buttons.

### Setup Parameters:

ScreenMaker 2000 allows several parameters to be assigned within the Generic ASCII driver.



Baud Rate: 300, 600, 1200, 2400, 4800, 9600, or 19200 bps

Parity: Even, odd, or none

Byte format: Fixed at 8 data bits, 1 stop bit

Poll Rate: 1 - 255. Determines how often Screen Trigger and Command Word are read.

### Command Word Reference:

Assigning a tag name to the Command Register Reference that specifies a *Read Request* string and *Read Response* string, allows the host device to control the three lock bits in the Command Word.

To control all three locks a data value having at least three digits must be returned by the host device in response to a Command Word read request. The three least significant digits of the returned value control the locks. The least significant digit is considered digit one.

Digit 1 = Lock PLC variable fields      Digit 2 = Lock screen navigation      Digit 3 = Lock function keys

! When a digit value is zero, the lock will be disabled.

! When the digit value is anything other than zero the lock is enabled.

Example: If the returned value is: 0101. PLC variable fields and function keys would be locked.

If these locks are not required then set the Command Register Reference to "NONE".

### **Screen Trigger Reference:**

The ASCII driver allows only a single screen to be displayed at any particular time. The triggering source for that screen is determined by the presents of absents of a Screen Trigger Reference and it format.

Here are the possibilities:

- ! No Screen Trigger Reference assigned.
- ! Screen Trigger Reference assigned, only Write Request/Response strings specified.
- ! Screen Trigger Reference assigned, only Read Request/Response strings specified.
- ! Screen Trigger Reference assigned, both Read and Write Request/Response strings specified.

### **No Screen Trigger Reference assigned:**

In this condition the 2800 will utilize an internal screen trigger register. The number of the current screen being displayed is held in this register. Screen navigation is controlled solely by the operator through the Next/Prev keys or by function keys configured as screen triggers.

### **Screen Trigger Reference assigned, only Write Request/Response string specified:**

In this condition the 2800 will operate the same as with no trigger assigned, however, each time a new screen is triggered the screen number can be sent to the host device. For this to occur the Write Request string must include a variable data insert at least three digits in length.

### **Screen Trigger Reference assigned, only Read Request/Response string specified:**

In this condition both the internal trigger and host trigger can be used. On power up the 2800 will find the lowest screen number in its memory, write that number to the internal trigger and display that screen. When the host trigger is read the value found in it will take precedence over the internal trigger unless it is zero. As long as the value in the host trigger is zero, the internal trigger will maintain control. If the operator uses the Next/Prev keys or a function key to trigger a new screen, the screen number will be placed in the internal trigger.

If a value other than zero is placed in the host trigger then that screen number will immediately be displayed, taking precedence over the screen number in internal trigger. The number in the internal trigger will not be destroyed. If the host trigger returns to zero then the screen number in the internal trigger will reappear. This is useful for temporarily displaying fault messages without destroying the current display status.

Assume the host trigger forces a new screen to be displayed. If the operator uses the Next/Prev keys or a function key to trigger a new screen, the screen number will be placed in the internal trigger and that new screen will take the place of the screen triggered by the host trigger. In fact, the host trigger will be ignored until it changes to some other value. In simple terms, this makes the host trigger operate in a one-shot fashion.

### **Screen Trigger Reference assigned, both Read and Write Request/Response strings specified.**

In this condition only the host trigger is used. On power up the 2800 will find the lowest screen number in its memory, write that number to the host trigger and display that screen. If the operator uses the Next/Prev keys or a function key to trigger a new screen, the screen number will be written to the host trigger. The number in the host trigger is always the number of the screen being displayed regardless of how it got there.

### Function Keys:

The function keys differ from the normal PLC drivers in several ways. They can still be redefined on each screen, however the definition choices are fewer.

A function key can be used as a screen trigger and will operate the same as the **Next** and **Prev** keys.

For example, it may be convenient to label one function key "Main Menu" then have it assigned as a screen trigger on every screen so that the operator can easily return to the "Main Menu" at any time.

The screenshot shows the 'User Keys' configuration window. It has two tabs: 'Display' and 'User Keys'. Under 'User Keys', there is a list of function keys: F1, F2, F3, and F4. F1 is currently selected. Below this list is a 'Set All' button. To the right, the 'Usage Options' section contains three radio buttons: 'Message Switch', 'Screen Trigger' (which is selected), and 'Set As Default'. Below the 'Usage Options' is an 'Options' section with a sub-section 'Dialog File Trigger:'. This section contains a table:

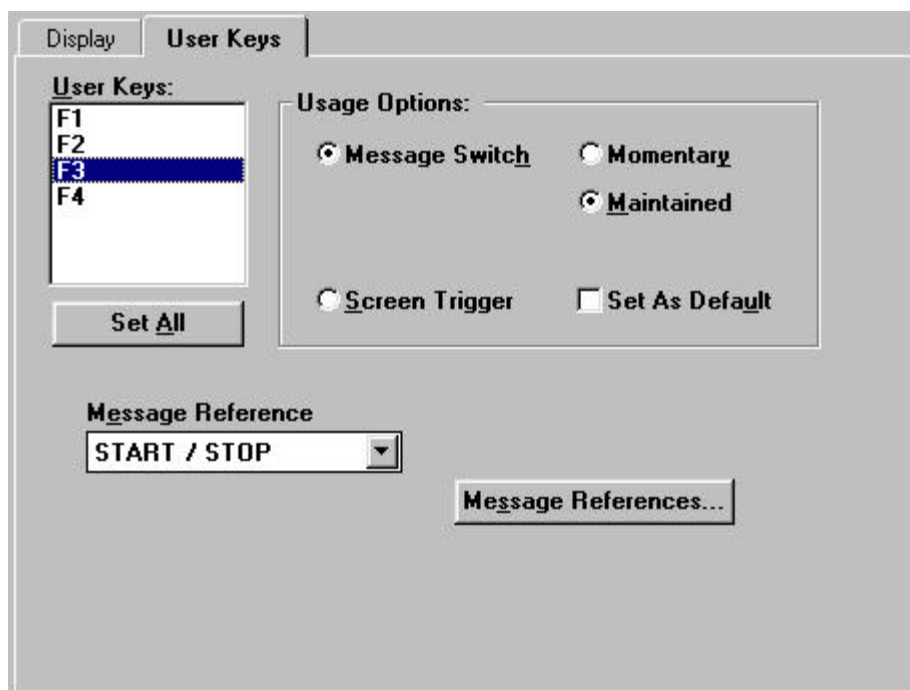
Trigger	Enabled	Screen Number
Int. Trigger 1	Yes	2

Below the table is a 'Browse...' button.

A function key can be used as a momentary pushbutton. In this configuration two tag references can be assigned, one for the contact make and one for the contact break. It is not necessary to assign both tag references. For example, a reference could be assigned for only the contact make. When the key is pressed the assigned string would be sent, but when the key is released no action would occur. In addition it is not necessary to have a *Read Request* or *Read Response* string since the momentary pushbutton will only write, never read.

The screenshot shows the 'User Keys' configuration window. It has two tabs: 'Display' and 'User Keys'. Under 'User Keys', there is a list of function keys: F1, F2, F3, and F4. F2 is currently selected. Below this list is a 'Set All' button. To the right, the 'Usage Options' section contains four radio buttons: 'Message Switch', 'Momentary' (which is selected), 'Maintained', and 'Screen Trigger'. Below the 'Usage Options' is a 'Set As Default' checkbox. Below the 'Usage Options' is a 'Make Reference:' dropdown menu with 'JOG ON' selected. Below that is a 'Break Reference:' dropdown menu with 'JOG OFF' selected. To the right of these dropdowns is a 'Message References...' button.

A function key can be used as a maintained pushbutton. In this configuration only one tag name reference is assigned but it must have a *Read Request*, *Read Response* and *Write Request* string specified. The *Write Response* string is optional. When the function key is pressed the read request will be executed. The least significant bit in the returned data will be complimented, inserted into the write request string and sent to the host device. When the key is released no action will occur.



For example, assume a memory byte in the host device holds an ASCII zero (30hex). When the function key is pressed the read request is executed and the ASCII zero is returned. The 2800 will compliment the least significant bit making the byte value an ASCII one (31hex). The ASCII one is then written back to the designated memory byte in the host device.

### **Field Notes:**

The PLC Data field allows numeric values to be displayed from and sent to the host device. Internally the 2800 uses double word, signed binary values. This allows values from -2,147,483,648 to +2,147,483,647 to be displayed and entered. Entry limit values within this range can be assigned to prevent an operator from entering an undesired value. The PLC Data field will allow a decimal point to be placed within the displayed value, however, the decimal point is not numerically relevant. A decimal point can not be passed between the host device and a PLC Data field.

If floating point values are required the PLC Text field can be used. The PLC Text field allows the operator to enter the numbers zero through nine, the minus sign, and the decimal point in any order. It is up to the host device to verify the validity of the entry and accept or reject it. Also the PLC Text field will allow strings up to sixteen characters to be displayed and entered.

The PLC Text field may also be used for displaying ASCII strings sent by the host device. All standard ASCII characters are valid (20hex - 7Ehex).

## Section 3: Slave ASCII Terminal Driver

The 2800 and 2900 Slave ASCII Terminal driver allows them to accept unsolicited ASCII data and control sequences. The operation is very similar to a dumb CRT terminal.

### Notable Features:

- ! Up to four pages can be open at any time. Complete cursor control is available.
- ! All keys except the number keys are software configured to send up to 12 characters
- ! Hardware interface is selectable. RS-232, RS-485 point-to-point, and RS-485 network.
- ! Control sequence codes are software definable

The OIT is configured using the ScreenMaker 2000 Configuration Editor for Microsoft Windows or NT. Support for the Slave ASCII Terminal driver begins with version 1.15. Both the configuration file and driver file are downloaded using the Configuration Editor. The Definable parameters are described in this document.

During power-up the OIT will display a number of screens that identify the product type, serial number, driver type with version level, and the communication setup parameters. The 2900 communication parameter screen looks like this:

The interface type, number of data bits, parity, and baud rate are displayed on the first line. The second line indicates if the keypad is enabled, if Clear-To-Send is used, and if the screen saver is enabled. A similar screen is displayed on the 2800.

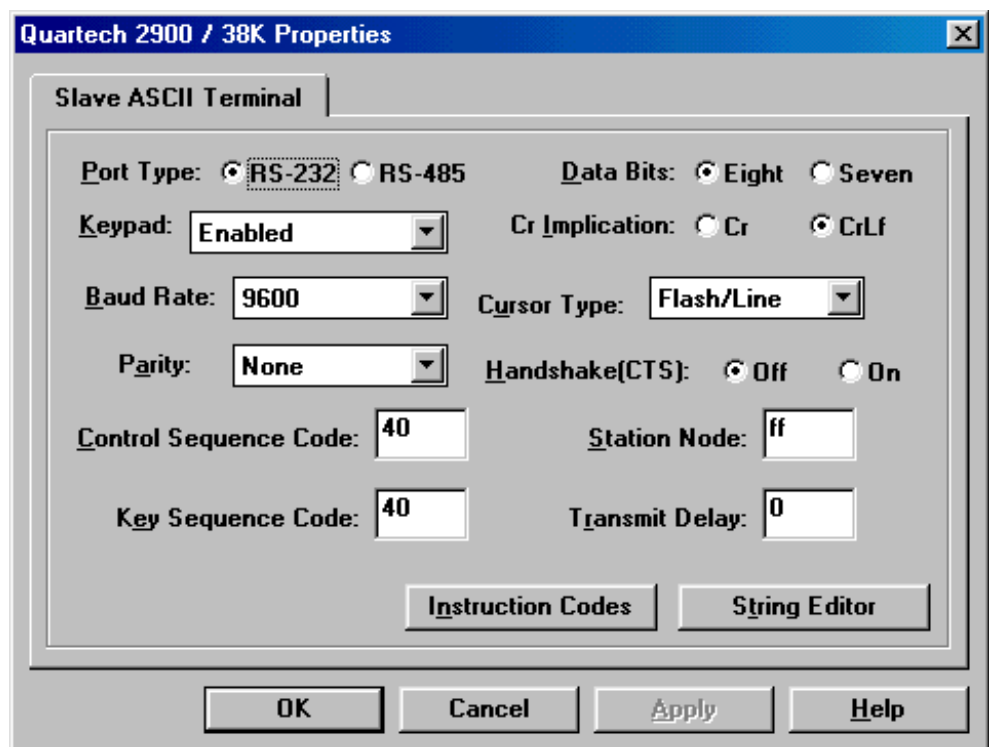
RS-232:8:N:9600  
KS:ON HS:OFF SS:OFF

### Communications:

The OIT must be configured to match the communication settings of the controlling device. The window shown here will allow most parameters to be set.

This window is selected from the editor menu bar via the Setup - OIT path.

This window also allows other operational parameters to be configured and provides the path to the instruction code and string editing windows.



- Port Type:** Select between RS-232 or RS-485. (RS-485 supports RS-422 devices)
- Baud Rate:** The communication speed (Baud) can be set for 300, 600, 1200, 2400, 4800, 9600, and 19200 bits per second.
- Data Bits:** Select between seven or eight data bits.
- Parity:** Select between odd, even, or none. The number of stop bits is always one.
- Transmit Delay:** This parameter allows a delay to be placed between data bytes that are transmitted to the master device. This delay is applicable to keypad transmissions and read responses. Each unit equals 25 milliseconds. Most devices will not require this delay so it may be set to zero.
- Station Node:** Multiple OITs may be wired to a single master device. To accomplish this the interface type must be set to RS-485 and each OIT must be assigned a unique station node address. Legal values are one to FEh (254). The address zero is reserved to disable all stations. The address FFh (255) is reserved to set the OIT for point-to-point operation.
- Handshake:** This function is only applicable when the interface type is RS-232. Setting this function On will prevent the OIT from transmitting data unless the Clear To Send (CTS) signal is on.
- Control Sequence Code:** This is always the first byte of a control sequence and may be set to any value from zero through 255. If a printable ASCII code is used (20h-7Eh) then that character may never be sent as a printable character since it will always be interpreted as the Control Sequence Code. It is also best to avoid using the predefined single byte cursor control characters: (08h, 09h, 0Ah, 0Bh, 0Dh, 7Fh).
- Key Sequence Code:** This code allows any of the definable keys to be used to clear the current page and set the cursor to zero. Any value from 01h through FFh may be used. The value zero will disable this function. When a programmable key is pressed the first string character is tested to see if it matches the Key Sequence Code. If a match occurs then the current page is cleared. No serial transmission will occur.
- CR Implication:** This flag determines how a Carriage Return (**CR**, 0Dh) is interpreted. If the flag is off then a received **CR** will move the cursor to the start of the current line. If the flag is on then the cursor will also be moved down one line (**CR LF**). The received **CR** will be ignored if the flag is on and the cursor is on the bottom line.
- Cursor Type:** The initial cursor type is selected using the configuration editor. The master device can change the cursor type at any time using the defined three byte control sequence. The following choices are available: Invisible, Invisible/Blinking, Line, Line/Blinking, Block, Block/Blinking
- Keypad:** The keypad may be completely disabled, completely enabled, or the number keys can be disabled while the definable keys are enabled. As discussed later in this document, the master device may change the power up setting.

### **Displaying Text:**

The OIT is always ready to accept ASCII data from the master device. All standard printable ASCII codes from 20 hexadecimal through 7E hexadecimal are allowed. When the OIT is first powered up the line or block cursor as selected will appear at the left most position of line one. If an invisible cursor is selected then the greater than symbol ">" will be flashing to indicate the unit is alive. The greater than symbol will only appear if the page is completely empty.

### Control Sequence:

A control sequence is a two or three byte instruction sent by the master device to the OIT. Instructions are used for such things as cursor control, page selection, and parameter setting. All control sequences start with a Control Sequence Code that is defined using the configuration editor window previously shown. All control sequences also have a second byte, the instruction byte, that is also definable. The third byte of a three byte control sequence is a data variable that must conform to requirements of the specific control sequence. The window shown here, which is accessed from the main properties window, is used to assign the a hexadecimal value for each of the instructions the OIT supports. The assigned code can be any value from zero to FFh, however, all codes must be a unique values. ScreenMaker 2000 assigns the following default values.

Blink On:	28	Set Cursor Type:	24	Move Cursor Down:	44
Blink Off:	29	Set Cursor Position:	3d	Move Cursor Home:	48
Clear Page:	53	Move Cursor Margin:	4d	Network Station:	23
Clear Line:	43	Move Cursor Back:	2d	Read Status:	56
BackSpace:	3c	Move Cursor Forward:	2b	Select Active Page:	50
Delete:	3e	Move Cursor Right:	52	Insert Mode:	49
Keypad On/Off:	4b	Move Cursor Left:	4c	Typeover Mode:	54
Transmit Delay:	57	Move Cursor Up:	55		

### Character Attributes:

An internal blinking attribute flag determines if the displayed ASCII data will flash or display steady. On power-up the flag is off, no blinking. If the flag is turned on then all ASCII data received from that point until the flag is turned off will flash. The blink flag is turned on or off using a separate two byte sequence.

### Cursor & Page Control:

The OIT allows the master device to completely control the cursor type and position. Received printable ASCII data is always written to the current cursor position, however, if the cursor is moved off screen or the screen is full then the received ASCII character is ignored.

#### Set Type:

The initial cursor type is selected using the primary properties window in the configuration editor. The master device can change the cursor type at any time using the defined three byte control sequence. The third sequence byte must be an ASCII or binary zero through five. Here are the cursor types: 0 = Invisible, 1 = Invisible/Blinking, 2 = Line, 3 = Line/Blinking, 4 = Block, 5 = Block/Blinking

**Set Position:** When the OIT is first powered the cursor is set to position zero which is the top, left position. The master device can set the cursor position at any time using the defined three byte control sequence. The third sequence byte is an eight bit binary value with a legal range from zero to 80, for the 2800, and zero to 40 for the 2900. The maximum value will move the cursor off screen. Assuming the default control codes are being used, the command to move the cursor to the start of the second display line would be: @ = ctrl T Equivalent hexadecimal: 40h 3Dh 14h Equivalent decimal: 64 61 20

**Display character positions: Unshaded are iteger notation, shaded are hexadecimal notation.**

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13
21	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39
14	15	16	17	18	19	1A	1B	1C	1D	1E	1F	20	21	22	23	24	25	26	27
40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59
28	29	2A	2B	2C	2D	2E	2F	30	31	32	33	34	35	36	37	38	39	3A	3B
60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
3C	3D	3E	3F	40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F

**Move Back:** This two byte sequence will decrement the cursor position by one unless the current position is at zero which is the first screen position.

**Move Forward:** This two byte sequence will increment the cursor position by one unless the current position is at the maximum screen position (off screen). The single ASCII **HT** code (09h) will also be accepted.

**Move Right:** This two byte sequence will increment the cursor position by one unless the current position is at the end of a line (19, 39, 59, 79).

**Move Left:** This two byte sequence will decrement the cursor position by one unless the current position is at the beginning of a line (0, 20, 40, 60).

**Move Up:** This two byte sequence will subtract the line width (20) from the cursor position unless the current position is less than 20. This will move the cursor up one line. The single ASCII **VT** code (0Bh) will also be accepted.

**Move Down:** This two byte sequence will add the line width (20) to the cursor position unless the current position is located within the bottom line. This will move the cursor down one line. The single ASCII **LF** code (0Ah) will also be accepted.

**Move Home:** This two byte sequence will set the cursor position to zero which is the first screen position.

**Move Margin:** This two byte sequence is the equivalent of a carriage return. The single ASCII **Cr** code (0Dh) will also be accepted. The interpretation of this instruction depends on the setting of the Cr Implication flag in the primary properties window. If **Cr** was chosen then the cursor will be moved to the start of the current line. If **CrLf** was chosen then the cursor will be moved to the start of the next line.

- Clear Page:** This two byte sequence will erase all data within the current page and set the cursor to zero.
- Clear Line:** This two byte sequence will erase all data on the line where the cursor currently sets and move the cursor to the left most position of that line.
- Backspace:** This two byte sequence will erase the character to the left of the current cursor position then shift the cursor and all data at and after the cursor to the left one position. The single ASCII **BS** code (08h) will also be accepted.
- Delete:** This two byte sequence will erase the character currently at the cursor position then shift all data after the cursor left one position. The single ASCII **DEL** code (7Fh) is also be accepted.
- Select Page:** The third byte of this three byte sequence specifies the target page that will receive the printable ASCII data and control instructions. Four pages are available, numbered zero through three. The third sequence byte may be a binary value or an ASCII value. Each page hold a cursor pointer, so switching to a new page will also load the cursor position for that page. The cursor type is not affected by a page change.
- Insert:** This two byte sequence will select data insert mode. When an ASCII printable character is accepted in insert mode all data at the cursor and after the cursor will be shifted right one position then the new character will be displayed at the cursor position.
- Typeover:** This two byte sequence will select data typeover mode. When an ASCII printable character is accepted in typeover mode it will be displayed at the current cursor position, then the cursor will be shifted right one position. The OIT will power-up in typeover mode.

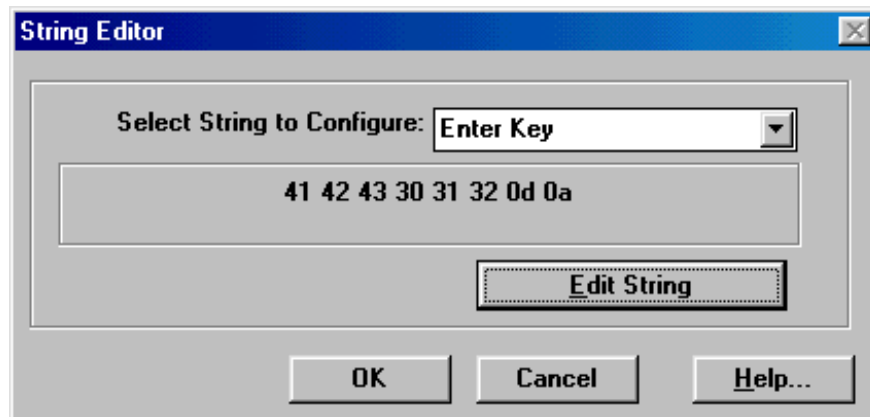
### **Read Status:**

The master device may request several parameters from the OIT through a three byte control sequence. The third byte of the sequence is an ASCII or binary value from zero through seven and determines what parameters will be returned. The response format is: Control Sequence Code, Instruction Code, Data.

- 0 = Read fail count** The data is a single binary byte that indicates the number of rejected bytes that have occurred since the last request. The maximum value is 128 of 255 depending on the number of data bits selected. The maximum value will never be exceeded but simply held at the maximum until a request resets the value.
- 1= Read page & cursor** The data consists of two binary bytes. The first is the current page number with a range from zero to three. The second is the cursor with a range from zero to the OIT maximum (40 or 80). The maximum value indicates the cursor is off screen.
- 2 = Last byte received** The data consists of a single ASCII byte. The data byte is equal to the last printable ASCII character received from the master device.
- 3, 4, 5** Not Assigned
- 6 = Firmware Version** The data consists of an ASCII string twelve characters long that includes the product identification and firmware version level. Example: .2800\_Ver1.00
- 7 = User Name** The data consists of an ASCII string twelve characters long. The string is defined using the String Editor. A zero within the string is sent as a space (20h).

### String Editor:

All keys except the numeric keys on the 2800 are defined using the string editor shown below. When a definable key is pressed the assigned string will be transmitted to the master device. On the 2800 the four function keys have a separate string for the make and break of the switch contact. On the 2900 the F1/YES and F2/No keys have a separate string for the make and break of the switch contact. The rest of the keys have only a contact make string.



Up to twelve characters may be included in a string. The characters may be any value from One to FF hexadecimal. If the first character is zero then the key is ignored. A zero is also used to terminate a string so that less than twelve characters can be transmitted.

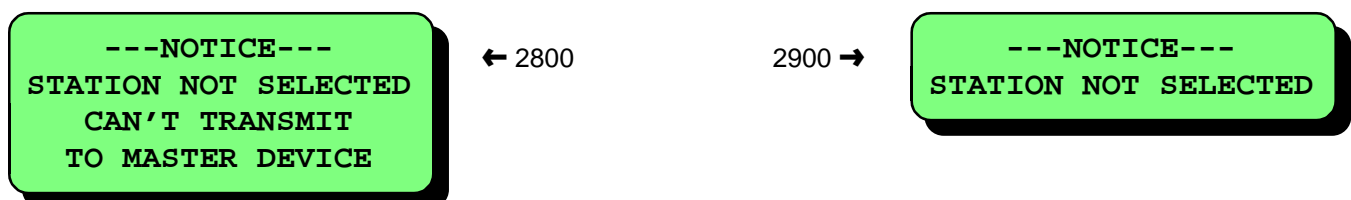
**Special case:** In the primary properties window the option was presented to set a value for the Key Sequence Code. When a key is pressed, if the first string character is nonzero then it is compared to the Key Sequence Code. If the bytes match then no serial transmission will occur and instead the key closure will be treated as a Clear Page command. The current page will be completely cleared and the cursor will be moved to position zero.

### Keypad Activation:

An internal control byte exists that determines if the keypad is enabled or disabled. A selection box is present in the primary properties window that allows the power-up state to be set. A three byte sequence allows the master device to change the control byte. The third byte may be an ASCII or binary zero, one, or two. A value of zero will disable the entire keypad. A value of one will enable the entire keypad. On the 2800 a value of two will enable only the definable keys, numeric keys will be disabled.

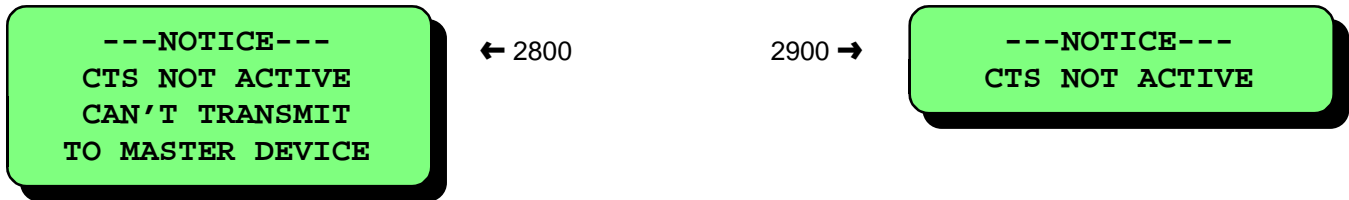
As discussed above, even when enabled, a definable key will do nothing unless an ASCII string has been created for it. When enabled, the numeric keys on the 2800 will always transmit their equivalent ASCII single byte value, 30h through 39h.

If the OIT is configured for network operation and a key is pressed while the station is not enabled then the following warning message will be displayed. The key will be ignored.



### CTS Handshaking:

A selection box is present in the primary properties window that allows the hardware handshaking to be enabled or disabled. If enabled the OIT will not transmit keypad data unless its CTS signal is detected in the on state. If the keypad data can not be transmitted the warning message shown here will be displayed for about two seconds. Handshaking is only applicable when the interface type is set for RS-232.



### Power-up String:

The 2800 and 2900 can be configured to transmit a string of up to twelve characters immediately after power up. The string is programmed using the string editor and may use any characters from one to FF hexadecimal. As with key strings, If the first character is zero then no transmission will occur. A zero is also used to terminate a string so that less than twelve characters can be transmitted. If Clear-To-Send handshaking is enabled, the string will not be sent unless the CTS signal is on. If networking is enabled the string will be ignored.

### Networking Multiple OITs:

Multiple OITs can be controlled by a single master device if the interface type is set as RS-485 and unique node addresses are assigned to each OIT. The RS-485 receiver pair (RxdA/RxdB) of each OIT must be wired together and connected to the master devices transmit pair (TxdA/TxdB). If the keypad on the OIT is to be used or the master device needs to execute a read command then the transmit pair of the OIT must be connected to the receiver pair of the master device. The RS-485 specification allows up to thirty two devices to be connected at a maximum distance of four thousand feet without repeaters.

**Note:** Some devices use the terminology Tx+/Tx-, Rx+/Rx-. Generally (A & -) and (B & +) are synonymous.

When the OIT is first powered the station node address is checked. If the address is not 255 (FFh) then the OIT will begin watching for a station enable control sequence. No other data or control sequence will be recognized until the station is enabled. Once a station is enabled, it will remain enabled until a different station is enabled or the global disable address zero is received.

### Transmission Line Termination:

The RS-485 network is required to be wired in a point-to-point fashion. All devices are wired in-line, one after another, forming a single trunk line. Drop lines off the main trunk line are not recommended and may degrade or prevent network operation. As the trunk line length increases two factors become increasingly important. These factors are line resistance and line capacitance. In most applications the use of line terminating resistors will improve or allow satisfactory performance. Two terminating resistors are required for a network regardless of the number of devices in the network. One terminating resistor is connected at each end of the network. The 2800 and 2900 includes a terminating resistor that is connected into to the receive via a DIP switch. If the OIT is at either end of the network the termination should be enabled. Devices which are not at either physical end of the network must not have termination resistors connected into the circuit. If your application requires more than one OIT to display identical information and the keypad is not used then you can connect the units without using the network commands. Simply connect the transmitter of the master device to the receiver of all the OITs. The RS-232 interface will work if only two or three OITs are required and the distance is less than fifty feet. The node address for all OITs must be FFh so they are always enabled.

The ASCII drivers allows the physical connection between the OIT and host device to be either RS-232 or RS-485. The pin assignments are shown here:

## 2800 & 2900 Series Communication Port 15 Pin Male D-Type

1	>))	N/A	
2	>))	TXD, RS-232 Transmit Data	(Output)
3	>))	RXD, RS-232 Receive Data	(Input)
4	>))	RTS, RS-232 Request To Send	(Output)
5	>))	CTS, RS-232 Clear To Send	(Input)
6	>))	TXDA, RS-485 Transmit Data "A"	(Output)
7	>))	SC, RS-232/485 Signal Common	
8	>))	N/A	
9	>))	N/A	
10	>))	N/A	
11	>))	N/A	
12	>))	RXDB, RS-485 Receive Data "B"	(Input)
13	>))	RXDA, RS-485 Receive Data "A"	(Input)
14	>))	TXDB, RS-485 Transmit Data "B"	(Output)
15	>))	N/A	

**The pins marked N/A must remain disconnected.**

The Generic ASCII driver running on the 2800 requires the Clear To Send signal to be active for normal operation. If it is not at the correct signal level then the "COMMUNICATION CABLE DISCONNECTED" fault will be displayed. Placing a jumper wire between the Request To Send and Clear To Send pins will allow normal operation.

Decimal Hex Character

Decimal Hex Character

Decimal Hex Character

# Appendix B: Standard ASCII Code Table

0	00	ctrl @	NUL	43	2B	+	86	56	V
1	01	ctrl A	SOH	44	2C	,	87	57	W
2	02	ctrl B	STX	45	2D	-	88	58	X
3	03	ctrl C	ETX	46	2E	.	89	59	Y
4	04	ctrl D	EOT	47	2F	/	90	5A	Z
5	05	ctrl E	ENQ	48	30	0	91	5B	[
6	06	ctrl F	ACK	49	31	1	92	5C	\
7	07	ctrl G	BEL	50	32	2	93	5D	]
8	08	ctrl H	BS	51	33	3	94	5E	^
9	09	ctrl I	HT	52	34	4	95	5F	_
10	0A	ctrl J	LF	53	35	5	96	60	'
11	0B	ctrl K	VT	54	36	6	97	61	a
12	0C	ctrl L	FF	55	37	7	98	62	b
13	0D	ctrl M	CR	56	38	8	99	63	c
14	0E	ctrl N	SO	57	39	9	100	64	d
15	0F	ctrl O	SI	58	3A	:	101	65	e
16	10	ctrl P	DLE	59	3B	;	102	66	f
17	11	ctrl Q	DC1	60	3C	<	103	67	g
18	12	ctrl R	DC2	61	3D	0	104	68	h
19	13	ctrl S	DC3	62	3E	>	105	69	i
20	14	ctrl T	DC4	63	3F	?	106	6A	j
21	15	ctrl U	NAK	64	40	@	107	6B	k
22	16	ctrl V	SYN	65	41	A	108	6C	l
23	17	ctrl W	ETB	66	42	B	109	6D	m
24	18	ctrl X	CAN	67	43	C	110	6E	n
25	19	ctrl Y	EM	68	44	D	111	6F	o
26	1A	ctrl Z	SUB	69	45	E	112	70	p
27	1B	ctrl [	ESC	70	46	F	113	71	q
28	1C	ctrl \	FS	71	47	G	114	72	r
29	1D	ctrl ]	GS	72	48	H	115	73	s
30	1E	ctrl ^	RS	73	49	I	116	74	t
31	1F	ctrl _	US	74	4A	J	117	75	u
32	20	Space		75	4B	K	118	76	v
33	21	!		76	4C	L	119	77	w
34	22	"		77	4D	M	120	78	x
35	23	#		78	4E	N	121	79	y
36	24	\$		79	4F	O	122	7A	z
37	25	%		80	50	P	123	7B	{
38	26	&		81	51	Q	124	7C	
39	27	`		82	52	R	125	7D	}
40	28	(		83	53	S	126	7E	→
41	29	)		84	54	T	127	7F	DEL
42	2A	*		85	55	U			