CRESTRON

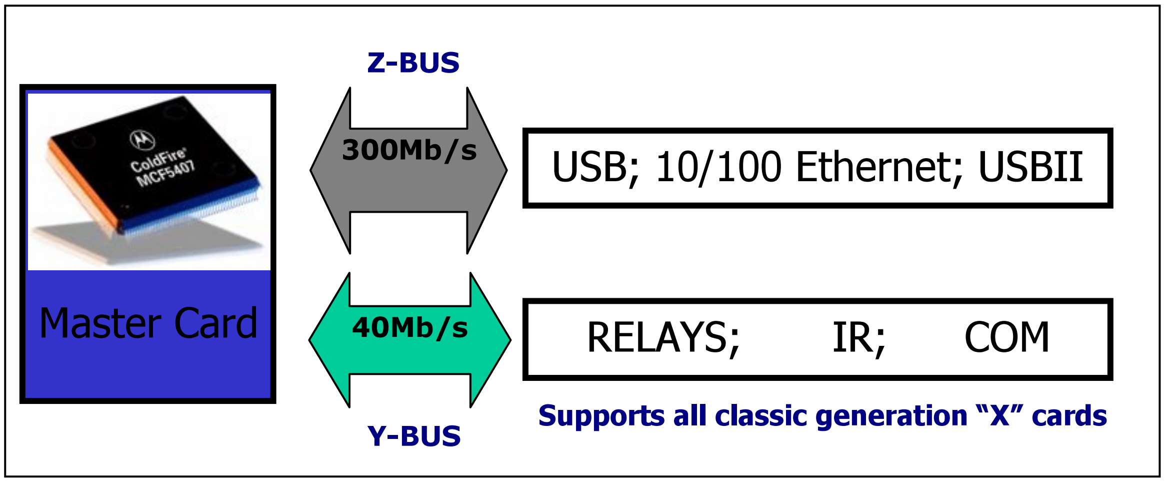
INTERMEDIATE TRAINING

crestron TECHNICAL INSTITUTE

DAY 1

1. 2-SERIES PROCESSOR

- ***Design***


Z-BUS
300Mb/s
USB; 10/100 Ethernet; USBII
Master Card
40Mb/s
RELAYS; IR; COM
Y-BUS
Supports all classic generation "X" cards

This is the latest generation of Crestron processors capable of multithreading and uses a Motorola ColdFire® Processor. This processor is now utilizing 32bit architecture enabling complex mathematics and string handling to be carried out at a significantly faster speed. The processor is rated at 257 MIPs.

The processor uses a CUZ for its firmware which sits over an upgradeable monitor Rom.

The processor is broken into two busses designed to provide a 300mbs bus for high speed communications to Ethernet based systems and a 40mbs bus for low speed communication over RS232 IR and relay control.

The processor can handle 1000 digital signals changing status at the same wave. (Wave will be explained later on). For analogue this value is 256 and for serial signals this value is 127.

Our processor is event driven, so the processor only actions whenever there is a change of status of a specific signal and will only process this signal and the related signals in the program. So in rest (no button presses on the TP), the communication between the TP and the processor is shut down.

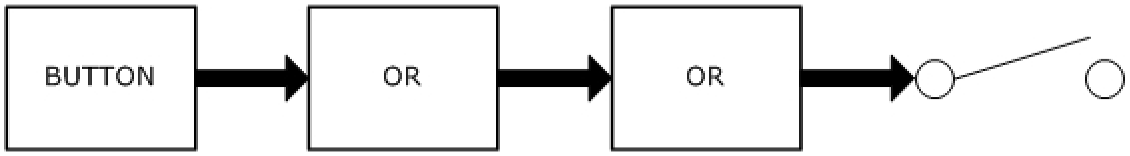
- ***2-Series processing speed***

1/100 = 1 TICK and is equal to 10 ms.

Every TICK can be divided into 255 parts or WAVES. (So every wave = 0.039 mS (10ms / 255))

In 1 wave, the processor is able to shift a digital signal trough a symbol. (So 1 wave is 1 action…)

Example: In how much time before the contact closes from the moment the button is pressed?



Solution: Time = 3 X 1 wave = 3 X 39 µS = 127 µS

The processor can do this for a maximum of 1000 digital, 256 analogue and 127 serial signals per wave.

Processor	**Processing speed (MIPS)**
QM-RMC	63
QM-RMCRX-BA	63
AES	66
AMS	66
AMS-AIP	66
PAC2M	66
MPS	66
AADS	100
PAC2	257
PRO2	257
AV2	257
CP2	257
MP2	257
MC2	257
RACK2	257

- ***Numeric Formats***

Numeric values can be expressed in a number of formats, where the character in parentheses represents the format identifier:

(d) Decimal
(h) Hexadecimal
(%) Percentage
(s) Seconds
(t) Ticks
(= 1/112.5 seconds = 8.8ms for Xgeneration Processors)
(= 1/100 seconds = 10ms for 2 Series Processors)
(' Character') (Single byte)

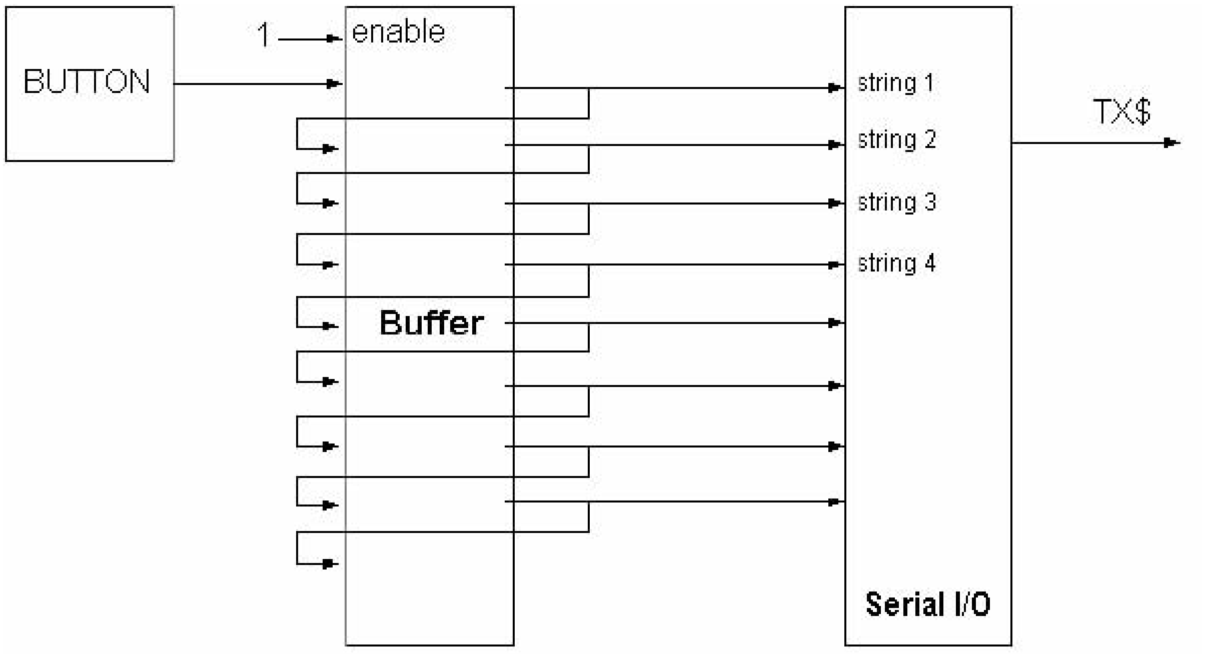
The allowable range of analogue values expressed in each format is as follows:

Format	Minimum	Maximum
Decimal	0d	65535d
Hexadecimal	0h	FFFFh
Percentage*	0%	100%
Seconds**	0s	582.53s
Ticks	0t	65535t
Byte	' ' (space, ASCII 20h)	'~' (tilde, ASCII 7Eh)

*Percentage and seconds formats can be expressed with precision of .01% or .01s.
**Double precision time values range from 0.0 seconds to 19,088,743 seconds.

- ***Buffer Power***

Suppose we need to send 100 serial commands to a broadcast switcher via serial control as fast as possible. So how can we send as many strings possible as fast as possible?

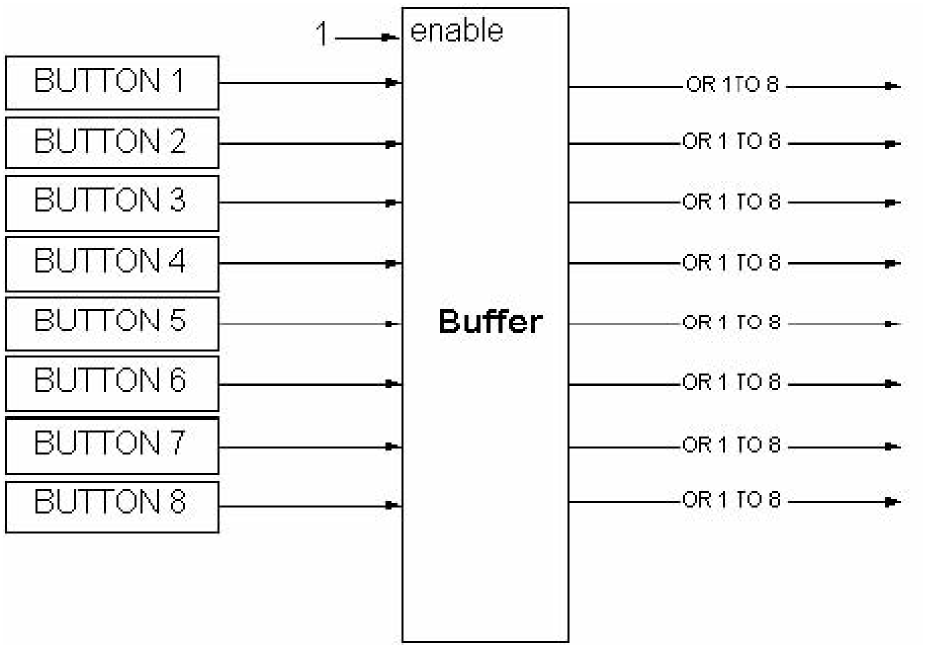


When using the oscillator symbol, the minimum value we can fill in is 1t (1 tick).

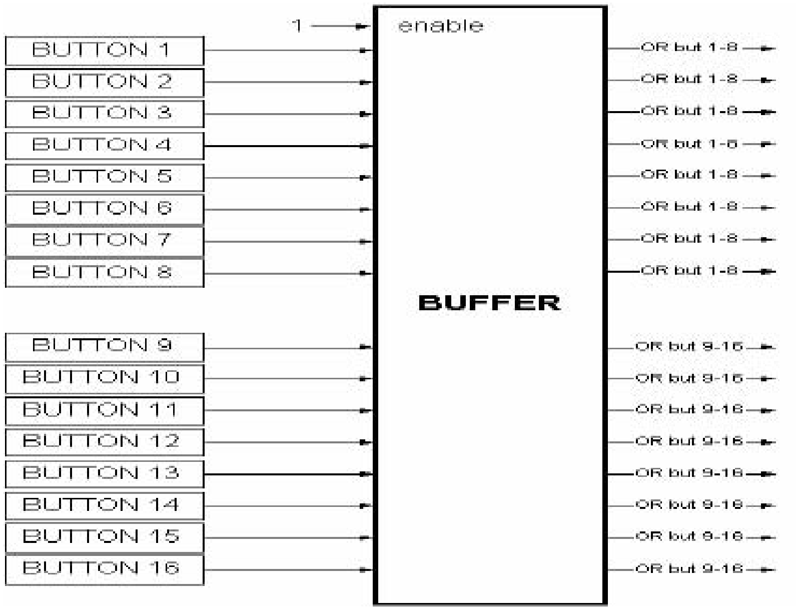
This means an oscillating speed of 2 X 8.88ms or 17.6 ms. For 100 strings this would be 1.76 seconds what's to slow. So the above solution is the fastest possible.

A buffer can be used as an oscillator, can be used to slow down a signal for a few waves and can also be used as an OR gate.

Example:



In the above example the BUFFER functions as an OR gate because the signal name for all the outputs are the same. We can even take all OR gates out of the total program and put them in 1 BUFFER that function as a multiple OR gate:



2. SIMPL WINDOWS SHORTCUT KEYS

- ***Display manipulation***

ALT + B	Hide / View both Symbol Library and Program View
ALT + P	Hide/ View Program View
ALT + Y	Hide/ View Symbol Library

- ***Signal manipulation***

ALT + 6	Add signal suffix from prompt
ALT + 1 to 5	Add signal suffix that were set in Preferences
ALT + / -	Add or delete 1 input or output to a symbol
ALT + SHIFT + PLUS	Add multiple inputs or outputs to a symbol
SHIFT + F4	Increment Signal name by First Numeric value
F4	Increment Signal name by last Numeric value
F6	Copy selected signal from one side to the other
F9	Search and replace

- ***General edit***

CTRL + C	Copy
CTRL + X	Cut
CTRL + V	Paste

3. Exercise: Analog scaling

New used symbols

- ***Analog Scaler [ascale]***

The Analog Scaler symbol scales the range of values for its analog input signal to the range defined by the <**span**> and <**offset**> parameters, using the following formula:

<**aout**> = (<**ain**> * <**span**> / <**divisor**>) + <**offset**>

Where <**span**> represents the scale factor, and <**offset**> the minimum value. The default divisor is 65535d unless defined.

- ***Analog Scaler without Zero Pass [ascale0]***

The Analog Scaler without Zero Pass symbol operates identically to the Analog Scaler symbol, except for the behavior when the input is set to 0d.

On the Analog Scaler, when <ain> = 0d, this results in <aout> also immediately being set to 0d (a "zero pass"), whereas in the Analog Scaler without Zero Pass the output does not go to 0d, it is set equal to the offset.

- ***Analog Compare - Full Set [acomp2]***

The Analog Comparison symbol compares the values of its two inputs and drives its outputs high based on the following traditional conditions:

Value 1 is <, <=, >=, >, =, != Value 2 the corresponding output will go high until another match is found.

4. Exercise: Analog RAM

New used symbols

- ***Analog RAM [ram]***

The Analog RAM symbol stores the current values of all its analog inputs in NVRAM (non-volatile RAM) whenever the <**store**> input goes high while any one or more <**select**> inputs are high.

The stored values are recalled:

1. Whenever a <**select**> input goes high while <**recall**> is high;

   or

2. Whenever <**recall**> goes high while a <**select**> input is high and <**store**> is low.

Every analog input has a corresponding analog output, and each input/output pair is independent of other input/output pairs. (In normal applications, each input/output pair will have the same signal name.)

5. Exercise: IR control

New used symbols

- ***Numeric Keypad [#pad]***

The Numeric Keypad symbol generates on its output the analog value of its digital inputs, keeping the value within the range given by the <**upper limit**> and optional <**lower limit**> parameters.

The <**0**> through <**9**> keypad inputs set the output value and are cumulative. That is, a rising edge of <**2**> sets the output to 2, a subsequent rising edge of <**5**> sets the output to 25, and so on. The output value also increments by 1 with each rising edge of <**+**>, and decrements by 1 with each rising edge of <**-**>.

- ***Decade [decade]***

The Decade symbol converts its analog input into groups of digital outputs representing the decimal value of each digit. Each group acts as a one-of-ten decoder. For example, given an input of 125, the following outputs would go high: <**hundreds1**>, <**tens2**> and <**units5**>.

The optional <**enable**> input enables the symbol when high and sets all outputs to 0 when low. In applications where the value of the analog input might change continually, as from an Analog Ramp symbol, <**enable**> can be pulsed in order to produce the desired output.

DAY 2

1. RS232 / 422 / 485

- ***What is RS232?***

It is a standard for serial interfacing (RS = Recommended Standard) that has been approved by the EIA (Electronics Industry Association) for inter-connecting serial devices. It dates from 1962 and is originally developed to achieve serial communication between peripherals on a network (such as printers, terminals,…).
It is unbalanced and uses a 9 or 25 pin connector.

There are 3 possible RS232 interconnections:
SIMPLEX: Only two conductors are used, TX and GND
HALF-DUPLEX: Bi-directional communication between the devices, but only one device can transmit at a time. Often Handshaking is used: RTS and CTS lines. So minimum of 3 conductors: TX, RX and GND this one is most often used in the AV industry.
FULL DUPLEX: Bi-directional communication between the devices, both the units can transmit at the same time (less used in AV industry….)

- 3 conductors: TX, RX, GND (explain this by showing a DB9 connector).
- Data is send between GND and TX, data is received between GND and RX.
- Distance between CPU and node is +/- 20 meters (60 feet).
- Databits: information is send in packages of 8 or 7 bits
- Startbit and stopbit's (normally 1 startbit and 1 or 2 stopbits).
- Transmission speed between CPU and node is Baudrate (bits/seconds).
  With RS-232 the maximum baudrate is 19200 bps.
- Parity control (datacheck in the package) Odd, even or None.
- Flow control also know as handshaking.

- ***Protocol and serial settings***

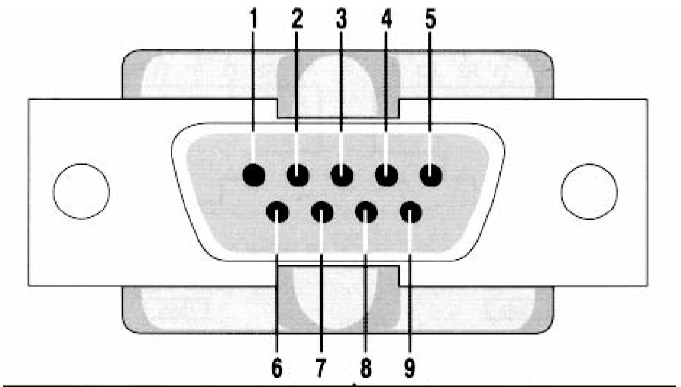
In serial communication, bytes are disassembled into series of bits, to be transmitted from one device to another over a single wire. Often the information is sent in packages of seven or eight **data bits**. To let the receiving device know when the package starts and stops, one **start bit** and one or two **stop bits** are added.

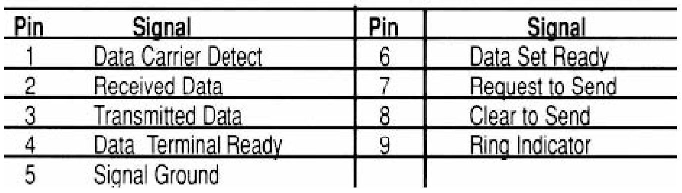
The most important variable in the asynchronous protocol is the transmission speed, called the **baud rate**, expressed in bits per second (bps). The maximum **baud rate** for RS232 is 115200 bps.
Two more variable settings are: **Parity** and **Flow Control** or **Handshaking**.

Parity can be Odd, even or None and determines whether to add a bit to each set of data bits used to validate the data package for transmission errors. **Flow Control** or **Handshaking** can be (Xon/ Xoff) Software Handshaking, (RTS/ CTS) Hardware handshaking, or none.
Handshaking is used by the receiver to tell the sender to stop transmitting as it is not ready to accept more data at that time. If done via hardware, two extra conductors are used, the RTS (Request to Send) and CTS (Clear to Send), if done via software, the code Xon and Xoff are sent. Most often used is NONE.

Typical serial settings: 9600 Bps, 8 data bits, 1 stop bit and no parity.

Pin assignment:





Pin	Signal	Pin	Signal
1	Data Carrier Detect	6	Data Set Ready
2	Received Data	7	Request to Send
3	Transmitted Data	8	Clear to Send
4	Data Terminal Ready	9	Ring Indicator
5	Signal Ground		

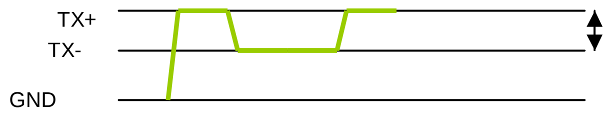
Pin 2 (RX), Pin 3 (TX) and pin 5 (GND) are always used, Pin 7 (RTS) and pin 8 (CTS) are optional dependant on what device you are talking to. A good source of this information is the Crestron Cable database.

- ***RS422***

The limitation of RS232 is the maximum distance between the two devices of **50 feet or 15 meters at a baud rate of 19200 bps**. The higher the baud rate, the shorter the distance that can be covered. The RS232 standard is an unbalanced way of sending information, thus very sensitive for noise. To be able to cover longer distances, the RS422 standard was developed, which is a balanced way of sending information and thus more noise tolerant. As a result of this each transmit and receive line uses two conductors (+TX, -TX, +RX, -RX).
The maximum distance between the devices is now **4000 feet or 1200 meters at a baud rate of 115200 bps**

Beside the difference in cable length the second big difference is the maximum baud rate that can go up to 40 Megabits per second in RS422.

- o Data is transported in a balanced form this makes it possible to increase the distance between CPU and node up to 1200 meters (4000 feet).
- o 5 conductors TX+ TX- RX+ RX- GND



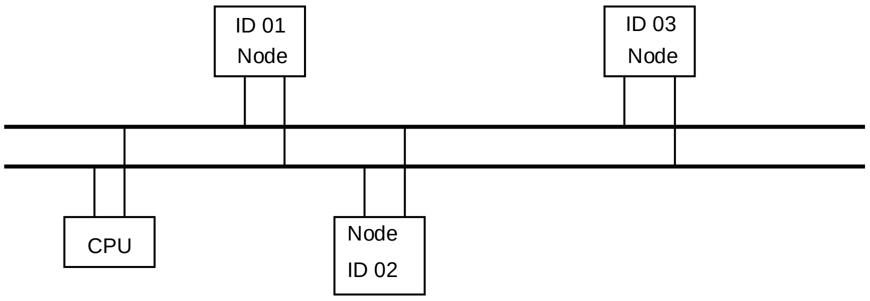
ΔTX: This is the value that will be send between the CPU and the node.
*same counts for the Receive

Baudrate is possible up to 115200 bit per second

- ***RS485***

This is a variation of the RS422 protocol, allowing multiple slave devices (up to 64) to be addressed by 1 master device. The master sends an address within the command to that makes it possible to control a specific slave. CRESTRON has it's own protocol based on the RS485 protocol, allowing the processor to communicate with a maximum of 252 devices on the CRESNET port.

This is a varation of the RS-422 protocol, only now 2 wires will handle all the datatransmition between CPU and nodes.



- Correct, our Cresnet is actually based on a RS-485 network.
- Distance +/- 5000 feet (+/- 1,2 km)
- Baudrate 115200 bps
- Instead of seperate serial cables coming from a CPU to each node,
- you can now daisy chain and give each node a separate ID.

- ***Pin out assignment CRESTRON COM PORTS:***

The pin-outs are as follows for Bi-Directional **RS232**:

PIN	DIRECTION	DESCRIPTION
1	No Connection	
2	To Crestron	(RXD) RS-232 Receive Data
3	From Crestron	(TXD) RS-233 Transmit Data
4	No Connection	
5	Ground	RS-232 Signal Common
6	No Connection	
7	From Crestron	* (RTS) RS-232 Request to Send, Only use if selected in SimplWindows program.
8	To Crestron	* (CTS) RS-232 Clear to Send, Only use if selected in SimplWindows program.
9	No Connection	

The pin-outs are as follows for Bi-Directional **RS422**:

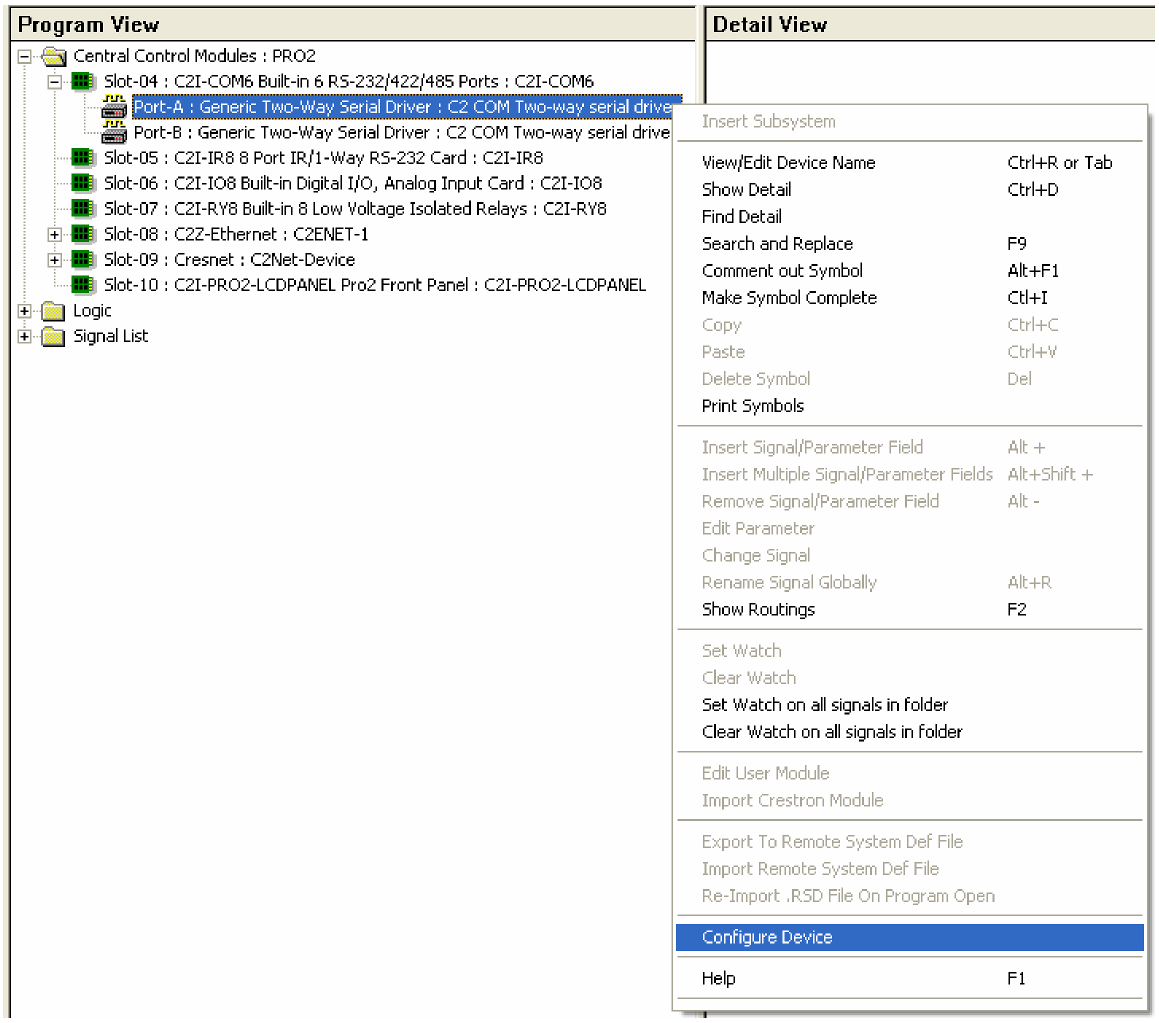
PIN	**DIRECTION**	**DESCRIPTION**
1	To Crestron	(RXD-) RS-422 Receive Data (Idles low)
2		
3		
4	From Crestron	(TXD+) RS-422 Transmit Data (Idles high)
5	Ground	RS-422 Signal Common
6	To Crestron	(RXD+) RS-422 Receive Data (Idles high)
7		
8		
9	From Crestron	(TXD-) RS-422 Transmit Data (Idles low)

*= RS-422 transmit and receive are balanced signals requiring two lines plus a ground in each direction. RXD+ and TXD+ should idle high (going low at start of data transmission). RXD and TXD- should idle low (going high at start of data transmission). If necessary, RXD+/RXD-and TXD+/TXD- may be swapped to maintain correct signal levels.

The pin-outs are as follows for Bi-Directional RS485. NOTE, not all com ports support support RS485, review the manual which is also available on-line as well as SimplWindows for specifics:

Com (DB9) Connector	**DIRECTION**	**DESCRIPTION**
Tie Pins 1 & 9 together	BiDirectional	(RS485 -) RS-485
Tie Pins 4 & 6 together	BiDirectional	(RS485 +) RS-485
Pin 5	Ground	(RS485 G) RS-485

- **Serial settings of the Crestron COM port:**


Program View
Central Control Modules : PRO2
Slot-04 : C2I-COM6 Built-in 6 RS-232/422/485 Ports : C2I-COM6
Port-A : Generic Two-Way Serial Driver : C2 COM Two-way serial drive
Port-B : Generic Two-Way Serial Driver : C2 COM Two-way serial drive
Slot-05 : C2I-IR8 8 Port IR/1-Way RS-232 Card : C2I-IR8
Slot-06 : C2I-IO8 Built-in Digital I/O, Analog Input Card : C2I-IO8
Slot-07 : C2I-RY8 Built-in 8 Low Voltage Isolated Relays : C2I-RY8
Slot-08 : C2Z-Ethernet : C2ENET-1
Slot-09 : Cresnet : C2Net-Device
Slot-10 : C2I-PRO2-LCDPANEL Pro2 Front Panel : C2I-PRO2-LCDPANEL
Logic
Signal List
Detail View
Insert Subsystem
View/Edit Device Name Ctrl+R or Tab
Show Detail Ctrl+D
Find Detail
Search and Replace F9
Comment out Symbol Alt+F1
Make Symbol Complete Ctl+I
Copy Ctrl+C
Paste Ctrl+V
Delete Symbol Del
Print Symbols
Insert Signal/Parameter Field Alt +
Insert Multiple Signal/Parameter Fields Alt+Shift +
Remove Signal/Parameter Field Alt -
Edit Parameter
Change Signal
Rename Signal Globally Alt+R
Show Routings F2
Set Watch
Clear Watch
Set Watch on all signals in folder
Clear Watch on all signals in folder
Edit User Module
Import Crestron Module
Export To Remote System Def File
Import Remote System Def File
Re-Import .RSD File On Program Open
Configure Device
Help F1

- ***ASCII codes***

The communication between a keyboard and the CPU is also via codes, but since you have more than 2 (binary) or 16 (hexa) characters on a keyboard they had to create something like the ASCII format. Every key on the keyboard will actually generate a 8 bit (1byte) code. Every Byte is representing a certain character.

Show with the help of the conversion table witch ASCII codes that you have.

Example: 182 decimal is Ā in ASCII

(See appendix for the ASCII table)

2. Exercise: Creating a serial signal

New used symbols

- **Analog to Serial [txa]**

The Analog to Serial symbol constructs a string from the **low bytes** or **high bytes** of its analog inputs, or from a combination of analog inputs and string constants, and then transmits the string with each rising edge of <**trig**>. In this way the symbol can update the serial data during run time.

The <**Format**> parameter determines whether data is taken from the low bytes or high bytes of the analog inputs. It also specifies what special formatting, if any, is to be applied to the string before it is transmitted.

- **Analog/Serial one shot [smv, s1shot]**

The Serial/Analog One-Shot symbol drives its output signal high with each transition of the input. The output goes high for the period specified by <**pulse_time**>, and low when <**pulse_time**> expires.

The Serial/Analog One-Shot symbol is retriggerable, meaning that it will recognize any changes in the input, even while <**out**> is high, causing it to start the count all over again. The output will not go low until the full <**pulse_time**> has elapsed without interruption.

3. Exercise: Creating a variable serial signal

New used symbols

- **Serial to Analog [rxa]**

The Serial to Analog symbol evaluates its serial input string until it finds an exact match with the string (or string fragment) that is defined by the <**p**> parameters. It then extracts selected characters from the string and transmits each character as a separate analog signal, such that the analog value of each output is the same as the ASCII value of the extracted character.

Consider a security system that transmits data in the format:

> \x2A {1-byte zone number} {2-byte zone status} {5-byte zone name} \x35 {1-byte checksum} \x26

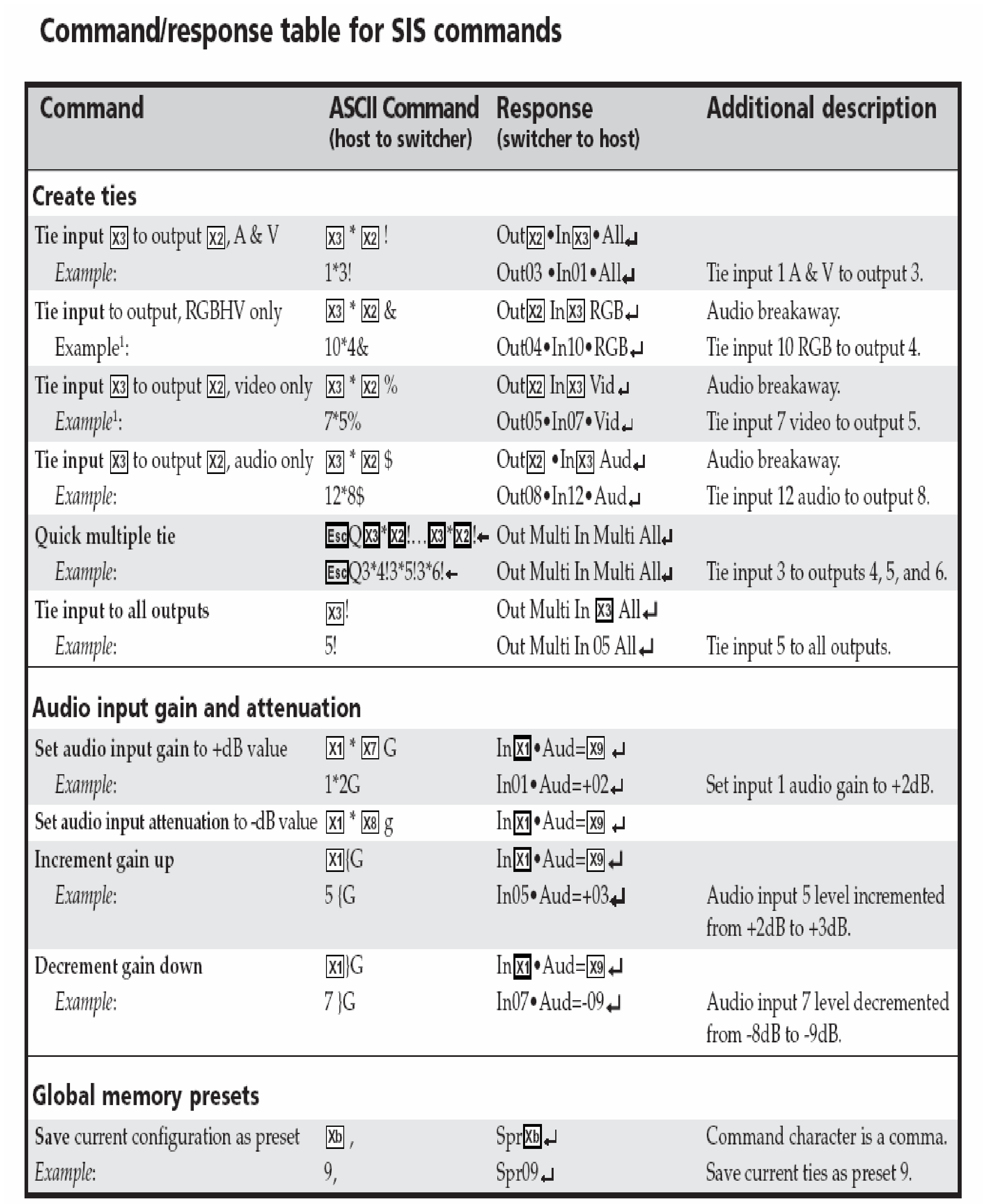
Where the hexadecimal values are constants and the fields in curly braces are variable values. Suppose that the two bytes representing the zone status must be extracted and routed to two analog outputs whenever the zone number equals \x01. Further suppose that the zone name and the checksum are irrelevant.

Since there are twelve bytes of input data there must be twelve <**p**> parameters, defined as follows:

<**p1**> (must match \x2A): 012Ah
<**p2**> (must match \x01): 0101h
<**p3**> (extracted and routed to <**byte1**>): 0200h
<**p4**> (extracted and routed to <**byte 2**>): 0200h
<**p5**> (ignore): 0000h
<**p6**> (ignore): 0000h
<**p7**> (ignore): 0000h
<**p8**> (ignore): 0000h
<**p9**> (ignore): 0000h
<**p10**> (must match \x35): 0135h
<**p11**> (ignore checksum): 0000h
<**p12**> (must match \x26): 0126h

Here <**p3**> and <**p4**>, the two bytes representing the zone status, will be extracted and routed to the <**byte 1**> and <**byte 2**> analog outputs.

4. Exercise: Audio/video switching



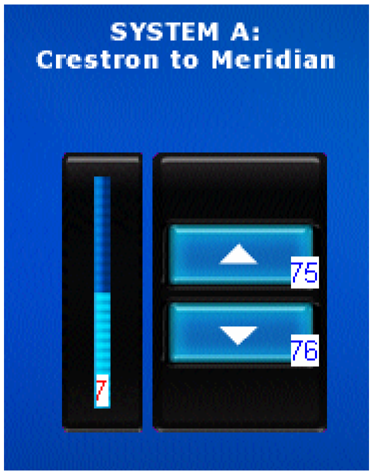
Command/response table for SIS commands

Command	ASCII Command (host to switcher)	Response (switcher to host)	Additional description
Create ties			
Tie input X3 to output X2, A & V	X3 * X2 !	OutX2•InX3•All↵	
Example:	1*3!	Out03 •In01•All↵	Tie input 1 A & V to output 3.
Tie input to output, RGBHV only	X3 * X2 &	OutX2 InX3 RGB↵	Audio breakaway.
Example[1]:	10*4&	Out04•In10•RGB↵	Tie input 10 RGB to output 4.
Tie input X3 to output X2, video only	X3 * X2 %	OutX2 InX3 Vid↵	Audio breakaway.
Example[1]:	7*5%	Out05•In07•Vid↵	Tie input 7 video to output 5.
Tie input X3 to output X2, audio only	X3 * X2 $	OutX2 •InX3 Aud↵	Audio breakaway.
Example:	12*8$	Out08•In12•Aud↵	Tie input 12 audio to output 8.
Quick multiple tie	EscQX3*X2!...X3*X2!←	Out Multi In Multi All↵	
Example:	EscQ3*4!3*5!3*6!←	Out Multi In Multi All↵	Tie input 3 to outputs 4, 5, and 6.
Tie input to all outputs	X3!	Out Multi In X3 All↵	
Example:	5!	Out Multi In 05 All↵	Tie input 5 to all outputs.
Audio input gain and attenuation			
Set audio input gain to +dB value	X1 * X7 G	InX1•Aud=X9 ↵	
Example:	1*2G	In01•Aud=+02↵	Set input 1 audio gain to +2dB.
Set audio input attenuation to -dB value	X1 * X8 g	InX1•Aud=X9 ↵	
Increment gain up	X1{G	InX1•Aud=X9↵	
Example:	5 {G	In05•Aud=+03↵	Audio input 5 level incremented from +2dB to +3dB.
Decrement gain down	X1}G	InX1•Aud=X9↵	
Example:	7 }G	In07•Aud=-09↵	Audio input 7 level decremented from -8dB to -9dB.
Global memory presets			
Save current configuration as preset	Xb ,	SprXb↵	Command character is a comma.
Example:	9,	Spr09↵	Save current ties as preset 9.

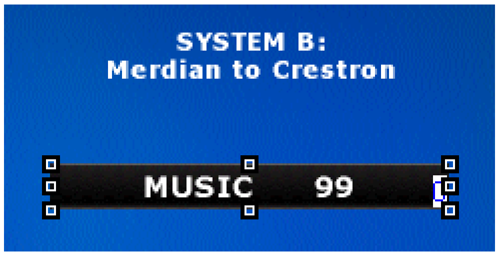
The protocol for the QM-MD switcher can be found in the Simpl Windows help file within the topic 'QM-MD 16x16'.

5. Exercise: Meridian

Work in groups of two people. One person acts as the CRESTRON system, one person as the MERIDIAN unit.

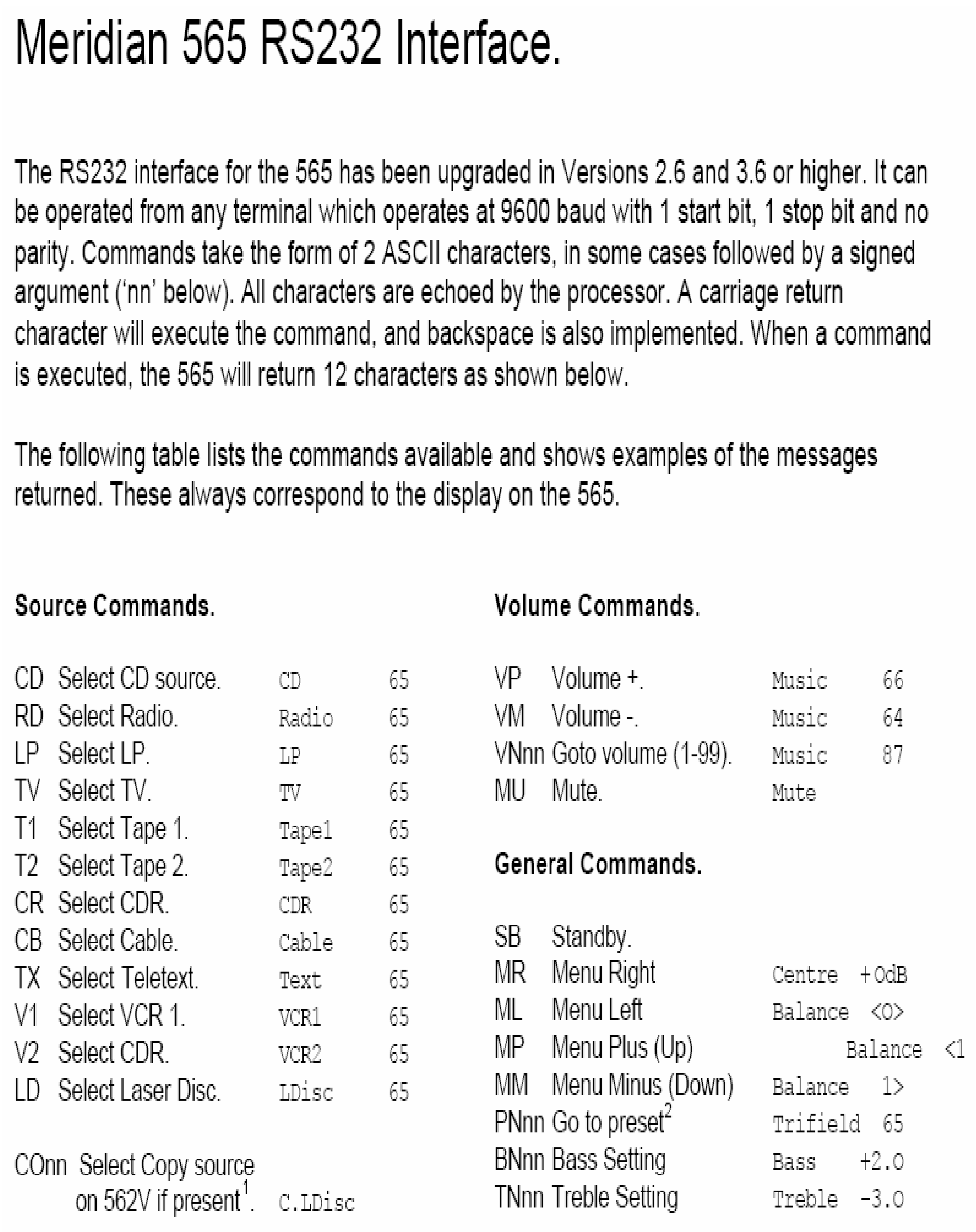


Crestron sends volume up and down commands **VP** and **VM** to the Meridian, using pulses of 0.2s. The Crestron processor and the Meridian are both linked to one another via serial port A.



The Meridian receives this string and sends the actual volume to the CRESTRON in the official format being: **MUSIC(5 spaces)00** to **MUSIC(5 spaces)99**.

On the Meridian touchpanel, the actual volume should be displayed. Crestron receives the reply from Meridian and takes out the volume information to send to a Gauge.



Meridian 565 RS232 Interface.

The RS232 interface for the 565 has been upgraded in Versions 2.6 and 3.6 or higher. It can be operated from any terminal which operates at 9600 baud with 1 start bit, 1 stop bit and no parity. Commands take the form of 2 ASCII characters, in some cases followed by a signed argument ('nn' below). All characters are echoed by the processor. A carriage return character will execute the command, and backspace is also implemented. When a command is executed, the 565 will return 12 characters as shown below.

The following table lists the commands available and shows examples of the messages returned. These always correspond to the display on the 565.

Source Commands.

CD	Select CD source.	`CD`	`65`
RD	Select Radio.	`Radio`	`65`
LP	Select LP.	`LP`	`65`
TV	Select TV.	`TV`	`65`
T1	Select Tape 1.	`Tape1`	`65`
T2	Select Tape 2.	`Tape2`	`65`
CR	Select CDR.	`CDR`	`65`
CB	Select Cable.	`Cable`	`65`
TX	Select Teletext.	`Text`	`65`
V1	Select VCR 1.	`VCR1`	`65`
V2	Select CDR.	`VCR2`	`65`
LD	Select Laser Disc.	`LDisc`	`65`
COnn	Select Copy source on 562V if present[1].	`C.LDisc`	

Volume Commands.

VP	Volume +.	`Music`	`66`
VM	Volume -.	`Music`	`64`
VNnn	Goto volume (1-99).	`Music`	`87`
MU	Mute.	`Mute`	

General Commands.

SB	Standby.		
MR	Menu Right	`Centre`	`+0dB`
ML	Menu Left	`Balance`	`<0>`
MP	Menu Plus (Up)	`Balance`	`<1`
MM	Menu Minus (Down)	`Balance`	`1>`
PNnn	Go to preset[2]	`Trifield`	`65`
BNnn	Bass Setting	`Bass`	`+2.0`
TNnn	Treble Setting	`Treble`	`-3.0`

6. Exercise: Console command

New used symbols

- **Console [console]**

The Console symbol transmits (<tx$>) and receives (<rx$>) serial data to and from the Viewport console. This allows us to send commands to the processor as if from viewport and also receive commands and responses from viewport.

- **Serial Gather [gather]**

The Serial Gather evaluates its serial input string until it finds the character specified in the <**delimiter**> parameter. Then it transmits the delimited portion of the string (including the <**delimiter**>). The remaining string fragment is stored in an internal buffer that is the size of the <**length**> parameter. If the input should contain multiple delimited strings, each delimited portion is transmitted in order until the symbol no longer finds a matching character.

- **Serial Substring [sub$]**

The Serial Substring evaluates it's serial input string until it find either the start character or start position and collects the string up to the end character or end position. This serial result is sent out of the **out$**.

DAY 3

1. Exercise: TCP/IP Client

The Soundserver is an Ethernet Device at IP address 10.0.0.40. The port number for communications is 5xx based on your seat position in the training room starting with 501 for processor 1 leading to 512 for processor 12.

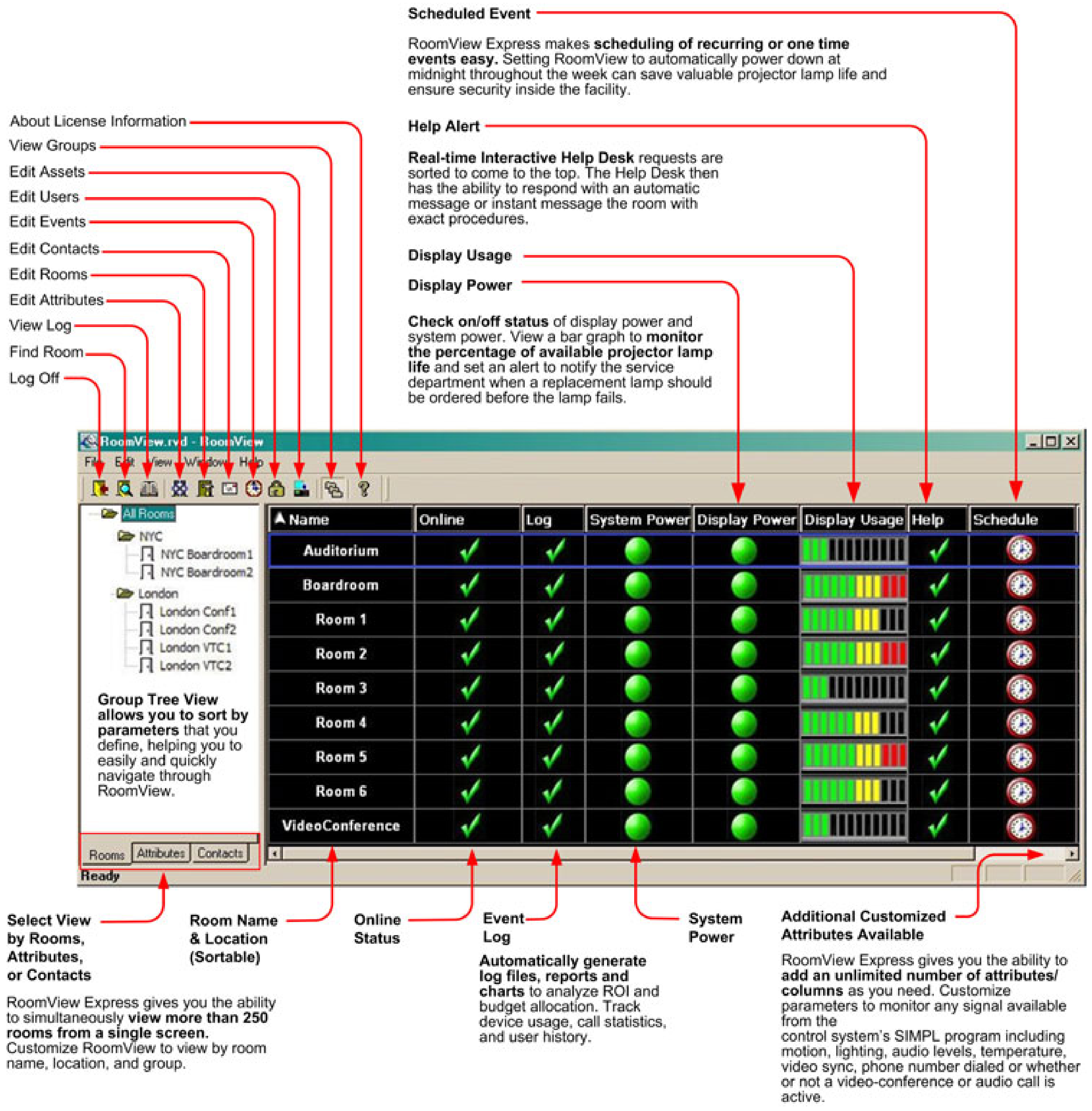
Protocol to control the soundserver:

• Previous disc:	CD –\r
• Next disc:	CD +\r
• Previous track:	TR –\r
• Next track	TR +\r

2. Exercise: Roomview

We will look at the implementation of the roomview symbol to allow control over the system power, projector power, lamp life etc.

The Roomview interface uses the internal loopback address for communication and therefore should have 127.0.0.1 as its IP address. The IP-ID should be set to 05.



Name	Online	Log	System Power	Display Power	Display Usage	Help	Schedule
Auditorium	✓	✓	●	●	🟩	✓	🕓
Boardroom	✓	✓	●	●	🟩🟨🟥	✓	🕓
Room 1	✓	✓	●	●	🟩🟨	✓	🕓
Room 2	✓	✓	●	●	🟩🟨🟥	✓	🕓
Room 3	✓	✓	●	●	🟩	✓	🕓
Room 4	✓	✓	●	●	🟩🟨	✓	🕓
Room 5	✓	✓	●	●	🟩🟨🟥	✓	🕓
Room 6	✓	✓	●	●	🟩🟨	✓	🕓
VideoConference	✓	✓	●	●	🟩	✓	🕓

Scheduled Event

RoomView Express makes **scheduling of recurring or one time events easy.** Setting RoomView to automatically power down at midnight throughout the week can save valuable projector lamp life and ensure security inside the facility.

Help Alert

Real-time Interactive Help Desk requests are sorted to come to the top. The Help Desk then has the ability to respond with an automatic message or instant message the room with exact procedures.

Display Usage

Display Power

Check on/off status of display power and system power. View a bar graph to **monitor the percentage of available projector lamp life** and set an alert to notify the service department when a replacement lamp should be ordered before the lamp fails.

Group Tree View allows you to sort by parameters that you define, helping you to easily and quickly navigate through RoomView.

Select View by Rooms, Attributes, or Contacts

RoomView Express gives you the ability to simultaneously **view more than 250 rooms from a single screen.** Customize RoomView to view by room name, location, and group.

Room Name & Location (Sortable)

Online Status

Event Log

Automatically generate log files, reports and charts to analyze ROI and budget allocation. Track device usage, call statistics, and user history.

System Power

Additional Customized Attributes Available

RoomView Express gives you the ability to **add an unlimited number of attributes/ columns** as you need. Customize parameters to monitor any signal available from the control system's SIMPL program including motion, lighting, audio levels, temperature, video sync, phone number dialed or whether or not a video-conference or audio call is active.

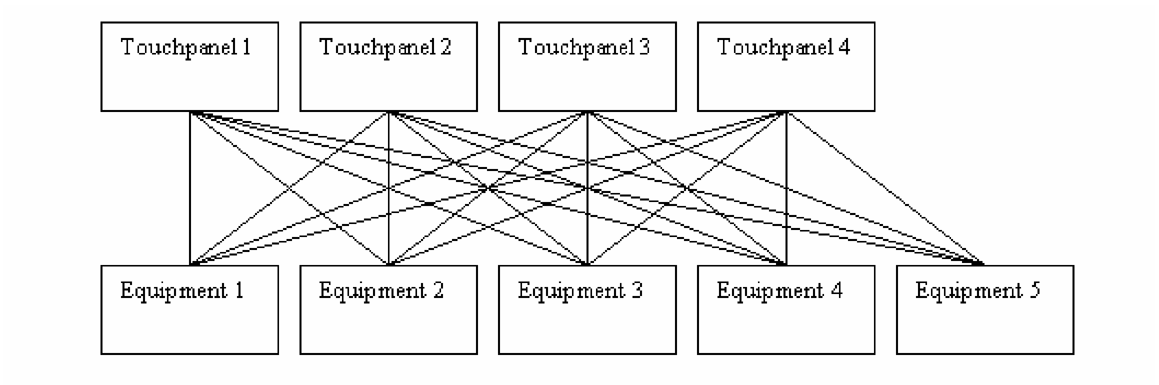
3. Exercise: Crosspoint routing

This exercise is designed to demonstrate the implementation of the Crosspoint router symbols and their respective effects on program speed and size.

Our exercise uses 2 touchpanels on our single TPS-4000 to represent separate rooms of a house. Each room in the house should be able to control the lights in any other room.

The Equipment Crosspoint Routing symbol works in combination with the Control Crosspoint Routing and Equipment/Control Crosspoint Connect symbols to facilitate signal routing in large AV applications where multiple interfaces control multiple devices.

For example, a non-crosspoint system that includes 4 touchpanels controlling 5 devices requires 20 control paths.



Implementing the above system requires a minimum of 60 gates. That is, each of the 20 control paths requires 3 buffers for routing digital, analog and serial signals.

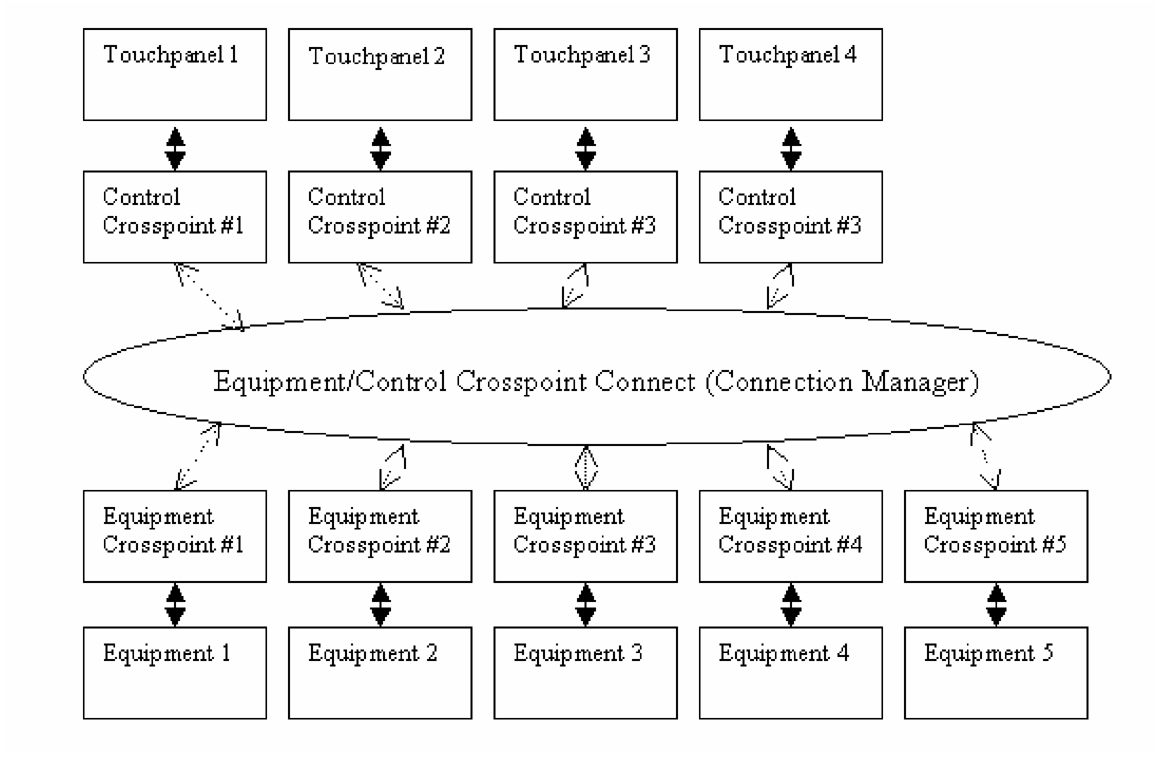
The use of Crosspoint symbols reduces the number of gates required to implement the system.

All the signals from a control source such as a touchpanel are routed to the Control Crosspoint symbol. All the signals from a device are routed to an Equipment Crosspoint symbol. The signals coming into the input side of one symbol type will be propagated to the output side of the other symbol type.

New used symbols

Equipment Crosspoint Routing [ecross]
Control Crosspoint Routing [ccross]
Equipment / Control Crosspoint Connect [eccon]

Each Control Crosspoint symbol has a unique Control ID; each Equipment Crosspoint symbol has a unique Equipment ID. The Control ID and the Equipment ID are fed into the Equipment/Control Crosspoint Connect symbol, which controls the connections between the two symbols.



4. Exercise: E-powerpoint

E-powerpoint enables the Crestron control system to control a Microsoft PowerPoint presentation running on a PC. The software detects loss of connection and automatically reconnects.

APPENDIX

ASCII	HEX	DEC	ASCII	HEX	DEC	ASCII	HEX	DEC	ASCII	HEX	DEC	ASCII	HEX	DEC	ASCII	HEX	DEC	ASCII	HEX	DEC	ASCII	HEX	DEC
NUL	00	0	SP	20	32	@	40	64	`	60	96	Ç	80	128	á	A0	160	+	C0	192	Ó	E0	224
SOH	01	1	!	21	33	A	41	65	a	61	97	ü	81	129	í	A1	161	-	C1	193	ß	E1	225
STX	02	2	"	22	34	B	42	66	b	62	98	é	82	130	ó	A2	162	-	C2	194	Ô	E2	226
ETX	03	3	#	23	35	C	43	67	c	63	99	â	83	131	ú	A3	163	+	C3	195	Ò	E3	227
EOT	04	4	$	24	36	D	44	68	d	64	100	ä	84	132	ñ	A4	164	-	C4	196	õ	E4	228
ENQ	05	5	%	25	37	E	45	69	e	65	101	à	85	133	Ñ	A5	165	+	C5	197	Õ	E5	229
ACK	06	6	&	26	38	F	46	70	f	66	102	å	86	134	ª	A6	166	ã	C6	198	µ	E6	230
BEL	07	7	'	27	39	G	47	71	g	67	103	ç	87	135	º	A7	167	Ã	C7	199	þ	E7	231
BS	08	8	(	28	40	H	48	72	h	68	104	ê	88	136	¿	A8	168	+	C8	200	Þ	E8	232
HT	09	9	)	29	41	I	49	73	i	69	105	ë	89	137	®	A9	169	+	C9	201	Ú	E9	233
LF	0A	10	*	2A	42	J	4A	74	j	6A	106	è	8A	138	¬	AA	170	-	CA	202	Û	EA	234
VT	0B	11	+	2B	43	K	4B	75	k	6B	107	ï	8B	139	½	AB	171	-	CB	203	Ù	EB	235
FF	0C	12	,	2C	44	L	4C	76	l	6C	108	î	8C	140	¼	AC	172	¦	CC	204	ý	EC	236
CR	0D	13	-	2D	45	M	4D	77	m	6D	109	ì	8D	141	¡	AD	173	-	CD	205	Ý	ED	237
SO	0E	14	.	2E	46	N	4E	78	n	6E	110	Ä	8E	142	«	AE	174	+	CE	206	¯	EE	238
SI	0F	15	/	2F	47	O	4F	79	o	6F	111	Å	8F	143	»	AF	175	¤	CF	207	´	EF	239
DLE	10	16	0	30	48	P	50	80	p	70	112	É	90	144	_	B0	176	ð	D0	208		F0	240
DC1	11	17	1	31	49	Q	51	81	q	71	113	æ	91	145	_	B1	177	Ð	D1	209	±	F1	241
DC2	12	18	2	32	50	R	52	82	r	72	114	Æ	92	146	_	B2	178	Ê	D2	210	_	F2	242
DC3	13	19	3	33	51	S	53	83	s	73	115	ô	93	147	¦	B3	179	Ë	D3	211	¾	F3	243
DC4	14	20	4	34	52	T	54	84	t	74	116	ö	94	148	¦	B4	180	È	D4	212	¶	F4	244
NAK	15	21	5	35	53	U	55	85	u	75	117	ò	95	149	Á	B5	181	ı	D5	213	§	F5	245
SYN	16	22	6	36	54	V	56	86	v	76	118	û	96	150	Â	B6	182	Í	D6	214	÷	F6	246
ETB	17	23	7	37	55	W	57	87	w	77	119	ù	97	151	À	B7	183	Î	D7	215	¸	F7	247
CAN	18	24	8	38	56	X	58	88	x	78	120	ÿ	98	152	©	B8	184	Ï	D8	216	°	F8	248
EM	19	25	9	39	57	Y	59	89	y	79	121	Ö	99	153	¦	B9	185	+	D9	217	¨	F9	249
SUB	1A	26	:	3A	58	Z	5A	90	z	7A	122	Ü	9A	154	¦	BA	186	+	DA	218	·	FA	250
ESC	1B	27	;	3B	59	[	5B	91	}	7B	123	ø	9B	155	+	BB	187	_	DB	219	¹	FB	251
FS	1C	28	<	3C	60	\	5C	92	|	7C	124	£	9C	156	+	BC	188	_	DC	220	³	FC	252
GS	1D	29	=	3D	61	]	5D	93	{	7D	125	ø	9D	157	¢	BD	189	¦	DD	221	²	FD	253
RS	1E	30	>	3E	62	^	5E	94	~	7E	126	£	9E	158	¥	BE	190	Ì	DE	222	_	FE	254
US	1F	31	?	3F	63	_	5F	95	DEL	7F	127	ƒ	9F	159	+	BF	191	_	DF	223		FF	255
ASCII	HEX	DEC	ASCII	HEX	DEC	ASCII	HEX	DEC	ASCII	HEX	DEC	ASCII	HEX	DEC	ASCII	HEX	DEC	ASCII	HEX	DEC	ASCII	HEX	DEC