

5. Computer Interface Commands in Binary

The Interface that is built into the Central Control Unit has the capability of working in either one of two modes. One is binary and is discussed briefly in this section of the booklet, and the other is ASCII which is covered in detail in the next section. Further information on the Binary mode and more examples of programming in BASIC are found in The Users Guide to the Märklin Digital System.

Computers can be easily programmed to talk to the trains and accessories through the Märklin Interface. Commands can be sent a number of ways, depending on the type of programming language being used and the interests of the programmer. Data can be sent as a string in BASIC such as: PRINT #1, CHR\$(10), or as a hexadecimal number: PRINT #1, HEX\$(10). Or, if assembly language routines are being used, bytes can be addressed as binary or hex notations; 0000 1010 (in binary) or 0A (in hex). The command "PRINT #1 " in BASIC directs the output to the serial port on the computer if it is set as port #1. When using this command be sure to end the command with a semicolon ";". Further information on this command and others mentioned in this section are given in Chapter Eighteen of The Users Guide to the Märklin Digital System.

The Interface is a serial peripheral to the computer. This means it has a baud rate and specific communication parameters. These parameters need to be set in the computer software so the computer can "talk" to the Interface. The communications port requirements for ASCII mode can be changed but for binary mode they must be:

Baud rate = 2400
Start bits (if requested by computer) = 1
Stop bits = 2
Parity = None
Word size = 8 bits

For the purposes of this discussion, all notations will be given in decimal numbers with examples showing the commands written in Microsoft BASIC being sent out serially with string notation (i.e. CHR\$(5)). The reason for this is that Microsoft BASIC is the most popular form of the language being used with microcomputers. It is the form included with the IBM, Radio Shack, and is also sold for the Macintosh, and its very close to forms of BASIC being used for the Apple II, Commodore, and Atari.

Commands will be presented for all types of control operations that are possible for controlling k83 decoders and decoders in the locomotives. Each set of commands going out the serial port needs to be separated from other commands that are sent by a fraction of a second. If this is not done, the signals

will jam up on the serial line and no action will result. For best results, it is advisable to use a loop command in between commands. For example:

```
10 FOR X=1 TO 500
20 NEXT X
```

This short loop should be set into a subroutine in BASIC. Then the subroutine can be called at the end of each set of commands. This pattern will be followed in the examples that follow.

Engine Commands

Engines are controlled by sending a **two byte command** to the serial port. The following is an example:

```
10 PRINT #1,CHR$(10);CHR$(1);:GOSUB 100
100 FOR X=1 TO 500
110 NEXT X
120 RETURN
```

NOTE: after the last CHR\$ command in line 10 there was a semi-colon (;). It is mandatory that each command line conclude with this semi-colon or the programs will not work.

In the example above, the first byte is the "10" in the command CHR\$(10). CHR\$ tells the computer that what is to be sent is the ASCII string for the decimal number "10". For those who understand computer programming the 10 is the same as the byte 0000 1010, or the hex number 0A. The second byte is the "1" in the command CHR\$(1).

In commanding engines, the two bytes are: 1) the command for speed and function in the first byte, and 2) the engine's address in the second byte. Engine speed can be any one of the 15 possible speed steps available, and the engine address can be any one of the addresses from 1 to 80. For example: you want engine #26 to travel at a speed of 5. The command would be:

```
10 PRINT #1,CHR$(5);CHR$(26);:GOSUB 100
20 END
100 FOR X=1 TO 500
110 NEXT X
120 RETURN
```


Or if engine #2 is to travel at speed 13, the command would be:

```
10 PRINT #1,CHR$(13);CHR(2);:GOSUB 100
20 END
100 FOR X=1 TO 500
110 NEXT X
120 RETURN
```

Speed "0" is the same as "STOP", the engine will not move. Reverse is set as speed "15". It will not only reverse the engine, but it will stop it also. That makes the number 14 the fastest possible speed for the engine. The following will send engine #5 moving at the speed 10, reverse it, then start again at speed 4, then stop the engine.

```
10 PRINT #1,CHR$(10);CHR$(5);:GOSUB 100
20 PRINT #1,CHR$(15);CHR$(5);:GOSUB 100
30 PRINT #1,CHR$(4);CHR$(5);:GOSUB 100
40 PRINT #1,CHR$(0);CHR$(5);:GOSUB 100
50 END
100 FOR X=1 TO 500
110 NEXT X
120 RETURN
```

Notice in the above example, that each command is separated with the loop to protect the data from being sent too fast. The amount of time between commands can be adjusted down by making the number "500" smaller. Doing so will increase the operating speed of the program, but be careful not to make it so small that the data jams. Newer interfaces such as the one built into the Central Control Unit have both RTS and CTS signals and so they do not need the delay loops in each of the statements as seen in this chapter. Refer to Appendix E for more information on these interface units.

Functions

Functions such as smoke, lights, or Telex couplers can be activated along with the speed command. Simply add 16 to the command, and the function will be active also. For example, take the series of commands given in the last example, if the function should be active when the engine begins the first speed (which is 10) and it should continue until the engine reverses in command line 20, the example would look like this:

```
10 PRINT #1,CHR$(26);CHR$(5);:GOSUB 100
20 PRINT #1,CHR$(15);CHR$(5);:GOSUB 100
```

```
30 PRINT #1,CHR$(4);CHR$(5);:GOSUB 100
40 PRINT #1,CHR$(0);CHR$(5);:GOSUB 100
50 END
100 FOR X=1 TO 500
110 NEXT X
120 RETURN
```

The only difference is that the speed "10" in line 10, has 16 added to it. The byte "26" tells the engine to turn on the function and travel at speed 10. If the function should be "on" through the entire series of commands, then 16 needs to be added to all of the first bytes. Notice that "stop" becomes 16 and "reverse" becomes 31.

```
10 PRINT #1,CHR$(26);CHR$(5);:GOSUB 100
20 PRINT #1,CHR$(31);CHR$(5);:GOSUB 100
30 PRINT #1,CHR$(20);CHR$(5);:GOSUB 100
40 PRINT #1,CHR$(16);CHR$(5);:GOSUB 100
50 END
100 FOR X=1 TO 500
110 NEXT X
120 RETURN
```

The decoder chips used in engines have four other functions built into them and they can be called up with the Control 80f Unit or through the interface. They can be called with the same two byte system that has been discussed previously, but, the first byte is the signal to turn on the functions. The second remains the address of the engine or the digital car. There is no relationship to speed control with the first byte. It is independent from speed and original function control. Engines will continue with their previous commanded activity, but will employ the new function command. These chips are employed in the Märklin 1 gauge engines and also in the HO Panorama and Dance cars at the present time. Other applications for these chips will be added to the Märklin line in the future.

The following codes enable the four other functions of the c80 chips. Each combination of the functions can be activated by one of the code numbers. For example, if you wanted the waiter to move up the aisle and also have the table and overhead lights on in the Panorama car, you would choose code 77 or 78. Each of these codes moves the waiter with function #1 and #2, and turns on lights with function #3 and #4. Of course, with the waiter, it is not possible to activate functions #1 and #2 at the same time. The waiter cannot move both up and down the aisle at the same time.

code functions on

64	all off
65	#1
66	#2
67	#1,#2
68	#3
69	#3,#1
70	#3,#2
71	#3,#2,#1
72	#4
73	#4,#1
74	#4,#2
75	#4,#2,#1
76	#4,#3
77	#4,#3,#1
78	#4,#3,#2
79	#4,#3,#2,#1

Any number not listed will turn off. For example "79" will turn on all functions, and if followed by "71", it will just turn off function #4. The others (#1,#2,and #3) will stay on. Commands will look like the following example which turns on functions #2 and #3 in engine 55.

```
10 PRINT #1,CHR$(70);CHR$(55);:GOSUB 100
20 END
100 FOR X=1 TO 500
110 NEXT X
120 RETURN
```

Switch Commands

Switching is accomplished in a similar fashion to controlling engines. They take a two byte command, the first one giving the switch directions, and the second is the switch address. Addresses are given simply as 1-256. There is no need to worry about Keyboard addresses, just remember to set your k83 Decoder addresses correctly. For example, a k83 with the address 1(1) will control switches 1,2,3, and 4. The next k83 in line would be 1(2) which will control switches 5,6,7, and 8. Remember that each Keyboard controls 16 switches and each k83 control 4 of those 16 switches. The second Keyboard controls k83 units 2(1) through 2(4), and the second set of 16 switches (numbers 17 - 32). The k83

unit with the address 2(1) has switches 17, 18, 19, and 20. And, k83 number 2(4) has switches 29, 30, 31, and 32. Only the switch number needs to be given in the computer program.

Switch settings are "33" to go straight and "34" for the curve or "branch" setting. An example of setting switch no. 3 to the branch would be:

```
10 PRINT #1,CHR$(34);CHR$(3);:GOSUB 100
20 END
100 FOR X=1 TO 500
110 NEXT X
120 RETURN
```

Setting the switch back to the straight position would look like this:

```
10 PRINT #1,CHR$(33);CHR$(3);:GOSUB 100
20 END
100 FOR X=1 TO 500
110 NEXT X
120 RETURN
```

When connecting uncouplers to the k83 Decoders, you will connect one to the red side and one to the green side. This means that when you want the one on the green side to activate, you need to send the "33" and to activate the red one, you send "34". When working with uncouplers in binary mode you will not be able to control the length of time the uncoupler is in the up position. It will always be a short time. Only in ASCII mode can you add a time delay.

System Commands

The Digital System can be started and stopped from the Control 80 Unit. The computer only needs to send a one byte command to initiate either action "go" or "stop". "Go" is sent with the number "96" and "stop" uses "97". The system would be started with this series of commands:

```
10 PRINT #1,CHR$(96);:GOSUB 100
20 END
100 FOR X=1 TO 500
110 NEXT X
```


And the system can be stopped with this command:

```
10 PRINT #1,CHR$(97);:GOSUB 100
20 END
100 FOR X=1 TO 500
110 NEXT X
```

Track Detection Modules

The s88 Decoders store data sent from the track detectors or the reed switches. Each s88 has room to store two bytes of data (16 separate contacts). In order for these Decoders to be read by the computer, they must first be told to "dump" their memories into the computer. Two options are available for requesting this "dump", depending on how you want the s88 units to feed their data into the computer.

Read one s88 unit - If only one s88 is to be read, you send the code "192" PLUS the number of the unit to be read in. For example: you only have one s88 on the layout - you read it with the command "193" (192 + 1). On the other hand, if you had 4 units on the layout and you wanted to read only number 3, then send the code 195 (192+3). The command would be:

```
10 PRINT #1,CHR$(195);:.....
```

The command is not finished because more instruction need to be given to the computer so it can accept the incoming data, but that will be covered later.

Read many s88 units - The other option for "dumping" s88 memory is for getting all the s88 units to dump memory up to a specific unit. For example: four s88 units are on the layout and you want information from the first 3 only. The command for this type of memory dump is the number 128 PLUS the number of the last unit to be read. In this case the command would be "131" (128+3). The command would look like:

```
10 PRINT #1,CHR$(131);:.....
```

In this case, the first s88 on line would dump its memory, then the second, and finally the third unit connected in the series.

When any of these commands are given, the s88 units will send data back to the computer. The program needs to be alerted so it can receive this data and report it back to the computer operator in some readable fashion. Most BASIC

programs require that two communications lines be opened in order to both send and receive data. The opening lines on a program would be:

```
10 OPEN "COM1:2400,N,8,2"FOR OUTPUT AS #1
20 OPEN "COM1:2400,N,8,2"FOR INPUT AS #2
```

The parameters mentioned earlier are seen in these statements. Notice the 2400 Baud, No parity, 8 data bits, 2 stop bits. One line is set for output and numbered #1. That is why all the print statements seen earlier said PRINT #1. They were only sending data on the output line. The other line is for input and is numbered as #2.

If data is expected, then it should be assigned a variable and the computer should be alerted immediately after sending the code to dump memory. The program would look like this:

```
10 OPEN "COM1:2400,N,8,2"FOR OUTPUT AS #1
20 OPEN "COM1:2400,N,8,2"FOR INPUT AS #2
30 PRINT #1,CHR$(193);:a$=INPUT$(2,#2)
```

Line 30 sends the command CHR\$(193) out on #1 telling s88 number 1 to dump memory. That memory will come into the variable "a\$". It will be assigned as a result of the command a\$=INPUT\$(2,#2). The first number 2 means that there will be 2 bytes coming in and the #2 means it is on the INPUT format as defined in statement 20. Those two bytes can now be separated and printed on the screen to let you know the status of the s88. This command will do that:

```
30 PRINT #1,CHR$(193);:a$=INPUT$(2,#2)
40 contact=ASC(LEFT$(a$,1)):PRINT contact
50 contact2=ASC(RIGHT$(a$,1)):PRINT contact2
```

A new variable called "contact" will be assigned the leftmost single byte of the two bytes (this is the first one from sockets 1-8 on the s88 unit) and then will be printed to the screen. Line 50 will take the second byte (sockets 9-16 on the s88) and assign it to variable "contact2" and print it.

The data printed to the screen can be confusing unless you understand how binary works. Each of the sockets stand for a binary number in the sequence 1,2,4,8,16,32,64, and 128. They are assigned as follows:

sockets binary number

1	=	128
2	=	64
3	=	32
4	=	16
5	=	8
6	=	4
7	=	2
8	=	1

If only the contact connected to socket #8 was triggered, then the number printed on the screen would be "1". If only #3 was triggered then it would be "32". If multiple numbers are triggered, then their assigned binary numbers are added together. For instance, contacts #7 and #8 would be reported as a "3" (2+1), while #1 and #8 would be 129 (128+1).

The second side of the s88 is read the same way, only it will be in the second byte of data received. Contact #9 would be the same as #1 on the opposite side of the s88 and socket #16 is the same as #8 (they are both "1") Programs can be written that will analyze these number totals and tell you the contacts that were active. It is easy to write such a program. If the number is greater than 128 then you know that contact #1 has been triggered. If this was true, then subtract 128 from the number and check it again. If it is now greater than 64, that means contact #2 was triggered. If that's right subtract 64 and check the next number, etc.

The s88 units can be told to reset its memory after a dump to the computer, or it can remain as it was and continue adding data to the byte. A reset would make each position in memory a "0" again. If you want it to reset send the single byte 192 without adding any numbers to it. If the reset mode is to be "off" then send the code 128. For example, the following would set the reset mode "on":

10 PRINT #1,CHR\$(192);

Don't forget to also add the time delay loop with these commands, unless your working with the newer interface with both RTS and CTS signals (see next section).

6. Computer Interface Commands in ASCII

NOTE: This 6023 Central Control Unit has an updated computer interface built into it. Some of the instructions found in Chapter Seventeen of THE USERS GUIDE TO THE MÄRKLIN DIGITAL SYSTEM is specific for the Märklin no. 6050 separate interface module which only works in binary mode. Most of the detailed information for the binary mode found in Chapter Seventeen does apply to the 6023 interface. The material in this booklet refers to the ASCII commands that are unique to the interface in the 6023 Central Control Unit and some elementary binary mode commands which are repeated in Chapter Seventeen of THE USERS GUIDE TO THE MÄRKLIN DIGITAL SYSTEM.

GENERAL INFORMATION

The Central Control Unit (Märklin No. 6023) offers many expanded interface commands in comparison to the 6050 interface which only allows communication in binary mode. The newer interface includes both an ASCII mode and a binary mode. When first initializing the interface in the Central Control Unit, the default mode will be the ASCII mode (this means when the interface is first turned on it is in ASCII mode). This can be changed by pressing the "stop", "go" and function "off" keys simultaneously at the control unit then letting go of the "stop" and "go" keys first, then wait 5 seconds and let go of the "off" key thereby resetting the interface to the binary mode as in the 6050 interface. Commands for controlling the 6050 interface are contained in Chapter Seventeen of THE USERS GUIDE TO THE MÄRKLIN DIGITAL SYSTEM. They will not be repeated in detail in these instructions, instead the reader is directed to that chapter.

CONNECTING CABLES

Pin assignments at the interface socket

	u	
5		1
	6	
4		2
	3	

u	top of plug
1	RD Data
2	CTS Handshake
3	GND Ground
4	TD Data
5	RTS Handshake

(looking at the socket on the interface)

Cables for connecting your computer to the interface are available from your Märklin dealer. Ask for the 5 wire cable for the 6023 interface. Be sure to specify the type of computer and the number of pins at your serial port. If you wish to

construct your own cable, refer to the pin assignments for the various computers in THE USERS GUIDE TO THE MÄRKLIN DIGITAL SYSTEM.

COMMAND FORMAT

The binary format used in the 6050 interface is still understood by the interface in the 6023 Central Control Unit, plus it has some added features. In the ASCII format, letters and decimal numbers are expected alternately (i.e. L 15 S 5 F 1). The numbers can range from 0 to 256, blank characters are ignored but can be listed for clarity.

Only numbers, letters, blank characters and carriage returns(CR) will be accepted by the 6023 interface. Capital letters and small letters are treated equally and are returned just as they are entered in the echo modes. A command will be interpreted only after the reception of the carriage return (CR) and executed if possible.

Commands consist of a maximum of 6 elements (letters or decimal numbers). The string entered into the computer can be as long as you want, but only the first 6 elements will be executed. Only one command line per entry is possible and linking strings is not possible.

Processing the data with the computer is based on the fact that only one line is being sent or received (BASIC: Print #1,"string" or Input #2, variable\$). Data being received in the serial buffer is treated as a character string with variable lengths depending on the command that generated the data coming in.

s88 MODULES

With the new 6023 interface only four s88 modules can be accessed instead of 31. If these are constantly being monitored, their data is stored in the memory of the Central Control Unit. When the computer requests data from the s88 units there is an immediate response from the memory of the 6023 rather than waiting for the s88 to respond. Access time is reduced by this function and status of the other contacts is not lost through the inquiry. It is possible to tell the Central Control Unit to watch for specific contacts to be activated, either turned on or off and to report this change. This command only works in the echo mode since the data has to be "echoed" back to the computer.

After logging on, only the first s88 module is activated. If others are to be monitored, especially in the Watch and Job modes, they must first be requested (with the D# command in ASCII mode).

BINARY MODE

The binary mode of the interface of the new Central Control Unit still recognizes the binary commands as used in the 6050 interface. The solenoid command was modified however. When the solenoid (in a switch or signal) is activated the interface will automatically send the shut off signal after about 200

milliseconds. This prevents accidentally leaving the solenoid on thereby causing the switch unit to run hot and possibly burn out. The time span before sending the off commands can be altered up to 12.5 seconds for older switches and uncouplers which may need the extra time to activate.

The s88 commands to reset (128 and 192) are no longer used. Once data is sent from the s88 to the interface, that data is stored in the interface and only changed if the s88 reports new data at that contact. Resetting is not necessary any longer. Also only four s88 units can be monitored, requests for units 5 thru 31 will be ignored.

Error reporting commands that are accessible in the ASCII mode (E1 or E2) will not work in binary mode. During an emergency stop, commands going through the Central Control Unit to the track will be ignored. Commands to other units (i.e. s88 monitoring) can still be executed.

ONE BYTE COMMANDS

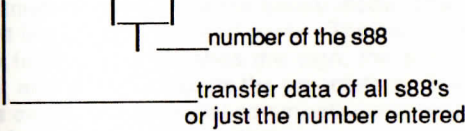
CLEAR	0 1 1 0 0 0 0	96	h'60'
EMERGENCY STOP	0 1 1 0 0 0 1	97	h'61'

• Switching Modes

0 1 1 0 0 1 x x


100	h'64'	direct ASCII mode	Echo mode 0	'd'	direct
101	h'65'	ASCII Mode	Echo mode 1	'e'	echo
102	h'66'	ASCII Mode	Echo mode 2	'f'	fast

• Reading s88 Data

1 x 0 0 0 x x x

 number of the s88 001 to 100 1 to 4
 transfer data of all s88's or just the number entered
 0 = all s88's up to no. entered
 1 = just the s88 # entered

Assignment of contacts on the incoming byte from the s88

value	128	64	32	16	8	4	2	1
	msb							lsb

1. byte	1	2	3	4	5	6	7	8
2. byte	9	10	11	12	13	14	15	16
if possible.....								
3. byte	1	2	3	4	5	6	7	8
4. byte	9	10	11	12	13	14	15	16
5. byte	etc. until requested module is given							

Example:	read in module #1	193 or 129	h'81 or h'c1'
	read in modules #1-#3	131	h'83'
	read in only module #3	195	h'c3'

2 BYTE COMMANDS

- **Locomotive control**

Byte one --- 0 0 0 x | x x x x
|
└─── speed levels
 special functions 0= off
 1=on

example:	function off	loco stop	0	h'0'
		speed levels	1..14	h'01'...h'0E'
		reverse	15	h'0F'
	function on	stop	16	h'10'
		speed levels	17..30	h'11'...h'1E'
		reverse	31	h'1F'


Byte two --- 0 x x x x x x x
|-----|
 loco number 1 thru 80

```

example:
      loco 25 in forward motion speed level 10 no function
            10, 25                                h'0A", h'19'
      loco 25 reversing with function on
            31, 25                                h'1F", h'19'
      loco 25 speed 3 with function on
            19, 25                                h'13", h'19'

```

- Solenoid control

Byte one 0 0 1 0 0 0 x x
 direction 10=red (curve on switches)
 01=green (straight on switches)

example:	
position curve, red, etc for approx 170-230 milliseconds	34 h'22'
position straight, green, etc.	33 h'21'

Byte two x x x x x x x x

 |—————|

 solenoid address

example:		
switch 55 on red, curve (branch) setting	34, 55	h'22', h'37'
switch 20 on green, straight setting	33, 20	h'21', h'14'

3 BYTE COMMANDS

- Initializing the UART

The RS232 chip (UART) in the 6023 interface can have its communication parameters changed in the binary mode. The adjustments include the baud rate, word length, stop bits and parity. The change is formatted as a 3 byte command. The first byte establishes the sign, the second contains the code for the baud rate, and the third adjusts the parameters. The command for initializing the UART was developed for serial communications limited to 7 bit words such as found on the Apple II line of computers.

Byte one 0 1 1 0 1 0 0 x or 1 x x 1 x x x x

optional "0" or "1"

example:

104...105	h'68'..h'6B'	this and any below must be used with Apple II
144...159	h'90'..h'9F'	
176...191	h'B0'..h'BF'	
208...223	h'D0'..h'DF'	
240...255	h'F0'..h'FF'	

Byte two x x x x x x x x

Baud rate (see table)

optional (no function)

Byte three x x x x x x x x

function	1	0	decimal
word length	8	7	1
Parity	odd	even	2
Stop bits	2	1	4
Parity on	off	on	8

no function

Baud rate table

Byte	value	baud-rate
0	h'00'	75
1	h'01'	110
2	h'02'	134.5
3	h'03'	150
4	h'04'	300
5	h'05'	600
6	h'06'	1200

Byte	value	baud-rate
7	h'07'	2000
8	h'08'	2400
9	h'09'	4800
10	h'0A'	1800
11	h'0B'	9600
12	h'0C'	19200

BEWARE !!

values 13-15 request a baud rate that the UART does not use and causes the system to "hang-up"

Examples:

Adjusting to 9600 baud, no parity, 1 stop bit =

104, 11, 11	h'68', h'0B', h'0B'
or 254, 27, 107	h'FE', h'1B', h'6B'
or 255, 251, 235	h'FF', h'FB', h'EB'

Adjusting to 75 baud, 7 bit, even parity, 2 stop bits =

255, 240, 228	h'FF', h'F0', h'E4'
---------------	---------------------

The default setting on the Central Control Unit interface is 8 bit, 2 stop bits, no parity. If other than this parameter is to be used (such as on an Apple II), it must be set from the binary mode before proceeding. Notice that the 8th bit is not used in any of the parameter setting bytes, thus allowing a 7 bit machine to be reset.

OVERVIEW OF BINARY COMMANDS AND ERRORS

0...32	Locomotive control
33, 34	solenoid accessories
96, 97	go, stop
100...102	switch to ASCII modes
104...105	UART setting
129...132	read all s88 units up to ...
193...196	read in just one s88 unit
254, 255	UART setting

Values not listed have no effect other than possibly hanging up the system. They will not lead to any of the functions mentioned.

IMPORTANT !!!

Binary commands should **never** be terminated with a carriage return (CR, 13 or h'0D'). These are automatically generated in BASIC with the PRINT instruction unless a semicolon (;) is inserted at the end of the print line. This carriage return (13) command is interpreted by the Central Control Unit as a speed level 13 and so the next command is interpreted as a loco number. The results are that the system will hang-up. So, use the semicolon at the end of BASIC commands !

Trouble-shooting If you should cause the system to hang-up, you can reboot the interface by simultaneously pressing the stop and go buttons and the function off button on the Central Control Unit for about 2 seconds. Let go of the stop and go buttons first, wait, then let go of the function off button.

Prevention Of course the logical thing to do to prevent this from occurring is to put semicolons at the end of your print lines. Remember however, that this is only needed in the binary mode. ASCII mode requires the carriage return at the end of the print line so do not use the semicolon.

Differences with the 6050 interface

The Central Control Unit was designed to be compatible with the 6050. The binary mode is only one operating mode of this interface. The main emphasis on the new unit was directed towards the 'ECHO' modes, and has resulted in some major modifications over the 6050 interface. These changes however, will not affect the running of programs that were originally designed for the 6050 unit. What is most important is to realize that the semicolon is now needed where it may not of been before. Other changes that may cause some minor modifications include:

1. Turning off the solenoid - Since the new interface automatically turn off the solenoids, the old command (32) is now obsolete. Leaving it in a program will not cause any problems, it is just not recognized.

2. s88 Feedback modules - The 6050 allowed for 31 s88 units to be connected for feedback to the computer. Since the new central control unit's interface feeds the s88 information into a buffer in the interface, memory became a problem. For this reason the system is limited to only four s88 modules allowing a total of 64 contact points on the tracks.

2.1. Access time - The s88 contacts are read once every 25 milliseconds and stored in the interface memory buffer. When requesting s88 data, the information is retrieved from the buffer and sent at the speed of the baud rate. Thus the information is never older than 25ms. plus the baud rate.

2.2. Reset commands - Data from s88 units residing in the memory is not deleted if that data is not requested by the computer. Only data requested is deleted. The old reset commands are therefore not needed. It is possible in ASCII mode to remove this memory with the "R" command.

2.3 Number of Active s88's - after logging on, only the first s88 is activated. When requesting the 2nd, 3rd, or 4th modules only "0" will be transmitted. It is necessary to activate the modules that are desired through the ASCII mode. There is no access in binary. A sequence of strings can be created that will switch to ASCII, request the modules and then return control to binary mode. In BASIC the commands would start with - Print #1,CHR\$(100);

example:	100	switches to ASCII
	68, xx, 13	xx=modules to be active (49-53, 0-4)
	81, 13	switches back to binary

This sequence must be inserted at the beginning of the program if that program requests information from any more than the first s88 unit.

3. Emergency stops/short circuits - During an emergency, commands are no longer sent to the Central Control Unit. Storing a command, as is possible with the 6050 interface, does not apply. A command to the central unit (loco, solenoid, function) is ignored while in the stop mode. All other commands not directed toward the tracks will continue to function.

ASCII MODE

Short overview of all ASCII commands

X	invalidate this command string	eXit
S {CR}	emergency stop	Stop
G {CR}	clear, go	Go
V {CR}	speed control is echoed to serial buffer use echo mode 2	View
L # S # [F #] {CR}	engine address, speed and function command loco 1...80 speed 1...14 function 0...1	Loco-Speed -Function
L # D {CR}	direction control	Direction
M # R [time] {CR} M # G [time] {CR}	magnetic solenoid, time in 0.1 seconds no time = 200 ms default time=0, solenoid is not turned off	Magnetic Red - Green
M {CR}	magnetic solenoid turned off	
D # {CR}	define number of s88 to be active	Define
A # {CR}	read one s88 in '10001111' format use echo modes	Ascii
C # {CR}	read one contact of the s88 use echo modes	Contact
R # {CR}	remove s88 memory from storage A, D = 1...4 Contact 1...64	Remove
J # W # {CR} J # W # P/N {CR}	job x to wait for 1 to 12.7 seconds wait for contact to be positive or negative p=pressed, n=released, use echo modes	Job Wait Positive Negative
K # {CR}	kill job job number 1...3 contact 1...64	Kill
W # {CR}	send all s88 changes to serial buffer must use echo modes	Watch

Q {CR}	ASCII mode off, binary mode on	Quit
E # {CR}	echo modes activated 0/1/2 echo mode 0...2, this invokes ASCII modes	Echo

OPERATING MODES IN THE ASCII FORMAT

• Echo mode 0 - Echo turned off

In this mode there are no prompts or error messages. Operation is similar to the 6050 interface with the exception that commands are sent in ASCII strings. Transmission of commands takes place only at the request of the computer (or the s88 triggering a computer command). There are no independent messages from the Central Control Unit.

Application:

Simple controlling programs that are similar to the ones used for the 6050 except that the ASCII commands allow for better comprehension when reading programs. The ASCII mode is slower than the binary mode, but easier to follow for novice programmers.

• Echo mode 1 - Terminal operation

Every character is sent back to the serial buffer. A blank character is transmitted before and after every letter of a command to increase readability. A program needs to be written to retrieve this data in the buffer and print it on the screen or test it in the program. Refer to THE USERS GUIDE TO THE MARKLIN DIGITAL SYSTEM for more information on programming with echo mode 1.

* Echo Mode 2 - Short or fast echo operation

Every command is acknowledged after processing by a prompt or short error message. Information and error messages can be transmitted at any time (see mode 1). Refer to THE USERS GUIDE TO THE MARKLIN DIGITAL SYSTEM for more information on programming with echo mode 2.

DESCRIPTION OF INDIVIDUAL ASCII COMMANDS

Locomotive control

L # D {CR}	change of direction
L # S # {CR}	forward command with no functions
L # S # F # {CR}	forward command with functions
	special function 0=off 1=on
	speed level 0...14
	loco number 1...80

Function:

With this command the locomotive decoder can be controlled. The conditions of the special function is stored in the interface and if it is not added with the command (F #) it is added during transmission to the central unit. Thus, for example, a loco light activated through the controller remains on when the engine receives a new speed command from the computer. The same is true for changes in engine direction. The speed level is not stored in memory and must be stated when changing functions.

Attention:

When changing a locomotive's direction more than once it is necessary to enter a speed command between the direction changes. That speed can be "0" if you wish. The loco will not change more than once without a speed command after the direction change.

Solenoid and magnetic accessories:

M {CR}	Magnetic items off
M # x {CR}	Magnetic item number change G or R
M # x # {CR}	
	switch time (0=don't switch off)
	in 0.1 seconds 1...127 (.1 to 12.7 sec)
	direction R=red, G=green
	Magnetic accessory number 1...256

Function:

Magnetic operated accessories such as track switches, uncouplers, and signals are controlled with this command. To eliminate programming steps, an automatic "switch off" command is sent approximately 200ms after the switching command. This can be changed from 0 (no switch off) to 12.7 seconds, in .1 second steps. If a "0" time is given, the magnet will stay on until the M command

is sent. This would allow uncouplers to remain on for whatever length of time was wanted to accomplish the uncoupling task. It would not work with the electro magnet in the Märklin crane because once another "M" command is entered, the previous one will shut off. So you could activate the electro magnet, but you couldn't lift it or move it sideways without it shutting off. To use this crane, you need the k84 decoder that turns on an internal relay or you need an external relay in addition to the k83 decoders.

s88 Feedback modules

R {CR}	deletes s88 storage in the interface
D # {CR}	defines the number of active s88 units
A # {CR}	input status of all contacts in unit x
	_____ module 1....4
C # {CR}	read in only one contact
	_____ contact number 1.....64

Function:

Unlike the binary mode, feedback information is sent in ASCII format. In addition to requesting a whole module, a single contact can be accessed.

A set contact is given as a "1", and on open contact as a "0". When requesting data on the whole module, positions within the string can be dealt with by using the string processing commands in BASIC (i.e. MID\$, RIGHT\$, LEFT\$).

The contact numbers of the first module are identical with the module's legend. With the 2nd thru 4th modules you need to add the numbers of the previous modules.

Module one	contacts 1 .. 16
Module two	contacts 17 .. 32
Module three	contacts 33 .. 48
Module four	contacts 49 .. 64

Only the first module is activated after logging on. If more than the first 16 contacts are wanted, you must activate the other modules with the "D" command. Four is the maximum number of modules that can be read, others will be ignored.

Attention:

After logging on incidental data may be in the memory buffer. It is necessary to clear this out with the command "R". If modules consistently transmit "0" as data, check to see if they have been activated.

Correction

X

the command just given is invalid

Function:

The last command is not executed. A command line can be interrupted anytime with the "X" command and then deleted. In echo mode 1 just a new prompt without OK is sent. In echo mode 2 a new '!' is sent. Then a new command can be entered. This is helpful if the terminal is being used that does not allow backspace corrections.

Attention:

"X" is the only command that does not need to be acknowledged with a {CR}. If a {CR} is sent, an additional OK message will be sent back to the screen.

Example:

Keyboard input
L 16 S X
L 15 S3 F1

	terminal output	
	1	echo mode 2
> L 16 S X		!
> L 15 S3 F1		!
OK		!
>		!

Emergency Stop:

S {CR}	stop command set
G {CR}	clear (go) command

Function:

The computer can activate the stop and go commands of the Central Control Unit with these commands. The status report % STOP and % GO will be echoed back by the interface. The commands take about 1 second to evaluate the emergency stop line because the electronics need this reaction time in case of a short circuit in the system.

Attention:

The LEDs of the locomotive or solenoid articles on the control panel will go off for a very short time while the stop and go commands are being executed.

Analog Input:

V {CR}	speed controller is queried
--------	-----------------------------

Function:

The position of the speed control knob is read into the terminal as the corresponding speed levels. This data can be stored and later "replayed" causing the engine to mimic the previous actions that were given by hand at the control knob. This command requires that the echo modes be invoked and the data can be retrieved at the serial port by the computer, analyzed, and assigned to variables for later recall.

Jobs:

K # {CR}	cancel job >no message
J # W # {CR}	report after waiting time
	waiting time in 0.1 sec.
	job number
J # W # x {CR}	wait for s88 contact
	wait for P (press) or N (release)
	contact number
	job number

Function:

The Central Control Unit provides the computer with a timer that will allow for time controlled processes to be accomplished outside of the computer without interfering with the computer program or the speed of the computer.

Where program loops have to be used with the 6050 to identify contacts at the s88 modules, this function is not accomplished by the interface. To make full use of this function, the programming language should have some type of serial interface interrupt such as the one used in BASIC: ON COM GOSUB..... Otherwise loops will have to be written to continually check to see if the data has come into the serial port.

Attention:

Be sure that the corresponding s88 unit addressed in the wait command has been activated with the "D" command. This command recognizes the change of status of the contact (either from 0 to 1 = 'P' or from 1 to 0 = 'N'). Make sure that if you are waiting for the pressing ('P') of a contact that the previously stored memory was deleted. This can be done with the 'C' command which will access just the contact being addressed and reset its value in the memory.

Changing operating modes:

Binary mode	Q {CR}	ASCII mode off, binary mode on
ASCII mode	E # {CR}	ASCII or echo modes 0, 1, 2

Function:

These two commands are provided for changing the operating mode of the Central Control Unit's Interface. They correspond to the binary commands 100, 101, 102 and are usually only needed at the beginning of the program in the initializing phase.

Attention:

When exiting from the ASCII mode, the last command ('Q') must be concluded with a {CR} because this command is still executed in the ASCII mode. It can also be cancelled with 'X'. The newly chosen mode is only active after the {CR}.

Watch:

W # {CR}	watch all s88 changes
	turning on or off - 0= off 1= on

The Watch operation will only work in echo modes 1 or 2, because only these allow for independent messages from the interface to the computer. At the time the event occurs, the message is prepared and sent at the first available opportunity when the serial interface is clear, (i.e. no other commands or transmissions such as A, I, V.. etc. are going on).

Every pressing of an s88 contact is reported to the computer if Watch ('W') is activated. Releasing of a contact is only noticeable with the Job ('J') command. Watch has a buffer that stores all contacts until they are transferred. Multiple pressings of a contact until transfer will cause just one message. The transfer occurs from the highest to the lowest contact number in descending order.

The internal storage of the contacts is deleted just as in the 'C' command and the Watch command will continue to remain active if the mode is changed.

Attention:

A false transmission can be caused by the commands 'A', 'C', and 'R' if the Watch is left running since the deletion of the contact in Watch causes the same command as if the contact was activated on the layout and is then reported by the 'A' or 'C' command. In the binary mode or the Echo 0 mode the correct processing of a program can be interfered with or destroyed by a Watch command that was left running erroneously.

Example:

Action	Echo mode 1	Echo mode 2
WATCH - on	W {CR}	> W OK !
contacts 1, 3, 5 close		> ^CONTACT 05 ^05 ^CONTACT 03 ^03 ^CONTACT 01 ^01 OK !
contact 2 closed		> ^CONTACT 02 ^02 OK !
3 and 5 open		
delete storage	R {CR}	> R
contact 1 closed		^ CONTACT 01 ^01
request s88	A1 {CR}	> A 1
contact 1 closed		^ CONTACT 01 ^01 *1000000000000000 *1000000000000000 OK !
		>

MESSAGES FROM THE INTERFACE IN ECHO MODES 1/2

1. Ready-report, software handshake

The RTS line states when the Central Control interface is ready for reception. In echo mode 1 or 2 there is another software handshake. After the operation is completed, the interface reports 'OK' or '!'. If several reports (W, T, J) and transmission requests (A, C, V, I) are lined up at the same time, they are transmitted according to the priority list

Priorities for transmission	1. WATCH
	2. jobs
	3. s88
	4. engine control requests
	5. error messages

These transmissions are usually completed one after another with no 'OK' or '!' reports after each one.

2. Status and Error Messages

echo 1	echo 2	meaning
% stop	%0	emergency stop was activated
% go	%1	go or clear confirmation
% syntax error	%2	wrong command syntax
% range error	%3	numerical value too small or large
% time-out error	%4	command not sent to central unit
% used by # (1...10)	%# (A...J)	that engine is used by Control 80(1..10)
% - IO - # (1...15)	%# (Q...)	error in RS-232 transfer

If there are several errors in a command string, only the one is reported that was detected first.

ACCESSING THE COMMANDS FROM THE INTERFACE

Since the Central Control Interface can transfer data at any time, even when the program hasn't requested any, there must be a way for the computer to handle this data simply and quickly. It is best if the computer and the programming language has an interrupt controller at the serial interface such as BASIC: ON COM GOSUB.

The echo mode 2 was designed specifically for processing data input in this manner. The incoming data can be easily recognized by the first character. The first character is the ID of the transmission. In echo mode 1 the information is presented as legible text, which makes the recognition of the data more difficult for the programming language. Mode 1 is less useful than mode 2 for fast and automatic programming.

ID	assignment	length	1	echo 2	remarks
*	individual contacts	3	2		always
	whole s88 modules	18	17		requested by
&	speed control knob	13	3		the program
%	status error	var	2		
<	job completed	3	2		
!	clear report	--	1		echo mode 2 only
OK	clear report	2	--		echo mode 1 only
>	prompt	var	--		echos commands given in mode 1