



Digital Satellite Equipment Control (DiSEqC™)



BUS FUNCTIONAL SPECIFICATION

VERSION 4.2

February 25, 1998

Reference Documents that define the DiSEqC System:

DiSEqC™ Bus Specification Version 4.2 (February 25, 1998)

DiSEqC™ Slave Microcontroller Specification Version 1.0 (February 25, 1998)

DiSEqC™ Logos and Their Conditions of Use (February 25, 1998)

Associated Documents:

Update and Recommendations for Implementation Version 2.1 (February 25, 1998)

Application Information for using a "PIC" Microcontroller in DiSEqC™ LNB and simple switcher Applications Version 1.0 (June 7, 1999)

Application Information for Tuner-Receiver/IRDs (April 12, 1996)

Application Information for LNBs and Switchers Version 2 (February 25, 1998)

Reset Circuits for the Slave Microcontroller (August 12, 1996)

Simple Tone Burst Detection Circuit (August 12, 1996)

Positioner Application Note Version 1.0 (March 15, 1998)

CONTENTS

0. Status	1
1. Introduction	1
2. System Overview	2
3. Slave device architecture	3
4. Bus Hardware Specification	4
5. Method of Data-Bit Signalling	5
5.1. Backwards Compatibility	6
5.2. Simple “Tone Burst” Control Signal	6
5.3. Combined Transmission of DiSEqC™ and Backwards-Compatible Signals	7
5.4. Repeated Messages	8
6. Message Data Format	8
6.1. Framing (first) byte	9
6.2. Address (second) byte	10
6.3. Command (third) byte	10
6.4. Data (subsequent) bytes	10
7. Initialisation	10
7.1. Power (Standby) Control	11
7.2. Bus Arbitration (DiSEqC™ Levels 2.x)	11
7.3. Bus Collision detection (DiSEqC™ Levels 2.x)	12
8. Definition of DiSEqC™ Bus Commands	13
8.1. Framing Byte	13
8.2. Address Byte	14
8.3. Command Byte	15
9. Description of DiSEqC™ Bus Commands	18
9.1. Write Port Group	18
9.2. Write Channel frequency	19
9.3. Positioner Operation and Commands	20
9.4. Write Polariser Skew	21
9.5. Read Positioner Status Byte (Level 2.x)	21
9.6. Read Local Oscillator Frequency (Level 2.x)	22
9.7. Read Status Byte (Level 2.x)	23
9.8. Read Configuration Byte (Level 2.x)	23

9.9. Read Committed Switches Byte (Level 2.x)	24
9.10. Read Uncommitted Switches Byte (Level 2.x)	24
10. Contact details.....	25

0. Status

This Specification contains minor corrections and some additions to the Version 4.1 Specification dated April 18th, 1997. Additions and significant changes are marked with a vertical revision bar in the left-hand margin. This version incorporates new commands for DiSEqC™ Level 1.2, supporting “one-way” control of motorised Positioners (antenna-movement actuators) and Polarisers, via the DiSEqC™ Bus. The original “two-way” Positioner commands are provisionally withdrawn and eventually will be deleted, or amended, unless there are requests for their re-instatement.

1. Introduction

This document describes the Functional Specification for the Digital Satellite Equipment Control (DiSEqC™) Bus. DiSEqC™ is an OPEN STANDARD with additions controlled by industry agreement.

The DiSEqC™ system is a communication bus between satellite receivers and satellite peripheral equipment, using only the existing coaxial cable. DiSEqC™ can be integrated into consumer satellite installations to replace all conventional analogue (voltage, tone or pulse width) switching and all other control wiring. A “Universal” Slave IC supports multiple applications by link-configuration to identify the peripheral hardware that it is controlling.

The main advantages of DiSEqC™ are:

- standardised digital system with non-proprietary commands
- enables switching in multi-satellite installations
- backwards compatible with 13/18 volt and 22 kHz tone switching
- potential for reduced power dissipation and thus cost reduction and improved reliability
- elimination of switching problems caused by incompatibility of system components
- easier receiver installation using device recognition via optional two-way communication

2. System Overview

The DiSEqC™ concept is based on extending the present 22 kHz tone-signalling method, thus minimising the changes required in Tuner-receiver or Integrated Receiver Decoder (IRD) units, and simplifying backwards compatibility. However, since the full DiSEqC™ protocol supports a return-signalling path and multiple peripheral devices, it is necessary to more closely define the impedances (at 22 kHz) on the bus, than the simple low impedance drive (supply voltage modulation) commonly employed with the present tone method.

DiSEqC™ is a single master, single or multi slave system, so communications may be initiated only by the master Tuner-receiver/IRD. This avoids the need for the software in the Tuner-receiver/IRD to continually monitor the bus (by polling or interrupts) when there may be other tasks in hand. In principle, the master can transmit messages by “chopping” an existing 22 kHz tone, generated either entirely by software or with some hardware support.

The DiSEqC™ slave function generally will be implemented in a simple microcontroller. When this is dedicated to DiSEqC™ bus support it is practicable to perform not only the control functions but also the tone decoding and encoding in software, thus eliminating many of the components currently used for 22 kHz tone detection.

It is expected that the single (mask-programmed) microcontroller will be able to support most applications. However, the microcontroller does not have sufficient input/output pins to support all possible requirements concurrently, and (in “two-way” systems) must also be able to report back to the master which facilities are actually available. Therefore, at power up or reset, the microcontroller software scans the peripheral hardware attached to its pins to determine the “family” which it represents, and the “resources” available.

In the more complex installations there is some risk of there being two “identical” slaves (e.g. Switchers or LNBs) on the same bus. The preferred method for dealing with this is a “loop through” structure (see *section 7.2.*) so that initially there is only one slave on the bus which the master can interrogate. When this device is first accessed, its address can be changed to one which is not normally used. It can then be instructed to loop the bus through to the next device, which can then be installed in a similar way.

To accommodate situations where identically-addressed devices do exist on the bus, a collision-detection and arbitration scheme is defined. Thus, if two devices have the same address, but some difference in their available “resources”, then they can be identified by the master on subsequent occasions. However, because the device which “wins” any arbitration is effectively determined at random, then the identity of any truly identical devices will be lost when power is next removed.

3. Slave device architecture

As far as practicable, the intention is to make the DiSEqC™ Bus Specification independent of the actual Slave hardware or microcontroller type employed. However, to assist in understanding the function of the various commands, the general structure of the DiSEqC™ Slave will be outlined. Full details will be found in the separate DiSEqC™ Slave Microcontroller Specification.

A prime function of the DiSEqC™ system is to remotely select between two or more switchable alternatives, such as the Polarisation of received signals, LNB Local Oscillator frequency, etc. The DiSEqC™ chip therefore has a number of logic output pins to select the various signal types, either by electronic or electro-mechanical (relay) methods.

In hardware terms, the actual switching output pins of the DiSEqC™ slave chip normally will be identical, so the actual control functions could be connected in any arbitrary manner. However, an aim of the DiSEqC™ concept is to rigidly fix not only the pin functions (e.g. Polarisation axis) but also their sense (e.g. Enabled = Horizontal Polarisation) so that system installation is greatly simplified. It is expected that the definitions will be observed not only in devices such as LNBs, but also in Switcher and SMATV installations, whenever possible.

In many applications of the DiSEqC™ slave chip, there are available further “Uncommitted” switching control pins which can be used for arbitrary selection functions, or, at a later date, may be allocated to any specific new facilities which may become desirable. A probable use is for the selection of up to 16 inputs in “SMATV” switching systems where the numbered input signals do not map directly to the recognised broadcast parameters of “Band”, “Polarisation”, etc.

To allow the DiSEqC™ chip to report back to the master (Tuner-receiver/IRD) which facilities are available, the slave software must scan a number of input pins. These pins can be dedicated inputs, or combined input/output pins (as are available in some families of microcontroller), or even “virtual” pins fixed in a particular version of the control software. It is of no concern to the Bus Specification or the Tuner-receiver/IRD software how the configuration data is obtained, the master software will simply access a group of formally-defined “status registers” to indicate the available functions and external condition of the slave hardware.

Some peripheral device functions require continuously variable, or analogue, control signals, such as Polarisation Skew or “Installer Aid” signal strength. This information is passed over the bus usually as a single byte data value from 0 to 255. It may then be processed in several different ways, for example by applying the individual bits to 8 separate pins for driving a digital-analogue converter, or driving a single pin with a variable duty cycle Pulse Width

Modulation (PWM) waveform, which may be low-pass filtered to a steady dc value. Again it is not relevant to the Bus Specification how the actual analogue control level is achieved.

Another peripheral device requiring support is the “Positioner”, to move a steerable antenna to point in a required direction. Small incremental movements of the dish are normally detected as pulses by a magnetic or optical sensor, etc. on the motor drive. A DiSEqC™ slave processor can monitor these pulses, and also endstop operation, etc. and employ simple software to drive the motor in the appropriate direction to achieve the required output-shaft position. The preliminary set of “two-way” commands (i.e. demanding the return of data, and employing control functions within the Tuner-receiver/IRD), which were defined in previous Bus Specifications, have been replaced in this version by a new set optimised for “one way” DiSEqC™ operation.

4. Bus Hardware Specification

To permit slave devices to communicate back to the master, it is necessary to make more specific recommendations for the operating impedances of the bus than is required for a simple 22 kHz continuous tone signalling. In addition, it is advantageous to be able to accept a wide range of DC operating voltages during the DiSEqC™ introductory phase, to permit compatible and/or hybrid operation with the present signalling methods. The recommendations for operation of the DiSEqC™ Bus are:

The long-term recommendation for the DC Supply voltage is 12 ± 1 volts, but during the introductory phase, voltages up to 20 volts should be tolerated by peripheral devices (i.e. not suffer catastrophic failures). For the short term, some manufacturers may choose to retain compatibility with 13/18 volt signalling levels, and this facility is supported in the first versions of the DiSEqC™ Slave integrated circuit.

It is recommended that Tuner-receiver/IRDs should support a DC Supply current drain of up to 500 mA for peripheral devices powered by the Bus.

The nominal 22 kHz signalling amplitude is 650 mV (± 250 mV) peak-peak. To accommodate tolerances and voltage drop in the cable, the detector should respond to amplitudes down to approximately 300 mV (± 100 mV). The maximum recommended amplitude to be applied to the bus is 1 volt peak to peak.

Designers of receivers and all peripherals (i.e. “Masters” and “Slaves”) should strive to avoid injecting “noise” or spurious signals onto the Bus. However, it is recognised that some disturbance may occur on a cable which

carries both power and data signals. Therefore, it is recommended that the DiSEqC™ tone reception interface circuits should not lead to detection of signals (at any frequency) having an amplitude of less than 100 mV peak to peak (either cyclical or “spikes”).

When back-channel signalling is to be supported (DiSEqC™ Implementation Level 2.0 and above), the Master transmitter (in the Tuner-receiver/IRD) should present a nominal source impedance of 15Ω to the bus, at 22 kHz. Typically, this termination will consist of a resistor, a parallel inductor to support the d.c. power supply current, and a capacitor (to ground) to shape the 22 kHz signal when the cable length and termination capacitance are both small.

To permit transport of the 22 kHz signal, it is recommended that the total load capacitance at the far end of the bus (cable) should not exceed 250 nF (0.25 μ F). This value is chosen to tolerate one LNB of existing design for a cable length of up to 50 meters. True DiSEqC™ peripherals should not load the bus by typically more than 100 nF, and for certain classes of device such as SMATV nodes and Installer Aids, a much lower value is preferred. It is expected that installation tables will be made available to indicate the maximum acceptable loading for various combinations of terminating device(s) and cable lengths.

5. Method of Data-Bit Signalling

DiSEqC™ uses base-band timings of $500\mu s$ ($\pm 100\mu s$) for a one-third bit PWK (Pulse Width Keying) coded signal period on a nominal 22 kHz ($\pm 20\%$) carrier. The end of each DiSEqC™ message is signalled by a minimum of 6 ms of silence. The following diagram shows the 22 kHz time envelope for each bit transmitted, with nominally 22 cycles for a Bit ‘0’ and 11 cycles for a Bit ‘1’:

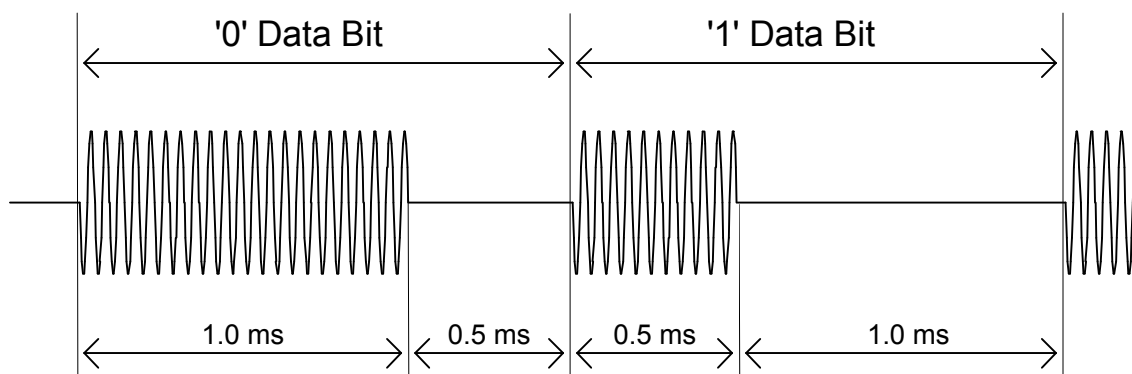


Figure 1: DiSEqC™ Modulation Scheme

5.1. Backwards Compatibility

DiSEqC™ commands are able to co-exist on the Bus (i.e. on the satellite download cable) with the established voltage and 22 kHz tone signalling methods. Thus the DiSEqC™ data may be superimposed on a supply voltage of either 13 or 18 volts (for H/V polarisation switching) and also may be inserted between a continuous 22 kHz tone (for Local Oscillator frequency or Satellite selection), provided there are short “quiet” gaps before and after the DiSEqC™ command burst. The recommended timing periods are defined in the next section.

There are two aspects to this compatibility:

Firstly, it is possible to add a new DiSEqC™ - controlled switch to select between one or more existing LNBs. The Tuner-receiver/IRD sends a DiSEqC™ message to the switch to select the required input, and then the Tuner-receiver/IRD can send the appropriate voltage and/or continuous tone to select the required output from the LNB.

Secondly, the introductory versions of the DiSEqC™ Slave microcontroller will initially (until they receive a DiSEqC™ command) respond to the established signalling methods so that they may be used under the control of non-DiSEqC™ Tuner-receiver/IRDs. This reduces the need for manufacturers to produce different versions of LNBs and Switchers for “old” and “new” systems, and increases the chances that a user will already have some DiSEqC™ accessories installed when he upgrades to a DiSEqC™ Tuner-receiver/IRD at some later date.

5.2. Simple “Tone Burst” Control Signal

To avoid the need to use a complete DiSEqC™ Slave microcontroller just to add a simple two-way switch to an existing system, a simple “Tone-Burst” command has been added to the DiSEqC™ Specification. This can be detected by simple analogue hardware. The Tone-Burst uses a 22 kHz carrier and is sent after any normal DiSEqC™ data burst, but before the resumption of the continuous tone (if this is being used for backwards-compatible signalling). Two types of Tone-Burst are used, one is Unmodulated to select “Satellite Position A”, and the other is Modulated to select “Satellite Position B”.

The nominal Modulated Tone-Burst corresponds to a single-byte DiSEqC™ command of nine ‘1’s (i.e. ‘FFh’), which has a 1: 3 duty cycle. Thus, low-pass filtering of the detected modulated carrier can give a DC level which is approximately 33 % of the peak carrier. The nominal burst duration is 12.5 ms (nine 0.5 ms pulses and eight 1.0 ms gaps) with the normal DiSEqC™ modulation tolerances of ± 20 %.

The Unmodulated Tone-Burst has a similar duration to the Modulated burst and can be detected as a d.c. level approximately 3 times larger than the Modulated burst. The detection circuitry may use either this difference in levels, or may detect the presence or absence of the modulation tone itself. In either case, the termination of the Tone-Burst can cause the appropriate control value to be latched into a flip-flop.

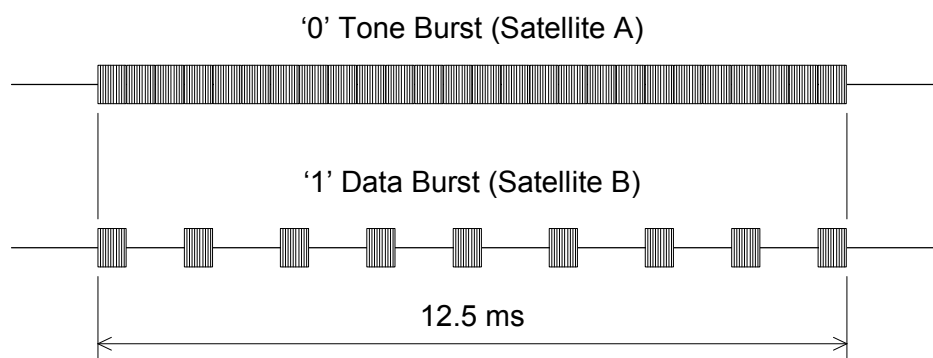


Figure 2: Timing Diagram for Tone Burst Control Signal

5.3. Combined Transmission of DiSEqC™ and Backwards-Compatible Signals

It is possible to combine both types of signalling on the bus, either to specifically control accessories of both types, or simply to avoid the need to configure the Tuner-receiver/IRD to a particular command protocol. Generally the control data carried in the two protocols will be the same, but this is not essential because a DiSEqC™ slave microcontroller should ignore any backwards-compatible signals once it has received a valid DiSEqC™ command.

The following figure shows the recommended timing of the combined signals. The minimum time gaps should be observed to avoid the risk of malfunctions (particularly of simple analogue decoder hardware). The voltage-signalling “step” should occur significantly before the Tone-Burst to avoid the risk of disturbing simple analogue hardware Tone-Burst detectors. Time-gaps much longer than the minimum are acceptable, but will produce more severe “glitches” in switch lines controlled by the continuous tone. In “two-way” DiSEqC™, the master may need to leave a gap of at least 150 ms between the message and the next tone, to wait for the slave’s reply. However, the reply normally will be received much sooner, and then the Tone-Burst can be sent after a pause of 15 ms.

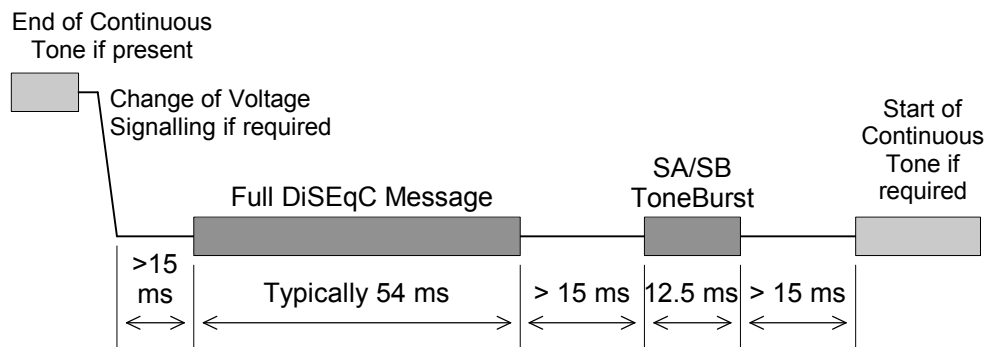


Figure 3: Timing Diagram for DiSEqC™ Level 1.0

5.4. Repeated Messages

When DiSEqC™ switches are “cascaded” it may be necessary to repeat commands for the “distant” device. The reason is that the “nearby” switch may have de-selected the distant one, and thus removed it from the bus. Therefore, it cannot receive the first transmission of a message, and may not even be receiving power. For the case of a distant slave powered via the bus, a pause of at least 100 ms must be allowed for the slave to initialise, before any command is sent to this device. Further information may be found in applications information such as *section 5.2. of the “Application Information for Tuner-Receiver/IRDs”*, and the *“Update and Recommendations for Implementation Version 2.1”*, published by Eutelsat.

6. Message Data Format

DiSEqC™ messages consist of one or more bytes of data, each formed from 8 bits as defined above, and with each byte followed by an odd parity bit. The most-significant bit (of each byte) is transmitted first, and then in descending sequence to the least-significant bit last.

The detailed message format is described in later sections, but the basic command structure (from the master) consists of 3 bytes: a start and run-in (framing) byte which includes some high-level control, a slave address byte and a command byte. For some commands, additional data is carried in one or more subsequent (data) bytes.

FRAMING	P	ADDRESS	P	COMMAND	P	DATA	P
---------	---	---------	---	---------	---	------	---

The return of a “reply” message from the slave is at the discretion of the master. Thus the bus can be kept clear if the master is unable to receive a reply (e.g. “one-way” DiSEqC™, Level 1.x), or wishes to send rapid updates (e.g. of an analogue polarisation or signal strength value). However, it is expected that most unique-address commands will request an acknowledgement because data-collection, communications validation and arbitration, cannot be performed without one. The reply will normally consist of a single acknowledge byte, which will be followed by one or more additional bytes when the command involves the return of data (or to supply an error-code). The slave should reply within a maximum of 150 ms from the end of the command message.

The handling of errors can be largely at the discretion of the master (Tuner-receiver/IRD) control software. In general it is recommended that any command which fails to receive a reply should be repeated once, but a different number of repeats might be used during the installation process (and as a result of the behaviour of the slave at this time).

To be compatible with possible future extensions, the message bytes normally should be fully decoded from their definitions in *section 8*.. However, as an introduction, their overall structure is outlined below.

6.1. Framing (first) byte

The first four bits of the first byte will be initially reserved as a “run in” and “framing” pattern ‘1110’ for DiSEqC™ commands. It is possible that future extensions may make use of these bits, but it may be assumed that the first two bits always will be identical (i.e. contribute even parity) for error-detection purposes.

The fifth bit initially will be reserved as ‘0’ but may be used in the future for “open” extensions to the protocol.

The sixth bit is cleared to ‘0’ if the message originates from the master and is set to ‘1’ if it is a reply from a slave.

The seventh bit is set to ‘1’ if a response is required and cleared to ‘0’ if no reply is expected. In the case of a slave, the set bit can request a re-transmission because an error has been detected.

In command messages, the eighth bit is set to ‘1’ if the message is a re-transmission of a previous message (for which a reply was expected and not received). In reply messages the eighth bit is used to qualify the reason for failure to execute a command (either erroneous or unsupported command).

6.2. Address (second) byte

The second byte indicates which slave device the message is intended for, and is divided into two nibbles of 4 bits each. The first nibble indicates the overall “family” to which the slave belongs (e.g. LNB, Positioner, etc.) and the second nibble allows the family to be divided into sub-types for some applications. In each nibble the value ‘0’ (binary ‘0000’) indicates a “wildcard” or “don’t care” address and the command should be executed by any slave within the family and/or sub-type.

Original Equipment Manufacturers may apply for an OEM address byte, which may then be followed by a moderate number of bytes (provisionally a maximum of 6), using any protocol which does not conflict with normal operation of the DiSEqC™ Bus.

6.3. Command (third) byte

The third byte specifies the required action(s) to be taken by the addressed slave(s). The byte generally should be fully decoded, although some related commands (by function or format) are grouped together to simplify decoding.

6.4. Data (subsequent) bytes

The fourth and subsequent command bytes (from the master) and second and subsequent reply bytes (from the slave) contain numeric data when necessary. This may be an “analogue” value (e.g. for a Polariser), a number (e.g. related to the LNB Local Oscillator frequency, or for Positioner operation, etc.) or a group of 8 separate flag bits.

7. Initialisation

When the master applies power to the bus, any slave which supports backwards compatibility takes up a state which is consistent with the voltage and tone signals on the cable. Then, if the slave detects a valid DiSEqC™ command on the bus, it abandons the backwards compatible mode and sets up its control lines in response to the DiSEqC™ commands. Although some slaves may take up a defined state at power-up, the master control software

must not make any assumptions, but should issue all relevant commands during the initialisation procedure.

7.1. Power (Standby) Control

The DiSEqC™ command set includes commands to power-down (into Standby) and power-up any or all of the accessories on the bus. However, to provide full backwards compatibility with existing LNBs (with no power-control functions), it is at present necessary for all IF switches to also interrupt the d.c. path. Thus, at most ONE D.C. path should exist between the Tuner-receiver/IRD and the selected LNB, to avoid overloading the master's power supply.

This demands that present versions of the slave microcontroller must start up into normal (power on) operating mode, so for DiSEqC™ level 1.0 it is not necessary to issue a “Standby Off” command.

However, for the longer term, and for SMATV applications, it is recommended that Tuner-receiver/IRDs should transmit appropriate “Standby On/Off” commands to reduce loading on the bus and unnecessary power consumption.

7.2. Bus Arbitration (DiSEqC™ Levels 2.x)

The preferred method for preventing contention on the bus during initialisation is to avoid having more than one device with the same family address “listening” to the bus at the same time. This can be achieved by using a “loop through” architecture, which is particularly relevant to switcher devices, but can include LNBs. In this architecture, one slave device connects directly to the bus (going to the master) and other slaves are then connected to the bus via the first device. At power-up (initialisation), the first device responds to the bus, but blocks the DiSEqC™ commands from propagating to the further devices (either specifically, or in conjunction with the normal IF signal switching). When the first device is being initialised, it can be allocated a unique address and is then commanded to switch the bus through to the next device in the chain. This process could be continued for several levels, if the master's control software is sufficiently sophisticated, but aspects of power management and response times will need careful consideration.

7.3. Bus Collision detection (DiSEqC™ Levels 2.x)

The single-master protocol and the limited number of slave devices expected on the bus should give a low risk of data collisions. However, to simplify the initialisation process, it is proposed to initially specify a relatively small number of different device families and device types. Also, because of the “mechanical” aspects of satellite position and plane of polarisation there may be legitimate occasions when slaves with identical addresses appear on the bus. A simple strategy for resolving conflicts is proposed:

Before a slave device replies to any command to its address (at least after power-up until completion of the system initialisation process) it should monitor the bus for any other slave responses, for a random time of between typically 15 ms and 115 ms. If it detects another DiSEqC™ reply message it should cancel its own attempt to reply and set an internal “Bus Contention” flag. Whilst this Contention flag remains set, the slave should either not reply at all, or wait for a period of typically 130 ms (i.e. slightly longer than the maximum random delay) and reply only if no other response is detected. This extended delay avoids any collisions with the reply from the slave which “won” the arbitration, but does not cause a slave to “disappear” from the system if the contention flag becomes set accidentally.

The master can detect the existence of the contention in various ways. It may be able to detect the extended response time from the slave, and/or issue a command which applies only to a device with its Contention flag set (e.g. the “return address” command ‘05’). Alternatively, the master can change the address of any slave known to it, and then test for any further responses. The “change address” commands are considered more useful than the “sleep” command defined in early specifications.

If the master does send a command to change the address of a slave, it should preferably check for a reply from the new address. To attempt to retain the identity of the conflicting devices after a power-down, the master should ideally interrogate the available “resource” registers of each device and store any distinguishing feature in its Non-Volatile Memory.

8. Definition of DiSEqC™ Bus Commands

The following tables list the currently-defined DiSEqC™ Bus Control Commands. Commands from the DiSEqC™ Master consist of 3 bytes, plus any ancillary data bytes, all followed by an odd parity check bit. Slave Reply messages consist of 1 byte, plus any ancillary data bytes, all followed by an odd parity check bit. The bits are transmitted as a continuous sequence until the message is complete.

Master Command:

FRAMING	P	ADDRESS	P	COMMAND	P	DATA	P
---------	---	---------	---	---------	---	------	---

Slave Reply:

FRAMING	P	DATA	P	DATA	P
---------	---	------	---	------	---

8.1. Framing Byte

The following framing bytes have so far been defined:

Hex Byte	Binary	Framing byte Function
E0	1110 0000	Command from Master, No reply required, First transmission
E1	1110 0001	Command from Master, No reply required, Repeated transmission
E2	1110 0010	Command from Master, Reply required, First transmission
E3	1110 0011	Command from Master, Reply required, Repeated transmission
E4	1110 0100	Reply from Slave, "OK", no errors <u>detected</u>
E5	1110 0101	Reply from Slave, Command not supported by slave
E6	1110 0110	Reply from Slave, Parity Error detected - Request repeat
E7	1110 0111	Reply from Slave, Command not recognised - Request repeat

8.2. Address Byte

The address byte is divided into two nibbles of four bits to define a family and sub-type:

FAMILY	SUB-TYPE
--------	----------

The following table lists the addresses which so far have been defined:

Hex. Byte	Binary	Family and Sub-type
00	0000 0000	Any Device (Master to all devices)
10	0001 0000	Any LNB, Switcher or SMATV (Master to all...)
11	0001 0001	LNB
12	0001 0010	LNB with Loop-through switching
14	0001 0100	Switcher (d.c. blocking)
15	0001 0101	Switcher with d.c. Loop-through
18	0001 1000	SMATV
20	0010 0000	Any Polariser (Master to all Polarisers)
21	0010 0001	Linear Polarisation (Skew) Controller
30	0011 0000	Any Positioner (Master to all Positioners)
31	0011 0001	Polar / Azimuth Positioner
32	0011 0010	Elevation Positioner
40	0100 0000	Any Installer Aid (Master to all Installer Aids)
41	0100 0001	Signal-strength analogue value
6x	0110 bbbb	Family reserved for address re-allocations
70	0111 0000	Any Intelligent Slave Interfaces (Master to all)
71	0111 0001	Interface for subscriber controlled headends
Fx	1111 bbbb	Reserved for OEM Extensions

8.3. Command Byte

The Command Bytes define the actions required of the addressed slave(s). The table below lists the commands which have so far been defined. The first column indicates the relative priority which should be placed on incorporating the commands in products at various DiSEqC™ Implementation Levels. ‘M’ indicates commands which must be included to qualify for the associated DiSEqC™ Implementation Level (and higher). ‘R’ Recommends commands for use at the stated Implementation Level, and ‘S’ Suggests commands which may be of value in the relatively short term.

The Level number has been added to the first column of the table for this version of the Bus Specification, but change bars are only shown beside the table where the commands themselves have been amended or added.

The final column defines the Reply data byte(s) which are expected to be received from an addressed slave in a “two-way” DiSEqC™ system.

Status / Level	Hex. Byte	Command Name	Command Function	Total trans. Bytes	Reply Data byte(s)
M*R/1.0	00	Reset	Reset DiSEqC™ microcontroller	3	
R / 2.0	01	Clr Reset	Clear the “Reset” flag	3	
R / 1.0	02	Standby	Switch peripheral power supply off	3	
R / 1.0	03	Power on	Switch peripheral power supply on	3	
S / 2.0	04	Set Contend	Set Contention flag	3	
S / 2.0	05	Contend	Return address only if Contention flag is set	3	Address
R / 2.0	06	Clr Contend	Clear Contention flag	3	
S / 2.0	07	Address	Return address unless Contention flag is set	3	Address
S / 2.0	08	Move C	Change address only if Contention flag is set	4	
R / 2.0	09	Move	Change address unless Contention flag is set	4	
	0F		Reserved		
R / 2.0	10	Status	Read Status register flags	3	Status
R / 2.0	11	Config	Read Configuration flags (peripheral hardware)	3	Config.
R / 2.0	14	Switch 0	Read Switching state flags (Committed port)	3	Switches
R / 2.0	15	Switch 1	Read Switching state flags (Uncommitted port)	3	Switches
	16	Switch 2	expansion option	3	
	17	Switch 3	expansion option	3	

Status / Level	Hex. Byte	Command Name	Command Function	Total trans. Bytes	Reply Data byte(s)
R / 1.0	20	Set Lo	Select the Low Local Oscillator frequency	3	
R / 1.0	21	Set VR	Select Vertical Polarisation (or Right circular)	3	
R / 1.0	22	Set Pos A	Select Satellite position A (or position C)	3	
R / 1.0	23	Set S0A	Select Switch Option A (e.g. positions A/B)	3	
R / 1.0	24	Set Hi	Select the High Local Oscillator frequency	3	
R / 1.0	25	Set HL	Select Horizontal Polarisation (or Left circular)	3	
R / 1.0	26	Set Pos B	Select Satellite position B (or position D)	3	
R / 1.0	27	Set S0B	Select Switch Option B (e.g. positions C/D)	3	
R / 1.1	28	Set S1A	Select switch S1 input A (deselect input B)	3	
R / 1.1	29	Set S2A	Select switch S2 input A (deselect input B)	3	
R / 1.1	2A	Set S3A	Select switch S3 input A (deselect input B)	3	
R / 1.1	2B	Set S4A	Select switch S4 input A (deselect input B)	3	
R / 1.1	2C	Set S1B	Select switch S1 input B (deselect input A)	3	
R / 1.1	2D	Set S2B	Select switch S2 input B (deselect input A)	3	
R / 1.1	2E	Set S3B	Select switch S3 input B (deselect input A)	3	
R / 1.1	2F	Set S4B	Select switch S4 input B (deselect input A)	3	
	30	Sleep	Ignore all bus commands except "Awake"	3	
	31	Awake	Respond to future bus commands normally	3	
M / 1.0	38	Write N0	Write to Port group 0 (Committed switches)	4	
M / 1.1	39	Write N1	Write to Port group 1 (Uncommitted switches)	4	
	3A	Write N2	expansion option	4	
	3B	Write N3	expansion option	4	
S / 2.0	40	Read A0	Read Analogue value A0	3	byte value
S / 2.0	41	Read A1	Read Analogue value A1	3	byte value
R / 1.2	48	Write A0	Write Analogue value A0 (e.g. Skew)	4	
S / 1.2	49	Write A1	Write Analogue value A1	4	
S / 2.0	50	LO string	Read current frequency [Reply = BCD string]	3	BCD bytes
R / 2.0	51	LO now	Read current frequency table entry number	3	F number
S / 2.0	52	LO Lo	Read Lo frequency table entry number	3	F number
S / 2.0	53	LO Hi	Read Hi frequency table entry number	3	F number

Status / Level	Hex. Byte	Command Name	Command Function	Total trans. Bytes	Reply Data byte(s)
M / 1.1	58	Write Freq	Write channel frequency (BCD string)	5 / 6	
	59	Ch. No.	Write (receiver's) selected channel number	5	
M / 1.2	60	Halt	Stop Positioner movement	3	
	61		Reserved		
	62		Reserved		
M / 1.2	63	Limits Off	Disable Limits	3	
R / 2.2	64	PosStat	Read Positioner Status Register	3	Pos Status
	65		Reserved		
M / 1.2	66	Limit E	Set East Limit (& Enable recommended)	3	
M / 1.2	67	Limit W	Set West Limit (& Enable recommended)	3	
M / 1.2	68	Drive East	Drive Motor East (with optional timeout/steps)	4	
M / 1.2	69	Drive West	Drive Motor West (with optional timeout/steps)	4	
M / 1.2	6A	Store nn	Store Satellite Position & Enable Limits	4	
M / 1.2	6B	Goto nn	Drive Motor to Satellite Position nn	4	
	6C		Reserved		
	6D		Reserved		
	6E	Goto x.x°	Drive Motor to Angular Position (°)	5	
R / 1.2	6F	Set Posns.	(Re-) Calculate Satellite Positions	(4) / 6	

M = Mandatory to qualify for the appropriate DiSEqC™ Level (and above).

R = Recommended for the stated DiSEqC™ Level (and above).

S = Suggested for the stated DiSEqC™ Level (and above).

M*R = In the case the where the Tuner-receiver/IRD has a loop-through facility then the Reset command is mandatory, in other cases it is Recommended.

9. Description of DiSEqC™ Bus Commands

The following subsections describe specific aspects of the commands defined in *section 8*.

Note that the sequence and numbering of these subsections have been re-arranged (compared with the previous DiSEqC™ bus specifications) so that the “one-way” Write (Level 1.x) commands are presented first. These are followed by the “two-way” Read commands (Level 2.x), where the Master reads data from the Slave.

9.1. Write Port Group

In addition to the specific command bytes (‘2x’) which set or clear individual switch states, there are commands which can control groups of 4 switching lines by employing an additional (data) byte. The new switching combination is defined by the data byte within the command, and is arranged such that any combination of individual switches can be either updated or left in their previous state. This is achieved by allowing the two separate nibbles of the data byte to determine the switches’ state in different ways. Any bits set in the high nibble CLEAR the corresponding switch control lines and any bits set in the low nibble SET the corresponding lines. To ensure predictable operation the high nibble should always effectively be applied first, followed immediately by the low nibble. Thus if a group of four lines is initially set to ‘WXYZ’ (where W, X, Y and Z are each an arbitrary binary value, ‘0’ or ‘1’) and a write data byte ‘0011 0101’ is sent, then the resulting switching lines become ‘W101’. To set up all four lines, it is usually most convenient to make the high nibble ‘1111’ and define the required switching state in the low nibble.

CLEAR	SET
data nibble	data nibble

Initial state	W X Y Z
CLEAR flags	0 0 1 1
Intermediate result	W X 0 0
SET flags	0 1 0 1
Result	W 1 0 1

9.2. Write Channel frequency

In some SMATV systems, the “tuning” function may be performed at the antenna end of the cable, so that whichever channel (broadcast signal) is selected, it is carried at the same frequency over the cable. The selected (broadcast) channel frequency may be requested in these “intelligent” systems by the following command format:

FRAMING	P	ADDRESS	P	COMMAND	P	DATA	P	DATA	P	DATA	P
---------	---	---------	---	---------	---	------	---	------	---	------	---

where typically the Framing byte is ‘E0’, the address byte ‘71’ and the command byte ‘58’. The required channel frequency is carried in the Data bytes in Binary Coded Decimal (BCD) format, starting with the 10’s and units GHz byte. Subsequent bytes contain decimal data until only trailing zeros remain. For example, the frequency 12.650 GHz would be transmitted in two Data bytes (with parity bits):

0001 0010	1	0110 0101	1
-----------	---	-----------	---

Further information may be found in application information such as the “*Update and Recommendations for Implementation Version 2.1*”.

Note to receiver manufacturers:

It is planned for future applications to allow the tuner in the receiver to be controlled via the DiSEqC™ bus, i.e. rather than take the frequency from the receiver’s internal look-up table, the tuner will read the frequency returned by the slave on the DiSEqC™ bus. This could be implemented by a set-up command, for instance in the installation menu of the receiver, which sets the tuner to either internal or external (bus) control. A useful extension of this concept might also allow all the programmable features of the receiver to be controlled by the bus so that one receiver can be directly programmed via an intelligent installation aid.

9.3. Positioner Operation and Commands

The Positioner Commands in this version of the Bus Specification have been optimised for “one-way” DiSEqC™ control applications. It is still Recommended that all Slaves (including Positioners) should support “Level 2.x” reply messages (e.g. “OK”, “Parity Error”, etc.) when requested, but these new commands do not demand the return of actual data values for the system to operate. Column 1 of the table in section 8.3 indicates which commands are Mandatory for inclusion in Tuner-receiver/IRDs for Implementation Level 1.2 and above.

It is recommended that all Positioner Motor Assemblies should protect themselves against damage when they reach internal or external physical limits of movement. Even when stalled, they must not take more than the maximum d.c. current rating from the DiSEqC™ Bus (presently 500 mA). In addition, they may optionally implement the “Soft Limits” described in this Specification. Soft Limits allow the user to restrict the arc of movement of the Positioner (e.g. the “Dish” Antenna) by commands sent over the DiSEqC™ Bus. When these Limits are “Disabled” then the Positioner motor still drives over the full arc, but once the limits are “Enabled” then the motor stops when reaching the positions which were stored by the East and West Limit commands.

The command bytes for the Positioner Motor to Drive East (‘68’) and Drive West (‘69’) include a timing parameter. The default value for this parameter is ‘00’ for which the Positioner Motor drives continuously in the required direction. Positive (non-zero) values of the parameter (i.e. ‘01’ to ‘7Fh’) indicate a “timeout” value in seconds, but Positioner Motors not supporting a timing function should drive continuously for these parameter values. Master controllers (Tuner-receiver/IRDs, etc.) must still transmit a “Stop” command and not rely on the timeout alone. Negative parameter values (i.e. ‘80h’ to ‘FFh’) indicate the number of “steps” the motor should make, and Positioner Motors should only drive if they support this movement process. This is described in the “*Positioner Application Note Version 1.0*”, at present in preparation.

Satellite Positions are defined by positive binary number parameters from 1 (‘01’) upwards. Thus the command byte / parameter ‘6A 01’ Stores the present motor position as “Position 1” and command byte / parameter ‘6B 01’ causes the motor to drive back to this stored position at any later time. The parameter ‘00’ is reserved as a “reference position”, and all Masters should be capable of transmitting this parameter with the above commands, when required. The “Store Position 00” command can be used just to “Enable Limits” and the “Goto Position 00” command can cause the motor to drive to a “Reference Position” which will normally correspond to due South, but may be the easterly “Hard Limit” (i.e. defined by physical switching within the motor unit) if so implemented in the Positioner Motor Unit.

Command byte ‘6F’ with various data parameters can give the user/installer various types of “short-cut” to setting up or aligning the system. The basic command has a (first) parameter byte ‘00’ to initiate an appropriate function, defined by the manufacturer of the Positioner Motor Unit. However, it is preferred that Masters (Tuner-receiver/IRDs) with Menu systems should support 3 (signed) binary parameter bytes. These typically may be used for, “Satellite Number”, “X-Value” (e.g. Site Longitude) and “Y-Value” (e.g. Site Latitude).

9.4. Write Polariser Skew

Polarisation Skew is an “analogue” value carried to any type of Polariser in the address family ‘2x’, by the command byte ‘48’. This command carries a single data byte and so the analogue value can be resolved to one part in 256. The numerical value may be used to give at least a full 90 degree rotation of the plane of polarisation of the received signal, or it may be used just to trim the LNB polarisation axis to the alignment of each satellite. Therefore, the polarisation flag (H/V) should always be transmitted in an associated message (usually command ‘38 xx’) to the Polarisation Skew message.

9.5. Read Positioner Status Byte (Level 2.x)

The Positioner Status Byte is returned in response to command byte ‘64’ and contains individual flag bits to indicate the operational conditions of the slave software and peripheral hardware related to dish Positioner functions.

Bit Number	Positioner Function
.7	Movement command has been completed
.6	Software Limits are Enabled
.5	Movement direction is (or last was) West
.4	Motor is running
.3	Software Limit (end-stop) has been reached
.2	Power is not available
.1	Hardware switch (Limit or Reference) is activated
.0	Position reference data has been lost or corrupted

9.6. Read Local Oscillator Frequency (Level 2.x)

The Local Oscillator frequency is returned in response to command bytes of the form '5x'. It is normally returned as a single reply byte which represents the entry number in the following list. The list may be added to in the future up to entry number 239. Should the need arise for further expansion then the values 240 to 255 represent pointers to up to 16 auxiliary tables of 256 entries each. An additional byte will indicate the entry number in the appropriate auxiliary table.

Frequency list number byte	Local Oscillator Frequency	L.O. Offset (relative to carrier)
0	None (switcher)	
1	Not known	
2	9.750 GHz	Low
3	10.000 GHz	Low
4	10.600 GHz	Low
5	10.750 GHz	Low
6	11.000 GHz	Low
7	11.250 GHz	Low
8	11.475 GHz	Low
9	20.250 GHz	Low
10	5.150 GHz	
11	1.585 GHz	
12	13.850 GHz	High
13	Not allocated	
14	Not allocated	
15	Not allocated	

To provide backwards compatibility with first generation DiSEqC™ systems, future additions to the frequency list should also be supported in packed BCD string form. In response to the master's request for the frequency string, by the command byte '50', the slave returns sufficient data bytes to define the frequency. The high nibble of the first data byte contains the tens of GHz digit and the low nibble contains the GHz digit in BCD form. Subsequent bytes contain BCD nibbles in descending order until only trailing zeros remain. For example 13.850 GHz would be returned in two bytes (with parity bits):

0001 0011	0	1000 0101	0
-----------	---	-----------	---

9.7. Read Status Byte (Level 2.x)

The Status Byte is returned in reply to command byte '10' and contains individual flag bits to indicate the operational conditions of the slave software and the peripheral hardware. The flag bits are allocated from high to low weighting as follows:

Bit Number	Status Function
.7	Bus-Contention flag is set
.6	Standby mode is selected
.5	
.4	Auxiliary power is currently available
.3	
.2	Bus voltage is above 15 volts threshold
.1	
.0	A reset has occurred since this flag was last cleared

9.8. Read Configuration Byte (Level 2.x)

The Configuration Byte is returned in reply to command byte '11' and contains flags as defined in the following table:

Bit Number	Configuration Function
.7	Device has Analogue output facility
.6	Device has Standby (power-down) facility
.5	Device has Positioner capability
.4	Device has external power detection capability
.3	Device has loop-through facilities
.2	Device has Polariser capability
.1	Device has Switcher capability (including LNBs)
.0	Device can supply LO frequency value(s)

9.9. Read Committed Switches Byte (Level 2.x)

The Committed Switches byte is returned in reply to command byte '14'. The low nibble contains flags to indicate whether each corresponding named signal is controllable, and the high nibble indicates the current switched state. The byte thus contains flags as follows:

Bit Number	Committed switches state
.7	Options switch position 'B' is selected
.6	Satellite Position 'B' is selected
.5	Horizontal Polarisation is selected
.4	High Local Oscillator is selected
.3	Options Switch is available
.2	Two (or more) satellites are switchable
.1	Linear Polarisation is switchable
.0	Local Oscillator is switchable

9.10. Read Uncommitted Switches Byte (Level 2.x)

The Uncommitted Switches byte is returned in reply to command byte '15'. The low nibble contains flags to indicate whether each corresponding switch is controllable, and the high nibble indicates the current switched state. The byte thus contains flags as follows:

Bit Number	Uncommitted switches state
.7	State of Uncommitted switch 4
.6	State of Uncommitted switch 3
.5	State of Uncommitted switch 2
.4	State of Uncommitted switch 1
.3	Uncommitted switch 4 is available
.2	Uncommitted switch 3 is available
.1	Uncommitted switch 2 is available
.0	Uncommitted switch 1 is available