

Radical Electronics RS485 Communication Protocols
Version 0.0 PRELIMINARY
March 17, 2001

Contents

- 1.0 What is RS485?
- 1.1 System Description
- 1.2 Definitions
- 2.0 Arbitration (Level 1 hardware)
- 3.0 Bus Commands (Level 2 software)
 - 3.1 ENUMERATE
 - 3.2 BLOCK
 - 3.3 SELECT
 - 3.4 MUTE ON/OFF/ALL/NONE
 - 3.5 INTERRUPT ON/OFF/ALL/NONE
 - 3.6 CRC ON/OFF/ALL/NONE
 - 3.7 RESEND
 - 3.8 MASTER

1.0 WHAT IS RS485

RS485 is a specification for a multidrop bus using differential signaling over a twisted pair of wire. The multidrop bus means that it can have a number of units hooked up to the bus at any time. This can be a combination of talkers and listeners. RS485 defines the electrical specifications for this bus. This application note is intended to further define how this bus is used in equipment manufactured by Radical Electronics (referred to as "the equipment").

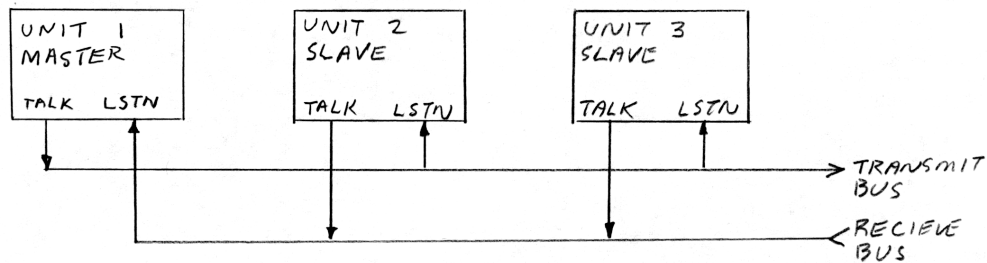
1.1 SYSTEM DESCRIPTION

In the bus setup used by the equipment there are 2 busses. The transmit bus is used for the bus master to communicate to the slave equipment. The receive bus is used for the slave equipment to communicate with the master. There is an arbitration system in place on some of the equipment to allow the use of scripting and interrupts.

A drawing of the typical setup is shown in figure 1.0.

The communication standard for the equipment is 8 bits, no parity, 1 stop bit. Multirate communication is not supported.

FIGURE 1.0 TYPICAL SETUP



1.2 DEFINITIONS

INTERRUPT - This is a predefined event that will cause a script to be run or a device to become a talker to send a message back to the master on the receive bus. This is defined by the unit specific commands sent to the slave unit.

MASTER - This is the unit that has control of the bus or busses. It determines which unit will become a talker and when it will become a talker. The most common application will have a PC or other piece of equipment defined as the master. There is only one master on the bus at one time. The duties of master may be switched from unit to unit (this is rare but can be used to allow units to talk to each other)

SLAVE - This unit has no control over the busses. It talks when it is instructed to talk or an interrupt occurs.

TALKER - This is the unit that is transmitting the information on the bus. On the transmit bus it is usually the master that is the talker. On the receive bus it is the slave units that are the talkers.

LISTENER - This is the unit that is receiving information off the bus. In most situations, the transmit bus will have the master as the talker, sending out commands to the other devices on the bus. The other devices on the bus are listeners. On the receive bus, usually it is the other way around. The slave devices (talkers) send their responses back to the master(listener) on this bus.

SCRIPTING - This is when a series of commands are sent to the unit that are executed upon a single command. This is useful if there is particular action that needs to be taken on an event. It can be also be used to simplify the communication between the talker and listener. An example of a script program usage could be a voltmeter and a switch. When the voltage reaches a certain point, the voltmeter can be instructed to become the master of the bus and send a command to the switch to turn on. More information on the scripting language is available in the application note scripting language.

2.0 BUS ARBITRATION

Due to the nature of the bus with multiple talkers and listeners on the bus, there has to be some sort of control over the bus to ensure that the messages from each talker and listener get through the bus without getting corrupted by other devices wanting to talk on the bus.

In the case of no arbitration, the equipment doesn't do anything to prevent the messages from two talkers from getting mixed up on the same bus. In this case, it is up to the master to wait for the messages from the talker finish before requesting information from another slave (and allowing it to talk). This method is adequate for a command/response type of system but it is not sophisticated enough to allow interrupts.

In the equipment that has arbitration, there are 2 levels of arbitration. The first level of arbitration consists of the first unit on the bus gets to send its message. The other one has to wait. The bus is considered to be busy on the rising edge of the first start bit until 2 byte times after the final high level signal. During this time no unit can become a talker. This is shown in figure 2.0.

Unfortunately, due to propagation delays and rise times, it is possible for 2 units to think that the bus is free and start talking. By the time the signals from one unit reach the other, they have both started to talk. In this case, both the units continue to talk until one of the units receives that it has an error (1 was transmitted but a 0 seen on the bus or a 0 transmitted and a 1 received) occurs. When the error occurs, the device in error drops out leaving the other unit talking. The unit in error then tries to transmit when the bus becomes free again. This condition is shown in figure 2.1.

Even this still has problems. Depending on line resistance and loading, it is possible for the 2 units not to detect an error as seen by the listener. In this case, the CRC error correction can be used to detect problems. In extreme cases, the listener will only hear one of the talkers and the other talker doesn't know it is not being heard. In this case the message may be lost.

The bus arbitration works best with short drop lengths. Long drop lengths will lead to more contentions due to resistance and delays.

FIGURE 2.0 FIRST START

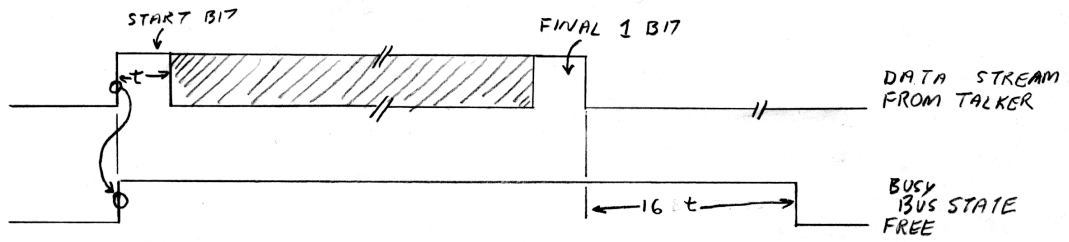
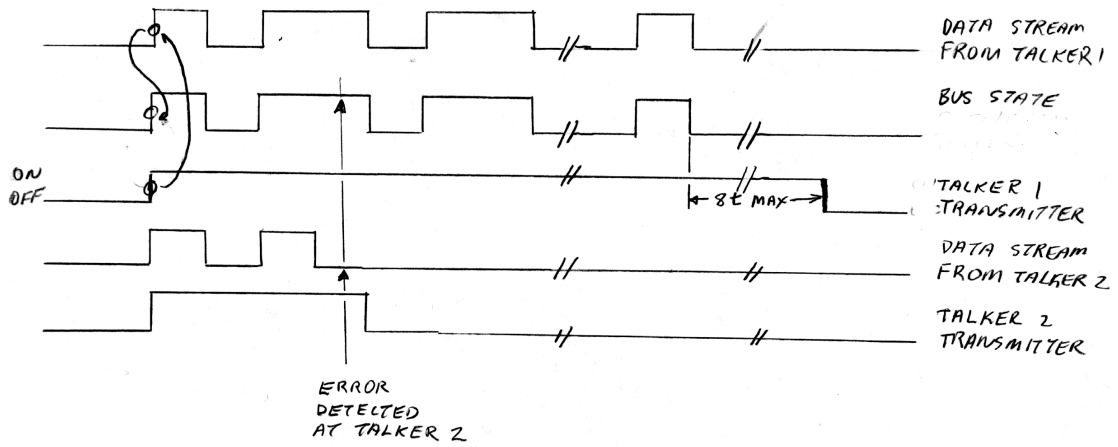


FIGURE 2.1 2 TALKERS STARTING



3.0 BUS COMMANDS

These commands are common to all RS485 products. They are used to control the state of the bus and to determine what response (if any) is to be returned from the remote unit to the master. Some of the commands do not have any effect on units that do not have flow control and they should not generate errors when they are received. Non printable ASCII characters are in brackets.

3.1 ENUMERATE

This command is used to identify the devices on the bus. Upon receipt of this command, the remote units send back the model identification and address of the unit. This can be useful when the user wants to check what equipment is hooked to the bus. If the function is associated with a particular address, it can be used to identify what functions can be carried out. The return order may change from query to query depending on how the bus contentions. The unit does not need to be selected for this command to be in effect. The responses from the units will be staggered in time depending on the model and address of the unit. The time will be counted in bit times the bus is free. The time is found using

$$\text{delay} = \text{model delay} * 16 + \text{address}$$

For example:

2 units on the bus. One has a model delay of 120 with an address of 10. The second has a model delay of 30 with an address of 0.

Once the bus has been free for 480 bit times, the second unit transmit its information back to the master. During the time of the transmission, the first unit will not be counting until the bus is free. After the second unit has finished its transmission, the first unit will resume counting until the bus has been free for 1930 bit times. At that time, it will transmit its information.

The devices will return the information in the following format
string "MODEL"
1 space [32]
the device model identifier string
a comma ","
string "UNIT"
1 space [32]
the devices address in ASCII
a carriage return [13]

Example

Sent on transmit bus

ENUMERATE[13]

Data returned

MODEL HFS13, UNIT 16[13]

MODEL HFS13, UNIT 3[13]

MODEL HPS10, UNIT 16[13]

In the example, there are 3 units on the RS485 bus. Two are model HFS13 units and one is HPS10.

IMPORTANT: This command is only implemented on the devices with flow control.

3.2 BLOCK

This command is used to indicate to the devices on the bus that a block of data is going to be sent down the RS485 bus that doesn't conform to any format. Any commands that are sent down the bus during this transfer are ignored. The only listener is the unit that is selected and no other units will be selected until the transmission is complete.

The format for this command is:
The device command that needs the data
A carriage return [13]
The BLOCK command
A space
The number of bytes in the block
A carriage return
The bytes for the block

Example

```
LOAD[13]  
BLOCK 1287[13]  
[1287 bytes of data]
```

In the example the device has a command called LOAD that requires a large chunk of data. Instead of trying to load it in ASCII form after the load command, the BLOCK command sends it over in a large binary block.

3.3 SELECT

This command is used to select the various devices on the RS485 bus. When one device on the bus is selected, it will automatically de select all the other devices on the bus. The selected device will send back an acknowledge to indicate that it has received the command and is now selected. The unit will not send back an acknowledge if the mute has been turned on for that device (using MUTE or MUTE ALL commends)

The format for this command is as follows:

```
string "SELECT"  
space  
string"MODEL"  
space  
the model identification string  
a comma ","  
string "UNIT"  
space  
the unit number in ASCII  
Carriage return [13]
```

Example:

Sent data

```
SELECT MODEL HFS13, UNIT 16[13]
```

return data from selected unit

```
ACKNOWLEDGE[13]
```

3.4 MUTE

This command is used to silence the talkers on the return bus. This can be used for compatibility with other equipment as well as using the return bus for interrupts when there is equipment with no arbitration on the bus.

There are 4 options for this command.

ON - Enables talking on the selected unit only.

OFF - Disables talking on the selected unit only.

ALL - Enables talking on all units.

NONE - Disables talking on all units.

The command format is as follows:

string "MUTE"

a space

option string "ON", "OFF", "ALL", "NONE"

a carriage return

This command does not affect the talkers in the case of an interrupt. None of these commands generate a response.

The default state of all talkers is OFF.

3.5 INTERRUPT

This command is used to enable an event triggered message from a device. There are usually setup commands for a device to determine the conditions of the trigger. This command enables the talker to transmit the message.

There are 4 options for this command.

ON - Allows the talker on the selected unit to send out interrupt generated messages.

OFF - Disallows the talker on the selected unit to send out interrupt generated messages.

ALL - Allows the talkers on all of the selected units to send out interrupt generated messages.

NONE - Disallows the talkers on all of the selected units to send out interrupt generated messages.

This command returns an ACKNOWLEDGE after the ON and OFF options. The ALL and NONE options do not return any message. If arbitration is not available, the message ERROR NO ARBITRATION is returned after the ON or OFF options.

When the interrupt occurs, the following information is returned:

string "INTERRUPT"

space

string "MODEL"

space

An ASCII string identifying the model of the device

a comma and space

string "UNIT"

space

An ASCII string identifying the address of the device

a carriage return

There is the option for more lines of information to identify the condition that caused the interrupt.

The format for this command is:

string "INTERRUPT"

space

string for option "ON", "OFF", "ALL", "NONE"

a carriage return

Example:

sent data

SELECT MODEL DVM01, UNIT 14[13]

SETMAXLIMIT 40V[13]

INTERRUPT ON[13]

return data

ACKNOWLEDGE

When the conditions for the interrupt occur the following message is returned:

INTERRUPT MODEL DVM, UNIT 14[13]

In the sent data, the following functions are performed.

The first line selects the unit

The second line sets the conditions of the interrupt

The third line enables the interrupt.

The default setting is INTERRUPT OFF.

3.6 CRC

This command enables or disables error checking on the received and transmitted data. This is used on the return bus for identifying errors caused by bus contentions or cable reflections. On the transmit bus, it is used for identifying errors caused by cable reflections.

When CRC is enabled, a CRC byte is inserted at the end of each line after the carriage return. CRC bytes are not inserted into blocks of data defined by the BLOCK command.

This command has 4 options:

ON - Enables the CRC error checking for the selected device.

OFF - Disables the CRC error checking for the selected device.

ALL - Enables the CRC error checking for all of the devices on the bus.

NONE - Disables the CRC error checking for all devices on the bus.

No messages are returned after the ALL or NONE options. The ACKNOWLEDGE response is sent back after the ON or OFF messages if the device has CRC error checking built in otherwise the error "ERROR NO CRC" is returned.

Subsequent commands require the CRC value to be correct before the command can be executed. The default value for this is CRC OFF.

When the CRC check is enabled, the talker sends the message with the CRC byte appended to the end. The listener then sends back a single byte acknowledge[TBD] or error[TBD] byte. If an error has occurred, the message is resent. There is no CRC check on the return value from the listener.

All remote devices are to wait for an acknowledge byte sent back from the master before attempting to assert control over the bus.

3.7 MASTER [DEFAULT]

This command allows the selected unit to be made the master of the bus. Most equipment doesn't have the need or ability to master the bus. If this command is sent to one of those devices it will return "ERROR UNABLE TO MASTER".

If the command is accepted the message "ACKNOWLEDGE" will be issued.

If the option "DEFAULT" follows the command then the power up state master will take control of the bus as the master again. The option DEFAULT is not required.

The function of this is the system can have a backup master for the bus in case of failure, and it permits a unit to take control of the bus to transfer data from unit to unit.

Only units that are capable of scripting are capable of mastering.