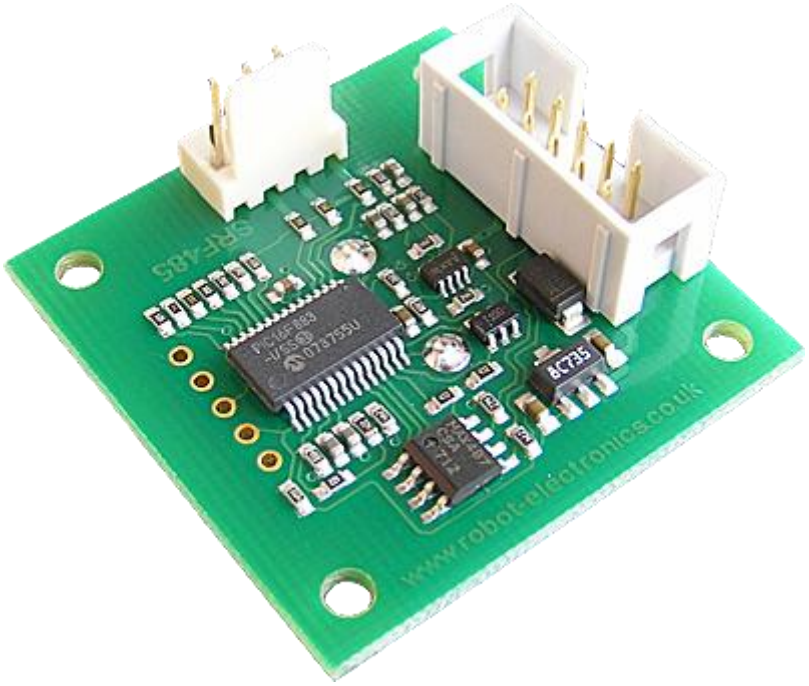
SRF485 Ultrasonic range finder

Technical Specification



Specification

Voltage -	12vdc (8vdc - 14vdc)
Current -	10mA
Range -	30cm - 5meters
Temperature Compensation -	-30 to +50 centigrade
Size -	38mm x 38mm (1.5" x 1.5")
Mounting -	30.5mm x 30.5mm (1.2" x 1.2")
Transducer -	16mm dia. centrally mounted.

Overview

The SRF485 is a single transducer ultrasonic ranger, based on our original SRF02 design. The SRF485 uses the RS485 standard for communications, for up to 127 modules on each RS485 bus. The SRF485 features an on-board 5v power regulator which can be supplied from 8vdc to 14vdc. Both power and RS485 signals are supplied to the module via a 10pin IDC connector, making cabling a large number of modules very easy. The Range of the SRF485 is 30cm to 5meters.

RS485 Communication

The SRF485 modules are connected using the RS485 bus. Up to 128 transceivers can be connected to the same bus, so you can have up to 127 SRF485 modules plus the controller.

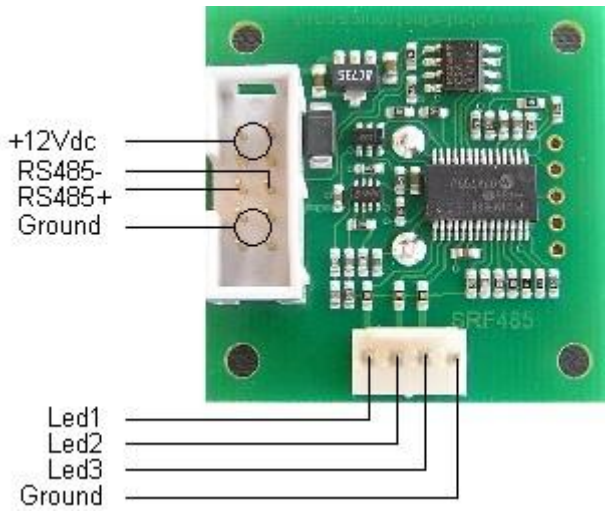
Serial data is fixed at 38400 baud 1 start, 2 stop and no parity bits. Control of the SRF485's is by sending frames of data to the module and then listening for the response. Each SRF485 has a unique 24-bit address which we program in during manufacture and is written on the module. The data frame you send to the SRF485 is:

Break	Command	AddressH	AddressM	AddressL	Data	Check Sum

Break - is defined as a continuous low in excess of 22 bit periods, followed by a high of 2 bit periods. Each bit is 26uS at 38400 baud, so 22 * 26uS = 572uS, it's ok to be longer.
Command - is one of a number of commands that the SRF485 will respond to. See below for details.
AddressH,M,L - is the 24-bit address of the module you wish to communicate with.
Data - Is the data you wish to send to the module, zero (0x00) if nothing is required by the command.
Chksum - is the 1's compliment (bitwise negation) of the sum of all the previous bytes (not counting the break).

The module will respond with a variable number of bytes, 0 to 4 depending on the command, but the transmit frame is always the same, a break followed by 6 bytes. See Commands section below for details.

Connections
The connections to the SRF485 are shown below. There are 4 pins for the 12vdc supply, these are connected together on the PCB. The two pins in the middle are the RS485 signals, + is nearest the edge of the PCB and - is on the inside. The last 4 pins are 0v Ground and are connected together on the PCB. There is a 4 pin connector which is designed to connect Led's. The Anode is connected to the Led(x) pin and the Cathode to ground. There is a 390 ohm resistor for each Led already on the PCB.



Commands
To send a command to the SRF485, you need to send a data frame as described above. There are three commands to initiate a ranging (80 to 82), to produce the result in inches, centimeters or microseconds. These three commands don't Transmit the result back to your controller. You should wait 70mS and then use command 94 to get the result of the ranging. Another set of three commands (83 to 85) do the same, but also transmits the temperature

compensated result of the ranging back to your controller as soon as it is available. Together, these six commands (80 - 85) are called "Real" because they perform a complete ranging. There is another set of six commands (86 - 91) called "Fake". They are the same as the "Real" commands except that they do not send the 8-cycle burst out. These are used where the burst has been transmitted by another sonar. It is up to you to synchronize the commands to the two sonar's. There is a command (92) to transmit a burst without doing the ranging.
Command 93 is used to get the SRF485 version. This command will return 4 bytes:
Byte1 is the module type, 0x01 for the SRF485.
Byte2 is the Hardware Version, 0x03 for the current PCB.
Byte3 is the Software Version, currently 10.
Byte4 is the Modules Group number, this will be in the range 0x00 to 0x7F (0-127).
Command 94 returns two bytes (high byte first) from the most recent ranging. Put them together to make a 16-bit result.

Command		Bytes Returned	Action
Decimal	Hex		
80	0x50	0	Real Ranging Mode - Result in inches
81	0x51	0	Real Ranging Mode - Result in centimeters
82	0x52	0	Real Ranging Mode - Result in micro-seconds
83	0x53	2	Real Ranging Mode - Result in inches, automatically Tx range back to controller as soon as ranging is complete.
84	0x54	2	Real Ranging Mode - Result in centimeters, automatically Tx range back to controller as soon as ranging is complete.
85	0x55	2	Real Ranging Mode - Result in micro-seconds, automatically Tx range back to controller as soon as ranging is complete.
86	0x56	0	Fake Ranging Mode - Result in inches
87	0x57	0	Fake Ranging Mode - Result in centimeters
88	0x58	0	Fake Ranging Mode - Result in micro-seconds
89	0x59	2	Fake Ranging Mode - Result in inches, automatically Tx range back to controller as soon as ranging is complete.
90	0x5A	2	Fake Ranging Mode - Result in centimeters, automatically Tx range back to controller as soon as ranging is complete.
91	0x5B	2	Fake Ranging Mode - Result in micro-seconds, automatically Tx range back to controller as soon as ranging is complete.
92	0x5C	0	Transmit an 8 cycle 40khz burst - no ranging takes place
93	0x5D	4	Get Version - sends 4 bytes back to the controller
94	0x5E	2	Get Uncompensated Range, returns two bytes (high byte first) from the most recent ranging.
100	0x64	1	Set Led's - Bit 0 controls Led1 - bit set and the Led is on. Bit 1 controls Led2. Bit 2 controls Led3
101	0x65	0	Set Search Mode - Used to search for SRF485's on the bus
102	0x66	0 or 1	Less Than - Used to search for SRF485's on the bus
103	0x67	0	Set Group - Sets the group number (0-127) that this sonar belongs to.
104	0x68	2	Get Temperature - returns the temperature in degrees centigrade, as a 16bit signed word,

			high byte first
105	0x69	2	Get temperature compensated range, returns two bytes (high byte first) from the most recent ranging.

Here is an example of sending a ranging command to a SRF485 located at address 0189AB.

Break	Command	AddressH	AddressM	AddressL	Data	Check Sum
600uS Low 50us High	0x51	0x01	0x89	0xAB	0x00	0x79

The check Sum is calculated as the low byte of ~(0x51+0x01+0x89+0xAB+0x00).
0x51+0x01+0x89+0xAB+0x00 = 0x0186
The bitwise not of 0x0186 is 0xFE79
We use the low byte of that, 0x79
The above command will start the SRF485 at 0x0189AB ranging in centimeters. After 70mS the result will be available. You should then use the GET RANGE command (0x5E) to retrieve the result.

Here is another example, this time to set Led1 on, and Led's 2,3 off.

Break	Command	AddressH	AddressM	AddressL	Data	Check Sum
600uS Low 50us High	0x64	0x01	0x89	0xAB	0x01	0x65

The SET LEDS command will return a single byte, 0x01, as an acknowledge.

Addressing
Each SRF485 has a unique 24-bit address programmed into it during manufacture. This address is written on the module, and can also be found by doing a search, see below. There are a number of addresses that selected commands will respond to. These are:
Its own unique address, an SRF485 will respond to most commands to its own address.
Address 0x000000, used where you need all SRF485's to respond to a command, such as start ranging, or setting search mode.
Address 0x000001, used where you need all SRF485's belonging to a specified group to respond to a command, such as start ranging,
Every Address, only the LESS_THAN command will respond to this, and only if its internal address is less than the one you send.

Groups
You can ignore the groups feature if you do not require it.
Each SRF485 sonar can belong to a "group". The purpose is to allow a selected range (or group) of SRF485's to all start ranging at the same time. It is similar to sending a ranging command to address zero, which causes all SRF485's to start ranging. A potential problem with this is if the SRF485's are too close together, they may interfere with each other. By arranging alternate SRF485's to be in a different group, you can have, say, units 1,3,5,7 etc ranging, and next time units 2,4,6,8 ranging.
Even if your SRF485's are not close enough to interfere, groups have the advantage of allowing you to do the ranging on one group, whilst getting the ranges from the previous

group, which makes more effective use of the busses bandwidth. To set the group number, send the SET_GROUP command (0x67) to the SRF485's actual address. The group number is stored in EEPROM, and only has to be sent once. This example sets the group number of the sonar at address 0x0189AB to 0x01.

Break	Command	AddressH	AddressM	AddressL	Data	Check Sum
600uS Low 50us High	0x67	0x01	0x89	0xAB	0x01	0x62

To start all SRF485's in group 0x01 ranging, send your chosen ranging command to address 0x000001 with the group number in the data byte, like this:

Break	Command	AddressH	AddressM	AddressL	Data	Check Sum
600uS Low 50us High	0x51	0x00	0x00	0x01	0x01	0xAC

The default group number is 0x00, so I would recommend that you avoid using that one in your application.

Searching the RS485 bus for Sonar's

The SET_SEARCH command (0x65) is used to place all SRF485's on the bus into a search mode, and must be used at address zero (0x000000) to do that. Only SRF485's that are in search mode will respond to the LESS_THAN command. The following will place all SRF485's into "search mode".

Break	Command	AddressH	AddressM	AddressL	Data	Check Sum
600uS Low 50us High	0x65	0x00	0x00	0x00	0x00	0x9A

The LESS THAN command is used to search for and find the addresses of all SRF485's on the bus. Every SRF485 whose unique internal address is less than the address supplied with the command will respond by sending back a single byte (0x00). They do this all at the same time, so you will just receive a single byte. Those SRF485's whose address is equal to, or greater than the supplied address will not respond. For example if you send this command frame:

Break	Command	AddressH	AddressM	AddressL	Data	Check Sum
600uS Low 50us High	0x66	0x80	0x00	0x00	0x00	0x19

Then all SRF485's whose internal address is in the range 0x000000 to 0x7FFFFF will send back the response byte. The value of the response byte is 0x00, but that is irrelevant. It is the presence or absence of a response that is important. The response, if there is one, will be immediate so if you have not received anything after 2mS, then you're not going to. Make

sure you time out in your receive routine.
So how do you use this to find the addresses of all SRF485's on the bus?
First, the SET_SEARCH command must be sent to address 0x000000 so that all SRF485's are placed into search mode.
This is where it gets a little harder to follow:
Send the LESS_THAN command with address 0x800000. If you get a response then there are one or more SRF485's whose address is between 0x000000 and 0x7FFFFF. Now you can narrow it down further by sending the LESS_THAN command with address 0x400000. If you get a response then you know there are one or more SRF485's between addresses 0x000000 and 0x3FFFFF. If you didn't get a response then the address would be between 0x400000 and 0x7FFFFF. If there was a response, do it again with address 0x200000. If not then address 0x600000. This is similar to the way an Analog to Digital converter works, by successive approximation. Using this technique any address can be found in 24 steps. The SRF485 found will be the one with the lowest address. Once found you should read the version of that SRF485 using the GET_VER command. This will reset the search mode for that SRF485, so it will take no further part in the search process.
Now you can repeat the process to find the next SRF485, and so on...

It's a lot harder to say than it is to do, the code is actually quite simple. Here is the core routine to do the job, written in c. It assumes the SET_SEARCH command has already been sent.

```
long Find_SRF485(void)
{
long Walk = 0x800000;              // used as a bit walker to calc next address to search
long Addr = 0x800000;              // start by looking for 0-7fffff range
char x;

  do {
    SendFrame(LESS_THAN, Addr, 0); // All SRF485's whose address is less than Addr,
will respond
    x = serin485();              // this times out after 1mS and returns 0xFF
    if(x==0) Addr ^= Walk;         // Must be greater or equal to Walk, so merge
    Walk >>= 1;                  // shift the bit walker
    Addr |= Walk;                // and generate next test address
  }
  while(Walk);                   // repeat until the full address of SRF485 is found

  SendFrame(GET_VER, Addr, 0);      // Get the version of this SRF485 to reset its search
mode.
  serin485();                    // module id - just get and discard these.
  serin485();                    // hardware version
  serin485();                    // software version
  serin485();                    // group number
  return Addr;                   // Return the SRF485's address, it will be 0xFFFFFF if none
found
}
```

We also have an example for the PC written in VC++ V6 here. You will need our USB-RS485 module to use it.