ISO18000-3 mode 2 RFID module

Developer's guide

Version	Date	By	Comment
01	April 07	G Price	Original Issue
02	December 07	G Price	Mechanical diagram updated to V6 hardware. RF field control added. Change baud rate command added.

FCC Statement

This device complies with Part 15 of the FCC Rules. Operation is subject to the following two conditions: (1) this device may not cause harmful interference, and (2) this device must accept any interference received, including interference that may cause undesired operation.

Electrical data

Pinout.

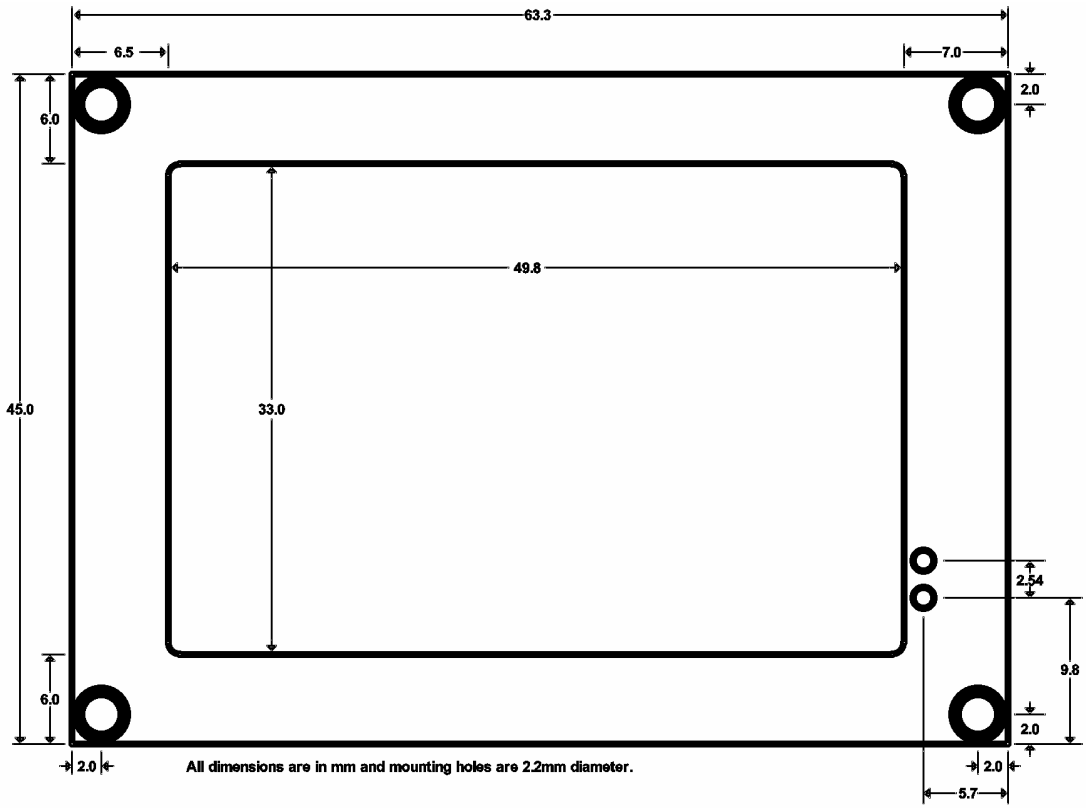
Pin	Description
1	Ground
2	+5V @ 150mA
3	Enable
4	3.3V (5V tolerant) TTL serial receive
5	3.3V (5V tolerant) TTL serial transmit
6	Antenna RF output
7	Antenna Ground

Current requirements

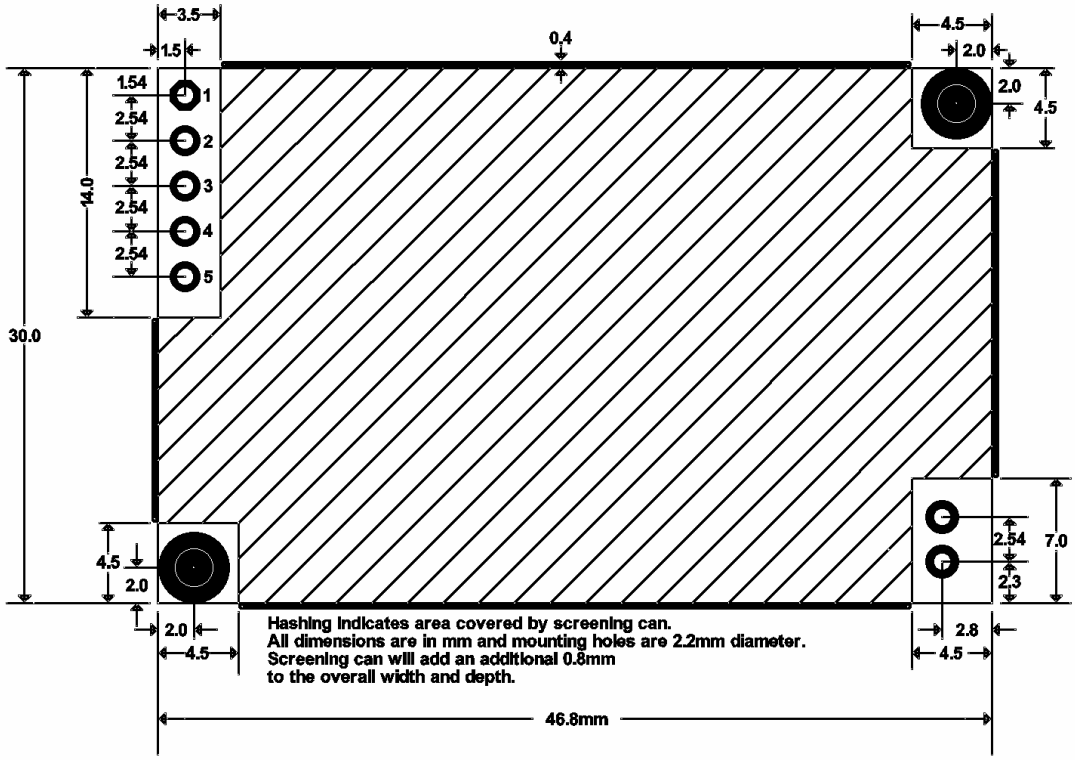
RF on	150mA
RF off	15mA
Shutdown	1.3mA

V6 Hardware mechanical diagram

Antenna



Module



Command structure

All data flow between the module and host is framed within packets. The packet structure is the same for data flow from the host as for data flow from the module and is shown below.

HDR	0xFF	Header, indicates the start of the packet
LEN	AA	Length, number of bytes that are to follow as a part of this packet
$Data_1$	0xXX	Data bytes to be transferred.
…	…	
$Data_{AA}$	0xXX	

It consists of a defined (0xFF) header, a length byte which is the count of all data bytes that are to follow, and then all the data bytes themselves.
Therefore if the data we wanted to send is 0x01, 0x02, 0x03, we would create the packet “0xFF, 0x03, 0x01, 0x02, 0x03”.
For multi-byte transfers i.e two byte words, the low byte is transferred first.
When transmitting a packet, the inter-byte delay must not exceed 500ms else this current packet will be abandoned and the complete packet must be resent.
Currently the baud rate is fixed at 9600.

Commands

Scan for a single tag

Returns the ID of a single tag, to be used when no more than one tag is in the field.

HDR	0xFF	
LEN	0x01	
CMD	0x01	

Successful operation – Tag detected

HDR	0xFF	
LEN	0x05	
CMD	0x01	
ID0		LSByte of unique ID
ID1		
ID2		
ID3		MSByte of unique ID

Unsuccessful operation - No tag detected

HDR	0xFF	
LEN	0x02	
CMD	0x01	
ERR	0x00	

Scan for multiple tags

Returns the IDs of multiple tags, use when more than one tag is expected in the field.

HDR	0xFF	
LEN	0x02	
CMD	0x02	
Data	AA	Maximum number of tags to detect, (0< AA <9)

Successful operation - Tags detected

HDR	0xFF	
LEN	BB	= (No of tags detected * 4) +1
CMD	0x02	
$ID1_0$		LSByte of unique ID, tag 1
$ID1_1$		
$ID1_2$		
$ID1_3$		MSByte of unique ID, tag 1
IDN_0		LSByte of unique ID, tag N
IDN_1		
IDN_2		
IDN_3		MSByte of unique ID, tag N

Unsuccessful operation - No tags detected

HDR	0xFF	
LEN	0x02	
CMD	0x02	
ERR	0x00	

Get tag's setup configuration

Returns the Hardcode, Time stamp, Lock pointer, Manufacturing code, ID, App group ID, Conditional ID and Configuration word of selected tag.

HDR	0xFF	
LEN	0x05	
CMD	0x03	
ID_0	AA	Least significant byte of Tag's ID to read
ID_1	BB	
ID_2	CC	
ID_3	DD	Most significant byte of Tag's ID to read

Successful operation - Tag responded

HDR	0xFF	
LEN	EE	= 17 + (2 * Hardcode words)
CMD	0x03	
[H]		Hardcode
[H]		
T		Timestamp
T		
L		Lock pointer
L		
M		Manufacturing Code
M		
SS		Unique ID
SS		
SS		
SS		
G		Application code
G		
Ci		Conditional ID
Ci		
Co		Configuration word
Co		

Unsuccessful operation - No tag response

HDR	0xFF	
LEN	0x02	
CMD	0x03	
ERR	0x0	

Read tag memory using 8 bit addressing

HDR	0xFF	
LEN	0x07	
CMD	0x04	
ID_0	AA	Least significant byte of Tag's ID to read.
ID_1	BB	
ID_2	CC	
ID_3	DD	Most significant byte of Tag's ID to read.
A	EE	Address of first word to read.
L	FF	Number of words to read (0< FF <51).

Successful operation - Tag responded

HDR	0xFF	
LEN	GG	= 1 + (FF * 2)
CMD	0x04	
$Data1_0$		LSByte of first word read
$Data1_1$		
$Data…_0$		
$Data…_1$		
$DataFF_0$		
$DataFF_1$		MSByte of last word read.

Unsuccessful operation - No tag response

HDR	0xFF	
LEN	0x02	
CMD	0x04	
ERR	0x0	

Read tag memory using 16 bit addressing

HDR	0xFF	
LEN	0x09	
CMD	0x05	
ID_0	AA	Least significant byte of Tag's ID to read
ID_1	BB	
ID_2	CC	
ID_3	DD	Most significant byte of Tag's ID to read
A_0	EE	Low byte of Address of first word to read
A_1	EE	High byte of Address of first word to read
L_0	FF	Low byte of number of words to read (0< FFFF <51).
L_1	FF	High byte of number of words to read (0< FFFF <51).

Successful operation - Tag responded

HDR	0xFF	
LEN	GG	= 1 + FFFF
CMD	0x05	
$Data1_0$		LSByte of first word read
$Data1_1$		
$Data…_0$		
$Data…_1$		
$DataFF_0$		
$DataFF_1$		MSByte of last word read

Unsuccessful operation - No tag response

HDR	0xFF	
LEN	0x02	
CMD	0x05	
ERR	0x0	

Write tag memory using 8 bit address and length, without password

HDR	0xFF	
LEN	AA	= 7 + (2 * GG)
CMD	0x08	
ID_0	BB	Least significant byte of Tag's ID to write to
ID_1	CC	
ID_2	DD	
ID_3	EE	Most significant byte of Tag's ID to write to
A	FF	Address of first word to write
L	GG	Number of words to write (0<= GG <5)
$Data1_0$		Low byte of first word to write
$Data1_1$		High byte of first word to write
$Data\ldots_0$		
$Data\ldots_1$		
$DataGG_0$		Low byte of last word to write
$DataGG_1$		High byte of last word to write

Notes;
GG must comply with block and sub-block boundary write constraints.
Setting GG to 0 and not sending any Data bytes will set the lock pointer to address FF.

Successful operation - data has been written and confirmed

HDR	0xFF	
LEN	0x02	
CMD	0x08	
OK	0x01	

Unsuccessful operation

HDR	0xFF	
LEN	0x02	
CMD	0x08	
ERR	0x0	

Write tag memory using 16 bit address and length, without password

HDR	0xFF	
LEN	AA	= 9 + (2 * GGGG)
CMD	0x09	
ID_0	BB	Least significant byte of Tag's ID to write to
ID_1	CC	
ID_2	DD	
ID_3	EE	Most significant byte of Tag's ID to write to
A	FF	Low byte of address of first word to write
A	FF	High byte of address of first word to write
L	GG	Low byte of number of words to write (0<= GGGG <5)
L	GG	High byte of number of words to write (0<= GGGG <5)
$Data1_0$		Low byte of first word to write
$Data1_1$		High byte of first word to write
$Data\ldots_0$		
$Data\ldots_1$		
$DataGG_0$		Low byte of last word to write
$DataGG_1$		High byte of last word to write

Notes;
GGGG must comply with block and sub-block boundary write constraints.
Setting GGGG to 0 and not sending any Data bytes will set the lock pointer to address FFFF.

Successful operation - data has been written and confirmed.

HDR	0xFF	
LEN	0x02	
CMD	0x09	
OK	0x01	

Unsuccessful operation

HDR	0xFF	
LEN	0x02	
CMD	0x09	
ERR	0x0	

Write tag memory using 8 bit addressing, with password

HDR	0xFF	
LEN	AA	= 13 + (2 * HH)
CMD	0x0A	
ID_0	BB	Least significant byte of Tag's ID to write to
ID_1	CC	
ID_2	DD	
ID_3	EE	Most significant byte of Tag's ID to write to
P	FF	Low byte of Password
P	FF	
P	FF	
P	FF	
P	FF	
P	FF	High byte of password
A	GG	Address of first word to write
L	HH	Number of words to write (0<= HH <5)
$Data1_0$		Low byte of first word to write
$Data1_1$		High byte of first word to write
$Data\ldots_0$		
$Data\ldots_1$		
$DataGG_0$		Low byte of last word to write
$DataGG_1$		High byte of last word to write

Notes;
HH must comply with block and sub-block boundary write constraints.
Setting HH to 0 and not sending any Data bytes will set the lock pointer to address GG.
When writing to an address less than 10 on a password protected tag, the module's own internal confirmation will always fail regardless of whether or not the write was successful. This is because addresses less than 10 cannot be read on a password protected tag. It is down to the host application to check the required operation against the data returned from a *Get tag's configuration memory* operation.

Successful operation - data has been written and confirmed.

HDR	0xFF	
LEN	0x02	
CMD	0x0A	
OK	0x01	

Unsuccessful operation

HDR	0xFF	
LEN	0x02	
CMD	0x0A	
ERR	0x0	

Write tag memory using 16 bit addressing, with password

HDR	0xFF	
LEN	AA	= 15 + (2 * HH)
CMD	0x0B	
ID_0	BB	Least significant byte of Tag's ID to write to
ID_1	CC	
ID_2	DD	
ID_3	EE	Most significant byte of Tag's ID to write to
P	FF	Low byte of Password
P	FF	
P	FF	
P	FF	
P	FF	
P	FF	High byte of password
A	GG	Address of first word to write
A	GG	
L	HH	Number of words to write (0<= HH <5)
L	HH	
$Data1_0$		Low byte of first word to write
$Data1_1$		High byte of first word to write
$Data\ldots_0$		
$Data\ldots_1$		
$DataGG_0$		Low byte of last word to write
$DataGG_1$		High byte of last word to write

Notes;
HHHH must comply with block and sub-block boundary write constraints.
Setting HHHH to 0 and not sending any Data bytes will set the lock pointer to address GGGG.
When writing to an address less than 10 on a password protected tag, the module's own internal confirmation will always fail regardless of whether or not the write was successful. This is because addresses less than 10 cannot be read on a password protected tag. It is down to the host application to check the required operation against the data returned from a *Get tag's configuration memory* operation.

Successful operation - data has been written and confirmed.

HDR	0xFF	
LEN	0x02	
CMD	0x0B	
OK	0x01	

Unsuccessful operation

HDR	0xFF	
LEN	0x02	
CMD	0x0B	
ERR	0x0	

Get firmware version

HDR	0xFF	
LEN	0x01	
CMD	0x0C	

Response

HDR	0xFF	
LEN	0x02	
CMD	0x0C	
OK	AA	Single byte version identifier.

Turn powering RF field

Normally there is no need to manually control the field.
The module will always ensure the field is turned on to perform any operation and will also automatically turn it off after the operation has completed.
If required, the field can be set to permanently on with this command.

HDR	0xFF	
LEN	0x02	
CMD	0x0D	
ARG	A	A = 1 to turn field on, A=0 field controlled automatically.

Response

HDR	0xFF	
LEN	0x02	
CMD	0x0D	
OK	A	A=1 successful operation

Set the baud rate

The communication baud rate can be controlled using this command.
If successful, the reply will be transmitted at the same baud rate it was received at and then the rate will be changed for all subsequent communications.
Only available on firmware version 4 or later.

HDR	0xFF	
LEN	0x02	
CMD	0x0E	
ARG	A	A = 0, set baud rate to 1200 A = 1, set baud rate to 2400 A = 2, set baud rate to 4800 A = 3, set baud rate to 9600 A = 4, set baud rate to 19200 A = 5, set baud rate to 28800 A = 6, set baud rate to 38400 A = 7, set baud rate to 56000 A = 8, set baud rate to 57600

Response

HDR	0xFF	
LEN	0x02	
CMD	0x0E	
OK	A	A=1 successful operation