RS232 serial & I²C InfraRed Communications Module

Technical Data

DS-IRCM
Firmware version 1

Features

- Register compatible with the DS-WCM range
- Features FWCM broadcast and extended data registers
- Network 255 OOPic (II, -R & -C) controllers / PC's with 100% data accuracy, just think 255 individually controlled robots !!
- Not just TX & RX modules, but a complete addressable network.
- Micro-sized module (14mm x 35mm high quality PCB) plugs directly into the OOPic (II, -R & -C) I2C programming connector.
- RS232* or I²C communication interface for simple connection to PC, OOPic, BASIC Stamp etc.
- Full bi-directional communication with packet checking, error correction and anti-collision.
- Infrared remote control support for Philips, Sony, SKY digibox and NEC format.

Description

The Designer Systems DS-IRCM is a highly integrated infrared wireless communications module that allows the formation of a local area bi-directional wireless network capable of transferring data over a distance of up to 10 metres. Specifically targeted at the OOPic (II) & OOPic-R embedded control module and the Personal Computer user the IRCM features I²C and RS232* communication.

Communication between IRCMs' is through self assigned node addresses with a maximum network consisting of 255 IRCM nodes. Data is transferred by simply writing a value (Nine (9) values in expanded data mode) and a destination node to the local IRCM and waiting for confirmation that the data has been sent successfully. The IRCM handles all the error correction, anti-collision and data confirmation protocol necessary for error free transfer.

In addition the DS-IRCM features the decoding of Infrared remote control data with support for Philips (RC5), Sony, SKY digibox and NEC format. This allows your robotic project to be easily controlled using most remote controls found in the modern home.

Applications

The DS-IRCM has applications in robotics, remote control, data transfer etc. Application notes for the OOPic (II) & OOPic-R controller are provided (loop test program & X_4 Rover remote control controller).

COMMS MODULES

Selection Guide

Description	Part Number
Infrared Communication Module	DS-IRCM
PC (DB9) to free end cable 1.5m*	DS-232CAB

* Connection must be self-soldered on to the module to support this operational mode.

Power requirements

The DS-IRCM takes the power necessary for operation from the OOPic or an external +5V supply.
Warning: Mis-connection of this connection may damage the DS-IRCM.

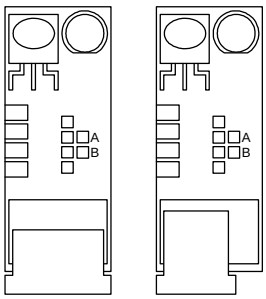
Normal Mode

The DS-IRCM when configured for normal mode allows two or more (255 max.) IRCMs' to form a wireless network capable of transmitting and receiving bytes of data to/from any other IRCM on the network. Each IRCM can allocate its own local node address, which allows peer-to-peer or server based transfer of data as required by the wireless application. Data is transferred a byte at a time (Nine using expanded data mode) with each transmission being checked and confirmed by the remote IRCM node before another can be sent. This protocol ensures that the data sent to the remote IRCM is always 100% correct, thus preventing errors that could cause robot failure or destruction.

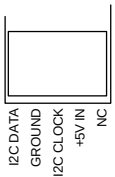
Broadcast mode…
If a write to remote node address 254 is undertaken the sending and receiving IRCM(s) automatically enter broadcast mode. This mode allows the sending node, i.e. PC, to broadcast data to multiple remote IRCM(s) simultaneously to allow, for example, the control of multiple rovers. In this mode the remote IRCM(s) will receive the broadcast data even if their local node address is not set to 254 and make it available to the connected device i.e. OOPic etc.
Broadcast mode differs from a standard mode transfer by the fact that the sending IRCM node does not wait for a confirmation from the remote IRCM(s) and the remote IRCM(s) do not send one. If data confirmation is required a timed write from each of the remote IRCM(s) can be undertaken.

I^2C connection

The I^2C connector (5pin header) allows the IRCM to be directly plugged onto the 'Prg' header on the OOPic, OOPic II & OOPic-R controllers without any additional cabling or the OOPic-C with an additional header. The IRCM may also be plugged onto a free 4 pin header on the DS-(F)WCM(+), DS-GPM etc. This allows a single or multiple modules to be easily integrated into any application with the minimum of connection. The following diagrams show the module plugged onto an OOPic 'Prg' header and DS-WCM 'I2C Output' header:

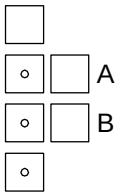


The 5pin header is pinned as follows:

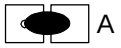


I^2C communication

Up to four IRCM modules may be connected to the same I^2C bus and accessed individually using their own individual address.
The address is configured with the following solder jumpers:



The jumpers are made by placing a blob of solder between the two pads for A or B as follows:



WARNING: If you are not competent at soldering you may not want to attempt this yourself.

The following table shows how the jumpers should be soldered for the different binary addresses:

Address xx	A	B
00 (default)	OFF	OFF
01	OFF	ON
10	ON	OFF
11	ON	ON

The binary address (xx) above is used in conjunction with the device ID 11000xxD to form the complete device address i.e. if both jumpers are left connected (default) then the device address would be $1100000D_{binary}$.
The 'D' bit determines if a read or a write to the IRCM is to be performed. If the 'D' bit is set '1' then a register read is performed or if clear '0' a register write.

To access individual registers a device write must be undertaken by the OOPic / I^2C Master which consists of a Start condition, device ID ('D' bit cleared), register to start write, one or more bytes of data to be written and a stop condition (see Figure 1.0 for I^2C write protocol).
There are 15 individual registers that can be written within the IRCM that control functions such as node address setup, data transmission control as follows:

	N_7	N_6	N_5	N_4	N_3	N_2	N_1	N_0
IRCM I2C address								
1.	1	1	0	0	0	X	X	0

XX = Address select jumpers A & B

Register address								
2.	U	U	U	B	B	B	B	B

B..B = 0 to 14
U..U = unused on this implementation

Local node address								
R0	A	A	A	A	A	A	A	A

A..A = 0 to 255 (Address to which this local node is to be allocated)

Remote node address to write too								
R1	A	A	A	A	A	A	A	A

A..A = 0 to 255 (Remote node address to which data is to be written)

Remote node timeout value and local / remote reset								
R2	A	U	U	U	T	T	T	T

T..T = 1 to 15 (Time in seconds to wait for remote IRCM to respond to write)
A = Local IRCM reset request bit (1 = Reset) *
U..U = unused on this implementation

Data to write to remote node								
R3	D	D	D	D	D	D	D	D

D..D = 0 to 255 (Data to be written to remote node)

Not implemented (kept for WCM compatibility)								
R4	X	X	X	X	X	X	X	X

Not implemented (kept for WCM compatibility)								
R5	X	X	X	X	X	X	X	X

Not implemented (kept for WCM compatibility)								
R6	X	X	X	X	X	X	X	X

Remote data register 1								
R7	D	D	D	D	D	D	D	D

D..D = 0 to 255 (Extended data value)

Remote data register 2								
R8	D	D	D	D	D	D	D	D

D..D = 0 to 255 (Extended data value)

Remote data register 3								
R9	D	D	D	D	D	D	D	D

D..D = 0 to 255 (Extended data value)

Remote data register 4								
R10	D	D	D	D	D	D	D	D

D..D = 0 to 255 (Extended data value)

Remote data register 5								
R11	D	D	D	D	D	D	D	D

D..D = 0 to 255 (Extended data value)

Remote data register 6								
R12	D	D	D	D	D	D	D	D

D..D = 0 to 255 (Extended data value)

Remote data register 7

R13	D	D	D	D	D	D	D	D

D..D = 0 to 255 (Extended data value)

Remote data register 8

R14	D	D	D	D	D	D	D	D

D..D = 0 to 255 (Extended data value)

** Warning: A node reset will immediately terminate all communication on the node being reset, use carefully.*

The transfer of data to a remote IRCM node is only activated when a write to Register 3 (Data to write to remote node) is processed. This allows other registers such as the remote node address, expanded data etc. to be stored before the data is transferred.

To read individual status registers a device write then read must be undertaken by the OOPic / I^2C Master. The write consists of a Start condition, device ID ('D' bit cleared), register to start read and a Stop condition.
This is followed by a read, which consists of a Start condition, device ID ('D' bit set), followed by data from the register specified and terminated with a Stop condition. The IRCM also auto increments the register specified for every additional read requested by the Master I^2C device, which allows more than one register to be read in one transaction. This allows for example Register 1 'Write completion flag' and Register 2 'Write time-out error flag' to be read together (see Figure 1.1 for I^2C read protocol).
There are 33 individual registers that can be read within the IRCM as follows:

	N_7	N_6	N_5	N_4	N_3	N_2	N_1	N_0

IRCM Address

1.	1	1	0	0	0	X	X	1

XX = Address select jumpers A & B

Remote write completion flag

R0	F	F	F	F	F	F	F	F

F..F = 0 (Write has not completed yet)
F..F = 255 (Write has completed successfully)

Remote write error flag register

R1	F	F	F	F	F	F	F	F

F..F = 0 (Write did not produce an error)
F..F = 255 (Write timeout error, remote IRCM did not respond within time specified)

Remote read status register

R2	F	F	F	F	F	F	F	F

F..F = 0 (No data has been received from remote node)
F..F = 255 (Data has been received and is waiting)

Remote read error flag register

R3	F	F	F	F	F	F	F	F

F..F = 0 (No error has occurred)
F..F = 255 (Data overflow condition has occurred) set if data register R5 has not been read and another byte of data has been received from remote IRCM.

Remote read node address

R4	A	A	A	A	A	A	A	A

A..A = 0 to 255 (Address from which data has been received)

Remote read data register

R5	D	D	D	D	D	D	D	D

D..D = 0 to 255 (Data received from remote node)

Not implemented (kept for WCM compatibility)

R6	X	X	X	X	X	X	X	X

Not implemented (kept for WCM compatibility)

R7	X	X	X	X	X	X	X	X

Not implemented (kept for WCM compatibility)

R8	X	X	X	X	X	X	X	X

Not implemented (kept for WCM compatibility)

R9	X	X	X	X	X	X	X	X

Not implemented (kept for WCM compatibility)

R10	X	X	X	X	X	X	X	X

Remote status validity flag register

R11	F	F	F	F	F	F	F	F

F..F = 0 (Remote values are invalid)
F..F = 255 (Remote values are valid and remote node is configured for normal mode)

Not implemented (kept for WCM compatibility)

R12	X	X	X	X	X	X	X	X

Not implemented (kept for WCM compatibility)

R13	X	X	X	X	X	X	X	X

Not implemented (kept for WCM compatibility)

R14	X	X	X	X	X	X	X	X

Not implemented (kept for WCM compatibility)

R15	X	X	X	X	X	X	X	X

Not implemented (kept for WCM compatibility)

R16	X	X	X	X	X	X	X	X

Not implemented (kept for WCM compatibility)

R17	X	X	X	X	X	X	X	X

DS-IRCM Status

R18	U	U	U	U	V	V	V	V

V..V = Firmware version number 1-15

Extended data read validity register

R19	X	X	X	X	X	X	X	X

X..X = 0 to 255 (Value indicates extended data register containing data e.g. = 5 then extended registers 1 and 3 contain data from remote node)

Extended data register 1

R20	D	D	D	D	D	D	D	D

D..D = 0 to 255 (Data received from remote node)

Extended data register 2

R21	D	D	D	D	D	D	D	D

D..D = 0 to 255 (Data received from remote node)

Extended data register 3

R22	D	D	D	D	D	D	D	D

D..D = 0 to 255 (Data received from remote node)

Extended data register 4

R23	D	D	D	D	D	D	D	D

D..D = 0 to 255 (Data received from remote node)

Extended data register 5

R24	D	D	D	D	D	D	D	D

D..D = 0 to 255 (Data received from remote node)

Extended data register 6

R25	D	D	D	D	D	D	D	D

D..D = 0 to 255 (Data received from remote node)

Extended data register 7

R26	D	D	D	D	D	D	D	D

D..D = 0 to 255 (Data received from remote node)

Extended data register 8

R27	D	D	D	D	D	D	D	D

D..D = 0 to 255 (Data received from remote node)

Remote control data read status register

R28	F	F	F	F	F	F	F	F

F..F = 0 (No data has been received from remote control)
F..F = 255 (Data has been received and is waiting)

Remote control type register

R29	U	U	U	U	U	D	D	D

D..D = 1 (Philips RC5 format)
D..D = 2 (Sony format)
D..D = 3 (SKY digibox format)
D..D = 4 (NEC format)

Remote control data register 1

R30	D	D	D	D	D	D	D	D

D..D = 0 to 255 (Data received from remote control)

Remote control data register 2

R31	D	D	D	D	D	D	D	D

D..D = 0 to 255 (Data received from remote control)

Remote control data register 3

R32	D	D	D	D	D	D	D	D

D..D = 0 to 255 (Data received from remote control)

Remote control data register 4

R33	D	D	D	D	D	D	D	D

D..D = 0 to 255 (Data received from remote control)
NOTE: This register must be read to clear register R28

Example.

To set the local IRCM node address to 1 and write a value of 45 to remote IRCM node 9 (Timeout value = 3 seconds, device address = default) write:

Byte 1 (IRCM Adr) 11000000_{binary}
Byte 2 (Set register) $0_{decimal}$, 00_{hex}
Byte 3 (Register 0) $1_{decimal}$, 01_{hex}
Byte 4 (Register 1) $9_{decimal}$, 09_{hex}
Byte 5 (Register 2) $3_{decimal}$, 03_{hex}
Byte 6 (Register 3) $45_{decimal}$, $2D_{hex}$

Once this command has been issued the local IRCM can be polled by the OOPic / I^2C Master to determine when the data transfer to the remote node has been successfully completed or a time-out error has occurred. This is accomplished by interrogating Register 1 (Write confirmation flag) followed by a Register 2 (Write time-out flag) as follows:

'Point to register 0
Byte 1 - 11000000_{binary}
Byte 2 – $0_{decimal}$, 00_{hex}

'Read register 0 & 1
Byte 1 - 11000001_{binary}
Byte 2 – $255_{decimal}$ if confirmed, $0_{decimal}$ if not
Byte 3 – $255_{decimal}$ if error, $0_{decimal}$ if not

Example.

To determine if a byte of data has been received from a remote IRCM:

'Point to register 2
Byte 1 - 11000000_{binary}
Byte 2 – $2_{decimal}$, 02_{hex}

'Read register 1 & 2
Byte 1 - 11000001_{binary}
Byte 2 – $255_{decimal}$ if received, $0_{decimal}$ if not
Byte 3 – $255_{decimal}$ if error, $0_{decimal}$ if not

Example.

To set the local IRCM node address to 1 and write nine (9) data values (10,20,30,40,50,60, 70,80,90) to remote IRCM node 9 (Timeout value = 3 seconds, device address = default) write:

Byte 1 (IRCM Adr) 11000000_{binary}
Byte 2 (Set register) $7_{decimal}$, 07_{hex}
Byte 3 (Register 7) $20_{decimal}$
Byte 4 (Register 8) $30_{decimal}$
Byte 5 (Register 9) $40_{decimal}$
Byte 6 (Register 10) $50_{decimal}$
Byte 7 (Register 11) $60_{decimal}$
Byte 8 (Register 12) $70_{decimal}$
Byte 9 (Register 13) $80_{decimal}$
Byte 10 (Register 14) $90_{decimal}$

Cont..

Then write:

Byte 1 (IRCM Adr) 11000000_{binary}
Byte 2 (Set register) $0_{decimal}$, 00_{hex}
Byte 3 (Register 0) $1_{decimal}$, 01_{hex}
Byte 4 (Register 1) $9_{decimal}$, 09_{hex}
Byte 5 (Register 2) $3_{decimal}$, 03_{hex}
Byte 6 (Register 3) $10_{decimal}$

The first write sequence loads the eight values into the extended registers the second sets the local / remote node address, time-out and activates transmission to the remote node.

Example.

To determine if remote control data has been received:

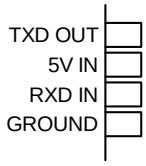
'Point to register 28
Byte 1 - 11000000_{binary}
Byte 2 – $28_{decimal}$, $1C_{hex}$

'Read register 28
Byte 1 - 11000001_{binary}
Byte 2 – $255_{decimal}$ if received, $0_{decimal}$ if not

'Read registers 29 to 33
Byte 3 – Register 29 returned
Byte 4 – Register 30 returned
Byte 5 – Register 31 returned
Byte 6 – Register 32 returned
Byte 7 – Register 33 returned

RS232 port & setup

The RS232 port allows connection to a Personal Computer or other serial RS232 device.
Connection is via. four (4) PCB pads which must have either wires ie. DS-232CAB or a 4pin 2.54mm pitch connector soldered to them.
The pads are identified as follows:

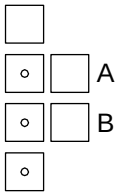


The connection is formatted to allow the maximum of four (4) DS-IRCM modules to be connected in a 'daisy chain' configuration similar to an I^2C bus.
This is accomplished by connecting all 'RXD INputs' together, all 'TXD OUTputs' together using diode steering and changing the IRCM address jumpers on each module.
The RS232 connection will support any modern RS232D/E compliant device and must be configured for the following protocol:

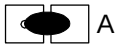
4800 baud (bps)
8 Data bits
1 Stop Bit
No Parity
No handshaking (if configurable)

RS232 communication

Up to four IRCM modules may be connected to the same RS232 device and accessed individually using their own individual address. The address is configured with the following solder jumpers:



The jumpers are made by placing a blob of solder between the two pads for A or B as follows:



WARNING: If you are not competent at soldering you may not want to attempt this yourself.

The following table shows how the jumpers should be soldered for the different binary addresses:

Address xx	A	B
00 (default)	OFF	OFF
01	OFF	ON
10	ON	OFF
11	ON	ON

The binary address (xx) above is used in conjunction with the device ID 11000xxD to form the complete device address i.e. if both jumpers are left un-connected (default) then the device address would be $1100000D_{binary}$.
The 'D' bit determines if a read or a write to the IRCM is to be performed. If the 'D' bit is set '1' then a register read is performed or if clear '0' a register write.

To access individual registers a device write must be undertaken by the RS232 device which consists of a Prefix character, device ID ('D' bit cleared), register to start write, one or more bytes of data to be written and two terminator characters.

There are 15 individual registers that can be written within the IRCM that control functions such as node address setup, data transmission control and remote data values as follows:

	N_7	N_6	N_5	N_4	N_3	N_2	N_1	N_0

RS232 command prefix
1. | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
ASCII character ' ['

IRCM RS232 address
2. | 1 | 1 | 0 | 0 | 0 | X | X | 0 |
XX = Address select jumpers A & B

Register address
3. | U | U | U | U | B | B | B | B |
B..B = 0 to 14
U..U = unused on this implementation

Local node address
R0 | A | A | A | A | A | A | A | A |
A..A = 0 to 255 (Address to which this local node is to be allocated)

Remote node address to write too
R1 | A | A | A | A | A | A | A | A |
A..A = 0 to 255 (Remote node address to which data is to be written)

Remote node timeout value and local / remote reset
R2 | A | U | U | U | T | T | T | T |
T..T = 1 to 15 (Time in seconds to wait for remote IRCM to respond to write)
A = Local IRCM reset request bit (1 = Reset) *
U..U = unused on this implementation

Data to write to remote node
R3 | D | D | D | D | D | D | D | D |
D..D = 0 to 255 (Data to be written to remote node)

Not implemented (kept for WCM compatibility)
R4 | X | X | X | X | X | X | X | X |

Not implemented (kept for WCM compatibility)
R5 | X | X | X | X | X | X | X | X |

Not implemented (kept for WCM compatibility)
R6 | X | X | X | X | X | X | X | X |

Remote data register 1
R7 | D | D | D | D | D | D | D | D |
D..D = 0 to 255 (Extended data value)

Remote data register 2
R8 | D | D | D | D | D | D | D | D |
D..D = 0 to 255 (Extended data value)

Remote data register 3
R9 | D | D | D | D | D | D | D | D |
D..D = 0 to 255 (Extended data value)

Remote data register 4
R10 | D | D | D | D | D | D | D | D |
D..D = 0 to 255 (Extended data value)

Remote data register 5
R11 | D | D | D | D | D | D | D | D |
D..D = 0 to 255 (Extended data value)

Remote data register 6
R12 | D | D | D | D | D | D | D | D |
D..D = 0 to 255 (Extended data value)

Remote data register 7
R13 | D | D | D | D | D | D | D | D |
D..D = 0 to 255 (Extended data value)

Remote data register 8
R14 | D | D | D | D | D | D | D | D |
D..D = 0 to 255 (Extended data value)

RS232 command terminator
n. | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 |
ASCII character '] '

RS232 command terminator
n. | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 |
ASCII character '] '

The RS232 protocol differs from the I^2C in that it is able to respond asynchronously (when ever it likes) to certain conditions such as confirming when data has been transferred or when data has been received. This is accomplished by the IRCM transmitting response packets to the

RS232 device in the following formats:

Write confirmation…
(sent upon write confirmation or time-out):

	N_7	N_6	N_5	N_4	N_3	N_2	N_1	N_0
RS232 command prefix								
1.	0	1	0	1	1	0	1	1

ASCII character ' ['

IRCM RS232 address								
2.	1	1	0	0	0	X	X	0

XX = Address select jumpers A & B

IRCM Status flag								
3.	F	F	F	F	F	F	F	F

F..F = 0 (Write confirmation / time-out identifier)

Remote write error flag register								
4.	F	F	F	F	F	F	F	F

F..F = 0 (Write did not produce an error)
F..F = 255 (Write timeout error, remote IRCM did not respond within time specified)

RS232 command terminator								
5.	0	1	0	1	1	1	0	1

ASCII character '] '

Read confirmation…
(Sent upon receipt of data from remote IRCM node):

	N_7	N_6	N_5	N_4	N_3	N_2	N_1	N_0
RS232 command prefix								
1.	0	1	0	1	1	0	1	1

ASCII character ' ['

IRCM RS232 address								
2.	1	1	0	0	0	X	X	0

XX = Address select jumpers A & B

IRCM Status flag								
3.	F	F	F	F	F	F	F	F

F..F = 255 (Data received from remote node identifier)

Remote read node address								
4.	A	A	A	A	A	A	A	A

A..A = 0 to 255 (Address from which data has been received)

Remote read data register								
5.	D	D	D	D	D	D	D	D

D..D = 0 to 255 (Data received from remote node)

Extended data read validity register								
6.	X	X	X	X	X	X	X	X

X..X = 0 to 255 (Value indicates extended data register containing data e.g. = 5 then extended registers 1 and 3 contain data from remote node)

Extended data register 1								
7.	D	D	D	D	D	D	D	D

D..D = 0 to 255 (Data received from remote node)

Extended data register 2								
8.	D	D	D	D	D	D	D	D

D..D = 0 to 255 (Data received from remote node)

Extended data register 3								
9.	D	D	D	D	D	D	D	D

D..D = 0 to 255 (Data received from remote node)

Extended data register 4								
10.	D	D	D	D	D	D	D	D

D..D = 0 to 255 (Data received from remote node)

Extended data register 5								
11.	D	D	D	D	D	D	D	D

D..D = 0 to 255 (Data received from remote node)

Extended data register 6								
12.	D	D	D	D	D	D	D	D

D..D = 0 to 255 (Data received from remote node)

Extended data register 7								
13.	D	D	D	D	D	D	D	D

D..D = 0 to 255 (Data received from remote node)

Extended data register 8								
14.	D	D	D	D	D	D	D	D

D..D = 0 to 255 (Data received from remote node)

RS232 command terminator								
15.	0	1	0	1	1	1	0	1

ASCII character '] '

Remote control data confirmation…
(Sent upon receipt of remote control data):

	N_7	N_6	N_5	N_4	N_3	N_2	N_1	N_0
RS232 command prefix								
1.	0	1	0	1	1	0	1	1

ASCII character ' ['

IRCM RS232 address								
2.	1	1	0	0	0	X	X	0

XX = Address select jumpers A & B

IRCM Status flag								
3.	F	F	F	F	F	F	F	F

F..F = 127 (Data received from remote control identifier)

Remote control type register								
4.	U	U	U	U	U	D	D	D

D..D = 1 (Philips RC5 format)
D..D = 2 (Sony format)
D..D = 3 (SKY digibox format)
D..D = 4 (NEC format)

Remote control data register 1								
5.	D	D	D	D	D	D	D	D

D..D = 0 to 255 (Data received from remote control)

Remote control data register 2								
6.	D	D	D	D	D	D	D	D

D..D = 0 to 255 (Data received from remote control)

Remote control data register 3								
7.	D	D	D	D	D	D	D	D

D..D = 0 to 255 (Data received from remote control)

Remote control data register 4								
8.	D	D	D	D	D	D	D	D

D..D = 0 to 255 (Data received from remote control)

RS232 command terminator								
9.	0	1	0	1	1	1	0	1

ASCII character '] '

Example:
To set the local IRCM node address to 1 and write a value of 45 to remote IRCM node 9 (Timeout value = 3 seconds, device address = default) write:

[#192 #0 #1 #9 #3 #45]]

Where #nn is a value

On confirmation that the value has been sent successfully or that a time-out error has occurred the IRCM will reply with:

Sent OK	[#192 #0 #0]
Timeout error	[#192 #0 #255]

To read the internal status registers a device read command must be received from the RS232 device i.e. PC etc.
The read request consists of a Prefix character, device ID ('D' bit set) and two terminator characters as follows:

	N_7	N_6	N_5	N_4	N_3	N_2	N_1	N_0
RS232 command prefix								
1.	0	1	0	1	1	0	1	1

ASCII character ' ['

IRCM RS232 address								
2.	1	1	0	0	0	X	X	1

XX = Address select jumpers A & B

RS232 command terminator								
3.	0	1	0	1	1	1	0	1

ASCII character '] '

RS232 command terminator								
4.	0	1	0	1	1	1	0	1

ASCII character '] '

Once this command has been received, the relevant IRCM responds by transmitting the contents of all the 33 registers in the following format:

	N_7	N_6	N_5	N_4	N_3	N_2	N_1	N_0
RS232 command prefix								
1.	0	1	0	1	1	0	1	1

ASCII character ' ['

IRCM RS232 address								
2.	1	1	0	0	0	X	X	0

XX = Address select jumpers A & B

Remote write completion flag								
R0	F	F	F	F	F	F	F	F

F..F = 0 (Write has not completed yet)
F..F = 255 (Write has completed successfully)

Remote write error flag register								
R1	F	F	F	F	F	F	F	F

F..F = 0 (Write did not produce an error)
F..F = 255 (Write timeout error, remote IRCM did not respond within time specified)

Remote read status register								
R2	F	F	F	F	F	F	F	F

F..F = 0 (No data has been received from remote node)
F..F = 255 (Data has been received and is waiting)

Remote read error flag register								
R3	F	F	F	F	F	F	F	F

F..F = 0 (No error has occurred)
F..F = 255 (Data overflow condition has occurred) set if data register R5 has not been read and another byte of data has been received from remote IRCM.

Remote read node address								
R4	A	A	A	A	A	A	A	A

A..A = 0 to 255 (Address from which data has been received)

Remote read data register								
R5	D	D	D	D	D	D	D	D

D..D = 0 to 255 (Data received from remote node)

Not implemented (kept for WCM compatibility)								
R6	X	X	X	X	X	X	X	X

Not implemented (kept for WCM compatibility)								
R7	X	X	X	X	X	X	X	X

Not implemented (kept for WCM compatibility)								
R8	X	X	X	X	X	X	X	X

Not implemented (kept for WCM compatibility)								
R9	X	X	X	X	X	X	X	X

Not implemented (kept for WCM compatibility)								
R10	X	X	X	X	X	X	X	X

Remote status validity flag register								
R11	F	F	F	F	F	F	F	F

F..F = 0 (Remote values are invalid)
F..F = 255 (Remote values are valid and remote node is configured for normal mode)

Not implemented (kept for WCM compatibility)								
R12	X	X	X	X	X	X	X	X

Not implemented (kept for WCM compatibility)								
R13	X	X	X	X	X	X	X	X

Not implemented (kept for WCM compatibility)								
R14	X	X	X	X	X	X	X	X

Not implemented (kept for WCM compatibility)								
R15	X	X	X	X	X	X	X	X

Not implemented (kept for WCM compatibility)								
R16	X	X	X	X	X	X	X	X

Not implemented (kept for WCM compatibility)								
R17	X	X	X	X	X	X	X	X

DS-IRCM Status								
R18	U	U	U	U	V	V	V	V

V..V = Firmware version number 1-15

Extended data read validity register								
R19	X	X	X	X	X	X	X	X

X..X = 0 to 255 (Value indicates extended data register containing data e.g. = 5 then extended registers 1 and 3 contain data from remote node)

Extended data register 1								
R20	D	D	D	D	D	D	D	D

D..D = 0 to 255 (Data received from remote node)

Extended data register 2
R21 | D | D | D | D | D | D | D | D |
D..D = 0 to 255 (Data received from remote node)

Extended data register 3
R22 | D | D | D | D | D | D | D | D |
D..D = 0 to 255 (Data received from remote node)

Extended data register 4
R23 | D | D | D | D | D | D | D | D |
D..D = 0 to 255 (Data received from remote node)

Extended data register 5
R24 | D | D | D | D | D | D | D | D |
D..D = 0 to 255 (Data received from remote node)

Extended data register 6
R25 | D | D | D | D | D | D | D | D |
D..D = 0 to 255 (Data received from remote node)

Extended data register 7
R26 | D | D | D | D | D | D | D | D |
D..D = 0 to 255 (Data received from remote node)

Extended data register 8
R27 | D | D | D | D | D | D | D | D |
D..D = 0 to 255 (Data received from remote node)

Remote control data read status register
R28 | F | F | F | F | F | F | F | F |
F..F = 0 (No data has been received from remote control)
F..F = 255 (Data has been received and is waiting)

Remote control type register
R29 | U | U | U | U | U | D | D | D |
D..D = 1 (Philips RC5 format)
D..D = 2 (Sony format)
D..D = 3 (SKY digibox format)
D..D = 4 (NEC format)

Remote control data register 1
R30 | D | D | D | D | D | D | D | D |
D..D = 0 to 255 (Data received from remote control)

Remote control data register 2
R31 | D | D | D | D | D | D | D | D |
D..D = 0 to 255 (Data received from remote control)

Remote control data register 3
R32 | D | D | D | D | D | D | D | D |
D..D = 0 to 255 (Data received from remote control)

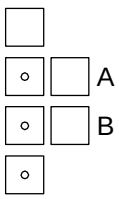
Remote control data register 4
R33 | D | D | D | D | D | D | D | D |
D..D = 0 to 255 (Data received from remote control)

SCP communication

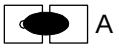
The Savage Innovations SCP allows a remote PC, Pocket PC, Palm Pilot, or any other device with a serial port to control and read the IRCM. The serial protocol is as follows:

4800 baud (bps), 8 Data bits
1 Stop Bit, No Parity
No handshaking (if configurable)

Up to four DS-IRCM units may be connected to the same RS232 port and accessed individually with their own node address. The address is configured with the following solder jumpers:



The jumpers are made by placing a blob of solder between the two pads for A or B as follows:



WARNING: If you are not competent at soldering you may not want to attempt this yourself.

The following table shows how the jumpers should be soldered for the different SCP node addresses:

Address xx	A	B
'0' (default)	OFF	OFF
'1'	OFF	ON
'2'	ON	OFF
'3'	ON	ON

The entire character set used by SCP is composed of human readable characters so that a serial terminal program can be used to manually control IRCM data transfer. The following commands are only briefly described, as the full SCP is not within the scope of this data sheet, a full explanation being available from Savage Innovations.

To enable SCP:
Send "\0V" replace 0 with node address 0 - 3
Receive "v" indicates IRCM is functional.

To set Memory type:
Send "128H" 128 + number of registers to access 0 = 1, 1=2
Receive "h" confirms set.

To set register location to start access:
Send "15J" 15 is register number 0 to 31.
Receive "j" confirms set.

To write to previously selected register:
Send "80N" 80 is sample hexadecimal value to write to register (must be in two character notation)
Receive "n" confirms write.

To read previously selected register:
Send "M" Request register read.

Receive "80m" 80 is sample hexadecimal value.

To read register location:
Send "I" Request register location.
Receive "16i" 16 is current register location.

To read memory type:
Send "G" Request memory type.

Receive "128g" 128 is current memory type.

To reset IRCM:
Send "W" IRCM is reset.

No response generated.

To query SCP buffer:
Send "Q" Request SCP buffer contents.
Receive "124q" Characters in buffer are returned e.g. 124 followed by "q". This command does not effect the buffers contents.

To disable SCP:
Send "X" Request exit from SCP.

Receive "x" SCP has exited.

If command format or value is not correct then a "!" response will be received and the command will not be executed.

Use the I^2C register set for SCP.

Example:
To enable SCP, setup memory type register, location register, set the local IRCM node address to 1 and write a value of 45 to remote IRCM node 9 (Timeout value = 3 seconds, device address = default) write:

Send "\0V"	Enable SCP.
Receive "v"	SCP enabled.
Send "131H"	Set memory type to 128 + 3=131 (4 registers to be accessed).
Receive "h"	Command confirmed.
Send "0J"	Set register location to 0 to allow access to R0 to R3.
Receive "j"	Command confirmed.
Send "0109032dN"	Write data to required registers.
Receive "n"	Write confirmed.

Remote control

The IRCM is capable of reading data from any standard infrared remote control of the following format:

- Philips RC5 (11bit format)
- Sony (12bit format)
- SKY Digibox (20bit format)
- NEC (16bit format)

Six (6) registers are associated with the remote control function (see registers above) which allow the format and data received to be used to control your application.
As the data from individual formats differs greatly the four (4) generic data registers (R30 to R33) will not be formatted as per the individual format specifications. It is therefore necessary to 'preset' your application with the format and data registers to use and the data values of specific remote control buttons.

Electrical Characteristics (T_A = 25°C Typical)

Parameter	Minimum	Maximum	Units	Notes
Supply Voltage (+5V) (Vcc)	4.5	5.5	V	1
Supply Current	1	90	mA	2
RS232 TX data output level	0.3	Vcc-0.8V	V	
RS232 RX data input level	-15	+15	V	
RS232 speed	-	4800	bps	
I^2C speed	-	400	kHz	
Distance	-	10	Metres	

Absolute Maximum Ratings

Parameter	Minimum	Maximum	Units
Supply Voltage (+5V)	-0.5	+6	V

Environmental

Parameter	Minimum	Maximum	Units
Operating Temperature	0	70	°C
Storage Temperature	-10	80	°C
Humidity	0	80	%
Dimensions	Length 35mm, Width 14mm, Height 15mm		
Weight	5g		
Immunity & emissions	EMC compliance to 89/336/EEC		

Notes:
1. Vcc is supply rail from OOPic or any other +5V supply.
2. Maximum value given is during byte transmission.

Example files for OOPic communication:

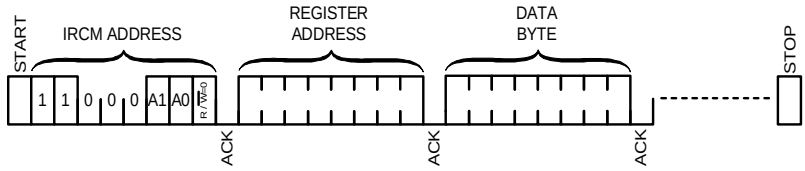
The following OOPic BASIC sub-routines can be used to communicate between OOPic controllers fitted with DS-IRCM modules. Please see the DS-IRCM demonstration diskette for code samples that include OOPic controller loop testing and TV remote control of the X4 rover.

```
' DS-IRCM application sub-routines
'
' Subroutines:
'
' SetUpIRCM – Sets up I2C communication to IRCM and configures local node address.
' WriteIRCM – Takes the value in 'WriteVal' and sends to the remote IRCM node 'RemNode' waiting 'Tout' seconds for transfer to
' complete. Sets flag 'Error' if data was unable to be sent and IRCM had tried for 'Tout' seconds.
' ReadIRCM – Checks for data from remote IRCM node, if received sets flag 'ReadOK', stores remote node IRCM node address in 'ReadAdr'
' and data in 'ReadVal'. Flag 'Oerror' is set if overflow error occurred.
' ReadRC – Checks for remote control data, if received sets flag 'RCOK', stores remote control format 'RCformat'
' and data in 'RCdata1' to 'RCdata4'.
'
' Created : 15/06/03                    Revision : 1.00              Written by : Designer Systems
'
Dim IRCM As New oi2c
Dim WriteVal As New oByte                          'Value to write to remote node
Dim LocNode As New oByte                           'Local node address
Dim RemNode As New oByte                           'Remote node address to write to
Dim ReadVal As New oByte                           'Value read from remote node
Dim ReadAdr As New oByte                           'Node address of remote node value came from
Dim RCformat As New oByte                          'Remote control format
Dim RCdata1 As New oByte                           'Remote control data register 1
Dim RCdata2 As New oByte                           'Remote control data register 2
Dim RCdata3 As New oByte                           'Remote control data register 3
Dim RCdata4 As New oByte                           'Remote control data register 4
Dim Error As New oBit
Dim OError As New oBit
Dim ReadOK As New oBit
Dim RCOK As New oBit
Dim Done As New oBit
Dim TOut As New oByte


Sub Main()

          LocNode = 1                              'Define local node address (0-255)
          RemNode = 2                              'Define remote node address (0-255)
          Const WriteTOut = 6                      'Define write timeout in seconds (0-15)
          Const IRCMAdr = &h60                     'IRCM A0 & A1 jumpers ON (Range &h60-&h63)
          Call SetUpIRCM                           'Setup IRCM ready for communication

          .
          . Main code goes in here
          .

End Sub
```

```
'
' Subroutine to setup I2C communication to DS-IRCM and
' set Local node address
'
Sub SetUpIRCM()

          'Set the DS-IRCM I2C address shifted right by 1 bit

          IRCM.Node = IRCMAdr                    'Setup I2C address for IRCM

          'Setup I2C addressing to IRCM

          IRCM.Width = cv8bit                    'Control Info is 1-byte
          IRCM.Mode = cv10bit                    'I2C mode is 10-Bit Addressing
          IRCM.NoInc = cvFalse                   'Increment on every read/write
          IRCM.Location = 0                      'Specify local node address location
          IRCM.Value = LocNode                   'and write local node address

End Sub
'
' Subroutine to write data to remote IRCM (RemNode) address
' On exit 'Error' = False if write completed OK
'
Sub WriteIRCM()

          IRCM.Location = 1                      'Start at remote node address
          IRCM.Value = RemNode                   'Set remote node address to write to
          IRCM.Value = WriteTOut                 'Set write timeout value 1-15 seconds
          IRCM.Value = WriteVal                  'Send data to remote node

          call      WaitIRCM                     'Wait for confirmation or error

End Sub
'
' Subroutine to check for data from remote IRCM(s) and upon
' confirmation that data has arrived store data in 'ReadVal' and
' remote IRCM address in 'ReadAdr'.
' On exit 'ReadOK' = True if data was read and 'OError' = True
' if internal data register has overflowed i.e. one or more data
' bytes may have been lost
'
Sub ReadIRCM()

          IRCM.Location = 2                      'Start at read status flag (R2)

                    If IRCM.Value = 255 then     'Has data been received (R2) ?

                    If IRCM.Value = 0 then       'If so has overflow error occurred (R3) ?
                              ReadAdr = IRCM.Value'If not then store IRCM address data came from
                              ReadVal = IRCM.Value'and store data
                              OError = cvFalse   'Indicate no overflow error
                              ReadOK = cvTrue    'Indicate data has been read and stored
                    else
                              OError = cvTrue    'Indicate overflow error has occurred
                              ReadOK = cvTrue    'Indicate data has been read
                              ReadAdr = IRCM.Value'Store IRCM address data came from
                              ReadVal = IRCM.Value'and store data
                    end if

                    else

                    OError = cvFalse             'If no data read then no error
                    ReadOK = cvFalse             'and no data

                    end if

End Sub
'
' Subroutine to check for data from remote controls and upon
' confirmation that data has arrived store remote control format in 'RCformat'
' and data in 'RCdata1' to 'RCdata4'.
' On exit 'RCOK' = True if data was read
'
Sub ReadRC()

          IRCM.Location = 28                     'Start at remote control read status flag (R28)

                    If IRCM.Value = 255 then     'Has data been received (R28) ?

                    RCformat = IRCM.Value        'If so then store remote control format
                    RCdata1 = IRCM.Value         'Store data
                    RCdata2 = IRCM.Value         'Store data
                    RCdata3 = IRCM.Value         'Store data
                    RCdata4 = IRCM.Value         'Store data
                    RCOK = cvTrue                'Indicate data has been read and stored

                    else

                    RCOK = cvFalse               'Otherwise no data

                    end if

End Sub
'
' Subroutine to wait for confirmation that data has been successfully sent to
' the remote node or that a timeout error has occurred
'
Sub WaitIRCM()

          Done = cvFalse
do
          IRCM.Location = 0                      'Start at write completion status flag (R0)

                    If IRCM.Value = 255 then     'Has write been completed (R0) ?

                    If IRCM.Value = 0 then       'Has write resulted in error (R1) ?

                    Error = cvFalse              'Indicate no error has resulted
                    Done = cvTrue                'and request exit from sub-routine
```

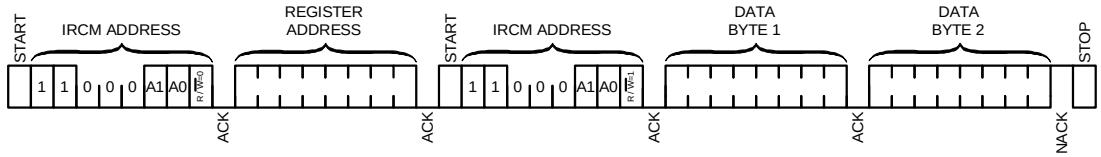
```
                    else

                    Error = cvTrue                'If so then indicate error
                    Done = cvTrue                 'and request exit from sub-routine

                    end if

                    else

                    If IRCM.Value = 255 then      'Has write resulted in error (R1) ?
                    Error = cvTrue                'If so then indicate error
                    Done = cvTrue                 'and request exit from sub-routine
                    end if

                    end if

loop until Done = cvTrue

End Sub
```

Figure 1.0 (I^2C write protocol)



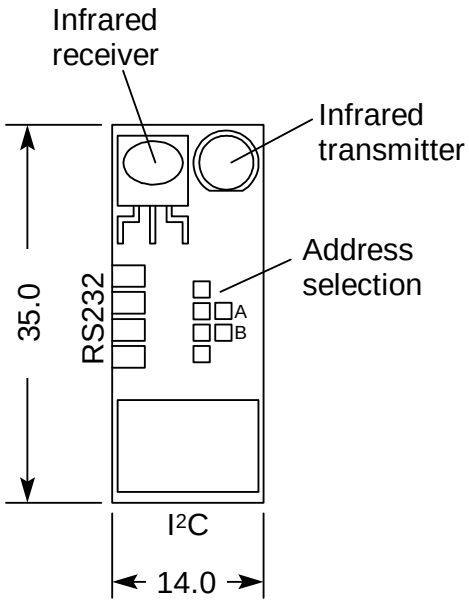
Multiple bytes may be written before the 'STOP' condition. Data is written into registers starting at 'REGISTER ADDRESS', then 'REGISTER ADDRESS' +1, then 'REGISTER ADDRESS' +2 etc.
Each byte transfer is acknowledged 'ACK' by the IRCM until the 'STOP' condition.

Figure 1.1 (I^2C read protocol)



'DATA BYTE 1 & 2' are register values returned from the IRCM. Each byte written is acknowledged 'ACK' by the IRCM , every byte read is acknowledged 'ACK' by the I^2C Master. A Not-acknowledge 'NACK' condition is generated by the I^2C Master when it has finished reading.

Mechanical Specifications – Units millimetres



Page intentionally left blank

Page intentionally left blank