INT-RS module - short technical description

The module INT-RS is dedicated to work with INTEGRA panels with firmware v1.06 2008-01-08 or above. It is an INTEGRA (LCD) bus to RS-232 converter.

To properly configure INT-RS module with INTEGRA panel, the following steps should be done:

1) Set the module address using DIP-switches 3..1 (3-MSB, 1-LSB). Allowed addresses are:
   - 0..3 - for INTEGRA 24 and 32 (i.e. DIP3='OFF')
   - 0..7 - for INTEGRA 64, 128 and WRL

   E.g. to set the 6 address = 110_{bin}, the DIP-switches should be moved to: DIP3='ON', DIP2='ON', DIP1='OFF'.
2) Set the module function using DIP-switches 8..4 (8-MSB, 4-LSB). Possible values are 0 to 31 = 00000_{bin} to 11111_{bin}, but only the first few functions are present (see description below).
3) Connect INT-RS module to INTEGRA LCD bus using 4-wire cable.
4) Enter the service mode, go into the *Structure* menu, enter the *Hardware* submenu, select the *Identification* position and invoke the *LCD keypads id.* function.

For more details refer to INTEGRA manuals.

Function 0 - DIP-switches 8..4 = 00000

The module RS-232 port acts as INT-KLCD keypad serial port. For details refer to INT-KLCD eng.pdf document.

Function 1 - DIP-switches 8..4 = 00001

The module is used by INTEGRA panel for the monitoring purposes. To activate monitoring through INT-RS module, set the *Mon.ETHM-1* option in panel service settings.
If the system contains ETHM-1 modules and INT-RS modules with function 1, setting the *Mon.ETHM-1* option will allow to monitor events only by one of these modules - the one with the lowest address (e.g. the system contains modules: ETHM-1 address 5, INT-RS with function 0 address 1 and INT-RS with function 1 address 3 modules. Monitoring will be processed only through INT-RS with function 1 address 3 module).

RS-232 serial port of INT-RS module is configured as 4800/8/1/N. The DB9-male connector on the PCB makes use of the following lines:

- RX (pin 2) - serial input
- TX (pin 3) - serial output
- DTR (pin 4) - output - active when INT-RS module has communication with INTEGRA
- GND (pin 5) - signal ground
- DSR (pin 6) - input - the module can use this signal only to generate 'No external device DTR signal' event

The GND lines between INT-RS module and external device must be tied together.
The RX and TX lines should be swapped.
The DTR and DSR lines should also be swapped, if they are used.
In INTEGRA service mode it can be set that INT-RS module does or does not generate 'No external device DTR signal' event. It can also be set that INT-RS module does or does not check '?',#13 command (see below). If set, a monitoring trouble arises if external device does not ask INT-RS with '?',#13 question for a time longer that 32 seconds.

Communication between INT-RS module and external device is arranged is such a way that external device should ask INT-RS module to check if a new event is ready to be send to a monitoring station. All data are ASCII chars ended with CR char (#13 = 0x0D byte). Data exchange is no time dependent.
Commands that INT-RS module understands:

- '?',#13 - a question if a new event is ready (2 bytes: 0x3F, 0x0D)
- '+',m,#13 - confirmation of sending event with marker m (3 bytes: 0x2B, m, 0x0D)
- '-',m,#13 - error sending event with marker m (3 bytes: 0x2D, m, 0x0D)

An answer is returned only on '?',#13 question. Possible answers are listed below:

- 'OK',#13 - no new event to send
- 'EN=m,s,iiii,cc'#13 - 4/2 event to sent: m - event marker, s - monitoring station number ('1' or '2'), iiii - event identifier, cc - event code
- 'EC=m,s,iiii,q,ccc,pp,nnn'#13 - Contact ID event to send: s - monitoring station number ('1' or '2'), m - event marker, iiii - event identifier, q and ccc - event code, pp - partition number, nnn - source number

Events format and what events should be sent (4/2 or Contact ID) are to be set in INTEGRA service mode.
Event marker m is a char between 'a' and 'z'. The current event and its marker remain unchanged upon successive '?',#13 questions, until the event is confirmed by '+',m,#13 command from the external device or if INTEGRA time-out occurs (75 seconds). The next event, if ready, will be submitted by INT-RS module with succeeding value of marker m.

Function 2 - DIP-switches 8..4 = 00010
The module is used by INTEGRA panel for the integration purposes.

RS-232 serial port of INT-RS module is configured as 19200/8/1/N. The DB9-male connector on the PCB makes use of the same lines as in the case of Function 1.

Communication between INT-RS module and external device is arranged is such a way that external device should ask (send command to) INT-RS module, and the module will answer immediately, if it is not marked otherwise.

Data exchange is no time dependent. The protocol uses the following frame structure (both ways - from and to INT-RS):

0xFE	0xFE	cmd	d1	d2	...	dn	crc.high	crc.low	0xFE	0x0D

The 16-bit crc sum is calculated as follows:

1) Set crc := 0x147A
2) For all successive bytes b = cmd, d1, d2, ..., dn perform the crc update steps:
    a) crc := rl(crc) - rotate crc 1 bit left (msb=bit.15 shifts into lsb=bit.0 position)
    b) crc := crc xor 0xFFFF
    c) crc := crc + crc.high + b, e.g. if crc=0xFEDC and b=0xA9 then: 0xFEDC + 0xFE + 0xA9 = 0x0083

The 0xFE byte is special value:

1) Two (or more) successive 0xFE mean frame synchronization - i.e. if device waits for any data-frame byte and it receives 0xFE, 0xFE - it should interrupt collecting the current frame and start waiting for cmd.
2) If device is waiting for the 1st byte of a frame (i.e. waiting for cmd), receiving 0xFE should not change it - device should be still waiting for cmd. So, cmd can not be 0xFE.
3) If any byte of the frame (i.e. cmd, d1, d2, ..., dn, crc.high, crc.low) to be sent is equal 0xFE, the following two bytes must be sent instead of single 0xFE byte: 0xFE, 0xF0. In such case only single 0xFE should be used to update crc.
4) If 0xFE, 0x0D are received, it means the frame is completed and it can be processed - i.e. check crc and analyze.
5) If other value after 0xFE is received - treat it as 0xFE, 0xFE (i.e. treat it as synchronization sequence).

If frame is corrupted (i.e. it has wrong crc sum or it was interrupted by 0xFE, 0xFE before completed) or cmd is not know or data length is not suitable for cmd - it is dropped and no answer is given back. **External device should act the same way.**

Part 1 - Reading INTEGRA state:

cmd	meaning	answer	
0x00	zones violation	0x00	+ 16 bytes
0x01	zones tamper	0x01	+ 16 bytes
0x02	zones alarm	0x02	+ 16 bytes
0x03	zones tamper alarm	0x03	+ 16 bytes
0x04	zones alarm memory	0x04	+ 16 bytes
0x05	zones tamper alarm memory	0x05	+ 16 bytes
0x06	zones bypass	0x06	+ 16 bytes
0x07	zones 'no violation' trouble	0x07	+ 16 bytes
0x08	zones 'long violation' trouble	0x08	+ 16 bytes
0x09	armed partitions (suppressed)	0x09	+ 4 bytes
0x0A	armed partitions (really)	0x0A	+ 4 bytes
0x0B	partitions armed in mode 2	0x0B	+ 4 bytes
0x0C	partitions armed in mode 3	0x0C	+ 4 bytes
0x0D	partitions with 1st code entered	0x0D	+ 4 bytes
0x0E	partitions entry time	0x0E	+ 4 bytes
0x0F	partitions exit time >10s	0x0F	+ 4 bytes
0x10	partitions exit time <10s	0x10	+ 4 bytes
0x11	partitions temporary blocked	0x11	+ 4 bytes
0x12	partitions blocked for guard round	0x12	+ 4 bytes
0x13	partitions alarm	0x13	+ 4 bytes
0x14	partitions fire alarm	0x14	+ 4 bytes
0x15	partitions alarm memory	0x15	+ 4 bytes
0x16	partitions fire alarm memory	0x16	+ 4 bytes
0x17	outputs state	0x17	+ 16 bytes
0x18	doors opened	0x18	+ 8 bytes
0x19	doors opened long	0x19	+ 8 bytes

0x1A	RTC and basic status bits	0x1A	+ 9 bytes (see description below)
0x1B	troubles part 1	0x1B	+ 47 bytes (see description below)
0x1C	troubles part 2	0x1C	+ 26 bytes (see description below)
0x1D	troubles part 3	0x1D	+ 60 bytes (see description below)
0x1E	troubles part 4	0x1E	+ 29 bytes (see description below)
0x1F	troubles part 5	0x1F	+ 31 bytes (see description below)
0x20	troubles memory part 1	0x20	+ 47 bytes (see description below)
0x21	troubles memory part 2	0x21	+ 39 bytes (see description below)
0x22	troubles memory part 3	0x22	+ 60 bytes (see description below)
0x23	troubles memory part 4	0x23	+ 29 bytes (see description below)
0x24	troubles memory part 5	0x24	+ 48 bytes (see description below)

Answers description:

RTC and basic status bits	-	7 bytes - time: YYYY-MM-DD hh:mm:ss - 0xYY, 0xYY, 0xMM, 0xDD, 0xhh, 0xmm, 0xss		
		1 byte -	.210	- day of the week (0=Monday, 1=Tuesday, ..., 6=Sunday)
			.7	- 1 = service mode
			.6	- 1 = troubles in the system (= flashing TROUBLE LED in keypad)
		1 byte -	.7	- 1 = ACU-100 are present in the system
			.6	- 1 = INT-RX are present in the system
			.5	- 1 = troubles memory is set in INTEGRA panel
			.3210	- INTEGRA type: 0 = 24, 1 = 32, 2 = 64, 3 = 128, 4 = WRL
troubles part 1	-	16 bytes		- troubles - technical zones
		8 bytes		- expanders AC trouble
		8 bytes		- expanders BATT trouble
		8 bytes		- expanders NO BATT trouble
		3 bytes		- system troubles (see description below)
		1 byte		- CA-64 PTSA modules AC trouble
		1 byte		- CA-64 PTSA modules BATT trouble
		1 byte		- CA-64 PTSA modules NO BATT trouble
		1 byte		- ETHM-1 monitoring trouble
troubles part 2	-	8 bytes		- proximity card readers head A trouble
		8 bytes		- proximity card readers head B trouble
		8 bytes		- expanders supply output overload
		2 bytes		- addressable zone expanders short circuit or jammed ACU-100 modules
troubles part 3	-	15 bytes		- ACU-100 modules jam level
		15 bytes		- radio devices with low battery
		15 bytes		- radio devices with no communication
		15 bytes		- radio outputs with no communication
troubles part 4	-	8 bytes		- expanders with no communication
		8 bytes		- switcherooed expanders
		1 byte		- LCD keypads with no communication
		1 byte		- switcherooed LCD keypads
		1 byte		- ETHM-1 modules with no LAN cable / INT-RS modules with no DSR signal
		8 bytes		- expanders tamper
		1 byte		- LCD keypads tamper
		1 byte		- LCD keypad initiation errors
troubles part 5	-	1 byte		- low battery in masters key fobs
		30 bytes		- low battery in users key fobs
troubles memory part 1	-	47 bytes		- memory of troubles part 1
troubles memory part 2	-	26 bytes		- memory of troubles part 2
		1 byte		- LCD keypads restart memory
		8 bytes		- expanders restart memory
		2 bytes		- GSM trouble code (high,low)
		2 bytes		- GSM trouble code memory (high,low)
troubles memory part 3	-	60 bytes		- memory of troubles part 3
troubles memory part 4	-	29 bytes		- memory of troubles part 4
troubles memory part 5	-	16 bytes		- long zones violation memory
		16 bytes		- no zones violation memory
		16 bytes		- zones tamper memory

System troubles:	1*st* byte -	.0	- OUT1 trouble
		.1	- OUT2 trouble
		.2	- OUT3 trouble
		.3	- OUT4 trouble
		.4	- +KPD trouble
		.5	- +EX1 or +EX2 trouble
		.6	- BATT trouble
		.7	- AC trouble
	2*nd* byte -	.0	- DT1 trouble
		.1	- DT2 trouble
		.2	- DTM trouble
		.3	- RTC trouble
		.4	- no DTR signal
		.5	- no BATT present
		.6	- external modem initialization trouble
		.7	- external model command (ATE0V1Q0H0S0=0) trouble
	3*rd* byte -	.0	- no voltage on telephone line (INTEGRA 24, 32, 64 and 128)
		.0	- auxiliary ST processor trouble (INTEGRA WRL)
		.1	- bad signal on telephone line
		.2	- no signal on telephone line
		.3	- monitoring to station 1 trouble
		.4	- monitoring to station 2 trouble
		.5	- EEPROM or access to RTC trouble
		.6	- RAM memory trouble
		.7	- INTEGRA main panel restart memory

Part 2 - INTEGRA control:

0x80 arm in mode 0: + 8 bytes - user code (with prefix, if required by INTEGRA), *e.g.:*
if code is '1234', no prefixes: 0x12, 0x34, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF
if code is '1234', prefix is '97': 0x97, 0x12, 0x34, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF
+ 4 bytes - partition list, *e.g.:*
if partition 1, 2, and 29 have to be armed: 0x03, 0x00, 0x00, 0x10
If function is accepted, function result can be checked by observe the system state

0x81 arm in mode 1 *data structure as above*
If function is accepted, function result can be checked by observe the system state

0x82 arm in mode 2 *data structure as above*
If function is accepted, function result can be checked by observe the system state

0x83 arm in mode 3 *data structure as above*
If function is accepted, function result can be checked by observe the system state

0x84 disarm *data structure as above*
If function is accepted, function result can be checked by observe the system state

0x85 clear alarm *data structure as above*
If function is accepted, function result can be checked by observe the system state

0x86 zones bypass + 8 bytes - user code - *see example for 0x80*
+ 16 bytes - zone list, *e.g.:*
if zone 1, 3, 62 and 120 have to be bypassed:
0x05, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x20,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x80, 0x00
If function is accepted, function result can be checked by observe the system state

0x87 zones unbypass *data structure as above*
If function is accepted, function result can be checked by observe the system state

0x88 outputs on + 8 bytes - user code - *see example for 0x80*
+ 16 bytes - output list - *see example for 0x86*
If function is accepted, function result can be checked by observe the system state

0x89 outputs off *data structure as above*
If function is accepted, function result can be checked by observe the system state

0x8A open door

+ 8 bytes - user code - *see example for 0x80*
+ 16 bytes - output list - *see example for 0x86* - outputs of a 101 type can be 'opened'
+ 8 bytes - expander list, *e.g.:*
if expander address 4 and 63 doors have to be opened:
0x10, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x80
If function is accepted, function result can be checked by observe the system state

0x8B clear trouble mem.

+ 8 bytes - user code - *see example for 0x80*
If function is accepted, function result can be checked by observe the system state

0x8C read event

+ 3 bytes - last event index. To start reading event log call this function with these 3 bytes equal 0xFF - the last event will be returned. To read previous event, call this function with event index returned by this function and so on.

Function result - 15 bytes in the following format:
1 byte - 0x8C
8 bytes - event record - see the table below
3 bytes - event index
3 bytes - event index used to call the function

Bit:	.7	.6	.5	.4	.3	.2	.1	.0
1st byte	Y	Y	Z	E	S2	S2	S1	S1
2nd byte	K	K	K	D	D	D	D	D
3rd byte	M	M	M	M	T	T	T	T
4th byte	t	t	t	t	t	t	t	t
5th byte	P	P	P	P	P	R	C	C
6th byte	c	c	c	c	c	c	c	c
7th byte	n	n	n	n	n	n	n	n
8th byte	S	S	S	u	u	u	u	u

YY - year marker (i.e. YEAR mod 4, e.g. 2007 mod 4 = 3, 2008 mod 4 = 0)
Z - 1 = record not empty
E - 1 = event present (normally ZE should be both 00 or 11)
S2 - monitoring to station 2 status:
00 - new event, not processed by monitoring service
01 - event sent
10 - should not occur
11 - event not monitored
S1 - monitoring to station 1 status - description as above
KKK - event class:
000 - zone and tamper alarms
001 - partition and expander alarms
010 - arming, disarming, alarm clearing
011 - zone bypasses and unbypasses
100 - access control
101 - troubles
110 - user functions
111 - system events
DDDDD - day of the month (1..31)
MMMM - month (1..12)
TTTTtttttttt - time in minutes (e.g. 17:53 = 17*60+53 = 1073)
PPPPP - partition number
R - 1 = restore
CCcccccccc - event code
nnnnnnnn - source number (e.g. zone number, user number)
SSS - object number (0..7)
uuuuu - user control number

Part 3 - users management:

General numbering scheme in INTEGRA is as follow:

1..240	- number of user (max. value depends on INTEGRA type)
241..248	- number of master (max. value depends on INTEGRA type)
255	- number of service

0xE0 read self-info + 4 bytes - user code (without prefix), *e.g.: if code is '1234': 0x12, 0x34, 0xFF, 0xFF*

Function result - 30 bytes:

- 1 byte - 0xE0
- 1 byte - user number:
  - 1..240 - user
  - 241..248 - master
  - 255 - service
- 2 bytes -
  - if user - user telephone code
  - if master - 0x00, 0x00
  - if service - 1*st* byte - existing masters, 2*nd* byte - 0x00
- 4 bytes - user partitions
- 1 byte - 0000TTTT: TTTT - user type:
  - 0 - normal
  - 1 - single
  - 2 - time renewable
  - 3 - time not renewable
  - 4 - duress
  - 5 - mono outputs
  - 6 - bi outputs
  - 7 - partitions temporary blocking
  - 8 - access to cash machine
  - 9 - guard
  - 10 - schedule
- 1 byte - user time
- 3 bytes - user rights:
  - 1*st* byte -
    - .0 - arming
    - .1 - disarming
    - .2 - alarm clearing in own partitions
    - .3 - alarm clearing in own object
    - .4 - alarm clearing in whole system
    - .5 - arm deferring
    - .6 - code changing
    - .7 - users editing
  - 2*nd* byte -
    - .0 - zones bypassing
    - .1 - clock setting
    - .2 - troubles viewing
    - .3 - events viewing
    - .4 - zones resetting
    - .5 - options changing
    - .6 - tests
    - .7 - downloading
  - 3*rd* byte -
    - .0 - can always disarm (i.e. even if armed by other user)
    - .1 - voice messaging clearing
    - .2 - GuardX using
    - .3 - access to temporary blocked partitions
    - .4 - entering 1*st* code
    - .5 - entering 2*nd* code
    - .6 - outputs control
    - .7 - clearing latched outputs
- 16 bytes - user name
- 1 byte -
  - if user - object number (0..7)
  - if master - object number (0..7)
  - if service - 0xFF

0xE1 read user
+ 4 bytes - user code
+ 1 byte - user number to read (1..240 - user, 241..248 - master)
Function result - 29 bytes:
1 byte - 0xE1
1 byte - user number:
    1..240 - user
    241..248 - master
    255 - service
4 bytes - user partitions
1 byte - XY00TTTT: TTTT - user type:
    0 - normal
    1 - single
    2 - time renewable
    3 - time not renewable
    4 - duress
    5 - mono outputs
    6 - bi outputs
    7 - partitions temporary blocking
    8 - access to cash machine
    9 - guard
    10 - schedule
  X - 1=user did not change own code after it was created
  Y - 1=other user tried to change own code to this user code
1 byte - user time
1 byte - user time - temporary value - valid only for schedule user
3 bytes - user rights - *see description for 0xE0*
16 bytes - user name
1 byte - if user - object number (0..7)
    if master - object number (0..7)
    if service - 0xFF

0xE2 read users list
+ 4 bytes - user code
+ 1 byte - user number (1..248) which users list is to be read
Function result - 62 bytes:
1 byte - 0xE2
1 byte - user number
30 bytes - list of all existing users
30 bytes - list of users that can be edited by this user

0xE3 read user locks
+ 4 bytes - user code
+ 1 byte - user number (1..248) which locks are to be read
Function result - 10 bytes:
1 byte - 0xE3
1 byte - user number
8 bytes - list of user locks

0xE4 write user locks
+ 4 bytes - user code
+ 1 byte - user number (1..248) which locks are to be written
+ 8 bytes - list of user locks

0xE5 remove user
+ 4 bytes - user code
+ 1 byte - user number (1..248) to remove

0xE6 create user
+ 4 bytes - user code
+ 1 byte - user number (1..248) to create, 255 - auto
+ 4 bytes - user-to-create code
+ 2 bytes - user-to-create telephone code - *4 x BCD or 0xFFFF*
+ 4 bytes - user-to-create partitions
+ 1 byte - user-to-create type
+ 1 byte - user-to-create time
+ 1 byte - user-to-create temporary time - valid only for schedule user
+ 1 byte - user-to-create 1*st* byte of rights
+ 1 byte - user-to-create 2*nd* byte of rights
+ 1 byte - user-to-create 3*rd* byte of rights
+ 16 byte - user-to-create name
+1 byte - user-to-create object - valid only if service is the creator

0xE7 change user

+ 4 bytes - user code
+ 1 byte - user number (1..248) to change
+ 4 bytes - user-to-change code - *will not be changed if equal 0xFFFFFFFF*
+ 2 bytes - user-to-change telephone code - *will not be changed if equal 0xFFFF*
+ 4 bytes - user-to-change partitions
+ 1 byte - user-to-change type
+ 1 byte - user-to-change time
+ 1 byte - user-to-change temporary time - valid only for schedule user
+ 1 byte - user-to-change 1*st* byte of rights
+ 1 byte - user-to-change 2*nd* byte of rights
+ 1 byte - user-to-change 3*rd* byte of rights
+ 16 byte - user-to-change name

0xEE read device name

+ 1 byte - device type to read:

0	- partition (1..32)
1	- zone (1..128)
2	- user (1..255)
3	- expander/LCD (129..192 - expander, 193..210 - LCD)
4	- output (1..128)

+ 1 byte - device number to read
Function result - 20 bytes:

1 byte	- 0xEE
1 byte	- device type - *see above*
1 byte	- device number - *see above*
1 byte	- device type/function:

if partition	- partition type - *see e.g. DloadX for partition types list*
if zone	- zone reaction - *see e.g. DloadX for zone reactions list*
if user	- 0
if expander	- expander type:

1	- CA-64 PP
2	- CA-64 E
3	- CA-64 O
4	- CA-64 EPS
5	- CA-64 OPS
6	- CA-64 ADR
7	- INT-ORS
8	- INT-S/SK
9	- INT-SZ/SZK
10	- CA-64 DR
11	- CA-64 SR
12	- ACU-100
13	- INT-IORS
14	- CA-64 Ei
15	- CA-64 SM
16	- CA-64 AV
17	- INT-IT
18	- CA-64 EPSi
19	- INT-SCR
20	- INT-ENT
21	- INT-RX

if LCD	- "LCD" type:

1	- INT-KLCD
2	- INT-KLCDR
3	- CA-64 PTSA
4	- INT-RS
5	- ETHM-1

if output	- output function - *see e.g. DloadX for output functions list*

16 bytes	- device name

If any function of 0xE0..0xE7, 0xEE does not return result or was not successful, the following result code is returned:

0xEF result + 1 byte - result code:

0x00	- ok
0x01	- requesting user code not found
0x02	- no access
0x03	- selected user does not exist
0x04	- selected user already exists
0x05	- wrong code or code already exists
0x06	- telephone code already exists
0x08	- other error
0x8?	- other errors
0xFF	- function accepted (i.e. data length and crc ok), will be processed next

Please pay attention that INT-RS module should return an answer on **every** request - function result or 0xEF result (described above), so after sending any request to the module please wait for answer before sending next request (or give the module e.g. 3 seconds time-out).