AR-7030 Computer remote control protocol.

Information for firmware releases 1.1A, 1.2A, 1.4A and 1.4B

1) Remote control overview.

The AR-7303 receiver allows remote control of all of its functions by means of a direct memory access system.
A controlling computer can read and modify the internal memory maps of the receiver to set required parameters and then call for the receiver's control program to process the new settings.
Commands to the receiver are byte structured in binary format, so it is not possible to control from a terminal.
All multi-byte numbers within the receiver are binary, stored msb first.

2) Receiver frequency configuration.

Receive frequency is set by two oscillators - local and carrier. In AM and FM modes the carrier oscillator is not used, and the final IF frequency is 455kHz. In Sync mode the carrier oscillator is offset by +20.29kHz before mixing with the IF.
The IF frequencies have a fixed inter-conversion frequency of 44.545MHz and, because of the high-side local oscillator, both IF's are inverted.

The receiver controller processes the following variables to establish the tuned frequency :-

[*local offset*] Frequency shift applied to local oscillator.
[*carrier offset*] 455.00kHz for LSB, USB, Data and CW modes / 434.71kHz for Sync mode.
[*filter offset*] IF Filter frequency at the (vestigial) carrier position as an offset from 455kHz.
[PBS] User set filter shift.
[BFO] User set offset between carrier position and frequency display.
[TUNE] Receiver tuned frequency as shown on display.

The relationship between these variables and the tuning is as follows :-

[*carrier offset*] + [*filter offset*] + [PBS] + [BFO] ——> Carrier oscillator
45.000MHz + [*filter offset*] + [PBS] ——> [*local offset*]
[TUNE] + [*local offset*] ——> Local oscillator

3) Serial data protocol.

All data transfers are at 1200 baud, No parity, 8 bits, 1 stop bit (1200 N 8 1). There is no hardware or software flow control other than that inherent in the command structure. The receiver can accept data at any time at full rate provided the IR remote controller is not used or is disabled. A maximum of one byte can be transmitted for each byte received, so data flow into a controlling computer is appropriately limited.
Each byte sent to the receiver is a complete command - it is best thought of as two hexadecimal digits - the first digit is the *operation code*, the second digit is 4-bits of *data* relating to the operation. Because the receiver operates with 8-bit bytes, intermediate 4-bit values are stored in *registers* in the receiver for recombination and processing. For example to write into the receiver's memory, the following steps would be followed :-

a) Send address high order 4-bits into *H-register*
b) Send address low order 4-bits and set *Address register*
c) Send first data byte high order 4-bits into *H-register*
d) Send first data byte low order 4-bits and execute *Write Data Operation*
e) Send second data byte high order 4-bits into *H-register*
f) Send second data byte low order 4-bits and execute *Write Data Operation*
g) Repeat (e) and (f) for each subsequent byte to be written.

4) Memory organisation.

Different memory areas in the receiver are referenced by selecting *Pages* - up to 16 pages are supported.
The memory is broadly divided into 3 sections :-

a) Working memory - where all current operating variables are stored and registers and stack are located. This memory is volatile and data is lost when power to the receiver is removed.
b) Battery sustained memory - where duplicate parameters are stored for retention when power is removed. This memory area is also used for storage of filter parameters, setup memories and squelch and BFO settings for the frequency memories and contains the real time clock registers.
c) EEPROM - where frequency, mode, filter and PBS information for the frequency memories is stored. Additionally S-meter and IF calibration values are stored here. This memory can be read or written to download and upload the receiver's frequency memories, but repetitive writing should be avoided because the memory devices will only support a finite number of write cycles.

5) Variations between A and B types and firmware revisions.

Type A firmware supports only basic receiver functions, type B extends operations and includes support for the Notch / Noise Blanker option. The whole of the type A memory map is retained in type B, but more memory and operations are added for the extended functions of type B.

In the following information, circled note numbers are included to indicate where items are specific to one type or
revision of the firmware:-

❶ Applicable to type B firmware only.

❷ Applicable to revision 1.4 only, types A and B

❸ Function is changed or added to in type B

6) Operation codes.

The high order 4-bits of each byte sent to the receiver is the *operation code*, the low order 4-bits is *data* (shown here as *x*) :-

Code	Ident		Operation	
0 *x*	NOP		No Operation	
3 *x*	SRH		Set H-register	*x* —> *H-register* (4-bits)
5 *x*	PGE		Set page	*x* —> *Page register* (4-bits)
4 *x*	ADR		Set address	0*Hx* —> *Address register* (12-bits) 0 —> *H-register*
1 *x*	ADH		Set address high	*x* —> *Address register* (high 4-bits)
6 *x*	WRD		Write data	*Hx* —> [Page, Address] *Address register* + 1 —> *Address register* 0 —> *H-register,* 0 —> *Mask register*
9 *x*	MSK	❶	Set mask	*Hx* —> *Mask register* 0 —> *H-register*
2 *x*	EXE		Execute routine *x*	
A *x*	BUT	❶	Operate button *x*	
7 *x*	RDD		Read data	[Page, Address] —> Serial output *Address register* + *x* —> *Address register*
8 *x*	LOC		Set lock level *x*	

Note that the *H-register* is zeroed after use, and that the high order 4-bits of the *Address register* must be set (if non-zero) after the low order 8-bits. The *Address register* is automatically incremented by one after a write data operation and by *x* after a read data operation.

When writing to any of the EEPROM memory pages a time of 10ms per byte has to be allowed. For this reason it is recommended that instructions SRH and WRD are always used together (even if the SRH is not needed) since this will ensure that the EEPROM has sufficient time to complete its write cycle.

Additionally to allow time for local receiver memory updates and SNC detector sampling in addition to the EEPROM write cycle, it is recommended to lock the receiver to level 2 or 3, or add a NOP instruction after each write. This is not required for firmware revision 1.4 but locking is still recommended.

The mask operation helps with locations in memory that are shared by two parameters and aids setting and clearing bits. The mask operates only in Page 0. If bits in the mask are set, then a following write operation will leave the corresponding bits unchanged. The mask register is cleared after a write so that subsequent writes are processed normally. Because it defaults to zero at reset, the mask is inoperative unless specifically set.

The operate button instruction uses the same button codes as are returned from routine 15 (see section 8), with an additional code of zero which operates the *power* button, but will not switch the receiver off. Also code 0 will switch the receiver on (from standby state).

7) Memory pages.

Page 0		Working memory (RAM)	256 bytes.
Page 1		Battery sustained memory (RAM)	256 bytes.
Page 2		Non-volatile memory (EEPROM)	512 bytes.
Page 3	❶	Non-volatile memory (EEPROM)	4096 bytes.
Page 4	❶	Non-volatile memory (EEPROM)	4096 bytes.
Pages 5 - 14		Not assigned.	
Page 15		Receiver Ident (ROM)	8 bytes.

The ident is divided into model number (5 bytes), software revision (2 bytes) and type letter (1 byte).

eg 7030_14A —> Model AR-7030, revision 1.4, type letter A.

8) Lock levels.

Level 0	Normal operation.
Level 1	IR remote control disabled. Front panel buttons ignored.

	Front panel spin-wheels logged but not actioned. Display update (frequency & S-meter) continues.
Level 2	As level 1, but display update suspended. In revisions before 1.4 squelch operation is inhibited, which results in no audio output after a mode change. In revision 1.4 squelch operation continues and mode changing is as expected.
Level 3	Remote operation exclusively.

Lock level 1 is recommended during any multi-byte reads or writes of the receiver's memory to prevent data contention between internal and remote memory access. See also EEPROM notes in section (6)

8) Routines.

Routine 0		Reset	Setup receiver as at switch-on.
Routine 1		Set frequency	Program local oscillator from *frequ* area and setup RF filters and oscillator range.
Routine 2		Set mode	Setup from *mode* byte in memory and display mode, select preferred filter and PBS, BFO values etc.
Routine 3		Set passband	Setup all IF parameters from *filter*, *pbsval* and *bfoval* bytes.
Routine 4		Set all	Set all receiver parameters from current memory values
Routine 5	❷	Set audio	Setup audio controller from memory register values.
Routine 6	❷	Set RF-IF	Setup RF Gain, IF Gain and AGC speed. Also sets Notch Filter and Noise Blanker if these options are fitted.
Routine 7		Not assigned	
Routine 8		Not assigned	
Routine 9		Direct Rx control	Program control register from *rxcon* area.
Routine 10		Direct DDS control	Program local oscillator and carrier oscillator DDS systems from *wbuff* area. The 32-bits at *wbuff* control the carrier frequency, value is 385674.4682 / kHz. The 32 bits at *wbuff+4* control the local osc frequency, value is 753270.4456 / MHz.
Routine 11		Display menus	Display menus from *menu1* and *menu2* bytes.
Routine 12		Display frequency	Display frequency from *frequ* area.
Routine 13		Display buffer	Display ASCII data in *wbuff* area. First byte is display address, starting at 128 for the top line and 192 for the bottom line. An address value of 1 clears the display. Data string (max length 24 characters) ends with a zero byte.
Routine 14		Read signal strength	Transmits byte representing received signal strength (read from AGC voltage). Output is 8-bit binary in range 0 to 255.
Routine 15		Read buttons	Transmits byte indicating state of front panel buttons. Output is 8-bit binary with an offset of +48 (ie ASCII numbers). Buttons held continuously will only be registered once.

Button codes :-

0 = None pressed	5 = RF-IF button
1 = Mode up button	6 = Memory button
2 = Mode down button	7 = * button
3 = Fast button	8 = Menu button
4 = Filter button	9 = Power button

Note that the work buffer *wbuff* area in memory is used continuously by the receiver unless lock levels 2 or 3 are invoked. Lock levels of 1 or more should be used when reading any front panel controls to prevent erratic results.

10) Battery sustained RAM (Memory page 1)

Address		Ident	Length	Description
0	00		13 bytes	Real time clock / timer registers :-
0	00	rt_con	1 byte	Clock control register
2	02	rt_sec	1 byte	Clock seconds (2 BCD digits)
3	03	rt_min	1 byte	Clock minutes (2 BCD digits)
4	04	rt_hrs	1 byte	Clock hours (2 BCD digits - 24 hr format)
5	05	rt_dat	1 byte	Clock year (2 bits) and date (2 BCD digits)
6	06	rt_mth	1 byte	Clock month (2 BCD digits - low 5 bits only)
8	08	tm_con	1 byte	Timer control register
10	0A	tm_sec	1 byte	Timer seconds (2 BCD digits)
11	0B	tm_min	1 byte	Timer minutes (2 BCD digits)
12	0C	tm_hrs	1 byte	Timer hours (2 BCD digits - 24 hr format)
13	0D		15 bytes	Power-down save area :-

13	0D	ph_cal		1 byte	Sync detector phase cal value
14	0E	pd_slp		1 byte	Timer run / sleep time in minutes
15	0F	pd_dly		1 byte	Scan delay value x 0.125 seconds
16	10	pd_sst		1 byte	Scan start channel
17	11	pd_ssp		1 byte	Scan stop channel
18	12	pd_stp		2 bytes	Channel step size
20	14	pd_sql		1 byte	Squelch
21	15	pd_ifg		1 byte	IF gain
22	16	pd_flg		1 byte	Flags (from *pdflgs*)
23	17	pd_frq		3 bytes	Frequency
26	1A	pd_mod	❸	1 byte	Mode (bits 0-3) and NB threshold (bits 4-7)
27	1B	pd_vol	❸	1 byte	Volume (bits 0-5) and rx memory hundreds (bits 6&7)
28	1C			26 bytes	Receiver setup save area :-
28	1C	md_flt		1 byte	AM mode : Filter (bits 0-3) and AGC speed (bits 4-7)
29	1D	md_pbs		1 byte	AM mode : PBS value
30	1E	md_bfo		1 byte	AM mode : BFO value
31	1F			3 bytes	Ditto for Sync mode
34	22			3 bytes	Ditto for NFM mode - except Squelch instead of BFO
37	25			3 bytes	Ditto for Data mode
40	28			3 bytes	Ditto for CW mode
43	2B			3 bytes	Ditto for LSB mode
46	2E			3 bytes	Ditto for USB mode
49	31	st_aud	❸	1 byte	Audio bass setting (bits 0-4)
					bit 5 Notch auto track enable
					bit 6 Ident search enable
					bit 7 Ident preview enable
50	32			1 byte	Audio treble setting (bits 0-3) and RF Gain (bits 4-7)
51	33			1 byte	Aux output level - left channel
52	34			1 byte	Aux output level - right channel
53	35	st_flg		1 byte	Flags (from *stflgs*)
54	36			26 bytes	Setup memory A (configured as above)
80	50			26 bytes	Setup memory B (configured as above)
106	6A			26 bytes	Setup memory C (configured as above)
132	84			24 bytes	Filter data area :-
132	84	fl_sel		1 byte	Filter 1 : selection bits and IF bandwidth
133	85	fl_bw		1 byte	Filter 1 : bandwidth (2 BCD digits, x.x kHz)
134	86	fl_uso		1 byte	Filter 1 : USB offset value x 33.19Hz
135	87	fl_lso		1 byte	Filter 1 : LSB offset value x 33.19Hz
136	88			4 bytes	Ditto for filter 2
140	8C			4 bytes	Ditto for filter 3
144	90			4 bytes	Ditto for filter 4
148	94			4 bytes	Ditto for filter 5
152	98			4 bytes	Ditto for filter 6
156	9C	mem_sq		100 bytes	Squelch / BFO values for frequency memories 0 to 99 (BFO for Data and CW modes, Squelch for others)

11) EEPROM (Memory page 2)

Address		Ident	Length	Description
0	000		4 bytes	Frequency memory data :-
0	000	mem_fr	3 bytes	Memory 00 : 24-bit frequency
3	003	mem_md	1 byte	bits 0 - 3 mode
				bits 4 - 6 filter
				bit 7 scan lockout
4	004		396 bytes	Ditto for memories 01 to 99
400	190	mem_pb	100 bytes	PBS values for frequency memories 0 to 99
500	1F4	sm_cal	8 bytes	S-meter calibration values :-
500	1F4		1 byte	RSS offset for S1 level
501	1F5		1 byte	RSS steps up to S3 level
502	1F6		1 byte	RSS steps up to S5 level
503	1F7		1 byte	RSS steps up to S7 level
504	1F8		1 byte	RSS steps up to S9 level
505	1F9		1 byte	RSS steps up to S9+10 level
506	1FA		1 byte	RSS steps up to S9+30 level
507	1FB		1 byte	RSS steps up to S9+50 level
508	1FC	if_cal	2 bytes	RSS offsets for -20dB and -8dB filter alignment

510	1FE	if_def		1 byte	Default filter numbers for narrow and wide (2 BCD digits)
511	1FF	option	❶	1 byte	Option information :-
					bit 0 Noise blanker
					bit 1 Notch filter
					bit 2 10 dB step attenuator (DX version)

12) EEPROM (Memory page 3) ❶

Address		Ident	Length	Description
0	000		4 bytes	Frequency memory data :-
0	000	mex_fr	3 bytes	Memory 100 : 24-bit frequency
3	003	mex_md	1 byte	bits 0 - 3 mode
				bits 4 - 6 filter
				bit 7 scan lockout
4	004		1196 bytes	Ditto for memories 101 to 399
1200	4B0		8 bytes	Timer memory data :-
1200	4B0	mtm_mn	1 byte	Timer memory 0 : minutes (2 BCD digits)
1201	4B1	mtm_hr	1 byte	hours (2 BCD digits)
1202	4B2	mtm_dt	1 byte	date (2 BCD digits)
1203	4B3	mtm_mt	1 byte	month (2 BCD digits)
1204	4B4	mtm_ch	2 bytes	rx channel (hundreds and 0-99)
1206	4B6	mtm_rn	1 byte	run time
1207	4B7	mtm_ac	1 byte	active (0 = not active)
1208	4B8		72 bytes	Ditto for timer memories 1 to 9
1280	500		16 bytes	Frequency memory data :-
1280	500	mex_sq	1 byte	Memory 0 : Squelch / BFO (not used for mems 0 to 99) (BFO for Data and CW modes)
1281	501	mex_pb	1 byte	PBS value (not used for mems 0 to 99)
1282	502	mex_id	14 bytes	Text Ident
1296	510		2800 bytes	Ditto for memories 1 to 175

13) EEPROM (Memory page 4) ❶

Address		Ident	Length	Description
0	000		16 bytes	Frequency memory data :-
0	000		1 byte	Memory 176 : Squelch / BFO (BFO for Data and CW modes)
1	001		1 byte	PBS value
2	002		14 bytes	Text Ident
16	010		3568 bytes	Ditto for memories 177 to 399
3584	E00	mex_hx	400 bytes	Frequency fast find index (1 byte for each memory 0 to 399) Index value is bits 9 to 16 of 24-bit frequency stored in each memory. Empty memories (frequency zero) should have a random index byte.
3984	F90		112 bytes	spare

14) Working memory (Memory page 0)

Areas not specifically addressed are used as workspace by the internal processor. - Keep out (by order).

Address		Ident	Length	Description
16	10	snphs	1 byte	Sync detector phase offset cal value
17	11	slptim	1 byte	Sleep time (minutes)
18	12	scnst	1 byte	Scan start channel
19	13	scnsp	1 byte	Scan stop channel
20	14	scndly	1 byte	Scan delay time value x 0.125 seconds
21	15	chnstp	2 bytes	16-bit channel step size, value is 376.6352 / kHz
23	17	sqlsav	1 byte	Squelch save value (non-fm mode)
24	18	ifgain	1 byte	IF gain value (zero is max gain)
26	1A	frequ	3 bytes	24-bit tuned frequency, value is 376635.2228 / MHz.
29	1D	mode	1 byte	Current mode :- 1 = AM 4 = Data 2 = Sync 5 = CW 3 = NFM 6 = LSB 7 = USB

30	1E			10 bytes				Audio control registers :-
30	1E	af_vol		1 byte				Main channel volume (6-bits, values 15 to 63)
31	1F	af_vll		1 byte				Left channel balance (5-bits, half of volume value above)
32	20	af_vlr		1 byte				Right channel balance (as above)
33	21	af_bas	❸	1 byte				Main channel bass (bits 0-4, values 6 to 25, 15 is flat)
						bit 5	nchtrk	Notch auto track enable
						bit 6	idauto	Ident auto search enable
						bit 7	idprev	Ident auto preview enable
34	22	af_trb	❸	1 byte				Main channel treble (bits 0-3, values 2 to 10, 6 is flat)
						bit 4	nb_opt	Noise blanker menus enabled
						bit 5	nt_opt	Notch Filter menus enabled
						bit 6	step10	10dB RF attenuator fitted
35	23	af_axl		1 byte				Left aux channel level (bits 0-5, values 27 to 63)
36	24	af_axr	❸	1 byte				Right aux channel level (bits 0-5, values 27 to 63)
						bit 7	nchsr	Notch search running
37	25	af_axs	❸	1 byte				Aux channel source (bits 0-3)
						bit 4	nchen	Notch filter active
						bit 5	nchsig	Notch filter signal detected
						bit 6	axmut	Aux output mute
						bit 7	nchato	Notch auto tune active
38	26	af_opt	❸	1 byte				Option output source (bits 0-3)
						bit 4	idover	Ident on LCD over frequency
						bit 5	idsrdn	Ident search downwards
						bit 7	idsrch	Ident search in progress
39	27	af_src		1 byte				Main channel source
						bit 6	afmut	Main output mute
40	28	rxcon		3 bytes				Receiver control register mapping :-
					byte 1	bit 0	rx_fs3	Filter select : FS3
					byte 1	bit 1	rx_fs2	Filter select : FS2
					byte 1	bit 2	rx_fs1	Filter select : FS1
					byte 1	bit 3	rx_fs4	Filter select : FS4
					byte 1	bit 4	rx_pre	Preamplifier enable
					byte 1	bit 5	rx_atr	Atten : 0 = 20dB / 1 = 40dB
					byte 1	bit 6	rx_rff	Input filter : 0 = HF / 1 = LF
					byte 1	bit 7	rx_atn	Attenuator enable
					byte 2	bit 0	rx_as1	AGC speed : 00 = Slow
					byte 2	bit 1	rx_as2	10 = Med
								11 = Fast
					byte 2	bit 2	rx_agi	AGC inhibit
					byte 2	bit 3	rx_en	LO and HET enable
					byte 2	bit 4	rx_aux	Aux relay enable
					byte 2	bit 5	rx_fs5	Filter select : FS5
					byte 2	bit 6	rx_fs6	Filter select : FS6
					byte 2	bit 7	rx_ibw	IF b/w : 0 = 4kHz / 1 = 10kHz
					byte 3	bit 0	rx_chg	Fast charge enable
					byte 3	bit 1	rx_pwr	PSU enable
					byte 3	bit 2	rx_svi	Sync VCO inhibit
					byte 3	bit 3	rx_agm	AGC mode : 0 = peak / 1 = mean
					byte 3	bit 4	rx_lr1	LO range : 00 = 17 - 30 MHz
					byte 3	bit 5	rx_lr2	10 = 10 - 17 MHz
								01 = 4 - 10 MHz
								11 = 0 - 4 MHz
					byte 3	bit 6	rx_sbw	Sync b/w : 0 = Wide / 1 = Narrow
					byte 3	bit 7	rx_car	Car sel : 0 = AM / 1 = DDS
43	2B	bits		3 bytes				General flags :-
					byte 1	bit 6	lock1	Level 1 lockout
					byte 1	bit 7	lock2	Level 2 lockout
					byte 2	bit 0	upfred	Update frequency display
					byte 2	bit 1	upmend	Update menus

					byte 2	bit 2	tune4x	Tune 4 times faster (AM & NFM)
					byte 2	bit 3	quickly	Quick tuning (fast AGC, Sync)
					byte 2	bit 4	fast	Fast tuning mode
					byte 2	bit 5	sncpt1	Auto sync - frequency lock
					byte 2	bit 6	sncpt2	Auto sync - phase lock
					byte 2	bit 7	sncal	Sync detector calibrating
					byte 3	bit 0	sqlch	Squelch active (ie low signal)
					byte 3	bit 1	mutsql	Mute on squelch (current setting)
					byte 3	bit 2	bscnmd	Scan mode for VFO B
					byte 3	bit 3	dualw	Dual watch active
					byte 3	bit 4	scan	Scan active
					byte 3	bit 5	memlk	Current memory scan lockout
					byte 3	bit 6	pbsclr	Enable PBS CLR from IR remote
				❷	byte 3	bit 7	memodn	MEM button scans downwards
46	2E	pdflgs	1 byte					Flags saved at power-down :-
						bit 0	power	Power on
						bit 1	flock	Tuning locked
						bit 2	batop	Battery operation (for fast chg)
				❶		bit 3	nben	Noise blanker active
				❶		bit 4	nblong	Noise blanker long pulse
47	2F	stflgs	1 byte					Flags saved in setup memories :-
						bit 0	mutsav	Mute on squelch (non-fm mode)
						bit 1	mutaux	Mute aux output on squelch
						bit 2	axren	Aux relay on timer
						bit 3	axrsql	Aux relay on squelch
						bit 4	snauto	Auto sync mode
						bit 5	snarr	Sync detector narrow bandwidth
						bit 6	scanmd	Scan runs irrespective of squelch
						bit 7	autorf	RF gain auto controlled
48	30	rfgain	1 byte					Current RF gain setting (0 to 5) (0=max gain)
49	31	rfagc	1 byte					Current RF AGC setting (added to above)
50	32	agcspd	1 byte					Current AGC speed : 0 = Fast 2 = Slow 1 = Medium 3 = Off
51	33	sqlval	1 byte					Squelch value (current setting)
52	34	filter	1 byte					Current filter number (1 to 6)
53	35	pbsval	1 byte					PBS offset (x33.19Hz)
54	36	bfoval	1 byte					BFO offset (x33.19Hz)
55	37	fltofs	1 byte					Filter centre frequency offset (x33.19Hz)
56	38	fltbw	1 byte					Filter bandwidth (2 BCD digits : x.x kHz)
57	39	ircode:	2 bytes					Current / last IR command code
59	3B	spnpos	1 byte					Misc spin-wheel movement } 0 = no movement
60	3C	volpos	1 byte					Volume control movement } +ve = clockwise
61	3D	tunpos	1 byte					Tuning control movement } -ve = anti-clockwise
62	3E	lstbut	1 byte					Last button pressed
63	3F	smval	2 bytes					Last S-meter reading (bars + segments)
65	41	mestmr	1 byte					Message time-out timer
66	42	rfgtmr	1 byte					RF gain delay timer
67	43	updtmr	1 byte					Sustained RAM update timer
68	44	agctmr	1 byte					AGC speed restore delay timer
69	45	snctmr	1 byte					Auto sync refresh timer
70	46	scntmr	1 byte					Scan delay timer
71	47	irdly	1 byte					IR remote auto repeat delay counter
72	48	runtmr	1 byte					Sleep mode timer
73	49	snfrq	1 byte					Sync detector frequency offset cal value
74	4A	frange	1 byte					Input / LO range

75	4B	menu1	❸	1 byte	Current left menu (type A and B menu numbers are different)
76	4C	menu2	❸	1 byte	Current right menu (type A and B menu numbers are different)
77	4D	memno		1 byte	Current memory number
78	4E	setno		1 byte	Setup / config selection - load / save
85	55	mempg	❶	1 byte	Memory page (hundreds - value 0 to 3)
86	56	nbthr	❶	1 byte	Noise blanker threshold (values 0 to 15)
87	57	hshfr	❶	1 byte	Current tuned frequ index value (during ident search)
88	58	nchtmr	❶	1 byte	Notch filter auto tune / search timer
90	59	wbuff		26 bytes	Work buffer
115	73	keymd		1 byte	IR remote +/- keys function
116	74	keybuf		20 bytes	IR remote key input buffer
136	88	frofs:		4 bytes	32-bit local osc offset
140	8C	carofs		4 bytes	32-bit carrier osc offset
144	90	smofs		1 byte	S-meter starting offset
145	91	smscl		7 bytes	S-meter segment values
152	98	ifcal		2 bytes	RSS offsets for -20dB and -5dB filter alignment
154	9A	ifdef		1 byte	Default filter numbers for narrow and wide (2 digits)
155	9B	vfo_b		22 bytes	VFO B storage area :-
155	9B			1 byte	B : Scan delay time
156	9C			2 bytes	B : Channel step size
158	9E			1 byte	B : Squelch save value (non-fm mode)
159	9F			1 byte	B : IF gain value
160	A0			1 byte	not used
161	A1			3 bytes	B : Tuned frequency
164	A4			1 byte	B : Mode
165	A5			1 byte	B : Volume
166	A6			1 byte	B : Left channel balance
167	A7			1 byte	B : Right channel balance
168	A8			1 byte	B : Bass response
169	A9			1 byte	B : Treble response
170	AA			1 byte	B : RF gain
171	AB			1 byte	B : RF AGC
172	AC			1 byte	B : AGC speed
173	AD			1 byte	B : Squelch value
174	AE			1 byte	B : Filter number
175	AF			1 byte	B : PBS offset
176	B0			1 byte	B : BFO offset
218	DA	savmnu	❶	1 byte	Saved menu 1 number during ident display
219	DB	srchm	❶	2 bytes	Ident search memory (page and number)
222	DD	idtmr	❶	1 byte	Auto ident search start timer
223	DE	nchfr	❶	2 bytes	16-bit notch filter frequency, value is 6553.6 / kHz

15) Sample routines (in MS QBASIC)

```
REM   Sample subroutines for communication with the AR-7030 A-type
REM   These subroutines use the following variables :-
REM        rx.freq#       frequency in kHz (double precision)
REM        rx.mode        mode number (1 to 7)
REM        rx.filt        filter number (1 to 6)
REM        rx.mem         memory number (0 to 99)
REM        rx.pbs         passband shift value (-4.2 to +4.2 in kHz)
REM        rx.sql         squelch value (0 to 255)
REM        ident$         model number, revision and type


REM Subroutine to open comms link to receiver
open.link:
       open “com1:1200,n,8,1,cd0,cs0,ds0,rs” for random as #1 len = 1
       field #1, 1 as input.byte$
       return
```

```
REM Subroutine to flush QBASIC serial input buffer
flush.buffer:
       print #1,”//”;
       do
          time.mark# = timer
          do while timer - time.mark# < 0.2
          loop
          if eof(1) then exit do
          get #1
       loop
       return

REM Subroutines to lock and unlock receiver controls
lock.rx:
       print #1,chr$(&H81);                    ‘ Set lockout level 1
       return
unlock.rx:
       print #1,chr$(&H80);                    ‘ Lockout level 0 (not locked)
       return

REM Subroutine to read byte from comms link
read.byte:
       read.value = -1                         ‘ Value assigned for read error
       time.mark# = timer
       print #1,chr$(&H71);                    ‘ Read byte command
       do while timer - time.mark# < 0.3
          if eof(1) = 0 then
                      get #1
                      read.value = asc(input.byte$)
                      exit do
                      end if
       loop
       return

REM Subroutine to set receiver frequency and mode
tune.rx:
       gosub lock.rx
       print #1,chr$(&H50);                    ‘ Select working mem (page 0)
       print #1,chr$(&H31);chr$(&H4A);         ‘ Frequency address = 01AH
       gosub send.freq                         ‘ Write frequency
       print #1,chr$(&H60+rx.mode);            ‘ Write mode
       print #1,chr$(&H24);                    ‘ Tune receiver
       gosub unlock.rx
       return

REM Subroutine to store data into receiver’s frequency memory
set.memory:
       mem.loc = rx.mem+156                    ‘ Squelch memory origin
       mem.h = int(mem.loc/16)
       mem.l = mem.loc mod 16
       print #1,chr$(&H51);                    ‘ Select squelch memory (page 1)
       print #1,chr$(&H30+mem.h);
       print #1,chr$(&H40+mem.l);              ‘ Set memory address
       print #1,chr$(&H30+int(rx.sql/16))
       print #1,chr$(&H60+rx.sql mod 16)       ‘ Write squelch value

       mem.loc = rx.mem*4                      ‘ Frequency memory origin
       mem.t = int(mem.loc/256)
       mem.loc = mem.loc mod 256
       mem.h = int(mem.loc/16)
       mem.l = mem.loc mod 16
       print #1,chr$(&H52);                    ‘ Select frequency memory (page
2)
       print #1,chr$(&H30+mem.h);
       print #1,chr$(&H40+mem.l);              ‘ Set memory address
       print #1,chr$(&H10+mem.t);
       gosub send.freq                         ‘ Write frequency
```

```
        print #1,chr$(&H30+rx.filt);
        print #1,chr$(&H60+rx.mode);                ‘ Write filter and mode

        mem.loc = rx.mem+400-256                    ‘ PBS memory origin
        mem.h = int(mem.loc/16)
        mem.l = mem.loc mod 16
        pbs.val = 255 and int(rx.pbs/0.033189+0.5)
        print #1,chr$(&H30+mem.h);
        print #1,chr$(&H40+mem.l);                  ‘ Set memory address
        print #1,chr$(&H11);
        print #1,chr$(&H30+int(pbs.val/16))
        print #1,chr$(&H60+pbs.val mod 16)          ‘ Write passband value
        return

REM Subroutine to read data from receiver’s frequency memory
read.memory:
        mem.loc = rx.mem+156                        ‘ Squelch memory origin
        mem.h = int(mem.loc/16)
        mem.l = mem.loc mod 16
        print #1,chr$(&H51);                        ‘ Select squelch memory (page 1)
        print #1,chr$(&H30+mem.h);
        print #1,chr$(&H40+mem.l);                  ‘ Set memory address
        gosub read.byte                             ‘ Read squelch value
        rx.sql = read.value

        mem.loc = rx.mem*4                          ‘ Frequency memory origin
        mem.t = int(mem.loc/256)
        mem.loc = mem.loc mod 256
        mem.h = int(mem.loc/16)
        mem.l = mem.loc mod 16
        print #1,chr$(&H52);                        ‘ Select frequency memory (page
2)
        print #1,chr$(&H30+mem.h);
        print #1,chr$(&H40+mem.l);                  ‘ Set memory address
        print #1,chr$(&H10+mem.t);
        gosub read.freq                             ‘ Read frequency
        gosub read.byte                             ‘ Read filter and mode
        if read.value < 0 then return
        rx.filt = int(read.value/16)
        rx.mode = read.value mod 16
        mem.loc = rx.mem+400-256                    ‘ PBS memory origin
        mem.h = int(mem.loc/16)
        mem.l = mem.loc mod 16
        print #1,chr$(&H30+mem.h);
        print #1,chr$(&H40+mem.l);                  ‘ Set memory address
        print #1,chr$(&H11);
        gosub read.byte                             ‘ Read passband value
        if read.value < 0 then return
        if read.value > 127 then read.value = 256-read.value
        rx.pbs = read.value*0.033189
        return

REM Subroutine to read receiver ident string
read.ident:
        print #1,chr$(&H5F);                        ‘ Select ident memory (page 15)
        print #1,chr$(&H40);                        ‘ Set address 0
        ident$=””
        for read.loop = 1 to 8
           gosub read.byte                          ‘ Read 8-byte ident
           if read.value < 0 then exit for
           ident$ = ident$+chr$(read.value)
        next read.loop
        return

REM Subroutine to send frequency  (Called only from other routines)
send.freq:
        fr.val# = int(rx.freq#*376.635223+0.5)      ‘ Convert kHz to steps
```

```
                                               ‘ Exact multiplicand is (2^24)/
44545
       print #1,chr$(&H30+int(fr.val#/1048576));
       fr.val# = fr.val# mod 1048576              ‘ Write frequency as 6 hex dig-
its
       print #1,chr$(&H60+int(fr.val#/65536));
       fr.val# = fr.val# mod 65536
       print #1,chr$(&H30+int(fr.val#/4096));
       fr.val# = fr.val# mod 4096
       print #1,chr$(&H60+int(fr.val#/256));
       fr.val# = fr.val# mod 256
       print #1,chr$(&H30+int(fr.val#/16));
       print #1,chr$(&H60+(fr.val# mod 16));
       return

REM Subroutine to read frequency  (Called only from other routines)
read.freq:
       fr.val# = 0
       for read.loop = 1 to 3
          gosub read.byte                         ‘ Read frequency as 3 bytes
          if read.value < 0 then exit for
          fr.val# = fr.val#*256+read.value
       next read.loop
       rx.freq# = fr.val#/376.635223              ‘ Convert steps to kHz
       return
```