

1 Introduction

The Icom IC-T90A is a handheld transceiver operating on 6 m, 2 m and 70 cm bands. It also receives over a wide range of frequencies (500 kHz to almost 1 GHz, with cellular bands blocked in the American model).

The HT supports a variant of the Icom CI-V protocol for cloning and programming its memory. For background on CI-V see [Ekki Plicht's page](#).

The transceiver uses the center connector of the 3.5 mm speaker jack for data input/output at 5 V TTL levels. When connected to a serial port of a computer (as opposed to another TTL device), a 'level converter' must be used. This is what the Icom programming cable (OPC-478) does. There are many designs on the Web – e.g. on the [site mentioned above](#), and my version described [here](#). Data transfer settings which worked for me were 9600 baud, 8/1/N.

2 Cloning protocol

When the unit is turned on while the SET and MR keys are pressed, it goes into a 'clone out' mode. Now pressing the PTT switch makes it dump the memory contents through the data port. This function can be used to upload the data to a computer, or to another unit connected directly to the master one. Because of the latter, the data dump is preceded by a CI-V command that puts the clone unit(s) in the 'receive' or 'clone in' mode.

This command is (in hex) FE FE EE EE E3 25 07 00 01 FD. In particular, the CI-V address of the T90A seems to be 0xEE.

One can also request the data dump from a unit while it is operating normally, i.e. without pressing the keys while powering on. This is done by sending it the CI-V command FE FE EE EF E2 25 07 00 01 FD (which will make the dump go to device with the address 0xEF, e.g. a computer).

The dump itself consists of CI-V commands FE FE EF EE E4 . . . FD where '.' stands for the command's payload (described next). The last piece of the dump is FE FE EF EE E5 . . . FD 00, in which '.' consists of the string "Icom Inc.", i.e. 49 63 6F 6D 20 49 6E 63 2E.

2.1 Command payload

All the FE FE EF EE E4 . . . FD commands seem to describe the contents of memory locations. The format of the payload is as follows:

- Four bytes representing the starting location of the data being sent. The location is found by converting the four hex numbers to their ASCII equivalents, e.g. the raw hex data 30 35 43 30 means ASCII '05C0', which in turn denotes location 1472 (decimal)¹. In my unit the locations go from '0000' to '2D1F', i.e. a total of 11552 addresses.
- Two bytes denoting the size of the data, typically 32 30. This again should be interpreted as the ASCII string '20', meaning decimal 32. Assuming the starting location as above, we will be setting locations 1472 through 1503 (decimal).
- Data to be stored in memory locations. In this case we now expect 32 bytes represented by ASCII characters; since each byte needs two characters, we will see 64 hex numbers in the data stream. For example, the raw data 30 37 30 41 44 30 will mean that the next three memory locations should be set to '07 0A D0'.
- Checksum computed over the previous blocks, i.e. starting location, size and data. It is computed by taking each pair of bytes in the raw data, converting them to two ascii characters, interpreting the pair as a hex number and subtracting that number as an unsigned char from 0xFF (i.e. taking its 2-s complement). The result from each pair of incoming bytes is added to what's already in the checksum for this payload, modulo 256. The final result, e.g. 0xD1, is as usual represented via its ASCII string, i.e. the raw data contains this checksum as two bytes, 44 31. The only exception seems to be the very last E4 command sent, which addresses 32 locations starting with 11552 (decimal) and contains data whose ASCII encoded bytes correspond to 16 ASCII spaces 0x20, and

¹This 'ASCII encoded hex' seems to be used by most Icom equipment.

then the string “IcomCloneFormat3”, but its checksum, 0x7E, doesn’t match the one computed according to the above recipe.

Since these commands are being sent during a cloning operation, it is reasonable to assume that to the receiving unit the CI-V command 0xE4 means ‘set these locations to these values’. Moreover, the length of the data can probably be anything from 0x00 to 0xFF, so one can most likely use the following command to set a single memory location, 1475 (decimal), to the value 0x4B:

FE FE EF EE E4 30 35 43 33 30 31 34 42 34 31 FD
 (assuming I haven’t miscomputed the checksum, 0x41...)

3 Memory layout

I’m missing a lot of information here...

address (decimal)	description
0000-7999	500 memories, 16 bytes each
8000-8799	25 scan edges, 16 bytes each
8800-9799	bank assignments, 2 bytes each
9800-9919	??? (looks the same as bank assignments)
9920-9935	VFO A settings ???
9936-9951	VFO B settings ???
9984-9999	more VFO A settings ???
10000-10015	more VFO B settings ???
10304-10879	TV channels
10880-10911	current bank/memory settings ???
11520-11551	status, flags, settings
11552-11567	16 bytes with 0x20 (ASCII space)
11568-11583	“IcomCloneFormat3” string

3.1 Memory descriptions

Locations 0000-7999 describe the 500 memories, M0-M499, using 16 bytes for each. We will denote the contents of those bytes by B0 through B15.

According to the manual the memories store: frequency, receive mode (modulation), tuning step, duplex direction (+ or -) and offset frequency, subaudible tone, tone squelch or DTCS squelch settings and frequencies, DTCS code with phase mode, memory name, and scan skip setting. Here are some of these.

- B3:** increment, used in frequency computations. Value of 0x00 means 5 kHz, 0x01 means 6.25 kHz
- B0-B2:** frequency represented as $(256 \cdot 256 \cdot B2 + 256 \cdot B1 + B0) \cdot \text{increment}$ indicated by B3
- B4:** modulation? it appears that 0x00 means FM, 0x10 is AM, and 0x08 is WFM; 0x21 is FM in a DUP-range, and 0x41 is FM in a DUP+ range
- B5:** pattern noted: 0x00 for lower frequencies, 0x64 for HF, 0x78 for VHF and 0xE8 for UHF; don’t know the cutoffs
- B6:** set to 0x00 in most cases, 0x03 for some UHF memories ???
- B7:** always 0x00 ???
- B8:** repeater tone frequency and tuning step; 0xD6 means 103.5 Hz, TS of 15 kHz; 0xE6 for 107.2 Hz, TS of 15 kHz; 0xE7 is 107.2 Hz, TS of 20 kHz
- B9:** no clear pattern; 0x20 for all AM, 0x40 for most (but not all?) UHF
- B10-B15:** alpha tag, 6 ASCII characters

3.2 Scan edges

Locations 8000-8799 describe the 25 frequency pairs used for scan edges, in the order 0A, 0B, 1A, 1B, etc. Each takes up 16 bytes, with B0-B3 having the same meaning as above, and the other bytes looking similar. For all practical purposes these are like extra 50 memories, I guess. Even the inactive ones have frequency info in them; whether they are displayed as active or not must be determined by the bank assignments (see below).

3.3 Bank assignments

Locations 8800-9799 indicate bank assignments for the 500 memories, using two bytes each. These look like 0xFF 0xFF for unused memory (nothing programmed in), 0x1F 0xFF for a memory storing a frequency but not assigned to a bank. When a memory is assigned to a bank, then the corresponding first byte here is set to the bank number (0x00 being A, 0x01 is B, etc.) and the second byte is set to the number within a bank, starting with 0x00.

The banks are A-H (0x00-0x07?), J (0x08), L (0x09), N-R (0x0A-0x0E), T (0x0F), U (0x10), Y (0x11). But there is probably more to this; I have a bunch of memories in bank T and nearly all are coded as 0x0F but two have 0x2F for some reason.

The next 60 pair of bytes, 9800-9919, appear to be similar but for the scan edge memories (first 50 pairs) and perhaps for the call channels? The two bytes corresponding to the scan edge memories which are set are 0x1F 0xFF, the others are 0xFF 0xFF.

[Added in May 2006] I pretty much abandoned this project. There is still not enough information from ICOM, and no open source software for managing this handheld. Maybe some day someone will write a piece of software using the excellent hamlib library, but for now I'll be programming the IC-T90A by hand - it's simple enough.