IR-Remote JP1 Atlas 1055

Autor:
Data de publicació: 18-01-2016

The History of the JP1 Project

There are several people who have made key contributions that have made what we can now do in the JP1 group possible and this page will attempt to give these folks some recognition and to give you an idea of the history of this project.
Dan Nelsen
None of this would be possible without the efforts of Dan Nelsen. He's the who discovered how to communicate with the EEPROM chip in these remotes. He invented the first "Classic" interface and wrote the original software used to read and write to the EEPROM. Dan also made the source code to his software available (under a free distribution license), which meant that other developers could build on the work that he'd already done.

If the JP1 Project had a birthday, it would be 10/06/2000, that's the date of Dan's first post (under the name HWHackr) over at Remote Central giving some early details of what he had discovered.
Here's the thread: RS 15-1994 JP1 details revealed

Rob Crowe
That's me. Once Dan got the first dump from his 15-1994, it was then my job to figure out what it all meant. I figured out how the upgrade section was put together, how the keymapping bytes work, etc. I put this info into an Excel spreadsheet which eventually materialized into the keymap-master spreadsheet. I also started the JP1 group at Yahoo Groups (which was called eGroups back then).

Mark Pauker
Mark took all of this info and built it into a fabulous Windows GUI program called IR.exe which opened up the JP1 group to the masses.

Kevin Timmerman
Kevin was the first one to identify the code used for protocols as assembler code. With this info we could start modifying protocols to better suit our purposes. Kevin also discovered the format used to store learned commands.

Nicola Salmoria
Words cannot describe the contributions that Nicola has made. Whatever shortcomings these remotes might have, Nicola has managed to cure most of them by writing some of his "special protocols" which allow you to do things like program device specific macros, add adjustable delays in macros and stack multiple setup codes onto a single device key. He even wrote one protocol modestly called the "Extender" which completely takes over the main control loop of the remote, thus allowing you to do almost anything, such as program functions to the device keys and the FAV/SCAN button, use the learning memory for upgrade codes, etc. (Nicola is the guy behind the popularMAME program).

Tommy Tyler
What Nicola is to software, Tommy is to hardware. When this project started the JP1 interface required good soldering skills and at least four hours of free time to put together, so Tommy decided to try and simplify it a bit. What he came up with was a design that called for 2 resistors and 1 diode and about 15 minutes of free time to build it. If Mark's IR.exe program made JP1 appealing to the masses, just imagine what Tommy's Simple Interface did for it.

And if that's not enough, Tommy has written up everything that he has done is documents that absolutely anybody can understand. He's also the author of the "JP1 For Beginners" document.

All of the JP1 members

These guys that I have just mentioned are not the only ones to have made contributions, many other JP1 group members have made contributions in many different ways, I can't possibly name everyone but some people who come to mind (in alphabetical order) include:

Glenn Buskirk,
Robert Eden,
Nishan Fernando,
Jim Henry,
Scott Johnson,
Dave Leggett,
Dave Levin,
Don Miller,
Bill Napier,
Chris Nappi,
Jerry Rubinow.

Before JP1

Before Dan made the discovery that started the JP1 effort, there were several people who were already putting in alot of effort to figuring out remote controls.

Barry Gordon

Barry's area of expertise is the Philips Pronto, for which he has written many detailed documents explaining the concepts of infrared, he has also written many useful utilities for generating new signals and analyzing existing one. I learned everything I know about infrared from Barry's documents.

John Fine

John has been a One For All remote enthusiast for many years. He painstakingly went through all the codes in his Cinema 6 using an oscilloscope documenting what the signals looked like. He discovered how advanced codes relate to the original button codes (he came up with the terms EFC and OBC). Once introduced to Barry Gordon and the Philips Pronto files, John wrote the ccf2efc program whick converts Pronto files into text and attempts to figure out what protocol and codes where used. Mark Pauker used John's routines to add this feature to his IR.exe program.

John Wasser

Long before One For All had anythign useful on their web site, John created a web site telling people how to program their remotes using things like advanced codes, etc. It was in direct response to his web site that OFA set about improving their own site. John's site is http://www.John-Wasser.com/OFA.

Gerald Pinzone

I knew nothing about all of the hidden power of the 15-1994 remote, which I already owned, until I found Gerald Pinzone's Cinema 7 FAQ web site.

Zig Zichterman

Long before we started hacking the remotes with the 6-pin connectors, Zig was hacking the old remotes with the 3-pin connectors. He has long since stopped messing with remotes, but I have an archive of his work on my site at http://www.hifi-remote.com/ziggr. You can catch up with what the Zigster's up to now at http://www.ziggr.com

I apologise for anyone that I have left out, as you can see from the number of people that I have credited, this has been a group effort that just shows you what the true power of the internet really is.

Topic:

RS 15-1994 JP1 details revealed

This thread has 99 replies. Displaying posts 1 through 15.

Post 1 made on Friday October 6, 2000 at 02:48

HW Hackr

Historic Forum Post

I've hacked the hardware details for the JP1 jumper in the RS 15-1994. I've successfully read the internal EEPROM, and monitored its reads/writes while pushing various buttons. I haven't tried writing the EEPROM, but it should be trivial.

There are 2 ICs on the board: U1 is the microcontroller, and U2 is a 2048x8 bit serial EEPROM (ATMEL 24C16N - datasheet available online). Only a few bytes in U2 are used, so I believe that the main code database is stored in a hardwired ROM in U1. U2 holds your customizations (device numbers, macros, key remaps etc.)

JP1 pin definitions:
1 - VCC(U1) (leave unconnected if using batteries)
2 - VCC(U2) (leave unconnected if using batteries)
3 - GND

4 - SDA (U2 data input/output)
5 - input to U1 (unknown function)
6 - SCL (U2 clock)

The SEEPROM is read/written by following the protocol in the datasheet. Until the software memory use is understood, we can't do much more than copy settings from one remote to another. I can already tell that there are various pointers, and it seems like the software calculates a checksum of the entire EEPROM contents after any operation that does a write.

I believe that a devices code upgrade stores the new codes in U2. The easiest way to crack this would be to look at an upgraded remote
(maybe the OFA remotes are close enough to be of help).

Reply

OP | Post 8 made on Monday October 9, 2000 at 17:45

HW Hackr

Historic Forum Post

So far, I've determined part of the EEPROM memory map and all keycodes. To make more progress, I'll need access to an upgraded remote (URC8800 or 9800) to see how the upgrade codes are stored.

Or maybe I'll release my software in the hopes that an advanced electonics hobbyist with an upgraded remote can post the memory dump. I've been using a Linux kernel driver I wrote to capture the EEPROM signals through the parallel port. I wrote another program to process the raw signal data and show the Serial-EEPROM operations (read/write address & data, etc.). I'm reluctant to do this though, because parallel port projects aren't for beginners and I don't want someone burning out their PC's parallel port (or remote).

RS 15-1994 EEPROM memory map (000-7ff):

000: EEPROM checksum
001: complement of checksum at 000

00a-00b: SAT device number (ex: 03 07 = 3*256+007 = 0775)

00c-00d: TV device number (ex: 10 b3 = 0*256+179 = 0179)
(ex: 14 7b = 4*256+123 = 1147)

00e-00f: VCR device number (ex: 20 43 = 0*256+067 = 0067)
(ex: 21 df = 1*256+223 = 0479)

010-011: CD device number (ex: 33 69 = 3*256+105 = 0873)

012-013: AUX1 device number (ex: 40 c3 = 0*256+195 = 0195)
(ex: 41 87 = 1*256+135 = 0391)

014-015: AUX2 device number (ex: 40 b0 = 0*256+176 = 0176)
(ex: 42 e8 = 2*256+232 = 0744)

016-017: PNP device number (ex: 30 a7 = 0*256+167 = 0167)
(ex: 31 a4 = 1*256+164 = 0420)

01b-0ff: advanced codes (fixed size, 5 bytes each) & macros
1st byte is keycode
3rd-4th bytes are copies of device number

100-105: pointers

3ff-???: learned code data structures (variable size, 13-14 bytes each for my examples)
1st byte is keycode
3rd byte is length of learned code
4th-nth bytes are the learned code

RS 15-1994 keycode map:

SLEEP ?

POWER 03
1S 21
2S 22
3S 23
4S 24
ARROW-UP 31
ARROW-DOWN 32
ARROW-LEFT 33
ARROW-RIGHT 34
SELECT 35

MENU 29
GUIDE 28
CENTER 2b
MUTE 08
LAST 13
SCAN ??
PIP 2d
DISPLAY 2a
SWAP 2f
MOVE 2e
VOL+ 04
VOL- 05
CH+ 06
CH- 07
1 15
2 16
3 17
4 19
5 1a
6 1b
7 1c
8 1d
9 1e
0 1f
TV/VIDEO 18
ENTER 12
REW 0b
PLAY 0c
FFWD 0d
REC 10
STOP 0e
PAUSE 0f

A JP1 remote is a type of universal remote, usually with a six-pin interface connector labeled "JP1" in the battery compartment, manufactured byUniversal Electronics Inc. The JP1 interface allows the remote to be reprogrammed, adding new code lists and functions. Home theater hobbyists use JP1 to avoid obsolescence.

Most JP1 remotes are capable of advanced functions like remapping keys and macros. Some models can be updated over the telephone to add new code lists.[1]

Contents
[hide]

1JP1 Remote controls
2Hardware interface
3Updating JP1 Remotes via an interface
4Software
5Extended Function Codes (EFC)
6Older UEI Remote controls
7Press
8See also
9References
10External links

JP1 Remote controls

All JP1 remotes are made by Universal Electronics, Inc.[2] UEI sells various models under their One For All brand name,[3] and supplies remotes to consumer electronic manufacturers such as Radio Shack, Sony, and Sky, as well as North American cable TV providers such as Comcast, Rogers, Cox,Shaw, Charter and Time Warner.[4]

On printed circuit boards the marking "JP1" is a common abbreviation of "Jumper 1", i.e. the first (and for most remotes, the only) jumper on the board. Perhaps in recognition of this custom, later models are labeled "JP1.x", where "x" is 1, 2, or 3, depending on the type of processor used.

Earlier JP1 circuit board designs employ an EEPROM memory chip. Later designs employ processors with flash memory.

Hardware interface

A JP1 interface cable connects a JP1 remote to a PC,[5] enabling the PC to read and write to the remote's user memory. Schematics for cabling parallel port, serial port or USBconnectors to various JP1 remote controls are freely available, and several vendors offer pre-built interface cables.

Updating JP1 Remotes via an interface

The data and software in many JP1 remote controls can be updated and extended using an interface cable connected to a PC running software such as IR, RemoteMaster, orKeyMap Master. Updates and extensions include new device code data, new IR protocols, advanced keymapping, and macros.[6]

Nicola Salmoria discovered how to add new functions by writing software "Extenders" (protocols which replace a JP1 remote's main processing loop). Extenders may depend on a particular JP1 remote's hardware capabilities. Typical extender features include longer macro length; fast command execution; nested macros; and the long key press (LKP) -- in which a key performs different actions depending on how long it is pressed.

Software

There are various software packages available. The core package at its simplest allows for the basic remote functionality to be updated via the PC. It also allows for installing new devices, protocols and extenders, though these are created separately and copied into the core package. It also permits the entire remote control's configuration to be saved as files on the PC for backup purposes or "cloning" remote controls.

Programs complementary to the core package cover such functionality as creating new devices, creating new protocols,

assembling the assembly languages of the processors in the remote control, and analysis of signals learned on the remote control to enable proper protocol support to be built to control new devices.

In October, 2000, at remotecentral.com's "General Consumer Remotes" forum, electronics hobbyists Dan "HW Hackr" Nelson and Rob Crowe[7] worked out how a JP1 connector could be used to examine and modify that portion of a remote control's memory containing user configuration data and user-updated devices.[8] The hobbyists were able toreverse engineer the layout of this memory area, then discovered how to apply updates directly. The JP1 Remotes Forum expanded on that work, and remains the foremost locus of discovery for exploiting new JP1 functionality.[9]

Extended Function Codes (EFC)
Earlier universal remote controls used a device code/protocol number and three-digit extended function codes for programming via the remote itself, or through the JP1 interface. These sometimes enable a universal remote control to be programmed to use some extra functions which may not have been made available even on the original equipment manufacturer's remote control. In later controls, these function codes were extended to a length of five digits.

There is a tool which assists in the lookup of these code sets at the JP1 group site,[10] and additional codes can also generally be obtained from the remote control manufacturer or supplier.

Older UEI Remote controls
Universal Electronics Inc. also supplied a previous series of universal remote controls, typified by the European control called the 'Big Easy'. This control can operate up to four consumer devices, with protocols and code sets normally limited to TV, analogue satellite and VCR. However, some terrestrial digital receivers and DVD players are using old protocols and code sets, typically those previously used by analogue satellite receivers. This means that these old controls can still be useful. Remote controls in this product range can normally be identified by the presence of three programming eyelets in the battery compartment. The codes to reprogram these remotes can be set down in the form of an algorithm, which can be freely downloaded and used to find extended control sets.[11]

Press
JP1 has been reported on by the Detroit Free Press [12][13] and Nuts and Volts magazine.[14]

See also

Universal Remote - Article discussing Universal Remote Controls in General
Logitech Harmony Remote - Logitech's range of programmable remote controls (non-JP1)

References

Jump up^ "OFA Remotes General Information". Hifi-remote.com. Retrieved 2010-06-03.
Jump up^ "Universal Electronics Website". Uei.com. Retrieved 2010-06-03.
Jump up^ "One-For-All Website". Oneforall.com. Retrieved 2010-06-03.
Jump up^ "UEI Cable Remotes Support Website". Urcsupport.com. Retrieved 2010-06-03.
Jump up^ "JP1 Hardware Interfaces". Hifi-remote.com. Retrieved 2010-06-03.
Jump up^ "JP1 Remote programming via an interface". Hifi-remote.com. Retrieved 2010-06-03.
Jump up^ "RS 15-1994 JP1 details revealed". Remote Central. Retrieved 2011-02-28.
Jump up^ "The History of the JP1 Project". Hifi-remote.com. Retrieved 2010-06-03.
Jump up^ "JP1 Technical Forum". Hifi-remote.com. Retrieved 2010-06-03.
Jump up^ "JP1 Device Lookup Tool". Remote Central. Retrieved 2012-01-07.
Jump up^ "Programming earlier UEI Remotes such as the ""Big Easy""". Kelvinadams.atspace.com. Retrieved 2010-06-03.
Jump up^ Newman, Heather (August 13, 2003). "All-in-one remotes make appliance control a cinch". Detroit Free Press.
Jump up^ Newman, Heather (October 8, 2003). "Remote makes all things possible". Detroit Free Press.
Jump up^ Weingarden, Michael (July 2003). "Discover a "secret" computer interface for your remote control.". Nuts and Volts.

External links

JP1 Community Wiki - JP1 history, technical details, help and how-to's
Description of JP1 at Remote Central website
Find extra five-digit EFCs by using a chart