Version 2.45

The IRP specification by Graham Dixon
Practical explanation of IR signals and IRP by Vicky Getz

48-NEC

If you get a decode whose protocol name is simply "48-NEC" that indicates the learned signal is not complete (usually caused by not holding the original remote's button long enough during learning). Enough of the signal is present to accurately determining the device, subdevice and OBC numbers. But not enough is present to determine whether the protocol is 48-NEC1 or 48-NEC2.

48-NEC1

IRP notation: {38.0k,564}<1,-1|1,-3>(16,-8,D:8,S:8,F:8,~F:8,E:8,~E:8,1,^108m,(16,-4,1,^108m)*)
EFC translation: LSB comp

This protocol signals repeats by the use of dittos.

48-NEC2

IRP notation: {38.0k,564}<1,-1|1,-3>(16,-8,D:8,S:8,F:8,~F:8,E:8,~E:8,1,^108m)+
EFC translation: LSB comp

AdNotam

IRP notation: {35.7Khz,895,msb}<1,-1|-1,1>(1,-2,1,D:6,F:6,^114m)+

Very similar to RC5, except AdNotam uses two start bits, and no toggle bit.

AirAsync

IRP notation: {37.7Khz,840}<1|-1>(N=0,(1,B:8:N,-2,N=N+8)+)

This protocol uses asynchronous data transmission that sends an 8-bit byte with 1 start bit, 8 data bits and 2 stop bits. The minimum signal is one byte. The protocol is reported as AirAsync*n*-xx.yy. ... where *n* is the number of bytes and xx, yy, ... are the byte values in hexadecimal notation.

AirB?-????

This is not a robust protocol, so spurious decodes are likely. If you see this decode from something other than an IR keyboard, you should probably ignore it.

Aiwa

UEI protocol: 005E or 009E
IRP notation: {38k,550}<1,-1|1,-3>(16,-8,D:8,S:5,~D:8,~S:5,F:8,~F:8,1,-42,(16,-8,1,-165)*)
EFC translation: LSB

This protocol signals repeats by the use of dittos.

The EFC numbering varies among the KM and RM versions of Aiwa. Using OBC numbers is less confusing.

When using a non-combo version of Aiwa in KM, you must combine the device and subdevice numbers as device+256*subdevice to get the number KM calls "Device Code". (Since subdevice is usually zero, that combination is trivial). In Aiwa combo in KM you use the subdevice as "parameter" (on the setup sheet) and put the device in the byte2 column on the functions sheet. RM follows DecodeIr's naming of Device and Subdevice.

AK

Documentation not written yet.

Akai

UEI protocol: 000D
IRP notation: {38k,289}<1,-2.6|1,-6.3>(D:3,F:7,1,^25.3m)+
EFC translation: LSB comp prefixed with last device bit

This is not a robust protocol, so spurious decodes are likely.

As of version 8.31, KM does not translate device to fixed data, nor OBC to EFC according to the same rules used by DecodeIr. RM does translate consistently with DecodeIr, so you may find it easier to use RM. If you use KM, you must change the device number as follows:

- Decoded device 0 or 4 --> KM device 3
- Decoded device 1 or 5 --> KM device 2
- Decoded device 2 or 6 --> KM device 1
- Decoded device 3 or 7 --> KM device 0

Also (in KM) you should use the EFC number from the decode not the OBC number. Akai protocol uses the same EFC numbering across all JP1 remotes, so use of EFC is safe. KM uses different OBC numbering than RM and DecodeIr, so use of OBCs isn't safe.

Amino

UEI protocol: 019C
IRP notation: {56.0k,268,msb}<-1,1|1,-1>[T=1] [T=0] (7,-6,3,D:4,1:1,T:1,1:2,0:8,F:8,15:4,C:4,-79m)+
-----Variant: {36.0k,268,msb}<-1,1|1,-1>[T=1] [T=0] (7,-6,3,D:4,1:1,T:1,1:2,0:8,F:8,15:4,C:4,-79m)+
{C =(D:4+4*T+9+F:4+F:4:4+15)&15} [the arithmetic sum of the first 7 nibbles mod 15]
T=1 for the first frame and T=0 for all repeat frames.
DecodeIR v2.43 checks T and will report in the Misc field if the start or end frame is missing. Amino equipment use both 36 and 56KHz, but the duration of the half-bit is always 268. Zaptor is a closely related protocol which for which the half-bit duration is 330. IRDecode v2.43 distinguishes between Amino and Zaptor in order of priority by 1)the position of the toggle bit, 2)the value of the next to last nibble, and 3)the measured duration of a half-bit.
EFC translation: MSB

Anthem

UEI protocol: 0123
IRP notation: {38.0k,605}<1,-1|1,-3>((8000u,-4000u,D:8,S:8,F:8,C:8,1,-25m)3, -75m)+ {C=~(D+S+F+255):8}
Anthem framing is very similar to NEC, and also uses 32 bits of data. However, the leadout is much shorter. The signal is sent at least 3 times. Anthem usually splits F into a 2 bit unit number, and a 6 bit function number.

Apple

UEI protocol: 01E0
IRP notation: {38.0k,564}<1,-1|1,-3>(16,-8,D:8,S:8,C:1,F:7,I:8,1,^108m,(16,-4,1,^108m)*) {C=~(#F+#PairID)&1)}
C=1 if the number of 1 bits in the fields F and I is even. I is the remote ID.

Apple uses the same framing as NEC1, with D=238 in normal use, 224 while pairing. S=135

Archer

IRP notation: {0k,12}<1,-3.3m|1,-4.7m>(F:5,1,-9.7m)+
EFC translation: 5-bit LSB

This is not a robust protocol, so spurious decodes are likely.

Async

This protocol uses asynchronous data transmission that sends an 8-bit byte with 1 start bit, 8 data bits and 1 stop bit. The minimum signal is four bytes. The protocol is reported as Async*n*:min-max:aa.bb...yy.zz where *n* is the number of bytes, min-max is the range of durations in microseconds that was taken as a single bit and aa.bb...yy.zz are the first two and last two byte values in hexadecimal notation. DecodeIR v 2.44 does not report Async decodes.

Barco

UEI protocol: 002A
IRP notation: {0k,10}<1,-5|1,-15>(1,-25, D:5,F:6, 1,-25,1,120m)+
EFC translation: 6-bit LSB comp

This is a moderately robust protocol, but spurious decodes are still possible.

Blaupunkt

IRP notation: {30.3k,528}<-1,1|1,-1>(1,-5,1023:10,-39,1,-5,1:1,F:6,D:3,-230)

Bose

UEI protocol: 010C
IRP notation: {38.0k,500,msb}<1,-1|1,-3>(2,-3,F:8,~F:8,1,-50m)+
EFC translation: MSB

Bryston

IRP notation: {38.0k,315} <1,-6|6,-1>(D:10,F:8, -18m)+

CanalSat

UEI protocol: 018C IRP notation: {55.5k,250,msb}<-1,1|1,-1>(T=0,(1,-1,D:7,S:6,T:1,0:1,F:7,-89m,T=1)+)
EFC translation: 7-bit MSB.

The repeat frames are not all identical. T toggles within a single signal, with T=0 for the start frame and T=1 for all repeats. DecodeIR v2.37 and later check T and will report in the Misc field if the start frame is missing. The official name for CanalSat is "ruwido r-step".

CanalSatLD

UEI protocol: unknown
IRP notation: {56k,320,msb}<-1,1|1,-1>(T=0,(1,-1,D:7,S:6,T:1,0:1,F:6,~F:1,-85m,T=1)+)

The official name for CanalSatLD is "ruwido r-step"

Denon, Denon{1} and Denon{2}

IRP notation: {38k,264}<1,-3|1,-7>(D:5,F:8,0:2,1,-165,D:5,~F:8,3:2,1,-165)+
EFC translation: LSB

A Denon signal is identical to Sharp, and has two halves, either one of which is enough to fully decode the information. A significant fraction of Denon learned signals contain just one half or have the halves separated so that DecodeIr can't process them together. When one half is seen separate from the other, DecodeIr will name the protocol Denon{1} or Denon{2} depending on which half is decoded. Denon, Denon{1} and Denon{2} all represent the same protocol when they are correct. But only Denon is robust. A Denon{1} or Denon{2} decode might be spurious.

Denon-K

UEI protocol: 00CD, 01C8
IRP notation: {37k,432}<1,-1,1,-3>(8,-4,84:8,50:8,0:4,D:4,S:4,F:12,((D*16)^S^(F*16)^(F:8:4)):8,1,-173)+
EFC translation: LSB comp

Denon-K is the member of the Kaseikyo family with OEM_code1=84 and OEM_code2=50.

Denon-K uses the same check byte rules as Panasonic protocol, but uses the data bits differently. The Panasonic Combo protocol in KM can be used with some difficulty to produce Denon-K signals. The Denon-K choice in RM uses the same protocol executor as Panasonic combo, but computes the hex commands based on Denon's use of the Kaseikyo data bits.

Dgtec

UEI protocol: 016A
IRP notation: {38k,560}<1,-1|1,-3>(16,-8,D:8,F:8,~F:8,1,^108m,(16,-4,1,^108m)+)
EFC translation: LSB comp

This protocol signals repeats by the use of dittos.

DirecTV

IRP notation: {38k,600,msb}<1,-1|1,-2|2,-1|2,-2>(5,(5,-2,D:4,F:8,C:4,1,-50)+) {C=7*(F:2:6)+5*(F:2:4)+3*(F:2:2)+(F:2)}
EFC translation: MSB

There are six variants of the DirecTV protocol, distinguished in RemoteMaster by the parameter "Parm" on the Setup page. The Parm value is shown in the Misc field of DecodeIR. The IRP notation above corresponds to the default Parm=3. The various Parm values correspond to three different frequencies (the 38k in the above) and two different lead-out times (the -50 in the above). The corresponding values are:

- Parm=0 : 40k, -15
- Parm=1 : 40k, -50
- Parm=2 : 38k, -15
- Parm=3 : 38k, -50
- Parm=4 : 57k, -15
- Parm=5 : 57k, -50

Portions of a dirty learn of a Sony signal may look like a DirecTV signal. So, if you get a DirecTV decode together with a plausible Sony decode, believe the Sony decode and ignore the DirecTV. If you get a DirecTV decode without a Sony decode for some functions of a Sony device, try relearning them; a DirecTV decode for a signal meant for a Sony device is not likely to be correct.

This protocol was called Russound in versions of DecodeIR earlier than 2.40.

Dishplayer

UEI protocol: 010F
IRP notation: {38.4k,535,msb}<1,-5|1,-3>(1,-11,(F:6,S:5,D:2,1,-11)+)
EFC translation: Not available in this version of DecodeIr

This is not a robust protocol, so spurious decodes are likely.

Dish_Network

UEI protocol: 0002
IRP notation: {57.6k,400}<1,-7|1,-4>(1,-15,(F:-6,S:5,D:5,1,-15)+)
EFC translation: MSB comp 6 function bits followed by LSB comp low 2 unit bits.

This is not a robust protocol, so spurious decodes are likely.

The unit number shows up in the Subdevice field of DecodeIr's output. In KM, the "unit code" is one greater than the unit number. So you must take the Subdevice from the decode and add one to it and use that as the "unit code" in KM.

There are two variants of the protocol executor for DishNetwork with different but compatible EFC numbering. The decoded EFC should work for both. But the results may be less confusing if you use OBC numbers in KM or RM.

EchoStar

UEI protocol: 0182
As of 2.42 DecodeIR shows this as RC5-7F

Elan

UEI protocol: Unknown
IRP notation: {40.2k,398,msb}<1,-1|1,-2>(3,-2,D:8,~D:8,2,-2,F:8,~F:8,1,^50m)+

See the JP1-forum for the executor.

Emerson

UEI protocol: 0065
IRP notation: {36.7k,872}<1,-1|1,-3>(4,-4,D:6,F:6,~D:6,~F:6,1,-39)+
EFC translation: 6-bit LSB comp with 2-bit mini-combo

Lists three different EFCs because this protocol is a mini combo.

F12

UEI protocol: 001A
IRP notation: (Toshiba specification) {37.9k,422}<1,-3|3,-1>(D:3,H:1,A:1,B:1,F:6),-80)2 for A=1 or B=1
{37.9k,422}<1,-3|3,-1>((D:3,H:1,A:1,B:1,F:6),-80)2,-128)+ for H=1.

Exactly one of H, A, or B can have a value of 1. If H=1 the signal can be sent repeatedly, and F can take any 6 bit value. If A or B=1, the signal is sent once only per button press, and only a single bit of F can be non-zero. Toshiba spec sheet

IRP notation: (JP1) {37.9k,422}<1,-3|3,-1>((D:3,S:1,F:8,-80)2,-128)+
A and B are subsumed into F, and the value of H is computed by the executor. H=A^B.
EFC translation: lsb, but not computed in DecodeIR.
DecodeIR reports H as the subdevice. This is useful when making a Pronto Hex file, or other description based on durations. Remotes with executors (e.g. UEI remotes) normally compute the value of H in the executor, and the "subdevice" is not needed as a parameter.

F32

UEI protocol: Unknown
IRP notation: {37.9k,422,msb}<1,-3|3,-1>(D:8,S:8,F:8,E:8,-100m)+

The meaning of the 32 bits of data is unknown, and the assignment to D, S, F, and E is arbitrary

Fujitsu

UEI protocol: 00F8
IRP notation: {37k,432}<1,-1|1,-3>(8,-4,20:8,99:8,0:4,E:4,D:8,S:8,F:8,1,-110)+
EFC translation: LSB comp

Fujitsu is the member of the Kaseikyo family with OEM_code1=20 and OEM_code2=99. There is no check byte, so the risk of an incorrectly decoded OBC is much higher than in other Kaseikyo protocols.

00F8 requires 2-byte hex commands, so the decoded EFC number is generally not useful. Use OBC number in upgrades or to compute Hex commands.

Fujitsu-56

IRP notation: {37k,432}<1,-1|1,-3>(8,-4,20:8,99:8,0:4,E:4,D:8,S:8,X:8,F:8,1,-110)+

G.I. Cable and G.I. Cable{1}

UEI protocol: 00C4
IRP notation: {38.7k,490}<1,-4.5|1,-9>(18,-9,F:8,D:4,C:4,1,-84,(18,-4.5,1,-178)*) {C = -(D + F:4 + F:4:4)}
EFC translation: LSB

This protocol signals repeats by the use of dittos. When the {1} is shown as part of the protocol name for G.I. Cable, it just means that the repeat part of the signal is not present. That doesn't indicate any difference in the actual protocol nor even any unreliability in the decode. It may indicate that use of the learned signal will be less reliable, so you have more than usual reason to replace it with a KeyMove, Upgrade or cleaned up version.

G.I.4DTV

UEI protocol: 00A4
IRP notation: {37.3k,992}<1,-1|1,-3>(5,-2,F:6,D:2,C:4,1,-60)+ {C= ((#(F&25) + #(D&5))&1) + 2*((#(F&43) + #(D&7))&1) + 4*((#(F&22) + #(D&7))&1) + 8*((#(F&44) + #(D&6))&1)}
EFC translation: NONE

This is a moderately robust protocol, but spurious decodes are still possible

Unit (device) numbers from 0 to 7 are supported. The check sum C is a Hamming Code, which can correct single bit errors. D:1:2 is encoded in the check sum. D encoding scheme The official (UEI) protocol executor for G.I.4DTV does not support EFC numbers. If you are creating an upgrade in KM or RM you should use OBC numbers, not EFC numbers. If you need the Hex Cmd for a KeyMove, you should use the functions sheet of KM or RM to compute it for you from the OBC and device number.

G.I.RG

UEI protocol: unknown
IRP notation: {37.3k,1000, msb}<1,-1|1,-3>(5,-3,F:6,S:2,D:8,1,-60)+
EFC translation:

This is a moderately robust protocol, but spurious decodes are still possible, especially SIM2 which has nearly identical timing.

Typical usage is the GI/Next Level/Motorola RG2x00 series

Grundig16 and Grundig16-30

UEI protocol: Grundig16 0112, Grundig16-30 00AB
IRP notation for Grundig16: {35.7k,578,msb}<-4,2|-3,1,-1,1|-2,1,-2,1|-1,1,-3,1> (806u,-2960u,1346u,T:1,F:8,D:7,-100)+
IRP notation for Grundig16-30: {30.3k,578,msb}<-4,2|-3,1,-1,1|-2,1,-2,1|-1,1,-3,1> (806u,-2960u,1346u,T:1,F:8,D:7,-100)+
EFC translation: MSB but with bit pairs translated data->hex by 00->00, 01->11, 10->01, 11-> 10 and off by one position.

These are two variants of the same protocol, differing only in frequency. The IRP notation is corrected from previous versions of this document, to bring it into line with what DecodeIR actually does.

GXB

IRP notation: {38.3k,520,msb}<1,-3|3,-1>(1,-1,D:4,F:8,P:1,1,^???)+

Decoder for a nonstandard Xbox remote.

Humax 4Phase

IRP notation: {56k,105, msb}<-2,2|-3,1|1,-3|2,-2>(T=0,(2,-2,D:6,S:6,T:2,F:7,~F:1,^95m,T=1)+)

The allocation of bits to device and subdevice (D:6, S:6) is a guess.

IODATA*n* and IODATA*n-x-y*

UEI protocol: not known.
IRP notation: {38k,550}<1,-1|1,-3>(16,-8,x:7,D:7,S:7,y:7,F:8,C:4,1,^108m)+ {n = F:4 ^ F:4:4 ^ C:4}
EFC translation: LSB

This is potentially a class of protocols distinguished by values of n, x and y with n = 0..15 and x, y = 0..127. If x and y are both zero, they are omitted. The only known example is IODATA1.

Jerrold

UEI protocol: 0006
IRP notation: {0k,44}<1,-7.5m|1,-11.5m>(F:5,1,-23.5m)+

This is not a robust protocol, so spurious decodes are likely.

JVC and JVC{2}

UEI protocol: 0034
IRP notation: {38k,525}<1,-1|1,-3>(16,-8,(D:8,F:8,1,-45)+)
EFC translation: LSB comp

JVC{2} indicates a JVC signal from which the lead-in is missing. The JVC protocol has lead-in on only the first frame, so it is quite easy to have it missing from a learned signal. So when JVC{2} is correct, it means the same as JVC. But JVC{2} is not robust, so spurious decodes are likely. It is also very similar in structure and timing to Mitsubishi protocol, so that DecodeIr has difficulty distinguishing one from the other. The device number, OBC and EFC are all encoded the same way between the two. So if you have JVC{2} decodes that you have reason to suspect should actually be Mitsubishi, you can try using them as Mitsubishi without changing the numbers. However, true Mitsubishi signals will not mis-decode as JVC, just as JVC{2}. So if some of the signals for your device decode as JVC and others as JVC{2}, you should trust all those decodes and not try Mitsubishi.

JVC-48

UEI protocol: 001F or 00C9 or 00CD
IRP notation: {37k,432}<1,-1|1,-3>(8,-4,3:8,1:8,D:8,S:8,F:8,(D^S^F):8,1,-173)+
EFC translation: LSB comp

JVC-48 is the member of the Kaseikyo family with M=3 and N=1.

Panasonic protocol uses the same check byte rules as JVC-48, so you might want use the (more flexible) Panasonic entries in KM or RM to produce a JVC-48 upgrade (by changing the OEM_code1(M) and OEM_code2(N) values). For simple JVC-48 upgrades you get exactly the same results by directly selecting the "JVC-48" protocol.

JVC-56

IRP notation: {37k,432}<1,-1|1,-3>(8,-4,3:8,1:8,D:8,S:8,X:8,F:8,(D^S^X^F):8,1,-173)+

JVC-56 is the member of the Kaseikyo56 family with M=3 and N=1.

Kaseikyo

UEI protocol: 00F8
IRP notation: {37k,432}<1,-1|1,-3>(8,-4,M:8,N:8,X:4,D:4,S:8,F:8,G:8,1,-173)+ {X=M:4:0^M:4:4^N:4:0^N:4:4}
EFC translation: LSB comp

Kaseikyo is a family of protocols that includes Panasonic, Mitsubishi-K, Fujitsu, JVC-48, Denon-K, Sharp-DVD and Teac-K. Each manufacturer is assigned a pair of identifiers, here identified as M and N. If an IR signal matches a known pair of OEM identifiers and has the correct checksum behavior, it will be decoded with appropriate protocol name. Otherwise it will be decoded as Kaseikyo.

Kaseikyo56

IRP notation: {37k,432}<1,-1|1,-3>(8,-4,M:8,N:8,X:4,D:4,S:8,E:8,F:8,G:8,1,-173)+ {X=M:4:0^M:4:4^N:4:0^N:4:4} Like Kaseikyo, each manufacturer is assigned a pair of identifiers identified as M and N. If an IR signal matches a known pair of OEM identifiers and has the correct checksum behavior, it will be decoded with appropriate protocol name. Otherwise it will be decoded as Kaseikyo56.

Kathrein

UEI protocol: 0066
IRP notation: {38k,540}<1,-1|1,-3>(16,-8,D:4,~D:4,F:8,~F:8,1,^105m,(16,-8,F:8,1,^105m)+)
EFC translation: LSB comp

This protocol signals repeats by the use of dittos. It is unusual in that the ditto frame carries part of the signal data, specifically the function code (OBC) but not the device code. Similar to Logitech, but both decodes give the same device number and OBC.

Konka

UEI protocol: 019B
IRP notation: {38k,500,msb}<1,-3|1,-5>(6,-6,D:8,F:8,1,-8,1,-46)+
EFC translation: MSB

Lumagen

IRP notation: {38.4k,416,msb}<1,-6|1,-12>(D:4,C:1,F:7,1,-26)+ {C = (#F+1)&1}
EFC translation: MSB prepended with C bit.

This is a moderately robust protocol, but spurious decodes are still possible.

Lutron

IRP notation: {40k,2300,msb}<-1|1>(255:8,X:24,0:4)+
EFC translation: MSB of decoded signal.

This is an unusual protocol in that an 8-bit device code and 8-bit OBC are encoded in a 24-bit error-correcting code as the X of the IRP notation. This is constructed as follows. First two parity bits are appended to the 16 data bits to give even parity for the two sets of 9 bits taken alternately. The resulting 18-bit sequence is then treated as 6 octal digits (0-7) expressed in 3-bit binary code. These are then re-coded in the 3-bit Gray code (also called, more descriptively, the reflected-binary code) with a parity bit to give odd parity, so giving 6 4-bit groups treated as a single 24-bit sequence. The whole thing allows any single-bit error in transmission to be identified and corrected.

Logitech

UEI protocol: 020B
IRP notation: {38k,127}<3,-4|3,-8>(31,-36,D:4,~D:4,F:8,~F:8,3,-50m)+

Logitech is used with their PS3 adapter. The IR signal is similar to Kathrein. If a Logitech signal is decoded as Kathrein, the device number and OBC are still correct.

Matsui

IRP notation: {38K,525}<1,-1|1,-3>(D:3,F:7,1,^30.5m)+
EFC translation: Not available in this version of DecodeIr

This is not a robust protocol, so spurious decodes are likely.

MCE (RC6-6-32)

IRP notation: {36k,444,msb}<-1,1|1,-1>(6,-2,1:1,6:3,-2,2,OEM1:8,OEM2:8,T:1,D:7,F:8,^107m)+

MCE is a member of the RC6 family. Technically it is RC6-6-32 with the standard toggle bit zero, with the OEM1 field equal to 128, and with a nonstandard (for the RC6 family) toggle bit added. If all those rules are met, DecodeIr will display the name as "MCE" and with the OEM2 field moved to the subdevice position. Otherwise it will display RC6-6-32.

As of version 8.31, KM does not have built-in support for this protocol, but there are KM format upgrade files available for Media Center (built by an expert who isn't limited to KM's built-in protocols). Those upgrades should be adaptable to any RC6-6-32 code set (by changing the fixed data), if the one you have doesn't already match the upgrade.

RM version 1.16 has support for RC6-6-32, which can be used for MCE upgrades. Version 1.17 will also have direct support for MCE

Metz19

IRP notation: (37.9K,106,msb)<4,-9|4,-16>(8,-22,T:1,D:3,~D:3,F:6,~F:6,4,-125m)+
The toggle bit T is inverted each time a new button press occurs.

Mitsubishi

UEI protocol: 0014
IRP notation: {32.6k,300}<1,-3|1,-7>(D:8,F:8,1,-80)+
EFC translation: LSB comp

This is not a robust protocol, so spurious decodes are likely. It is also very similar in structure and timing to JVC{2} protocol, so that DecodeIr has difficulty distinguishing one from the other. The device number, OBC and EFC are all encoded the same way between the two. So if you have Mitsubishi decodes that you have reason to suspect should actually be JVC, you can try using them as JVC without changing the numbers.

Mitsubishi-K

IRP notation: {37k,432}<1,-1|1,-3>(8,-4,35:8,203:8,X:4,D:8,S:8,F:8,T:4,1,-100)+
EFC translation: not available yet

Mitsubishi-K is the member of the Kaseikyo family with OEM_code1=35 and OEM_code2=203.

NEC

NEC is a family of similar protocols including NEC1, NEC2, Tivo, Pioneer, Apple, NECx1 and NECx2. If you get a decode whose protocol name is simply "NEC" that indicates the learned signal is not complete (usually caused by not holding the original remote's button long enough during learning). Enough of the signal is present to accurately determine the device, subdevice and OBC numbers. But not enough is present to determine whether the protocol is NEC1 or NEC2.

Difference between NEC1 and NEC2

The difference between NEC1 and NEC2 only affects the signal sent by a long keypress. A short press sends the same signal in NEC1 and NEC2.

Variant IRstreams in NEC protocols

For NEC1, NEC2, NECx1, and NECx2 protocols, the IRstream contains D:8,S:8,F:8,~F:8
However, some manufacturers (especially Yamaha and Onkyo) are breaking the "rule" that the 4th byte should be ~F:8
Version 2.42 decodes these variants by adding suffixes to the protocol name depending on the IRstream:
-y1: D:8,S:8,F:8,~F:7,F:1:7 (complement all of F except the MSB)
-y2: D:8,S:8,F:8,F:1,~F:7:1 (complement all of F except the LSB)
-y3: D:8,S:8,F:8,F:1,~F:6:1,F:1:7 (complement all of F except MSB and LSB)
-rnc: D:8,S:8,F:8;~F:4:4,~F:4 (complement F and reverse the nibbles)
-f16: D:8,S:8,F:8,E:8 (no relationship between the 3rd and 4th bytes)

NEC1

IRP notation: {38.0k,564}<1,-1|1,-3>(16,-8,D:8,S:8,F:8,~F:8,1,^108m,(16,-4,1,^108m)*)
EFC translation: LSB comp

A few devices use NEC1 protocol at 40Khz, rather than the typical frequency. When getting a decode of NEC1, if you notice that the frequency is closer to 40Khz than to 38Khz, examine multiple learns from the same device to estimate whether the 40Khz frequency is a learning error or a true characteristic of the device. If the 40Khz is correct, there are methods in JP1, or MakeHex (whichever you are using) to reproduce NEC1 at 40Khz rather than the usual frequency.

NEC2

IRP notation: {38.0k,564}<1,-1|1,-3>(16,-8,D:8,S:8,F:8,~F:8,1,^108m)+
EFC translation: LSB comp

Pioneer is distinguished from NEC2 only by frequency. So if your learning system does not learn frequency accurately, it won't accurately distinguish Pioneer from NEC2. All Pioneer signals should have a device number in the range 160 to 175 and no subdevice. No NEC2 signal should fit those rules. So you usually can determine whether the decision (by frequency) was wrong by checking the device numbers.

NECx

If you get a decode whose protocol name is simply "NECx" that indicates the learned signal is not complete (usually caused by not holding the original remote's button long enough during learning). Enough of the signal is present to accurately determining the device, subdevice and OBC numbers. But not enough is present to determine the exact protocol, which is probably NECx1 or NECx2. This incomplete learn also makes it harder to distinguish NEC from NECx, so a decode of "NECx" might be NEC1 or NEC2 or even Tivo or Pioneer.

NECx1

IRP notation: {38.0k,564}<1,-1|1,-3>(8,-8,D:8,S:8,F:8,~F8,1,^108m,(8,-8,D:1,1,^108m)*)
EFC translation: LSB comp
Most, but not all NECx1 signals have S=D

NECx2

IRP notation: {38.0k,564}<1,-1|1,-3>(8,-8,D:8,S:8,F:8,~F8,1,^108m)+
EFC translation: LSB comp
Most, but not all NECx2 signals have S=D

Nokia

IRP notation: {36k,msb}<164,-276|164,-445|164,-614|164,-783>(412,-276,D:8,S:8,F:8,164,^100m)+
EFC translation: MSB

Nokia12

IRP notation: {36k,msb}<164,-276|164,-445|164,-614|164,-783>(412,-276,D:4,F:8,164,^100m)+
EFC translation: MSB

Nokia32

UEI protocol: 0173
IRP notation: {36k,msb}<164,-276|164,-445|164,-614|164,-783>(412,-276,D:8,S:8,T:1,X:7,F:8,164,^100m)+
EFC translation: MSB

The Nokia32 protocol is one variation of the RCMM 1.5 protocol developed by Philips. RCMM refers to X as "System" and to D:2,S:4:4 as "Customer"

NRC16

Documentation not written yet.

NRC17

UEI protocol: 00BD
irp={500,38k,25%}<-1,1|1,-1>(1,-5,1:1,254:8,255:8,-28, (1,-5,1:1,F:8,D:8,-220)+, 1,-5,1:1,254:8,255:8,-200)
Details of NRC17

OrtekMCE

UEI protocol: not known.
IRP notation: {38.6k,480}<1,-1|-1,1>([P=0][P=1][P=2],4,-1,D:5,P:2,F:6,C:4,-48m)+{C=3+#D+#P+#F}
EFC translation: 6-bit LSB

The repeat frames are not all identical. P is a position code: 0 for the start frame of a repeat sequence, 2 for the end frame and 1 for all frames in between. C is a checksum, 3 more than the number of 1 bits in D, P, F together. DecodeIR v2.37 and later check P and will report in the Misc field if either the start or end frame, or both, is/are missing.

Pace MSS

IRP notation: {38k,630,msb}<1,-7|1,-11>(1,-5,1,-5,T:1,D:1,F:8,1,^120m)+
EFC translation: Not available in this version of DecodeIr

This is a moderately robust protocol, but spurious decodes are still possible.

Panasonic

UEI protocol: 001F or 00C9 or 00CD
IRP notation: {37k,432}<1,-1|1,-3>(8,-4,2:8,32:8,D:8,S:8,F:8,(D^S^F):8,1,-173)+
EFC translation: LSB comp

Panasonic protocol is the most commonly seen member of the Kaseikyo family

OEM_code1 is 2 and OEM_code2 is 32 (or DecodeIr won't display the name as "Panasonic"). So those values in KM or RM can be changed from the default 2 and 32 only when using the Panasonic entry in KM or RM to produce some Kaseikyo variant OTHER THAN Panasonic. When creating a Panasonic upgrade, don't change those from the default values.

Panasonic2

IRP notation: {37k,432}<1,-1|1,-3>(8,-4,2:8,32:8,D:8,S:8,X:8,F:8,(D^S^X^F):8,1,-173)+

Panasonic_Old

UEI protocol: 0000 or 0087
IRP notation: {57.6k,833}<1,-1|1,-3>(4,-4,D:5,F:6,~D:5,~F:6,1,-44m)+
EFC translation: 6-bit LSB comp with 2-bit mini-combo

Lists three different EFCs because this protocol is a mini combo.

PCTV

IRP notation: {38.4k,832}<0,-1|1,-0>(2,-8,1,D:8,F:8,2,-???)

pid-0001

UEI protocol: 0001
IRP notation: {0k,msb}<24,-9314|24,-13486>(24,-21148,(F:5,1,-28m)+)
EFC translation: 5-bit MSB comp

As of version 8.31 KM has seriously wrong OBC translation for pid-0001, so use only EFC's with KM.

pid-0003

UEI protocol: 0003
IRP notation: {40.2k,389}<2,-2|3,-1>(F:8,~F:8,^102k)+
EFC translation: LSB

pid-0004

UEI protocol: 0004
IRP notation: {0k,msb}<12,-130|12,-372>(F:6,12,-27k)+
EFC translation: 6-bit MSB comp

pid-0083

UEI protocol: 0083
EFC translation: 5-bit MSB comp
IRP notation: {42.3K, 3000}<1,-3,1,-7|1,-7,1,-3>(F:5,1,-27)+

This protocol is a very limited design. We have seen it used only in the UEI setup code TV/0159, which is for some TVs brand named Fisher, Sanyo and Sears. It is not likely that any other code set uses this protocol. So if you get a correct decode of pid-0083 you probably have a TV that can be controlled by the TV/0159 setup code.

As of version 8.31, KM does not translate OBC to EFC according to the same rules used by DecodeIr. RM does translate consistently with DecodeIr, so you may find it easier to use RM. If you use KM, you should use the EFC number from the decode not the OBC number. Pid-0083 protocol uses the same EFC numbering across all JP1 remotes, so use of EFC is safe. KM uses different OBC numbering than RM and DecodeIr, so use of OBCs isn't safe.

Pioneer

IRP notation: {40k,564}<1,-1|1,-3>(16,-8,D:8,S:8,F:8,~F:8,1,^108m)+
EFC translation: LSB comp

Pioneer is distinguished from NEC2 only by frequency. So if your learning system does not learn frequency accurately, it won't accurately distinguish Pioneer from NEC2. All Pioneer signals should have a device number in the range 160 to 175 and no subdevice. No NEC2 signal should fit those rules. So you usually can determine whether the decision (by frequency) was wrong by checking the device numbers.

Many Pioneer commands are sent as combinations of two different Pioneer signals. This version of DecodeIr does not associate the two signals together into one command. It decodes them separately. If you get more than one of the same OBC from decoding a learned signal, that just means the learning system failed to understand the repeat pattern. It does not mean a two part signal. But if there are two different OBCs (with the same or different device numbers) you have a two part Pioneer signal.

Including a two part Pioneer signal in a KeyMove or upgrade is a complex process that requires a good understanding of Pioneer signals and of the Pioneer support in KM. The signals don't vary much among related Pioneer models. So the best way to get an upgrade or hex cmd including such signals is to look through existing Pioneer upgrades in the JP1 group and find one that already includes the same (or nearly same) signal.

Proton

UEI protocol: 005C
IRP notation: {38k,500}<1,-1|1,-3>(16,-8,D:8,1,-8,F:8,1,^63m)+
EFC translation: LSB comp

This is not a robust protocol, so spurious decodes are likely.

RC5

UEI protocol: 00E8
IRP notation: {36k,msb,889}<1,-1|-1,1>(1,~F:1:6,T:1,D:5,F:6,^114m)+
EFC translation: 6-bit MSB comp with 2-bit mini-combo

Lists three different EFCs because this protocol is a mini combo.

What we call "device" is really the "System" and what we call "OBC" is really the "Command". If you are using ProntoEdit to create the RC5 signals directly, that GUI uses that correct (System and Command) Philips terminology.

RC5-7F

UEI protocol: 0182
IRP notation: {36k,msb,889}<1,-1|-1,1>(1, ~D:1:5,T:1,D:5,F:7,^114m)+
EFC translation: 7-bit MSB comp

RC5-7F-57

UEI protocol: 0182
IRP notation: {57k,msb,889}<1,-1|-1,1>(1, ~D:1:5,T:1,D:5,F:7,^114m)+
EFC translation: 7-bit MSB comp

RC5x

UEI protocol: 00F2
IRP notation: {36k,msb,889}<1,-1|-1,1>(1,~S:1:6,T:1,D:5,-4,S:6,F:6,^114m)+
EFC translation: NONE

The official (UEI) protocol executor for RC5x does not support EFC numbers. If you are creating an upgrade in KM or RM you should use OBC numbers, not EFC numbers. If you need the Hex Cmd for a KeyMove, you should use the functions sheet of KM or RM to compute it for you from the OBC and subdevice number.

In the functions sheet in KM you must put the subdevice number in the byte2 column, which KM calls "unit code".

What we call "Device" is really the "System". What we call Subdevice is really the "Command". What we call "OBC" is really the "Data". If you are using ProntoEdit to create the RC5 signals directly, that GUI uses the correct (System, Command and Data) Philips terminology.

RC5-?-??

Just ignore this decode. It is almost certainly spurious. In case there is a new protocol I don't know about yet in the family with RC5 and StreamZap, it will decode in this form producing data to help me understand that protocol.

RC6

UEI protocol: 0058
IRP notation: {36k,444,msb}<-1,1|1,-1>(6,-2,1:1,0:3,<-2,2|2,-2>(T:1),D:8,F:8,^107m)+
EFC translation: MSB

RC6 is the name used for the first member of the RC6 family of protocols. Technically this is RC6-0-16, but DecodeIr will always display that as simply "RC6"

RC6-6-20

IRP notation: {36k,444,msb}<-1,1|1,-1>(6,-2,1:1,6:3,<-2,2|2,-2>(T:1),D:8,S:4,F:8,-???)+
EFC translation: MSB

This protocol is commonly used in Sky and Sky+ remotes. As of version 8.31, KM does not have built-in support for this protocol, but there are KM format upgrade files available for Sky and Sky+ (built by an expert who isn't limited to KM's built-in protocols). RM has built-in support for RC6-M-20n protocol, which can be used to make Sky and Sky+ upgrades (or any other RC6-6-20 upgrades as long as the T bit is the same for all learned signals, as it is with Sky). To use RC6-M-20n for RC6-6-20, you must leave the M field in RM's setup sheet with its default value of 6, and you must set or leave the T field (if present) to the value shown in all the decodes (which I assume will be 0).

RC6-?-??

IRP notation: {36k,444,msb}<-1,1|1,-1>(6,-2,1:1,M:3,<-2,2|2,-2>(T:1),???:??)+

This is the generic form for a decode of protocols in the RC6 family. DecodeIr uses this form for all RC6 decodes, except RC6-0-16 which is displayed

as simply "RC6", RC6-6-24 which is displayed as "Replay" and some RC6-6-32 which display as MCE.

The first ? in the protocol name is the M value in the RC6 spec. The ending ?? represents the number of data bits in the signal.

RCA and RCA(Old)

UEI protocols: 00AF (RCA), 002D(RCA(Old)) and 0114 (RCA Combo)
IRP notation for RCA: {58k,460,msb}<1,-2|1,-4>(8,-8,D:4,F:8,~D:4,~F:8,1,-16)+
IRP notation for RCA(Old): {58k,460,msb}<1,-2|1,-4>(32,(8,-8,D:4,F:8,~D:4,~F:8,2,-16)+)
EFC translation: MSB

These are two very similar forms of RCA protocol which differ only in that RCA(Old) has an extended lead-in and a double-length ON pulse before the lead-out. They are so similar that most RCA devices will accept either. But some RCA devices only accept the one that really matches their own remote. In versions of DecodeIR prior to v2.40, RCA(Old) was decoded as a frame of RCA{1} followed usually by a frame of RCA. The second frame now no longer appears, so the protocol has been renamed to correspond to that used in KM and RM.

RCA-38 and RCA-38(Old)

UEI protocol: not known
IRP notation for RCA-38: {38.7k,460,msb}<1,-2|1,-4>(8,-8,D:4,F:8,~D:4,~F:8,1,-16)+
IRP notation for RCA-38(Old): {38.7k,460,msb}<1,-2|1,-4>(32,(8,-8,D:4,F:8,~D:4,~F:8,2,-16)+)
EFC translation: MSB

These are recently discovered variants of the RCA protocol. They differ from RCA and RCA(Old) only in the frequency, which is 38.7kHz instead of the standard 58kHz.

RECS80

UEI protocol: 0045, 0068, 0090 or ???
IRP notation for 0045: {38k,158,msb}<1,-31|1,-47>(1:1,T:1,D:3,F:6,1,-45m)+
IRP notation for 0068: {33.3k,180,msb}<1,-31|1,-47>(1:1,T:1,D:3,F:6,1,^138m)+
EFC translation: 6-bit MSB comp

RECS80 is a family of related protocols with the same structure, but different timing. See also Velleman

These are moderately non robust protocols, so spurious decodes are possible.

The timing differences are not definitive enough for DecodeIr to identify which RECS80 version is which. Instead it displays the timing information in the "Misc" field of the output. That will be three numbers formatted as in this example: (157/5048/7572).

Using those three numbers and the frequency, you should be able to determine whether the signals fit the 0045 version, the 0068 version, the 0090 version or none of them. You should look at all the learned signals for your device together when doing that. A single device won't mix versions of RECS80, so any differences in frequency or timing between learns is due to the IR learning process, not due to any differences among the correct signals. You should find one RECS80 version that is a good enough fit for all signals of the device.

For 0045,

- frequency should be between 37000 and 39000
- first timing number between 100 and 200
- second timing number between 4500 and 5500
- third timing number between 6800 and 8300

For 0068,

- frequency should be between 32300 and 34300
- first timing number between 130 and 250
- second timing number between 5100 and 6300
- third timing number between 7700 and 9500

For 0090,

- frequency should be 0
- first timing number between 0 and 40
- second timing number between 4500 and 5500
- third timing number between 6800 and 8300

You may find decodes that don't quite fit either. If it almost fits, it may be worth testing to see if it works, but it's most unlikely to work if the second timing number is above the suggested max or the third timing number is below the suggested min. For example, I found a decode with frequency 41879 and timing numbers (132,5092,7652). The three timing numbers are perfect for protocol 0045, but the frequency is quite wrong. I have no device to test with, but my guess is that it would work anyway. For protocol 0068, the third number 7652 is below the minimum of 7700 making it quite unlikely to work. I found a different device with frequency 33333 and timing (450,5770,8656). For 0068 all but the first number are perfect and I would be quite surprised if it didn't work. For 0045 the second number 5770 is too high for the max of 5500, so it's unlikely to work.

The decodes for RECS80 all report EFCs for protocol 0045. These are not correct EFCs if you select a protocol other than 0045, so it is better to use

OBC numbers when creating a JP1 upgrade based on these decodes.

Replay

UEI protocol: 0092
IRP notation: {36k,444,msb}<-1,1|1,-1>(6,-2,1:1,6:3,<-2,2|2,-2>(T:1),D:8,S:8,F:8,-100m)+
EFC translation: MSB

Replay is a member of the RC6 family. Technically it is RC6-6-24, but DecodeIr will always display the name as "Replay". ProntoEdit calls this protocol "RC6 mode 6A" and KM has it under the alternate name "RC-6a" as well as its primary name "Replay". RM has it under the alternate name "RC6-M-24n" as well as its primary name "Replay".

DecodeIr's Subdevice field in is called "unit" in KM

In ProntoEdit, DecodeIr's "Device" is called "Customer Code"; DecodeIr's "Subdevice" is called "System"; and DecodeIr's "OBC" is called "Command".

Revox

UEI protocol: 00A0
IRP notation: {0k,15u}<1,-9|1,-19>(1:1,-10,0:1,D:4,F:6,1:1,-10,1:1,-100m)+

Note that Revox uses no modulation. Programs which require a modulation frequency might work with f= 66.7KHz

Samsung20

IRP notation: {38.4k,564}<1,-1|1,-3>(8,-8,D:6,S:6,F:8,1,^???)+
EFC translation: LSB

This is a moderately robust protocol, but spurious decodes are still possible.

Samsung36

UEI protocol: 01B5
IRP notation: {38k,500}<1,-1|1,-3>(9,-9,D:8,S:8,1,-9,E:4,F:8,-68u,~F:8,1,-118)+
EFC translation: LSB

Sampo

IRP notation: {38.4k, 833}<1,-1|1,-3>(4,-4,D:6,F:6,S:6,~F:6,1,-39)+
EFC translation: Not available in this version of DecodeIr

This is a moderately robust protocol, but spurious decodes are still possible.

ScAtl-6

UEI protocol: 0078
IRP notation: {57.6k,846}<1,-1|1,-3>(4,-4,D:6,F:6,~D:6,~F:6,1,-40)+
EFC translation: 6-bit LSB comp

ScAtl-6 is distinguished from Emerson only by frequency. So if you are using a learning system that doesn't record the frequency accurately, then DecodeIr can't accurately select between Emerson and ScAtl-6.

In KM, this protocol is named "Scientific Atlanta". Most Scientific Atlanta cable tuners use Panasonic_Old protocol, not this protocol.

Sejin-*M*-38 and Sejin-*M*-56

UEI protocol: 0161
IRP notation for Sejin-*M*-38: {38.8k,310,msb}<-1|1>(<8:4|4:4|2:4|1:4>(3,3:2,Dx:8,Fx:8,Fy:8,E:4,C:4,-L))+
IRP notation for Sejin-*M*-56: {56.3k,310,msb}<-1|1>(<8:4|4:4|2:4|1:4>(3,3:2,Dx:8,Fx:8,Fy:8,E:4,C:4,-L))+
In both cases E is a checksum seed (0 in all known examples) and C is a checksum given by
C = Dx:4 + Dx:4:4 + Fx:4 + Fx:4:4 + Fy:4 + Fy:4:4 + E.
EFC translation: For Sejin-1, MSB. For Sejin-2, EFC translation not available.

The parameter *M* is either 1 or 2. It distinguishes two styles of this protocol that have different purposes and very different lead-out times L. The 8-bit parameter Dx is a signed integer. If Dx > 0 then the style is Sejin-1, used for normal buttons of a remote control. If Dx < 0 then the style is Sejin-2, used for signals of an associated 2- or 3-button pointing device. E is a checksum seed, E=0 in the only known examples. The checksum formula reflects that in the UEI executor, so is presumed correct.

The protocol parameters Dx, Fx, Fy translate into device parameters in different ways corresponding to the different uses of the protocol. In Sejin-1 the device parameters are a Device Code, a SubDevice code and an OBC as is common for many protocols. Sejin-2 has two sub-styles. One corresponds to the displacement of a cursor or other pointer with device parameters (X, Y) that give the horizontal and vertical components of the displacement (and which can be positive or negative). The other signals Button Up or Button Down for any of the three buttons of the pointing device. The Misc field of the

DecodeIR output displays these device parameters for the Sejin-2 signals. The relationship between these and the protocol parameters is beyond the scope of this document. The Misc field also displays an RMOBC value for Sejin-2 signals, which is an artificial OBC value that can be used as input to RemoteMaster to create the signal concerned.

The protocol parameters for Sejin-1 include a bit that marks the end frame of a repeat sequence. DecodeIR v2.37 and later check this and will report in the Misc field if the end frame is missing. This will normally be due to the key still being held when the learning process ends, so that the end frame gets omitted from the learned signal. For Sejin-2 signals that represent button operations the signal does not repeat. A single frame is sent on button down, a different frame is sent once on button up. Both frames can be detected and distinguished by DecodeIR v2.37 and later but the button up frame will not normally be present in a learned signal.

Sharp, Sharp{1} and Sharp{2}

IRP notation: {38k,264}<1,-3|1,-7>(D:5,F:8,1:2,1,-165,D:5,~F:8,2:2,1,-165)+
EFC translation: LSB

A Sharp signal, which is identical to Denon, has two halves, either one of which is enough to fully decode the information. A significant fraction of Sharp learned signals contain just one half or have the halves separated so that DecodeIr can't process them together. When one half is seen separate from the other, DecodeIr will name the protocol Sharp{1} or Sharp{2} depending on which half is decoded. Sharp, Sharp{1} and Sharp{2} all represent the same protocol when they are correct. But only Sharp is robust. A Sharp{1} or Sharp{2} decode might be spurious.

SharpDVD

UEI protocol: 00F8
IRP notation: {38k,400}<1,-1|1,-3>(8,-4,170:8,90:8,15:4,D:4,S:8,F:8,E:4,C:4,1,-48)+ {E=1,C=D^S:4:0^S:4:4^F:4:0^F:4:4^E:4}
EFC translation: LSB comp

SharpDVD is the member of the Kaseikyo family with OEM_code1=170 and OEM_code2=90.

SIM2

IRP notation: {38.8k,400}<3,-3|3,-7>(6,-7,D:8,F:8,3,-60m)

Nearly identical timing to G.I. RG, so either decode is likely.

Solidtek16

IRP notation: {38k}<-624,468|468,-624>(S=0,(1820,-590,0:1,D:4,F:7,S:1,C:4,1:1,-143m,S=1)3) {C= F:4:0 + F:3:4 + 8*S }

This is a KeyBoard protocol. The make/break bit is decoded into the subdevice field. Checksum is only known to be correct for D = 0

Solidtek20

IRP notation: {38k}<-624,468|468,-624>(1820,-590,0:1,D:4,S:6,F:6,C:4,1:1,-???)

This is a mouse protocol. The button press info is included in the Device field. The horizontal motion is in the Subdevice field, and the vertical motion is in the OBC field.

The decode interface does not support returning negative Subdevice or OBC. So negative motions are represented by adding 64 to them. The numbers 1 to 31 represent positive motion. The numbers 32 to 63 are each 64 larger than the true negative motion, so 63 represents -1 and 32 represents -32.

Somfy

IRP notation: {35.7k}<308,-881|669,-520>(2072,-484,F:2,D:3,C:4,-2300)+
C is reported as SubDevice. It is probably a check nibble {C = F*4 + D + 3}.
F = 1 for UP or 2 for DOWN.
D = 1, 2 or 3 for the three observed devices, or D = 0 to control all devices together.

Sony8

IRP notation: {40k,600}<1,-1|2,-1>(4,-1,F:8,^22200)
EFC translation: LSB.

Sony12

UEI protocol: 00CA
IRP notation: {40k,600}<1,-1|2,-1>(4,-1,F:7,D:5,^45m)+
EFC translation: LSB.

Sony15

UEI protocol: 00CA
IRP notation: {40k,600}<1,-1|2,-1>(4,-1,F:7,D:8,^45m)+
EFC translation: LSB.

Sony20

UEI protocol: 00DE
IRP notation: {40k,600}<1,-1|2,-1>(4,-1,F:7,D:5,S:8,^45m)+
EFC translation: LSB.

StreamZap

IRP notation: {36k,msb,889}<1,-1|-1,1>(1,~F:1:6,T:1,D:6,F:6,^114m)+
DecodeIR V2.43 decodes this as RC5-7F
EFC translation: 6-bit MSB comp

StreamZap-57

IRP notation: {57k,msb,889}<1,-1|-1,1>(1,~F:1:6,T:1,D:6,F:6,^114m)+
DecodeIR V2.43 decodes this as RC5-7F-57
EFC translation: 6-bit MSB comp

Sunfire

IRP notation: (38k,560,msb)<1,-1|3,-1>(16,-8, D:4,F:8,~D:4,~F:8, -32)+
EFC translation: Not available in this version of DecodeIr

TDC-38 and TDC-56

UEI protocol: 01BB (38KHz) 01BD (56KHz)
IRP notation for TDC-38: {38k,315,msb}<-1,1|1,-1>(1,-1,D:5,S:5,F:7,-89m)+
IRP notation for TDC-56: {56.3k,213,msb}<-1,1|1,-1>(1,-1,D:5,S:5,F:7,-89m)+
EFC translation: 7-bit MSB.

There are two variants of this protocol, with different frequencies but with the same number of carrier cycles in each burst, which makes the duration of a burst also differ. TDC-38 has a 38kHz carrier and is used by Danish TDC IPTV. TDC-56 has a 56.3kHz carrier and is used by Italian ALICE Home TV box. These implementations effectively use a 6-bit OBC as bit 0 of F is always the complement of bit 1, but there are other implementations which do not follow that pattern.

Teac-K

UEI protocol: 00BB
IRP notation: {37k,432}<1,-1|1,-3>(8,-4,67:8,83:8,X:4,D:4,S:8,F:8,T:8,1,-100,(8,-8,1,-100)+ {T=D+S:4:0+S:4:4+F:4:0+F:4:4}
EFC translation: LSB comp, two parts

Teac-K is the member of the Kaseikyo family with OEM_code1=67 and OEM_code2=83.

Teac-K uses different repeat rules and a different check byte than other Kaseikyo protocols.

00BB requires 2-byte hex commands. DecodeIr returns both hex cmd bytes through the interface that usually means one or the other (for mini combos) but in this case it means both.

This protocol signals repeats by the use of dittos.

Thomson

UEI protocol: 004B
IRP notation: {33k,500}<1,-4|1,-9>(D:4,T:1,D:1:5,F:6,1,^80m)+
EFC translation: 6-bit LSB comp, or that prepended with extra device bit.

This is not a robust protocol, so spurious decodes are likely.

DecodeIR2.42 deprecates Thompson (5 bits of device, and 6 bits of function) and reports these signals as Thompson7 (4 bits of device and 7 bits of function).

Thomson includes a toggle bit so using learned signals will have operational problems. You should use KeyMoves or Upgrades based on the decoded values, rather than continue to use the learned signals.

There are two different variants of UEI protocol 004B which have different EFC numbering. The decode lists both possible EFCs so you could experiment to discover which is right for your model. But, if you are creating an upgrade (rather than just KeyMoves) it is better to use RM and use the OBC numbers from the decode (which are stable across models of JP1 remote). As of version 8.31, KM does not have support for Thomson protocol,

so if you must make an upgrade in KM you need to use pid:004B. For the URC-8040 and 8060 the second decoded EFC should be right and the OBC values in KM should be wrong. For most (maybe all) other models, the first decoded EFC should be right and KM's default EFC to OBC translation should also be right.

Thomson7

UEI protocol: 004B
IRP notation: {33k,500}<1,-4|1,-9>(D:4,T:1,F:7,1,^80m)+
EFC translation: 7-bit LSB comp

DecodeIR2.42 deprecates Thompson (5 bits of device, and 6 bits of function) and reports these signals as Thompson7 (4 bits of device and 7 bits of function).

Tivo

IRP notation: {38.4k,564}<1,-1|1,-3>(16,-8,133:8,48:8,F:8,U:4,~F:4:4,1,-78,(16,-4,1,-173)*)
EFC translation: LSB comp

Velleman

IRP notation: {38k,msb}<700,-5060|700,-7590>(1:1,T:1,D:3,F:6,1,-55m)+
EFC translation: 6-bit MSB comp

Very similar to RECS80-0045, except on duration is longer

Velodyne

IRP notation: {38k,136,msb}<210,-760>
(<0:1|0:1,-1|0:1,-2|0:1,-3|0:1,-4|0:1,-5|0:1,-6|0:1,-7|0:1,-8|0:1,-9|0:1,-10|0:1,-11|0:1,-12|0:1,-13|0:1,-14|0:1,-15>
(T=0,(S:4:4,~C:4,S:4,15:4,D:4,T:4,F:8,210u,-79m,T=8)+)) {C=(8+S:4+S:4:4+15+D+T+F:4+F:4:4)&15}
Velodyne is a close relative of XMP.

Viewstar

UEI protocol: 0021
IRP notation: {50.5k,337}<1,-8|1,-5>(F:5,1,-17)+
EFC translation: 5-bit LSB comp

This is not a robust protocol, so spurious decodes are likely.

X10 and X10.*n*

UEI protocol: 003F (X10.*n*), 01DF (X10)
IRP notation for X10: {40.8k,565}<2,-12|7,-7>(7,-7,F:5,~F:5,21,-7)+
IRP notation for X10.*n*: {40.8k,565}<2,-12|7,-7>(F:5,N:-4,21,-7,(7,-7,F:5,~F:5,21,-7)+)
EFC translation: LSB of 2*OBC+1

These are two variants of the same Home Automation protocol. They differ in that X10.*n* has a distinctive start frame that carries a sequence number, the *n* of the protocol name, in addition to the OBC. The repeat frames, and all frames of the X10 version, only carry the OBC. The value of *n* runs from 0 to 15 (or some lower value) and then restarts again at 0. It is incremented on each successive keypress. A valid X10.*n* signal must have at least one repeat frame. If this is missing then the Misc column shows "invalid signal".

RemoteMaster has a single protocol, named X10 with PID 003F, that sends X10.*n* signals. This is the same as the UEI protocol with that PID. There is no control over the value of *n*, this is handled automatically by the remote. The newer UEI protocol, with PID 01DF, sends X10 signals.

XMP, XMP-1 and XMP-2

UEI protocol: 016C
IRP notation (without final frame): {38k,136,msb}<210,-760>
(<0:1|0:1,-1|0:1,-2|0:1,-3|0:1,-4|0:1,-5|0:1,-6|0:1,-7|0:1,-8|0:1,-9|0:1,-10|0:1,-11|0:1,-12|0:1,-13|0:1,-14|0:1,-15>(T=0,
(S:4:4,C1:4,S:4,15:4,OEM:8,D:8,210u,-13.8m,S:4:4,C2:4,T:4,S:4,F:16,210u,-80.4m,T=8)+)){C1=-(15+S+S::4+15+OEM+OEM::4+D+D::4),C2=-
(15+S+S:4+T+F+F::4+F::8+F::12)}
IRP notation (with final frame):{38k,136,msb}<210,-760>
(<0:1|0:1,-1|0:1,-2|0:1,-3|0:1,-4|0:1,-5|0:1,-6|0:1,-7|0:1,-8|0:1,-9|0:1,-10|0:1,-11|0:1,-12|0:1,-13|0:1,-14|0:1,-15>(T=0,
((S:4:4,C1:4,S:4,15:4,OEM:8,D:8,210u,-13.8m,S:4:4,C2:4,T:4,S:4,F:16,210u,[-80.4m][-80.4m][-13.8m],T=8)+,T=9)2)){C1=-
(S+S::4+15+OEM+OEM::4+D+D::4),C2=-(S+S:4+T+F+F::4+F::8+F::12)}
XMP uses one burst pair to encode numbers 0 to 15, with an on duration of 210uS, and off duration of 760uS + n*136uS where n takes on values of 0 to 15. The checksum nibble is the complement of 15 plus the sum of the other 7 nibbles, mod 16

The Device code is D, the SubDevice code is S and there are two OBC values. OBC1 is the high byte of F, OBC2 is the low byte of F. The OEM code is normally 0x44 and is reported in the Misc field only if it has a different value. The XMP-1 protocol is XMP with OBC2 = 0. The OBC field in DecodeIR

then shows OBC1. The XMP-2 protocol is XMP with OBC1 = 0. The OBC field in DecodeIR then shows OBC2.

This protocol has a 4-bit toggle T that is 0 for the first frame and normally 8 for all repeat frames. There is, however, a variant in which a further frame with T=9 is sent after the button is released, separated from the preceding frame by the short leadout of 13.8m that is used between two half-frames rather than the long lead-out of 80.4m used at the end of all other frames. When this frame is detected then the Misc field displays "With Final Frame". For this to be shown in a learned signal, the button must be released before the learning process times out, so a short button press is needed.

These are problem decodes because JP1 remotes don't typically learn these signals accurately enough for a correct decode. NG Prontos also do a rotten job of learning these signals. Older Prontos seem to do fairly well. DecodeIR v2.40 includes algorithms that attempt to reconstruct a valid XMP signal from a corrupt learn, but it is impossible to correct all learning errors and there can be no certainty that a reconstruction is actually correct.

In a correctly learned or fully reconstructed signal there will be an "XMP", "XMP-1" or "XMP-2" decode with device, subdevice and OBC values that can be used with RemoteMaster or any similar program to regenerate a clean signal. The Misc field shows which algorithms, if any, have been applied, as a list in brackets after any decode data in this field. There are notes below on the reliability of the various algorithms. When the protocol shows as (unqualified) XMP, both OBC values are non-zero. The OBC and Hex fields show OBC1. The corresponding values for OBC2 are shown in the Misc field.

The learned signal itself will certainly not be valid if any reconstruction algorithms have been applied and it may not be so even if it has been decoded without reconstruction. The possible algorithm indicators in the Misc field are as follows:

- End (= Endpoint): The lead-out burst is missing and has been inserted. This is almost certainly correct.
- Rec (= Recovery): Look-ahead has been used to recover a missing burst from the following repeat frame. This is very likely to be correct.
- Cor (= Correction): Two bursts have been coalesced in the learning process, e.g. those for hex digits C and D, causing a C to appear as D or vice versa. The error has been identified and corrected. This is probably correct.
- Cal (= Calculated): A missing digit has been calculated from a checksum. The digit is probably correct but it may be in the wrong place. The most likely error in the reconstruction is that the two digits of the OBC are the wrong way round.
- Cal2 (= Calculated 2) Two consecutive missing zero digits have been identified, corresponding to a zero OBC. When this happens, the signal will always be shown as XMP-1. The most likely error in the reconstruction is that it should actually be XMP-2.

If a learned signal is good enough to be recognised as XMP but not good enough to be fully reconstructed, the protocol will display with a name of the form

XMP:136.218-0F0F441A0A800F00

In IR.exe you'll need to widen the Protocol column to see the whole thing. This represents intermediate data from an unsuccessful attempt to decode a XMP signal. The number in the position where the 136 is in this example represents the time scale. A number (like this example) that is near 137 is reasonable. A number much further from 137 indicates a more serious learning or decoding problem. The number in the position where the .218 is in this example (it is not part of the 136) represents the level of inconsistency in the individual hex digit decodes. A value greater than .100 means the hex digits aren't very reliable.

The hex string, where the 0F0F441A0A800F00 is, is the decoded data. At least one digit is almost certainly wrong or the whole decode wouldn't be displayed in this form. With a JP1 learning remote, the most common errors are that a digit is actually missing, in which case the string will have fewer than 16 hex digits, or that two or more digits which are decoded the same are actually different, so some of them are correct and some are one value higher or lower. Although the reconstruction algorithms attempt to correct these types of errors, it is not always possible. In this example I happen to know the correct signal. One of the three F's is really an E and one of the two A's is really a 9. The correct string is 0E0F441A09800F00.

Almost all examples we've seen start with "0E0F441A0" or "060F44120". But we've also seen upgrades from UEI for "0D1F441A0" and "0C2F441A0" and "0B3F441A0". The last 4 digits of the whole 16 digit string (if they are correct) represent the Hex command needed to reproduce the signal in a JP1 upgrade or KeyMove. DecodeIR shows them as two 8-bit OBC values, as described with the IRP notation above.

XX

Documentation not written yet.

Zaptor

UEI protocol: unknown
IRP notation: {36k,330,msb}<-1,1|1,-1>[T=0] [T=0] [T=1] (8,-6,2,D:8,T:1,S:7,F:8,E:4,C:4,-74m)+ {C = (D:4+D:4:4+S:4+S:3:4+8*T+F:4+F:4:4+E)&15}
where T=0 for all frames except the last, T=1 for last frame, E is a checksum seed. There is a 56KHz variant.
EFC translation: MSB

A protocol so far seen only in the Motorola Zaptor. See also Amino

Zenith

UEI protocol: 0022 IRP notation: {40k,520,msb}<1,-10|1,-1,1,-8>(S:1,<1:2|2:2>(F:D),-90m)+
Before Version 2.43, this document has shown the IRP notation as {40k,520,msb}<1,-1,1,-8|1,-10>(S:1,<1:2|2:2>(F:D),-90m)+
EFC translation: MSB

An unusual protocol, in that the number of bits in the function code is variable. It is represented in DecodeIR as the device code. There are also two lead-in styles, decoded as subdevice values 0 and 1. Style 1 aka "double-start" is usually used in TV's, other appliances use 0 aka "single start". If the device code is >8 then the bytes given in the Misc field as E = ... follow the OBC in the function code value.

?1-??-??-??

An experimental decode I added based on the thread at JP1-forum"Unknown Protocol"

(from: http://www.hifi-remote.com/johnsfine/DecodeIR.html)