Before you start. When connecting to the Arduino Nano interface, pins 1&2 have “ATT” on the Printed Circuit Board. “JP1” is next to pins 5&6.

EXTENDER FOR URC-8910

Basic Extender features for URC-8910:

- You can put a Macro or KeyMove on any key.
- You have room for lots more KeyMoves and Macros.
- Macros can be Nested.
- Macros are fast.
- Very flexible device selection.
- Support for logical devices (w/o key moves)
- Control of device displayed on LCD.
- Improved (very different) Fav list processing.
- Configurable backlight timer.
- Optional second shift key.
- Automatic PC-to-Remote clock synchronization
- Included special protocols for:
  Pause, Device Specific Macros, ToadTog, Long/Double Key Press, Custom Mode Names on LCD

No longer available with Extender:

- No Learned signal support.
- No Scan support.
- No commercial skip support.
- No direct keying of EFC codes on the remote.
- No long-press-setup functions

INSTALLATION AND ACTIVATION

Run RMIR, and download the contents from the remote. On the RMIR menu, do File > Install Extender… and select the most recent extender Hex file: 8910ex1.hex — and RMIR will now contain the extended settings for your remote. The next time you upload to the remote, RMIR will report that the checksum and/or the device signature doesn’t match. This will happen first time because when we add the extender, the device signature is changed. Don’t worry, go ahead with the upload!

SPECIAL PROTOCOLS

This extender includes the following special protocols:

- MISC/1103, portocol 1FC = Device specific Macro
- MISC/1104, protocol 1FB = Pause
- MISC/1106, protocol 1F9 = Long/Double Key Press
- MISC/1107, protocol 1F8 = Custom Mode Name
- MISC/1800, protocol 181 = ToadTog

These protocols are explained in this document. The Pause protocol is described in the FAST MACROS section in this document.

DEVICE SELECTION:

The device selection mechanism supports the following:

- 8 physical device indexes: TV, VCR, CBL, SAT, CD, DVD, AUX, and AUD
- 7 logical device indexes: db-08, db-09, db-10, db-11, db-12, db-13, and db-14
- 6 key sets: Menu, Channel, Volume, Transport, Other and PIP

There is no normal device selection. There is no VPT. There is no Transport Punch Through. There is no Home Theater Mode. Instead there is a device selection mechanism by key set that has a superset of the power of all the above; But you must define macros to use and customize it.

The device keys have no built in meaning. You must define macros to have any device selection at all.

There are 15 DEV_ commands that can be used in macros to temporarily set the active device. There is a separate command for each of the 8 physical devices, as well as for each of the 7 logical devices.

Each command specifies the Device Index that will be used. For example, DEV_TV sets the TV device to the active device. DEV_ commands apply to all keys, but only last until the outermost macro (see Nested Macros) completes or another DEV_ command is encountered. DEV_ commands do not permanently set the active device. See Temporary Device Selection for more details.

There are 6 SET_xxx_KEYS commands that can be used in macros to tie the current active device (as specified by a DEV_ command) to a particular key set. Once a key set is tied to a device, it remains tied to that device until the same SET_xxx_KEYS command is used to tie it to another device. Keys bound by a SET_xxx_KEYS command remain bound, even outside of the macro.

For example, to set the transport keys to the VCR device, you would use the macro "DEV_VCR; SET_TRANS_KEYS". This would set the transport keys to the VCR device until another macro set them to something else.

There is also a SET_LCD command that can be used in macros to display the name of the current active device (as specified by a DEV_ command) on the LCD. Once the LCD is tied to a device, it remains tied to that device until the SET_LCD command is used to tie it to another device. The LCD will remain set, even outside of the macro.

Note that the SET_LCD command will always set the LCD to "MySYS" if the current active device is DEV_DB08. See the Custom Mode Name special protocol to customize the names associated with each device.

To achieve simple device selection (no VPT, TPT or HT) and set the LCD, you would put an 8 command macro on each device key, for example the TV device key would have the macro:

DEV_TV; SET_TRANS_KEYS; SET_VOL_KEYS; SET_CHAN_KEYS; SET_MENU_KEYS; SET_PIP_KEYS; SET_OTHER_KEYS; SET_LCD

The extender supports both the physical and logical devices. You can use any keys you want for the device selection macros (and you can use device keys for other purposes).

To achieve VPT, replace the SET_VOL_KEYS command within the macro for each device that should have VPT enabled with the commands to set the VPT keys to the desired device. For example, to enable TV VPT on your VCR, the VCR key macro might be:

DEV_TV; SET_VOL_KEYS; DEV_VCR; SET_TRANS_KEYS; SET_CHAN_KEYS; SET_MENU_KEYS; SET_PIP_KEYS; SET_OTHER_KEYS; SET_LCD

To achieve crude TPT (such as most of the UEIC remotes) replace the SET_TRANS_KEYS command within the macro for each device that should have TPT enabled with command sequence to set the transport keys. For example, your TV key macro might be:

DEV_VCR; SET_TRANS_KEYS; DEV_TV; SET_VOL_KEYS; SET_CHAN_KEYS; SET_MENU_KEYS; SET_PIP_KEYS; SET_OTHER_KEYS; SET_LCD

To achieve dynamic TPT (similar to the URC7800). Omit the SET_TRANS_KEYS command from the macro for each non Transport device and include it in the macro for each transport device (VCR, DVD etc.). When you select a non transport device, the transport keys will all remain associated with whatever transport device was most recently selected.

To achieve Home Theater, put a device selection macro on the Home Theater key. Include specific DEV_ and SET_xxx_KEYS to select your HT settings. To duplicate the UEIC version of HT mode, omit the SET_OTHER_KEYS command from the HT macro so that the Other keys remain associated with whatever device they were associated with before. (UEIC HT mode seems to support the Power key only as a special macro. With this extender the Power key is either an ordinary macro or an ordinary Other key, whichever you programmed it as).

Shifted keys all go into the key set containing their unshifted counterpart. For example SHIFT-Stop is in the Transport set. Phantom keys are in the Other set. Device keys are in the Other set; But most people will put macros on device keys. If a key is a macro, it doesn't matter which set it is in. The set only matters for KeyMoves and for keys defined by setup code.

Temporary Device Selection:

Within a macro, you often want to issue a key to a specific device regardless of the device the key is tied to and without disturbing that relationship. You can use the DEV_ commands to do that. For example, the sequence:

DEV_TV; 0; 3

in a macro would send the 0 and 3 keys to the TV regardless of the device tied to the 0 and 3 keys.

The DEV_ selection is automatically cancelled when the outermost macro (see nested macros) completes.

There is also DEV_Cancel command which will cancel the current DEV_ command in a macro. If the above example were intended for use as a top level macro, there would be no need for it to explicitly cancel its DEV_ command. If the above example were in a general purpose macro that might be called by other macros, you probably should change it to:

DEV_TV; 0; 3; DEV_Cancel

While a DEV_ command is active, it applies to all keys. Any assignments made by the SET_xxx_KEYS commands are ignored.

Logical Devices

Logical devices behave the same as the physical devices, with the following exceptions:

- You can't put KeyMoves on logical devices. This is because there are only 3 bits available to specify the device in a keymove and all of the logical devices numbers take 4 bits.

If you use the SET_LCD command on with logical device db-08, it will display "MySys" on the LCD regardless of the device type associated with it.

NESTED MACROS:

You can nest macros to any depth. Any key that is a macro when used from the keyboard is the same macro when used inside a macro.

This is very different from the base remote and from other extenders. Pay careful attention to this detail when converting a configuration for use with this extender. There is no protection from infinite loops when a macro nests into itself.

Many people have used the fact that macros don't nest (in the basic remote) for things like a Power macro that uses the normal Power key. Find and change anything like that in your configuration (see "Cloaking with Shift").

There is a 21 byte macro buffer (the same 15 byte buffer used by SelNestMac on other remotes). That doesn't change the limit on an individual macro (still 15 commands) nor does it set a limit on the total number of commands executed by one macro (virtually unlimited). It limits the number of commands "pending" at any one moment.

To understand "pending" commands, imagine 4 macros, A, B, C, and D:

- A = B; C; D
- B = 1; 2; 3
- C = 4; B; 5; 6
- D = 7; 8; 9

when you press A, you get 1 2 3 4 1 2 3 5 6 7 8 9, which is 12 commands, but in executing those 12 commands, there were never 12 commands "pending".

- When the extender processes the first B there are 5 command pending: 1; 2; 3; C; D;
- Later it process the C and there are again 5 commands pending: 4; B; 5; 6; D
- When it processes the second B there are 6 commands pending: 1; 2; 3; 5; 6; D

The whole 12 commands are sent with a maximum of 6 ever pending. You should be able to design ridiculously long macros without ever hitting the limit of 32 pending commands.

FAST MACROS:

I reduced both the hold time and the delay time for commands in a macro. I think that is necessary to make macros useful. There are situations in which you need to add back some hold time or delay time.

For delay, you can use:

1) For a very small delay, use a redundant device selection command. If you know that a DEV_ selection won't be in use at the relevant point in macro execution, you can use a redundant DEV_Cancel as a tiny delay. If you know or use any other device selection, you can use it again as a delay. For example, if you want a delay between digits in the macro "DEV_TV; 0; 3" you could use "DEV_TV; 0; DEV_TV; 3".

2) For a slightly longer delay, use an undefined key code. The actual amount of delay will depend on the number of items in your KeyMove and Macro area. For example, if you have no KeyMove or Macro for xs_Phantom1 you could use xs_Phantom1 as a delay.

3) For a long delay, use a KeyMove connected to the Pause protocol (MISC/1104). The hex command is the amount of delay from 01 (smallest) to FF.

For adding back hold time, I haven't provided anything in this version of the extender, except for the last step of a macro.

HOLDING LAST STEP OF A MACRO:

If the last step of a macro transmits a signal, and you held down the original key that started the macro through the entire macro execution, the extender will continue the last signal while that button is held, just as the remote normally does for a signal that is the only action of a button.

This feature acts the same for ToadTogs and DSMs as for ordinary macros. It acts the same for mini-macros (from a Fav list) as well, except that all mini-macros are really exactly 3 commands long, rather than being 1 to 3 commands long as they seem. Any shorter mini-macros will be padded with NULL commands to make them 3 commands long. If a mini-macro is less than 3 commands, its last command is a NULL command, so holding the key won't make a difference. If you want holding the key to make a difference, try padding the beginning or middle of the mini-macro with redundant device selection commands rather than letting RMIR pad the end with NULL commands.

In rare cases, you want to defeat this feature and avoid having the last step of a macro continued if the user holds the key. You can most easily do that by adding an X_ command to the end.

SHIFTED KEYS:

Pressing the Setup key causes the next key to be "shifted". The shift only affects the lookup of the key as a KeyMove or Macro. If no KeyMove or Macro is found for a shifted key, the remote then checks whether the unshifted version of the key is defined by the setup code.

The Setup key only acts as a shift key when used from the keyboard. When used in a macro the Setup key is just an ordinary (Other set) key. When used from the keyboard the Setup key is both a shift key and an ordinary key, so you can get some confusing or interesting (depending on your intent) behaviour by defining a KeyMove or Macro for the Setup key.

To use a shifted key in a macro use the "Add Shift" option of IR — Do not try to make it shifted by preceding it with the Setup key.

Changing the Shift Key:

In the General pane of RMIR you can configure shift to be a different key or two keys.

"Shift Button Keycode" defines the primary shift key. The value is displayed as a decimal number. The default is 2, which is the Set key. You can change it to any key you prefer. If you look up the keycode in 8910-KeyCodes.pdf, that value will be in hex. You will have to convert it into a decimal value before typing it into the settings area.

"Alt Shift Button Keycode" defines an optional second shift key. If that is the same as the primary shift keycode then the extender will use only the primary. If it is a different valid code, that code will be a second shift.

"Alt Shift" selects whether the second shift key performs an xshift operation or a ordinary (same as the first shift code) shift operation. You can select ordinary to have two shift keys that perform the same operation. Using xshift you can have up to three key moves on other keys by having an unshifted keymove, a shifted keymove, and an xs-keymove.

The second shift key can be a code shifted by the first shift key. In fact it can be the first shift key shifted by itself. For example, I set the first shift keycode to $02 and the second one to $82 and selected "xshift". Then if I press Set once, the next key is shifted; If I press P twice, you get the xs version of the next key; If I press P three times, the next key is back to normal.

Cloaking with Shift:

If you want to define a macro for a key (such as Power) but also use that key as defined by the setup code (probably in that macro) the trick is to use the shifted version of the key. For this to work, you must not define a KeyMove for the shifted key (in the current device index) nor define a macro for it. Then put shift-Power inside the Power macro. When the extender fails to find a KeyMove or macro for shift-Power, it looks in the setup code for a definition of Power and finds the one that you couldn't access directly because the macro is in the way.

PC-TO-REMOTE CLOCK SYNCHRONIZATION

The extender will automatically synchronize the time on the remote to the time on the PC if it is invoked within one minute of the download. If you set the time on the remote or wait longer than a minute after the download, it will not change the time on the remote.

In reality, this code works by checking the current time on the remote, adding the hours and minutes, and if the total is 12 (i.e. 12:00), it will reset the time. Otherwise it will not. This means that there are other times (i.e. 1:11, 2:10, etc.) that will reset the time when it shouldn't. I took the shortcut to save some memory.

KEY SETS:

Trans = Rew, Play, FFwd, Rec, Stop, Pause
Vol = Vol+, Vol-, Mute
Chan = Ch+, CH-, digits, Enter, Last, Sleep, Info, TV/Video
Menu = Menu, Guide, Up, Down, Left, Right, Select, Exit
PIP = Pip, Freeze, Swap, Move, +100
Other = P{Setup}, {Light}, Power, Fav/Scan, device keys, phantoms

FAV/SCAN:

You can use RMIR to define a Fav list of up to 15 mini-macros of up to 5 keys each. RMIR handles all 15 mini-macros together as one line item in its Scan/Fav tab. Within that line item, the mini macros are delimited by "{pause}". RMIR also allows you to create additional whole Fav lists; But the remote can't use them.

Not that your Fav list can only be used on one device and will only work when that device is selected.

The extender operates the Fav list differently than the unextended remote in these two important aspects:

1) The extender executes just one mini-macro each time you press the Fav key (advancing through them circularly). The unextended remote will continue executing the mini-macros with 3 second pauses in between until you press another key to stop it.

2) The unextended remote will do a Scan operation instead of a Fav list operation whenever the Fav key is used in a device index that doesn't match. The extender will never do a Scan operation.

If you prefer, you can use the Fav key as a normal key for each device, by using Key Moves.

BACKLIGHT AND SHIFT TIMER:

When the backlight is enabled, it comes on at the later of the end of each operation or the release of the key that started the operation. It then stays on for up to ten seconds or until you press another key.

You can change that ten second timer to another value. Multiply the number of seconds you want by 7.6 and round to the nearest integer. Scroll the "Other Settings" pane of IR down to show "Backlight timer" and type the computed value there. For example, if you want 12 seconds, you would compute 12 * 7.6 = 91.2 and enter the value 91. The value must be in the range 1 to 255.

Whether or not the backlight is enabled, the extender also uses the same timer to cancel shift operations. After you release the shift key, you have only the length of time programmed as the backlight timer to press the next key in order for that key to be shifted.

TROUBLE SHOOTING:

This is a complicated extender that may still have some bugs in it, and it lets you define very complex behaviours for your remote, which will probably have errors on first try.

The major method of trouble shooting is to break complex operations down into their parts and see if the parts work individually.

Example: You have a macro that does some device selection and does two phantom codes and each phantom code does a ToadTog and it doesn't work. Assign those two ToadTogs to pressable keys (temporarily for trouble shooting). If necessary, define some simple macros to duplicate the device selection of the complex macro. Test the individual pieces of the complex macro and see which work.

Example: You have a ToadTog that doesn't work: Make a simple macro or DSM out of each command list of the ToadTog. Test the two sequences that way.

A moderate fraction of macro, DSM and ToadTog problems are due to one of two timing issues:

1. The extender reduces (to the minimum value) the duration of signals which are not the very last step in its macro buffer and signals which are processed after the user has released the key that started the whole operation. If you suspect you have a duration problem, you should confirm it with two tests. Put the single function on a pressable key and press and hold that key. If that doesn't work, something more basic than a duration problem is wrong. Put a macro or DSM on a pressable key that is just the problem function followed by an X_Cancel. When you press that key, the function is not last, so the duration will be minimum. If that introduces the failure you know it's a duration problem (there are then a variety of approaches to fixing it).

2. The extender reduces the time between signals within any automated sequence of signals. If functions work when manually sent in sequence, but don't work when automatically sent in sequence, and you determine it's not a duration problem, then it's time to try delays. Define a KeyMove for a moderately long use of the pause protocol. Insert that before and/or after the problem function(s) in your sequence. If that fixes it, you've confirmed that it is a timing problem and you can then experiment to fine tune the correction to fix the problem with minimum increase in the time it takes the whole operation to complete.

Rarely, a device is sensitive to any signal at all right before or right after its signal. In those cases, the best you can do is find the smallest added delay that fixes the problem.

More often, a device just needs time to do the operation you gave it before it is ready for the next. In a complex macro, you may have an alternative to adding delay. If you're sending two commands to device A, then one command to device B, try sending the command to device B in between the two commands to device A. That probably takes one or two extra DEV_ commands beyond doing it the simple way, but probably takes less total execution time than simply adding delay.

CUSTOM MODE NAME PROTOCOL

This special protocol is packaged as an upgrade protocol (1F8) and an upgrade device (MISC/1107) that are automatically installed when you install the extender. If you don't need this protocol and want to conserve upgrade memory, you can delete it.

By default, the remote displays the name of the current active device as the name of the current mode. This special protocol allows you to customize the name of the mode displayed on the LCD screen.

Setup

On the Special Functions tab, click New from the options at the bottom. Select the Device Button and the Key for which you want to define the custom name. Under “Type” select “ModeName” and in the box that pops up alongside, type your 9 character name. Bear in mind that the first 5 characters will appear on the first line in the remote’s screen, and the other 4 in the second line.

When you execute a macro that includes that Device and Key, the text you defined will, from then on, appear in the remote’s screen, until changed to something else.

DEVICE SPECIFIC MACROS SPECIAL PROTOCOL

This is an enhanced version of the Device Specific Macros protocol. It is packaged as an upgrade protocol (1FC) and an upgrade device (MISC/1103) that are automatically installed when you install the extender. If you don't need this protocol and want to conserve upgrade memory you can delete those upgrades.

Setup

On the Special Functions tab, click New from the options at the bottom. Select the Device Button and the Key to execute the macro. Under “Type” select “DSM”. You can now define your sequence of keys for your macro.

LONG/DOUBLE KEY PRESS SPECIAL PROTOCOL

This special protocol allows you to put two different macro sequences on a single button and execute either one of them based on the key press pattern, Short vs. Long key press or Single vs. Double press.

Similar to the macros in the extenders this special protocol supports, these are fast, can be nested (each macro comes back to the caller when it finishes) and the last step in the entire macro repeats the signal while you hold down the button.

The way you set up a keypress, is very similar to the Device Specific Macro above, except that you define two macros, one will be executed on a short press, one on a long press. The time to set to wait to see if it's a long or short press, or a single or double press, can also be set.

If you want to set a L/DKP to work on any device, use a phantom key then call it from the actual key. For example, put your L/DKP on TV-Phantom3 and put a macro DEV_TV;Phantom3 on the desired key.

TOADTOG SPECIAL PROTOCOL

This is an enhanced of the ToadTog special protocol. It is packaged as an upgrade protocol (181) and an upgrade device (MISC/1800) that are automatically installed when you install the extender. If you don't need a ToadTog protocol and want to conserve upgrade memory you can delete this device.

ToadTog lets you construct toggle functions from discrete functions, and/or construct discrete functions from toggle functions, and/or track the state of device toggles so that other functions may be sent differently depending on that state.

Setup

On the Special Functions tab, click New from the options at the bottom. Select the Device Button and the Key to execute the function. Under "Type" select "ToadTog". You then have to select which "bit" will store the current state of the toggle, there are 8 available, numbered 0-7. There are three available functions, plus a Test Only function. They are Toggle, Force On and Force Off.

In Toggle mode, you should define a macro that will play if the current state is off, called Off>On, and another macro that will play if the current state is on, called On>Off. This will enable you to create a toggle where one doesn't exist on the original remote. As well as turning things on and off, another use is for a "Radio" button on a receiver, alternately selecting, say, FM and DAB.

In Force On mode, you have a macro to send if the current state is already on (often blank), and one to send if it's off. This might be a code that normally toggles the power supply, but here is only sent if it's off, thus turning it on.

In Force Off mode, you have a macro to send if the current state is already off (often blank), and one to send if it's on. This might be a code that normally toggles the power supply, but here is only sent if it's on, thus turning it off.

A disadvantage of ToadTog is that it's possible that the "current state" in the remote may get out of sync with the actual device, for example, if the device fails to receive a signal that has been sent. There should be a way to straighten it out, for example, use an XShifted key to toggle the state of the device without affecting the state as stored in the remote.