

Integration of iLoveControl into AMX Control Systems

MODULE DOCUMENTATION AND INSTALLATION INSTRUCTIONS

Version 0.9.1 RC1

Contents

Preface.....	2
<i>iLoveControl.....</i>	2
AMX.....	2
The ilc2amx software components.....	2
Main features of ilc2amx.....	2
System requirements.....	3
ilc2amx package content.....	3
System architecture.....	4
Overview.....	4
Module documentation.....	5
General definitions	5
<i>Module configuration files</i>	5
<i>Exception handling</i>	6
Module wc_ilcserver_dr1_0_0.....	7
<i>Module configuration</i>	7
Module wc_ilcconnector_dr1_0_0.....	8
<i>Module configuration</i>	8
Helper module wc_ilcconfig_dr1_0_0.....	8
<i>Module configuration</i>	8
Required global constants.....	10
Operation modes (work in progress).....	11
Operation mode 1.....	11
Operation mode 2.....	12
Operation mode 3.....	12
Appendix.....	13
ilcfuncmap: default G4API definition.....	13
ilcfuncicsp: default G4API definition.....	15

Preface

iLoveControl

iLoveControl is an iPhone application that enables you to control a wide range of devices using your iPhone or iPod Touch. Information about iLoveControl can be found on the product website <http://www.ilovecontrol.com>

AMX

AMX is a home and general purpose control system. AMX systems consists of one or several central controllers and one or many remote controls, keypads or touch panels. Information about AMX can be found on the product website <http://www.amx.com>.

The ilc2amx software components

ilc2amx is the glue between iLoveControl and AMX. The ilc2amx software components make it possible to use an iPhone or iPod Touch as a touch panel for AMX control systems. Because every AMX installation is unique the ilc2amx modules are highly configurable making it possible to meet even very special integration requirements. Furthermore ilc2amx supports different modes of operation. Depending on the complexity of the AMX installation and the structure of the existing AMX program the modules can be setup to operate in an appropriate way.

Main features of ilc2amx

ilc2amx offers the following key features:

- Fully configurable and thus highly flexible.
- Extremely easy to integrate into AMX installations.
- Includes out-of-the-Box functionality for a wide range of commonly used functions.
- Full support of all iLoveControl systems and features.
- Bidirectional state management for channels and levels.
- Multi device support: can handle multiple iPhones and/or iPod Touches at the same time.
- Offers different modes of operation for easy integration into different AMX environments.
- Fully compatible to the AMX SNAPI and G4API standards.
- Complete documentation with sample code for common use cases.

System requirements

System	Minimal requirement
AMX Controller	NetLinx Controller with Duet compatible firmware (like NI2XXX, NI3XXX, etc. with current firmware)
iLoveControl	Version 1.2
iPhone or iPod Touch	Firmware 2.X

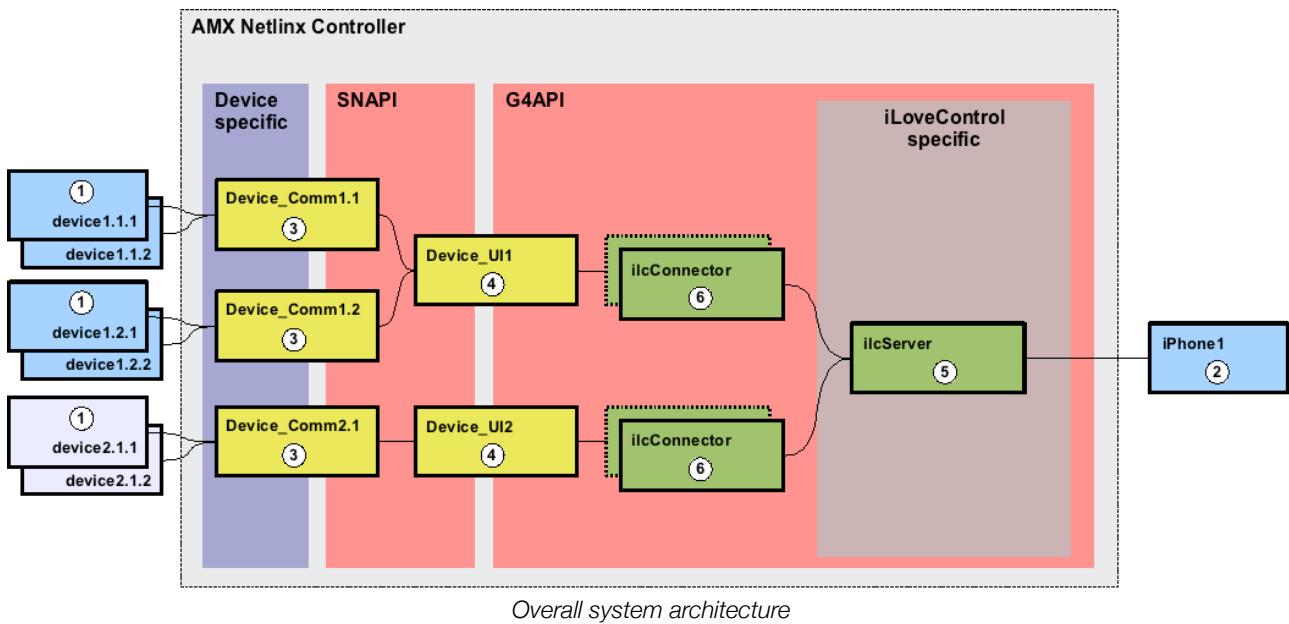
ilc2amx package content

File name	Type	Remarks
wc_ilcserver_dr1_0_0.jar	Duet module	Core module (binary)
wc_ilcconnector_dr1_0_0.jar	Duet module	Core module (binary)
wc_ilcconfig_dr1_0_0.axs	Netlinx module	Helper module (source)
wc_ilcconfig_1_0_0.axi	Netlinx include file	Global function definitions for helper module
wc_constants_1_0_0.axi	Netlinx include file	Global constants for helper module
ilcsample-mode1.axs	Netlinx program	Sample program for mode 1
ilcfuncmap-g4api.xml	Configuration file	Default G4API definitions
ilcfuncicsp-g4api.xml	Configuration file	Default G4API definitions
ilccustmap-lite.xml	Configuration file	Sample file for iLoveControl Lite application
ilcconfig-lite.xml	Configuration file	Sample file for iLoveControl Lite application
ilc2amx_documentation_1_0_0.pdf	Documentation	This documentation file

System architecture

Overview

The overall system architecture as shown in schematics 1 consists of different software components:



ID	Description
1	Physical devices which are to be controlled (i.e. TV, Light, etc.)
2	iPhone or iPod Touch to control the devices.
3	Typically a device is controlled using a device specific AMX module. It can be thought of as something like a device driver. The same module can be used for all existing devices of the same type (i.e. all JVC-HD100 video projectors can use the same communication module).
4	AMX recommends to separate user interface code from device specific code. In order to simplify this task, AMX published the so called SNAPI (Standard Netlinx API) and G4API standard. SNAPI and G4API standardize the mapping of commonly used functions to AMX channels and levels. Similar to the device communication module, a device user interface module is reusable for the same kind of device (i.e. all DVD players can share the UI module DVD_UI).

ID	Description
5	The ilcServer module is the iLoveControl specific part of the ilc2amx components. Its task is to translate between the iLoveControl protocol and the AMX specific communication protocol ICSP. It's also responsible for keeping channel and level states. For each ilc device (iPhone, iPod Touch) which is to be used in an installation one must implement an instance of this module. Every ilcServer module listens on a different port to data coming from ilc devices.
6	The ilcConnector modules are forwarding the (now AMX compatible) data from a specific ilcServer module to the ICSP bus of the AMX controller. It is the ilcConnector module that is addressed from within the AMX program. In fact the ilcConnector modules act like a regular AMX touch panel (TP) and can be used exactly like a TP. Of course the functionality of such a virtual TP may differ from a real TP. In case of iLoveControl the supported functionality can be seen in the Appendix. Depending on the number of devices and functions which are to be implemented the ilc2amx modules can be set up to operate in several ways. Some commonly used configuration scenarios are described in this paper (see Chapter „Operation modes“). The mode of operation determines how many instances of ilcConnector modules must be used.

Module documentation

General definitions

Module configuration files

As previously stated the ilc2amx components are highly configurable providing utmost flexibility for different project demands and installation sizes. Besides the two core modules (ilcServer ,ilcConnector), which are implemented in Java as Duet modules, there is also a helper module included which is available as Netlinx source code. The helper module eases the configuration steps for the core modules. Finally there is a sample program for each operation mode in the ilc2amx package which can be used as a reference for your own implementations.

Most of the configuration is kept in the following XML files:

Configuration command	Sample / default file	Description
ilcconfigfile	ilcconfig-lite.xml	Contains the configuration of your ilc device and can be downloaded from the iLoveControl website. The provided sample file corresponds to the iLoveControl Lite Application.
ilcfuncmapfile	ilcfuncmap-g4api.xml	Contains the mapping of ilc button names to virtual function names. This implements an abstraction layer between the ilc API and the AMX API. The default configuration contains a mapping of all ilc buttons to the corresponding G4API function names. It can usually be used without modification.
ilcfuncicspfile	ilcfuncicsp-g4api.xml	Defines the mapping of virtual function names to specific channel values. Again, the default configuration is fully G4API compatible and should usually not be modified.

Configuration command	Sample / default file	Description
ilccustmapfile	ilccustmap-sample1.xml	Allows to define mappings for customer specific functions to channel values. This can be used to map ilc scenes to channel numbers.

Parameters are passed to the module using module properties that are sent to the module using SEND_COMMAND statements as follows:

```
SEND_COMMAND virtualDeviceId, PROPERTY-configCommand[,configParamter1][,configParameterN]
```

Exception handling

The ilc2amx modules are designed to work as reasonable as possible out-of-the-Box. This is accomplished by using standardized commands like the rich set of predefined SNAPI/G4API commands. However, in some cases this is not possible - mostly because iLoveControl does not only support predefined actions but also customer defined scenes. Without telling the system what actions to take in case of such a customer scene, the module has a hard time to figure out a reasonable thing to do.

There are some exception handling rules defined in the module execution logic that care about „not defined“ actions. The following rules are implemented:

Exception situation	Description
No command found in ilcconfigfile or command is not mapped to any valid ICSP channel number.	<p>Sending button name and (if available) current room name as ICSP command (SEND_COMMAND) to the configured virtual device of the current system.</p> <p>Command definition: PUSH-<i>ilcbuttonname</i>[,<i>currentRoomName</i>]</p>
Activation of a scene and no mapping in ilccustmapfile found.	<p>Sending scene name as ICSP command (SEND_COMMAND) to the configured virtual device of the current system.</p> <p>Command definition: SCENE-<i>sceneName</i></p>
The module property ilcchanneloffset is set for the current system	<p>Sending current ICSP channel number (found via configuration files) increased by a room offset to the configured virtual device of the current system.</p> <p>The room offset is defined as: <i>roomIndex</i> * <i>offset</i></p> <p>The new channel number is defined as: <i>channel number</i> + <i>roomOffset</i> whereas offset has to be set in the module configuration process (ilcchanneloffset) and roomIndex is the zero based index of the „room“ setup parameter.</p> <p>Sample: <code>ilcchanneloffset = 50 ilcroom = { 'Kitchen','Family','Office' } base channel number: 9 (= toggle power)</code></p> <p>The channel number sent by ilc2amx is now recalculated for every room:</p> <ul style="list-style-type: none"> • Kitchen = 9 + (0 * 50) = 9 • Family = 9 + (1 * 50) = 59 • Office = 9 + (2 * 50) = 109

Module wc_ilcserver_dr1_0_0

Module configuration

The following ilc2amx parameters can be used to configure the module.

configCommand	configParameter	Description Sample
ilcport	<i>portNumber</i>	Port number of ilc device. PROPERTY-ilcport,49795
ilcchanneloffset	<i>offsetValue</i>	Offset for alternate channel calculation mode (see chapter „Exception handling“). PROPERTY-ilcchanneloffset,50
ilcconfigfile	<i>filename</i>	See table above. PROPERTY-ilcconfigfile,ilcconfig-lite.xml
ilcfuncmapfile	<i>filename</i>	See table above. PROPERTY-ilcfuncmapfile,ilcfuncmap-g4api.xml
ilcfuncicspfile	<i>filename</i>	See table above. PROPERTY-ilcfuncicspfile,ilcfuncicsp-g4api.xml
ilccustmapfile	<i>filename</i>	See table above. PROPERTY-ilccustmapfile,ilccustmap-sample1.xml

In addition to these ilc2amx specific parameters the following SNAPI commands are valid:

SNAPI command	Parameter	Description Sample
DEBUG	<i>debugLevel: 1 to 4</i>	Verbosity level of module: 1 = least information, 4 = most information DEBUG-1
REINIT	<i>NONE</i>	Initializes the module using the previously set properties. REINIT

Module wc_ilcconnector_dr1_0_0

Module configuration

The following ilc2amx parameters can be used to configure the module.

configCommand	configParameter	Description / Sample
ilcsystem	<i>ilc system name</i>	Name of ilc system to be controlled by this module instance. The ilc system name must match the definition in the ilc config file (see parameter ilcconfigfile. if the ilcroom parameter (see next line) is not set the ilcConnector instance will be connected to all existing rooms (based on the ilcconfigfile settings) of the respective ilc device. PROPERTY-ilcsystem,Lights
ilcroom	<i>ilc room names</i>	The ilcroom parameter is optional. If set it binds the module instance (in addition to the system mapping) also to one or several specific ilc room. To bind the module to more than one room (but not all) one can specify all desired room names separated by a comma. PROPERTY-ilcroom,Theater,Living

In addition to these ilc2amx specific parameters the following SNAPI commands are valid:

SNAPI command	Parameter	Description / Sample
DEBUG	<i>debug level 1 to 4</i>	Verbosity level of module: 1 = least information, 4 = most information DEBUG-1
REINIT	<i>NONE</i>	Initializes the module using the previously set properties. REINIT

Helper module wc_ilcconfig_dr1_0_0

Module configuration

The module wc_ilcconfig_dr1_0_0 simplifies the configuration process a great deal. The following parameters have to be passed to the module:

Parameter type	Description / Sample
(variable names are only samples, any valid identifier name is possible)	

DEV vdvlcServer

Virtual device of the ilcServer module. Since the ilcServer module is a Duet device, the virtual device Id must be within the range from 41000 up to 42000

Parameter type (variable names are only samples, any valid identifier name is possible)	Description / Sample
DEV ilcdevices [ILCMAXDEVICES]	<p>Array of DEV structures (= virtual device IDs) of all virtual devices used by the ilcConnector module. The order of the devices in this array must match the one of the system/room names (ilcConfig).</p> <p>Sample:</p> <pre>DEV dvILC_Lights = [41001:1:0] DEV dvILC_AudioOffice = [41010:1:0] DEV dvILC_DrapesFamily = [41020:1:0] DEV dvILC_DrapesOffice = [41021:1:0] DEV ilcdevices [ILCMAXDEVICES] = { dvILC_Lights , dvILC_AudioOffice , dvILC_DrapesFamily , dvILC_DrapesOffice }</pre>
CHAR ilcConfig [ILCMAXDEVICES] [ILCMAXROOMS] [ILCMAXNAMELEN]	<p>Three dimensional array containing (one) system name and (none, one or many) room names per virtual device used by the ilcConnector module. The order of the devices must match the ones of the DEV array (ilcdevices). The first string of each device defines the system name, all (optionally) following strings are treated as room names.</p> <p>Sample:</p> <pre>DEV ilcdevices [] = {...} // see sample definition above ilcConfig [ILCMAXDEVICES] [ILCMAXROOMS] [ILCMAXNAMELEN] = { { 'Lights' }, { 'Audio','Office' } , { 'Drapes','Family','Office' } }</pre> <p>This ilcConfig array defines the following mappings:</p> <pre>Ilc system „Lights“ = virtual device [41001:1:0] (for all rooms) Ilc system „Audio“ = virtual device [41010:1:0] for room „Office“ Ilc system „Drapes“ = virtual device [41020:1:0] for room Family and virtual device [41021:1:0] for room Office</pre>
INTEGER port	Port number of ilc device (see chapter „Module wc_ilcserver_dr1_0_0“)
CHAR configfile [ILCMAXFILENAMELEN]	Filename of configfile (see chapter „Module wc_ilcserver_dr1_0_0“)
CHAR funcmapfile [ILCMAXFILENAMELEN]	Filename of funcmapfile (see chapter „Module wc_ilcserver_dr1_0_0“)

Parameter type (variable names are only samples, any valid identifier name is possible)	Description / Sample
CHAR funcicspfile [ILCMAXFILENAMELEN]	Filename of funcicspfile (see chapter „Module wc_ilcserver_dr1_0_0“)
CHAR custmapfile [ILCMAXFILENAMELEN]	Filename of custmapfile (see chapter „Module wc_ilcserver_dr1_0_0“)

Required global constants

The following global variables must be declared and instantiated for the wc_ilcconfig_dr1_0_0 to work properly:

Name	Type	Description Sample
ILCMAXFILENAMELEN	INTEGER	Constant defining the maximal length of config file names.
ILCMAXNAMELEN	INTEGER	Constant defining the maximal length of system and room names.
ILCMAXROOMS	INTEGER	Constant defining the maximal number of rooms to configure.
ILCMAXDEVICES	INTEGER	Constant defining the maximal number of systems to configure.

Operation modes (work in progress)

The possibility to configure the ilc2amx modules in the previously described way together with the built in „exception handlers“ make it possible to use the modules in very different ways. Three typical configuration scenarios are described in more detail in this chapter. These commonly used configuration scenarios are referenced as „operation modes“.

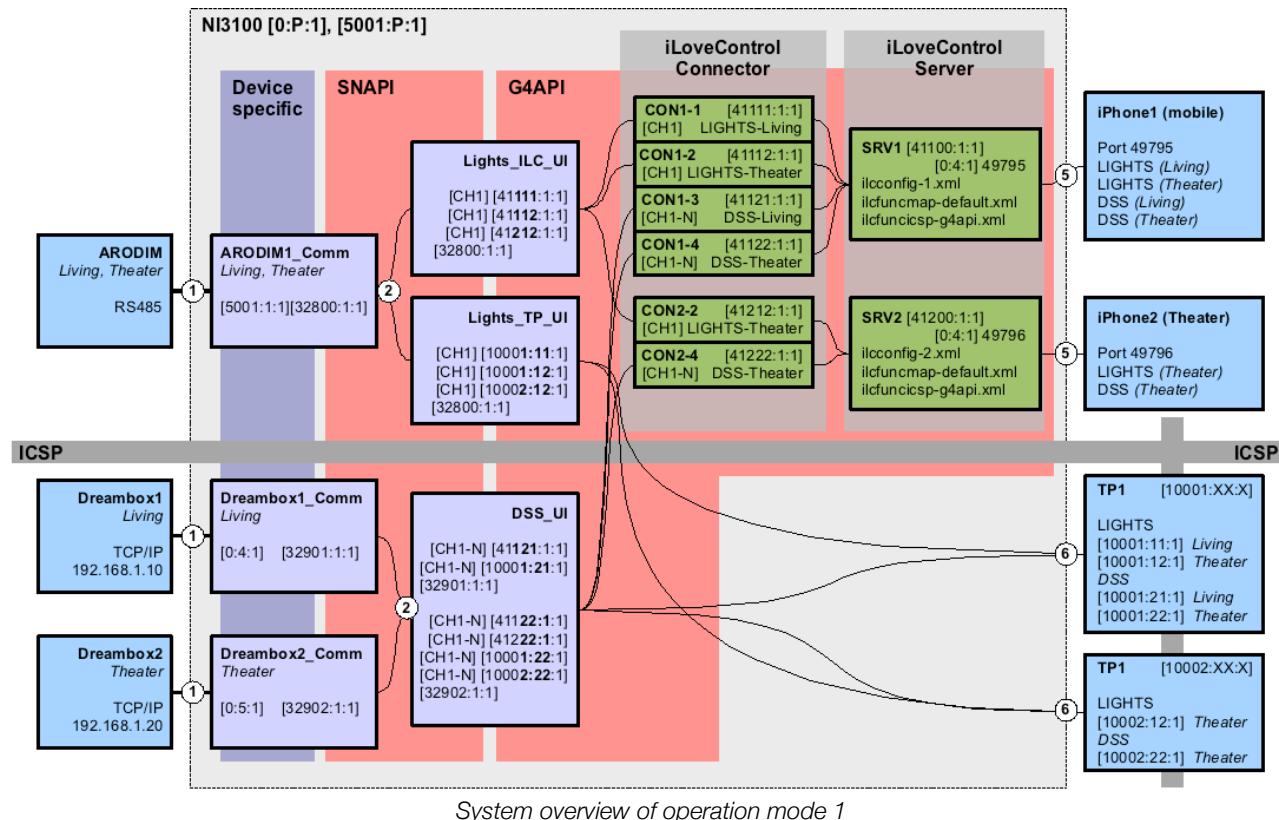
Operation mode 1

Remark: this chapter is not finished yet. To do: Finalize description.

Operation mode 1 is ideally suited to integrate one or more ilc devices in an existing AMX installation. Especially if the existing implementation follows the SNAPI standard and uses separated Comm and UI modules this mode should be chosen.

In operation mode 1 there must be created one ilcConnector module for each combination of system and room so that there is a direct connection between an AMX virtual device ID and a system/room combination of the ilc configuration.

The following schematics depicts a sample system overview when running the module in operation mode 1.



Operation mode 2

Remark: this chapter is not finished yet. To do: Finalize description and adding schematics

- Systematics: Alternate channel calculation mode with channel offset parameter.
- Purpose: reducing the number of virtual devices for ilcConnector instances.
- When to use: big installations, where memory consumption of mode 1 might be a problem or too many virtual devices are getting confusing.

Operation mode 3

Remark: this chapter is not finished yet. To do: Finalize description and adding schematics

- Systematics: Omitting channel mappings completely and therefor forcing module to use SEND_COMMANDS.
- Purpose: max. reducing the number of virtual devices for ilcConnector instances. Forcing command handling instead of channel handling
- When to use: similar to mode 2 or if command handling is for some reason preferred to channel handling.

Appendix

ilcfuncmap: default G4API definition

key	string	key	string	key	string
functype	channel	ilcfunc	On	g4func	PWR_ON
functype	channel	ilcfunc	Off	g4func	PWR_OFF
functype	channel	ilcfunc	CH+	g4func	BTN_CHAN_UP
functype	channel	ilcfunc	CH-	g4func	BTN_CHAN_DN
functype	channel	ilcfunc	Vol+	g4func	BTN_VOL_UP
functype	channel	ilcfunc	Vol-	g4func	BTN_VOL_DN
functype	channel	ilcfunc	Mute	g4func	BTN_VOL_MUTE
functype	channel	ilcfunc	Aspect	g4func	BTN_ASPECT_RATIO
functype	channel	ilcfunc	LightsTgl	g4func	BTN_KEYPAD_BUTTON_1
functype	channel	ilcfunc	DrapesTgl	g4func	BTN_KEYPAD_BUTTON_1
functype	channel	ilcfunc	Raise	g4func	BTN_HVAC_HEAT_UP
functype	channel	ilcfunc	Lower	g4func	BTN_HVAC_HEAT_DN
functype	channel	ilcfunc	1	g4func	BTN_KEYPAD_BUTTON_1
functype	channel	ilcfunc	2	g4func	BTN_KEYPAD_BUTTON_2
functype	channel	ilcfunc	3	g4func	BTN_KEYPAD_BUTTON_3
functype	channel	ilcfunc	4	g4func	BTN_KEYPAD_BUTTON_4
functype	channel	ilcfunc	5	g4func	BTN_KEYPAD_BUTTON_5
functype	channel	ilcfunc	6	g4func	BTN_KEYPAD_BUTTON_6
functype	channel	ilcfunc	7	g4func	BTN_KEYPAD_BUTTON_7
functype	channel	ilcfunc	8	g4func	BTN_KEYPAD_BUTTON_8
functype	channel	ilcfunc	9	g4func	BTN_KEYPAD_BUTTON_9
functype	channel	ilcfunc	0	g4func	BTN_KEYPAD_BUTTON_0
functype	channel	ilcfunc	Clear	g4func	BTN_MENU_CLEAR
functype	channel	ilcfunc	Enter	g4func	BTN_MENU_SELECT
functype	channel	ilcfunc	Play	g4func	BTN_PLAY
functype	channel	ilcfunc	Pause	g4func	BTN_PAUSE

key	string	key	string	key	string
functype	channel	ilcfunc	Stop	g4func	BTN_STOP
functype	channel	ilcfunc	Rew	g4func	BTN_REW
functype	channel	ilcfunc	Fwd	g4func	BTN_FFWD
functype	channel	ilcfunc	Prev	g4func	BTN_SREV
functype	channel	ilcfunc	Next	g4func	BTN_SFWD
functype	channel	ilcfunc	Replay	g4func	BTN_MENU_INSTANT_REPLY
functype	channel	ilcfunc	Rec	g4func	BTN_RECORD
functype	channel	ilcfunc	Live	g4func	BTN_MENU_LIVE_TV
functype	channel	ilcfunc	Fav	g4func	BTN_MENU_FAVORITES
functype	channel	ilcfunc	List	g4func	BTN_MENU_LIST
functype	channel	ilcfunc	Disc-	g4func	BTN_DISC_PREV
functype	channel	ilcfunc	Disc+	g4func	BTN_DISC_NEXT
functype	channel	ilcfunc	Menu	g4func	BTN_MENU_FUNC
functype	channel	ilcfunc	Audio	g4func	BTN_MENU_AUDIO
functype	channel	ilcfunc	Subt	g4func	BTN_MENU_SUBTITLE
functype	channel	ilcfunc	Info	g4func	BTN_MENU_INFO
functype	channel	ilcfunc	Return	g4func	BTN_MENU_RETURN
functype	channel	ilcfunc	Up	g4func	BTN_MENU_UP
functype	channel	ilcfunc	Dwn	g4func	BTN_MENU_DN
functype	channel	ilcfunc	Left	g4func	BTN_MENU_LT
functype	channel	ilcfunc	Right	g4func	BTN_MENU_RT
functype	channel	ilcfunc	Select	g4func	BTN_MENU_SELECT
functype	channel	ilcfunc	Red	g4func	RED
functype	channel	ilcfunc	Green	g4func	GREEN
functype	channel	ilcfunc	Yellow	g4func	YELLOW
functype	channel	ilcfunc	Blue	g4func	BLUE
functype	channel	ilcfunc	A	g4func	BTN_KEYPAD_BUTTON_A
functype	channel	ilcfunc	B	g4func	BTN_KEYPAD_BUTTON_B
functype	channel	ilcfunc	C	g4func	BTN_KEYPAD_BUTTON_C
functype	channel	ilcfunc	Pg+	g4func	BTN_MENU_PAGE_UP
functype	channel	ilcfunc	Pg-	g4func	BTN_MENU_PAGE_DN
functype	channel	ilcfunc	Day+	g4func	BTN_MENU_PAGE_UP
functype	channel	ilcfunc	Day-	g4func	BTN_MENU_PAGE_DN

key	string	key	string	key	string
functype	channel	ilcfunc	Pwr	g4func	BTN_POWER
functype	channel	ilcfunc	Exit	g4func	BTN_MENU_EXIT
functype	channel	ilcfunc	Last	g4func	BTN_TUNER_PREV
functype	channel	ilcfunc	Guide	g4func	BTN_MENU_GUIDE
functype	channel	ilcfunc	Repeat	g4func	BTN_MEDIA_REPEAT
functype	channel	ilcfunc	Shuffle	g4func	BTN_MEDIA_RANDOM
functype	channel	ilcfunc	Preset 1	g4func	BTN_TUNER_PRESET_1
functype	channel	ilcfunc	Preset 2	g4func	BTN_TUNER_PRESET_2
functype	channel	ilcfunc	Preset 3	g4func	BTN_TUNER_PRESET_3
functype	channel	ilcfunc	Preset 4	g4func	BTN_TUNER_PRESET_4
functype	channel	ilcfunc	Preset 5	g4func	BTN_TUNER_PRESET_5
functype	channel	ilcfunc	Preset 6	g4func	BTN_TUNER_PRESET_6
functype	channel	ilcfunc	Preset 7	g4func	BTN_TUNER_PRESET_7
functype	channel	ilcfunc	Preset 8	g4func	BTN_TUNER_PRESET_8
functype	channel	ilcfunc	Preset 9	g4func	BTN_TUNER_PRESET_9
functype	channel	ilcfunc	Cat	g4func	BTN_TUNER_BAND
functype	channel	ilcfunc	Band	g4func	BTN_TUNER_BAND
functype	channel	ilcfunc	Movies	g4func	BTN_MEDIADB_RETURN_DISC
functype	channel	ilcfunc	Music	g4func	BTN_MEDIADB_RETURN_AUDIO
functype	channel	ilcfunc	Covers	g4func	BTN_MEDIADB_RETURN_PLAYLIST
functype	channel	ilcfunc	Collec	g4func	BTN_MEDIADB_RETURN_PLAYLIST

ilcfuncicsp: default G4API definition

key	string	key	string	key	string
functype	channel	g4func	BTN_PLAY	icspfunc	1
functype	channel	g4func	BTN_STOP	icspfunc	2
functype	channel	g4func	BTN_PAUSE	icspfunc	3
functype	channel	g4func	BTN_FFWD	icspfunc	4
functype	channel	g4func	BTN_REW	icspfunc	5
functype	channel	g4func	BTN_SFWD	icspfunc	6
functype	channel	g4func	BTN_SREV	icspfunc	7
functype	channel	g4func	BTN_RECORD	icspfunc	8
functype	channel	g4func	BTN_POWER	icspfunc	9
functype	channel	g4func	BTN_DIGIT_0	icspfunc	10

key	string	key	string	key	string
functype	channel	g4func	BTN_DIGIT_1	icspfunc	11
functype	channel	g4func	BTN_DIGIT_2	icspfunc	12
functype	channel	g4func	BTN_DIGIT_3	icspfunc	13
functype	channel	g4func	BTN_DIGIT_4	icspfunc	14
functype	channel	g4func	BTN_DIGIT_5	icspfunc	15
functype	channel	g4func	BTN_DIGIT_6	icspfunc	16
functype	channel	g4func	BTN_DIGIT_7	icspfunc	17
functype	channel	g4func	BTN_DIGIT_8	icspfunc	18
functype	channel	g4func	BTN_DIGIT_9	icspfunc	19
functype	channel	g4func	BTN_MENU_PLUS_10	icspfunc	20
functype	channel	g4func	BTN_MENU_ENTER	icspfunc	21
functype	channel	g4func	BTN_CHAN_UP	icspfunc	22
functype	channel	g4func	BTN_CHAN_DN	icspfunc	23
functype	channel	g4func	BTN_VOL_UP	icspfunc	24
functype	channel	g4func	BTN_VOL_DN	icspfunc	25
functype	channel	g4func	BTN_VOL_MUTE	icspfunc	26
functype	channel	g4func	PWR_ON	icspfunc	27
functype	channel	g4func	PWR_OFF	icspfunc	28
functype	channel	g4func	BTN_SOURCE_VIDEO1	icspfunc	31
functype	channel	g4func	BTN_SOURCE_VIDEO2	icspfunc	32
functype	channel	g4func	BTN_SOURCE_VIDEO3	icspfunc	33
functype	channel	g4func	BTN_SOURCE_AUX1	icspfunc	39
functype	channel	g4func	BTN_TUNER_BAND	icspfunc	40
functype	channel	g4func	BTN_MENU_CANCEL	icspfunc	43
functype	channel	g4func	BTN_MENU_FUNC	icspfunc	44
functype	channel	g4func	BTN_MENU_UP	icspfunc	45
functype	channel	g4func	BTN_MENU_DN	icspfunc	46
functype	channel	g4func	BTN_MENU_LT	icspfunc	47
functype	channel	g4func	BTN_MENU_RT	icspfunc	48
functype	channel	g4func	BTN_MENU_SELECT	icspfunc	49
functype	channel	g4func	BTN_MENU_EXIT	icspfunc	50
functype	channel	g4func	BTN_DISC_NEXT	icspfunc	55
functype	channel	g4func	BTN_DISC_PREV	icspfunc	56

key	string	key	string	key	string
functype	channel	g4func	BTN_MENU_THUMBS_DN	icspfunc	58
functype	channel	g4func	BTN_MENU_THUMBS_UP	icspfunc	59
functype	channel	g4func	BTN_MENU_LIVE_TV	icspfunc	62
functype	channel	g4func	BTN_MENU_CLEAR	icspfunc	80
functype	channel	g4func	BTN_MENU_ADVANCE	icspfunc	83
functype	channel	g4func	BTN_MENU_LIST	icspfunc	86
functype	channel	g4func	BTN_MENU_PLUS_100	icspfunc	97
functype	channel	g4func	BTN_MENU_DISPLAY	icspfunc	99
functype	channel	g4func	BTN_MENU_SUBTITLE	icspfunc	100
functype	channel	g4func	BTN_MENU_INFO	icspfunc	101
functype	channel	g4func	BTN_MENU_FAVORITES	icspfunc	102
functype	channel	g4func	BTN_MENU_CONTINUE	icspfunc	103
functype	channel	g4func	BTN_MENU_RETURN	icspfunc	104
functype	channel	g4func	BTN_MENU_GUIDE	icspfunc	105
functype	channel	g4func	BTN_MENU_PAGE_UP	icspfunc	106
functype	channel	g4func	BTN_MENU_PAGE_DN	icspfunc	107
functype	channel	g4func	BTN_MENU_AB_REPEAT	icspfunc	112
functype	channel	g4func	BTN_MENU_HELP	icspfunc	113
functype	channel	g4func	BTN_MENU_TITLE	icspfunc	114
functype	channel	g4func	BTN_MENU_TOP_MENU	icspfunc	115
functype	channel	g4func	BTN_MENU_ZOOM	icspfunc	116
functype	channel	g4func	BTN_MENU_ANGLE	icspfunc	117
functype	channel	g4func	BTN_MENU_AUDIO	icspfunc	118
functype	channel	g4func	BTN_DISC_TRAY	icspfunc	120
functype	channel	g4func	BTN_DISC_RANDOM	icspfunc	124
functype	channel	g4func	BTN_DISC_REPEAT	icspfunc	125
functype	channel	g4func	BTN_LIGHT_ELEMENT_1	icspfunc	261
functype	channel	g4func	BTN_KEYPAD_BUTTON_1	icspfunc	1
functype	channel	g4func	BTN_KEYPAD_BUTTON_2	icspfunc	2
functype	channel	g4func	BTN_KEYPAD_BUTTON_3	icspfunc	3
functype	channel	g4func	BTN_KEYPAD_BUTTON_4	icspfunc	4
functype	channel	g4func	BTN_KEYPAD_BUTTON_5	icspfunc	5
functype	channel	g4func	BTN_KEYPAD_BUTTON_6	icspfunc	6

key	string	key	string	key	string
functype	channel	g4func	BTN_KEYPAD_BUTTON_7	icspfunc	7
functype	channel	g4func	BTN_KEYPAD_BUTTON_8	icspfunc	8
functype	channel	g4func	BTN_KEYPAD_BUTTON_9	icspfunc	9
functype	channel	g4func	BTN_KEYPAD_BUTTON_0	icspfunc	10
functype	channel	g4func	BTN_KEYPAD_BUTTON_A	icspfunc	11
functype	channel	g4func	BTN_KEYPAD_BUTTON_B	icspfunc	12
functype	channel	g4func	BTN_KEYPAD_BUTTON_C	icspfunc	13
functype	channel	g4func	BTN_MEDIA_RANDOM	icspfunc	124
functype	channel	g4func	BTN_MEDIA_REPEAT	icspfunc	125
functype	channel	g4func	BTN_MENU_INSTANT_REPLAY	icspfunc	218
functype	channel	g4func	BTN_TUNER_PREV	icspfunc	235
functype	channel	g4func	BTN_TUNER_PRESET_1	icspfunc	261
functype	channel	g4func	BTN_TUNER_PRESET_2	icspfunc	262
functype	channel	g4func	BTN_TUNER_PRESET_3	icspfunc	263
functype	channel	g4func	BTN_TUNER_PRESET_4	icspfunc	264
functype	channel	g4func	BTN_TUNER_PRESET_5	icspfunc	265
functype	channel	g4func	BTN_TUNER_PRESET_6	icspfunc	266
functype	channel	g4func	BTN_TUNER_PRESET_7	icspfunc	267
functype	channel	g4func	BTN_TUNER_PRESET_8	icspfunc	268
functype	channel	g4func	BTN_TUNER_PRESET_9	icspfunc	269
functype	channel	g4func	RED	icspfunc	1
functype	channel	g4func	GREEN	icspfunc	2
functype	channel	g4func	YELLOW	icspfunc	3
functype	channel	g4func	BLUE	icspfunc	4
functype	channel	g4func	BTN_MEDIADB_RETURN_PLAYLIST	icspfunc	426
functype	channel	g4func	BTN_MEDIADB_RETURN_DISC	icspfunc	428
functype	channel	g4func	BTN_MEDIADB_RETURN_AUDIO	icspfunc	429