SPACE PHILIPS

Software Document

jointSPACE API Reference Manual

Version 2.2

February 8, 2011

Status: Proposal

Document reference: AGT034-2010-197

Table of Contents

Abstract

The scope of this document is to explain the jointSPACE API. This API includes device discovery, key injection and graphical content push.

Because jointSPACE is based on DirectFB technology, part of the API described below is related to DirectFB.

More information on how to enable jointSPACE and which TV ranges are supporting it can be found on the jointSPACE web site.

Device initialisation

To be able to communicate with other jointSPACE devices, DirectFB needs to be initialised.

Therefore, the following call should be the first call made by the application:

- Call **once** the function **`DirectFBInit()`**

Device discovery

To be able to discover and identify jointSPACE devices on the network, the application should make use of the **`IVoodooPlayer`** interface to check for device topology changes.

- Create **once** the player, using the function **`voodoo_player_create()`**
- Once the player is created, the application should check at regular times (e.g. every 2 seconds) if new TV's have been added to the network or existing TV's have been removed from the network. This can be done using the functions **`voodoo_player_broadcast()`** and **`voodoo_player_enumerate()`**.

Example code for *device discovery*:

- In the main routine, call **`DirectFBInit()`** and create a thread:

```
/* DirectFB init */
DFBCHECK( DirectFBInit( (void*)&argc, (void*)&argv ) );
```

```
pthread_mutex_init(&wait_mutex, NULL);
pthread_create( &threadId, NULL, &DiscoveryTask, NULL );
```

Remark:
DFBCHECK() is a macro that allows you to check the return value of every DFB call.
The parameter **DiscoveryTask** is the thread function that will be called, once the task is created.

- In the thread function **DiscoveryTask()**, create a player and start a **while** loop that scans every 2 seconds for a change in the topology of the network:

```
static void* DiscoveryTask( void* param )
{
    .
    .
    .

    direct_snputs( info.name, "myplayer", VOODOO_PLAYER_NAME_LENGTH );
    DFBCHECK( voodoo_player_create( &info, &player ) );
    while ( true )
    {
            .
            .
            .
            DFBCHECK(voodoo_player_broadcast( player ));
            usleep( 100000 );
            DFBCHECK(voodoo_player_enumerate( player, player_callback, NULL ));
            .
            .
            wait_2_seconds();
    }

    return( NULL );
}
```

In the above code, **info** is of type **VoodooPlayInfo** (see **play.h**). A name is passed to the player at creation time and then a **while** loop is started that will:

- broadcast the player itself to other devices of the network: the application also announces itself on the network using the function **voodoo_player_broadcast()**
- enumerate the other devices discovered on the network: a call-back function, **player_callback**, is passed as one of the parameters to the function **voodoo_player_enumerate()**.

The call-back function **player_callback()** is called in a synchronous way for *each and every device* found on the network

- The call-back function receives the following important information from the player(s):
    - player information (through parameter **info** of type **VoodooPlayInfo**), that contains the following sub-information:
        - UUID of the set (16 bytes, 32 HEX characters): serial number of the TV. This parameter identifies uniquely a device on the network.
        - Name of the player (e.g.: “PhilipsTV” for Philips TV’s)
        - Model of the player (e.g.: “2k9” or “2k10”, depending of the type of set used). The model name will be extended with the complete set type information in the future.
        - Pls. refer to the section TV diversity for the exact naming of the parameters.
    - address information, which is the IP address of the set, represented as a string according the following template: “**xxx.xxx.xxx.xxx**”

- The application should keep administration of the incoming data and maintain the list of sets that are added or removed from the network.

Maintaining a list of active players is the responsibility of the application (e.g. devices not discovered after x retries should be removed from the list, linking IP address with UUID...).

To accomplish this, the user can create an array of structures of the following type:

```
typedef struct PlayerProperties
{
    unsigned char player_uuid[ 16 ];
    char          player_name[ VOODOO_PLAYER_NAME_LENGTH ];
    char          player_model[ VOODOO_PLAYER_MODEL_LENGTH ];
    char          player_address[ 16 ];
};
```

The values for **VOODOO_...** can be found in **play.h**.

Device selection

If an application wants to switch the control from device A to device B, the function **DirectFBSetOption()** must be called.

Example code for *key injection*:

To switch the key injection from one TV to another one, the following sequence should be used:

```
jslibrc_Exit();
DirectFBSetOption( "remote", <IP_address> );
jslibrc_Init( NULL, NULL );
```

The IP address can be found in the data that has been collected in the call-back function (the IP address is directly related to the UUID of the device in the above mentioned **PlayerProperties** structure).

When calling **DirectFBSetOption()**, the application must first take care that the previous connection is closed. Then a new connection should be started.

Example code to destroy/create a *Remote Control connection*:

```
jslibrc_Exit(); //Close a Remote Control connection
jslibrc_Init(); //Restart a new Remote Control connection
```

Once **jslibrc_Init()** is called, the system will connect to the new IP address that was set with the function **DirectFBSetOption()**.

TV Diversity

	2k9	2k10	2k11
player_name	"PhilipsTV"	"PhilipsTV"	"PhilipsTV"
player_model	"2k9"	"2k10"	"2k11"

- If the detected "**player_name**" and the detected "**player_model**" are both "**unknown**", then "**2k9**" is assumed.

API description

Discovery

Functions from directfb.h (#include "directfb.h")

Library to link in is **libdirectfb.so**.

```
DFBResult DirectFBInit( int  *argc      /* pointer to main()'s argc */
                      , char *(*argv[]) /* pointer to main()'s argv */
                      );
```

- **argc**: pointer to **main()**'s **argc**
- **argv**: pointer to **main()**'s **argv**

This function **must** be called first. It initialises the DFB stack.

`DFBResult DirectFBSetOption( const char *name, const char *value )`

- **`name`** : must be **`"remote"`**
- **`value`**: the corresponding IP address for the targeted set, according the template "**`xxx.xxx.xxx.xxx`**"

This function allows the application to change "runtime" DirectFB arguments.

Example: change from one remote device to another.

Functions from `ivoodooplayer.h` (`#include <voodoo/ivoodooplayer.h>`)

Library to link in is **`libdirectfb.so`**.

`DirectResult VoodooPlayerCreate( IVoodooPlayer **ret_interface )`

- **`ret_interface`**: interface returned by the call.

The interface returned by the function **`VoodooPlayerCreate`** allows you to access all functions of a player, as defined in the corresponding header file **`ivoodooplayer.h`**.

Key injection

Functions from `jslibrc_client.h` (`#include <jslibrc_client.h>`)

Library to link in is **`libjslibclient.so`**.

`int jslibrc_Init( int *argc, char **argv[] )`

- **`argc`**: pointer to **`main()`**'s **`argc`**
- **`argv`**: pointer to **`main()`**'s **`argv`**

This function should be called with parameters **`( NULL, NULL )`**.
This function will set up a Remote Control connection to the first device found on the network, unless the IP address of a device was previously defined using **`DirectFBSetOption()`**. In that case, it will connect to the device with the IP address given in the call to **`DirectFBSetOption()`**.

`void jslibrc_Exit( void )`

This function should be called:

- to close a Remote Control connection
- when exiting the application
- before a new TV is to be selected from the TV network.

`void jslibrc_KeyDown( int src, int sys, int cmd )`

- **`src`**: can be **`keySourceRc5`** (RC5 mode) or **`keySourceRc6`** (RC6 mode)
- **`sys`**: should always be 0 for TV mode or 3 for External sources (see RC command mapping table)
- **`cmd`**: the possible RC commands are defined in the file **`jslibrc_types.h`**.
  Note:
  Not all commands can be used. See RC command mapping table for more details.

This function can be used to inject keys to a remote device. It should be called at every key press, if relevant.

`void jslibrc_KeyUp( int src, int sys, int cmd )`

- **`src`**: can be **`keySourceRc5`** (RC5 mode) or **`keySourceRc6`** (RC6 mode)
- **`sys`**: should always be 0 for TV mode or 3 for External sources (see RC command mapping table)
- **`cmd`**: the possible RC commands are defined in the file **`jslibrc_types.h`**.
  Note:
  Not all commands can be used. See RC command mapping table for more details.

This function can be used to inject keys to a remote device. It should be called at every key release, if relevant.

```
int  jslibrc_RequestActivity( amLib_EnumActivityId act
                            , amLib_EnumActivation mode
                            , int cookie
                            )
```

This function is obsolete. It should not be used.

Graphical content push

If an application wants to create graphics on the TV, the DirectFB graphical interface has to be used.

This is done in a few steps:

- Create a DirectFB handle (**`DirectFBCreate()`**)
  - This initiates a graphical connection to the TV
- Obtain the graphical layer handle (**`GetDisplayLayer()`**)
- Create a graphical window (**`CreateWindow()`**) and add it to the window layout (**`SetOpacity()`** and **`RequestFocus()`**)
  Note: ***the maximum resolution of windows is 1280x720x16bits!***
- Obtain the graphical surface (**`GetSurface()`**)
- Draw in the graphical surface (**`Clear()`**, **`Write()`**)
- Commit the changes on screen (**`Flip()`**)
- Release the DirectFB handle when the picture has to be removed (**`Release()`**)
  - This ends the graphical connection to the TV

Example code:

```
#include <directfb.h>
#define DFBCHECK(x...)                                  \
do {                                                    \
   DFBResult err;                                       \
   err = x;                                             \
   if (err != DFB_OK) {                                 \
     printf ("Fail!! err!=DFB_OK");                     \
     DirectFBError (#x, err);                           \
   }                                                    \
} while(0);

// To show content on TV
static IDirectFB               *dfb;
static IDirectFBDisplayLayer   *layer;
static DFBWindowDescription    wdesc;
static IDirectFBWindow         *gwindow;
static IDirectFBSurface        *gsurface;
static unsigned short          PixelBuffer[ 1280 * 720 ];
static DFBRectangle            rect;

/* DirectFB init */
DFBCHECK(DirectFBInit( (void*)&argc, (void*)&argv ));
.
.
.
DFBCHECK(DirectFBCreate( &dfb ));
/* Obtain the layer */
DFBCHECK(dfb->GetDisplayLayer(dfb, DLID_PRIMARY, &layer));
/* Setup the Graphical window */
wdesc.flags  = ( DWDESC_WIDTH
               | DWDESC_HEIGHT
               | DWDESC_OPTIONS
               | DWDESC_PIXELFORMAT
               | DWDESC_STACKING
               );
wdesc.width       = 1280;
wdesc.height      = 720;
wdesc.pixelformat = DSPF_RGB16; // DSPF_ARGB4444 or DSPF_ARGB
wdesc.stacking    = DWSC_MIDDLE;
wdesc.options     = DWOP_NONE;
```

```
DFBCHECK(layer->CreateWindow(layer, &wdesc, &gwindow));
DFBCHECK(gwindow->GetSurface(gwindow, &gsurface));
DFBCHECK(gsurface->Clear(gsurface, 0x00,0x00,0x00, 0x00)); //R,G,B,A
DFBCHECK(gwindow->SetOpacity(gwindow, 0xff ));
DFBCHECK(gwindow->RequestFocus( gwindow ));
DFBCHECK(gsurface->Flip( gsurface, NULL, DSFLIP_NONE ));

// write local pixel buffer (e,g, JPEG decoded data)
rect.x = 0;
rect.y = 0;
rect.w = 1280;
rect.h = 720;
DFBCHECK(gsurface->Write( gsurface
                        , &rect
                        , (void *)PixelBuffer /* image buffer pointer */
                        , ( 1280 * 2 )        /* stride */
                        )
         );
DFBCHECK(gsurface->Flip( gsurface, NULL, DSFLIP_NONE ));

// to remove GFX content from TV
if (dfb)
{
    DFBCHECK(dfb->Release( dfb ) );
}
```

Remote Control command mapping according to TV diversity

Key Events	RC codes (src - sys,cmd)	RC #define's for cmd	2k9	2k10	2k11
Standby	RC6 - 000,012	rc6S0Standby			
Previous channel / P<P / Back	RC6 - 000,010	rc6S0PreviousProgram	"Back"	"Back"	"Back"
Browse / Guide / Find	RC6 - 000,204	rc6S0EpgGuide	"Guide"	"Browse"	"Find"
Red colour	RC6 - 000,109	rc6S0Red			
Green colour	RC6 - 000,110	rc6S0Green			
Yellow colour	RC6 - 000,111	rc6S0Yellow			
Blue colour	RC6 - 000,112	rc6S0Cyan			
Home	RC6 - 000,084	rc6S0MenuOn			
Volume up / V+	RC6 - 000,016	rc6S0VolumeUp			
Volume down / V-	RC6 - 000,017	rc6S0VolumeDown			
Mute	RC6 - 000,013	rc6S0MuteDemute			
Options / Adjust	RC6 - 000,064	rc6S0ContextualOptions	"Options"	"Options"	"Options"
Dot	RC6 - 000,217	rc6S0Dot			
Digit [0..9]	RC6 - 000, 000 … 009	rc6S0Digit0 … rc6S0Digit9			
Info / i+	RC6 - 000,015	rc6S0Display			
Cursor up	RC6 - 000,088	rc6S0StepUp			
Cursor down	RC6 - 000,089	rc6S0StepDown			
Cursor left	RC6 - 000,090	rc6S0StepLeft			
Cursor right	RC6 - 000,091	rc6S0StepRight			
Confirm / OK	RC6 - 000,092	rc6S0Acknowledge			
Next	RC6 - 000,076				
Previous	RC6 - 000,077				
Ambilight mode / Experience	RC6 - 000,144	rc6S0AmbLightMode	"Ambilight Mode"	"Experience"	"Adjust"
Tuner A / Watch TV / Watch Last Preset / Exit	RC6 - 000,159		"Watch TV"	"Watch TV"	"Watch TV"
Viewmode / Picture format	RC6 - 000,245	rc6S0MovieExpand			

Teletext	**RC6 - 000,060**	**rc6S0TxtSubmode**	"Source"	"Source"	"Source"
Subtitle	**RC6 - 000,075**	**rc6S0TvTextSubtitle**			
ChannelStepUp / P+	**RC6 - 000,032**	**rc6S0Next**			
ChannelStepDown / P-	**RC6 - 000,033**	**rc6S0Previous**			
Next source / Source	**RC6 - 000,056**	**rc6S0External1**			
Ambilight on/off	**RC6 - 000,143**	**rc6S0AmbLightOnOffDim**			
Play/Pause	**RC6 - 000,044**	**rc6S0Play**			
Pause	**RC6 - 000,048**				
Fast forward / ffw	**RC6 - 000,040**	**rc6S0FastForward**			
Stop	**RC6 - 000,049**	**rc6S0Stop**			
Rewind / rew	**RC6 - 000,043**	**rc6S0ScanReverse**			
Record	**RC6 - 000,055**	**rc6S0Record**			
Online / Net TV	**RC6 - 000,190**	**rc6S0DisplayBrowser**			
Side/Front	**RC5 - 003,005**				
Ext1	**RC5 - 003,004**				
Ext2	**RC5 - 000,057**				
Ext3	**RC5 - 003,056**				
Ext4	**RC5 - 003,057**				
Ext5	**RC5 - 003,123**				
Ext6	**RC5 - 003,006**				
Ext7	**RC5 - 003,007**				
Ext8	**RC5 - 003,008**				
USB mode (USB content browser)	**RC5 - 003,019**				
Mediaserver select (DLNA content browser)	**RC5 - 003,096**				

The pictures below illustrate how the RC key mapping is used for **2k9** TV's. The detailed key mapping per TV model that is discovered, is described in the section Key mapping table.

The picture on the left is the original remote control; the picture on the right represents the same remote control with reference numbers, going from 1 to 41.

The Key mapping table indicates which RC command from the file **`jslibrc_types.h`** corresponds with a specific button on the remote control.

Key mapping table

Remote Key (reference number)	RC6 code (**cmd** parameter)
1	**rc6S0Standby**
2	**rc6S0Red**
3	**rc6S0Green**
4	**rc6S0Yellow**
5	**rc6S0Cyan**
6	**rc6S0MenuOn**
7	**rc6S0EpgGuide**
8	**rc6S0ContextualOptions**
9	**rc6S0StepUp**
10	**rc6S0StepLeft**
11	**rc6S0Acknowledge**
12	**rc6S0StepRight**
13	**rc6S0StepDown**
14	**rc6S0PreviousProgram**
15	**rc6S0Display**
16	**rc6S0ScanReverse**
17	**rc6S0Play**
18	**rc6S0FastForward**
19	**rc6S0Stop**
20	**rc6S0Record**
21	**rc6S0VolumeUp**
22	**rc6S0VolumeDown**
23	**rc6S0MuteDemute**
24	**rc6S0MovieExpand**
25	**rc6S0Next**
26	**rc6S0Previous**
27	**rc6S0DisplayBrowser**
28	**rc6S0TxtSubmode**
29	**rc6S0Digit1**
30	**rc6S0Digit2**
31	**rc6S0Digit3**
32	**rc6S0Digit4**
33	**rc6S0Digit5**
34	**rc6S0Digit6**
35	**rc6S0Digit7**
36	**rc6S0Digit8**
37	**rc6S0Digit9**
38	**rc6S0TvTextSubtitle**
39	**rc6S0Digit0**

40	**rc6S0External1**
41	**rc6S0AmbLightOnOffDim**

Multitap entry tables

The tables below describe the multitap key sequence to be sent for each possible character in each possible language, supported by the TV.

A multitap key sequence is just a successive number of the same digit key presses to reach the wanted character.

- In between two keys, a delay of **10ms** is required.
- At the end of a sequence, there's no need for an additional delay, but following keys should be sent:
  - 2k9: **(keySourceRc6, 0, rc6S0StepUp)**
  - 2k10/: **(keySourceRc6, 0, rc6S0Acknowledge)**

Example: to send the "**!**" character for the *Dutch* language, the sequence should be:

- 2k9:
  - **(keySourceRc6, 0, rc6S0Digit0) – 10ms delay**
  - **(keySourceRc6, 0, rc6S0Digit0) – 10ms delay**
  - **(keySourceRc6, 0, rc6S0Digit0) – 10ms delay**
  - **(keySourceRc6, 0, rc6S0Digit0)**
  - **(keySourceRc6, 0, rc6S0StepUp)**
- 2k10/2k11:
  - **(keySourceRc6, 0, rc6S0Digit1) – 10ms delay**
  - **(keySourceRc6, 0, rc6S0Digit1) – 10ms delay**
  - **(keySourceRc6, 0, rc6S0Digit1) – 10ms delay**
  - **(keySourceRc6, 0, rc6S0Digit1) – 10ms delay**
  - **(keySourceRc6, 0, rc6S0Digit1) – 10ms delay**
  - **(keySourceRc6, 0, rc6S0Digit1) – 10ms delay**
  - **(keySourceRc6, 0, rc6S0Digit1) – 10ms delay**
  - **(keySourceRc6, 0, rc6S0Digit1) – 10ms delay**
  - **(keySourceRc6, 0, rc6S0Digit1)**
  - **(keySourceRc6, 0, rc6S0Acknowledge)**

Multitap tables for 2k9

Key	Bulgarian (bg)	Brazilian_Portuguese (bpt)	Croatian (hr)
1	_-1	_-1	_-1
2	АБВГабвг2ABCabc	ABCabc2	ABCabc2ĆČćč
3	ДЕЖЗдежз3DEFdef	DEFdef3	DEFdef3Đđ
4	ИЙКЛийкл4GHIghi	GHIghi4	GHIghi4
5	МНОПмноп5JKLjkl	JKLjkl5	JKLjkl5
6	РСТУрсту6MNOmno	MNOmno6	MNOmno6
7	ФХЦЧфхцч7PQRSpqrs	PQRSpqrs7	PQRSpqrs7Šš
8	ШЩЪЬЫшщъьы8TUVtuv	TUVtuv8	TUVtuv8
9	ЬЭЮЯьэюя9WXYZwxyz	WXYZwxyz9	WXYZwxyz9Žž
0	.@0!"#$%&'()*+,/:;<=>?[\]^`{|}~	.@0!"#$%&'()*+,/:;<=>?[\]^`{|}~	.@0!"#$%&'()*+,/:;<=>?[\]^`{|}~

Key	Czech (cs)	Danish (da)	Dutch (nl)
1	_-1	_-1	_-1
2	ABCabc2ÁČáč	ABCabc2ÅÆåæ	ABCabc2
3	DEFdef3ĎÉĚďéě	DEFdef3	DEFdef3
4	GHIghi4ÍÍ	GHIghi4	GHIghi4
5	JKLjkl5	JKLjkl5	JKLjkl5
6	MNOmno6ŇÓňó	MNOmno6Øø	MNOmno6
7	PQRSpqrs7ŘŠřš	PQRSpqrs7	PQRSpqrs7
8	TUVtuv8ŤÚŮťúů	TUVtuv8	TUVtuv8
9	WXYZwxyz9ÝŽýž	WXYZwxyz9	WXYZwxyz9
0	.@0!"#$%&'()*+,/:;<=>?[\]^`{|}~	.@0!"#$%&'()*+,/:;<=>?[\]^`{|}~	.@0!"#$%&'()*+,/:;<=>?[\]^`{|}~

Key	English (en)	Estonian (et)	Finnish (fi)
1	_-1	_-1	_-1
2	ABCabc2	AÄBCaäbc2	ABCabc2ÄÅäå
3	DEFdef3	DEFdef3	DEFdef3
4	GHIghi4	GHIghi4	GHIghi4
5	JKLjkl5	JKLjkl5	JKLjkl5
6	MNOmno6	MNOÕÖmnoõö6	MNOmno6Öö
7	PQRSpqrs7	PQRSŠpqrsš7	PQRSpqrs7
8	TUVtuv8	TUÜVtuüv8	TUVtuv8
9	WXYZwxyz9	WXYZŽwxyzž9	WXYZwxyz9
0	.@0!"#$%&'()*+,/:;<=>?[\]^`{|}~	.@0!"#$%&'()*+,/:;<=>?[\]^`{|}~	.@0!"#$%&'()*+,/:;<=>?[\]^`{|}~

Key	French (fr)	German (de)	Greek (el)
1	_-1	_-1	_-1
2	ABCabc2ÀÁÇàáç	ABCabc2Ää	ABΓαβγ2ABCabc
3	DEFdef3ÈÉÊèéê	DEFdef3	ΔΕΖδεζ3DEFdef
4	GHIghi4	GHIghi4	ΗΘΙηθι4GHIghi
5	JKLjkl5	JKLjkl5	ΚΛΜκλμ5JKLjkl
6	MNOmno6	MNOmno6Öö	ΝΞΟνξο6MNOmno
7	PQRSpqrs7	PQRSpqrs7ß	ΠΡΣπρσς7PQRSpqrs
8	TUVtuv8	TUVtuv8Üü	ΤΥΦτυφ8TUVtuv
9	WXYZwxyz9	WXYZwxyz9	ΧΨΩχψω9WXYZwxyz
0	.@0!"#$%&'()*+,/:;<=>?[\]^`{|}~	.@0!"#$%&'()*+,/:;<=>?[\]^`{|}~	.@0!"#$%&'()*+,/:;<=>?[\]^`{|}~

Key	Hungarian (hu)	Italian (it)	Kazakh (kk)
1	_-1	_-1	_-1
2	AÁBCaábc2	AÀBCaàbc2	AӘБВГaәбвг2ABCabc
3	DEÉFdeéf3	DEÈFdeèf3	ҒДЕЖ3ғдежз3DEFdef
4	GHIÍghií4	GHIÌghìì4	ИЙКҚЛийккқл4GHIghi
5	JKLjkl5	JKLjkl5	МНОӨПмноөп5JKLjkl
6	MNOÓÖŐmnoóöő6	MNOÒmnoò6	РСТУҰрстуұ6MNOmno
7	PQRSpqrs7	PQRSpqrs7	ҮФХЦЧүфхцч7PQRSpqrs
8	TUÚÜŰVtuúüűv8	TUÙVtuùv8	ШЩЪЫІшщъыі8TUVtuv
9	WXYZwxyz9	WXYZwxyz9	ЬЭЮЯьэюя9WXYZwxyz
0	.@0!"#$%&'()*+,/:;<=>?[\]^`{|}~	.@0!"#$%&'()*+,/:;<=>?[\]^`{|}~	.@0!"#$%&'()*+,/:;<=>?[\]^`{|}~

Key	Latvian (lv)	Lithuania (lt)	Norwegian (no)
1	_-1	_-1	_-1
2	AĀBCČaābcč2	AĄBCaąbc2	ABCabc2ÅÆåæ
3	DEĒFdeēf3	DEĘĖFdeęėf3	DEFdef3
4	GĢHIĪgģhiī4	GHIĮghiį4	GHIghi4
5	JKĶLĻjkķlļ5	JKLjkl5	JKLjkl5
6	MNŅOmnņo6	MNOmno6	MNOmno6Øø
7	PRSŠprsš7	PQRSŠpqrsš7	PQRSpqrs7
8	TUŪVtuūv8	TUŲŪVtuųūv8	TUVtuv8
9	ZŽzž9	WXYZŽwxyzž9	WXYZwxyz9
0	.@0!"#$%&'()*+,/:;<=>?[\]^`{|}~	.@0!"#$%&'()*+,/:;<=>?[\]^`{|}~	.@0!"#$%&'()*+,/:;<=>?[\]^`{|}~

Key	Polish (pl)	Portuguese (pt)	Romanian (ro)
1	_-1	_-1	_-1
2	ABCabc2ĄĆąć	ABCabc2ÁÃÇáãç	ABCabc2ĂăÂâ
3	DEFdef3Ęę	DEFdef3ÉÊéê	DEFdef3
4	GHIghi4	GHIghi4Íí	GHIghi4ÎîÏï
5	JKLjkl5Łł	JKLjkl5	JKLjkl5
6	MNOmno6ŃÓńó	MNOmno6ÓÕóõ	MNOmno6
7	PQRSpqrs7Śś	PQRSpqrs7	PQRSpqrs7Şş
8	TUVtuv8	TUVtuv8Úú	TUVtuv8Ţţ
9	WXYZwxyz9ŹŻźż	WXYZwxyz9	WXYZwxyz9
0	.@0!"#$%&'()*+,/:;<=>?[\]^`{|}~	.@0!"#$%&'()*+,/:;<=>?[\]^`{|}~	.@0!"#$%&'()*+,/:;<=>?[\]^`{|}~

Key	Russian (ru)	Serbian (sr)	Slovak (sk)
1	_-1	_-1	_-1
2	АБВГабвг2ABCabc	ABCabc2ĆČćč	ABCabc2ÁÄČáäč
3	ДЕЁЖ3деёжз3DEFdef	DEFdef3Đđ	DEFdef3ĎÉďé
4	ИЙКЛийкл4GHIghi	GHIghi4	GHIghi4Íí
5	МНОПмноп5JKLjkl	JKLjkl5	JKLjkl5ĹĽÍľ
6	РСТУрсту6MNOmno	MNOmno6	MNOmno6ŇÓÔňóô
7	ФХЦЧфхцч7PQRSpqrs	PQRSpqrs7Šš	PQRSpqrs7Šš
8	ШЩЪЫшщъы8TUVtuv	TUVtuv8	TUVtuv8ŤÚťú
9	ЬЭЮЯьэюя9WXYZwxyz	WXYZwxyz9Žž	WXYZwxyz9ÝŽýž
0	.@0!"#$%&'()*+,/:;<=>?[\]^`{|}~	.@0!"#$%&'()*+,/:;<=>?[\]^`{|}~	.@0!"#$%&'()*+,/:;<=>?[\]^`{|}~

Key	Slovenian (sl)	Spanish (es)	Swedish (sv)
1	_-1	_-1	_-1
2	ABCabc2ĆČćč	ABCabc2Áá	ABCÄÅabcää2
3	DEFdef3	DEFdef3Éé	DEÉFdeéf3
4	GHIghi4	GHIghi4Íí	GHIghi4
5	JKLjkl5	JKLjkl5	JKLjkl5
6	MNOmno6	MNOmno6ÑñÓó	MNOÖmnoö6
7	PQRSpqrs7Šš	PQRSpqrs7	PQRSpqrs7
8	TUVtuv8	TUVtuv8ÚÜúü	TUÜVtuüv8
9	WXYZwxyz9Žž	WXYZwxyz9	WXYZwxyz9
0	.@0!"#$%&'()*+,/:;<=>?[\]^`{|}~	.@0!"#$%&'()*+,/:;<=>?[\]^`{|}~	.@0!"#$%&'()*+,/:;<=>?[\]^`{|}~

Key	Turkish (tr)	Ukrainian (uk)	Argentinian Spanish (eas)
1	_-1	_-1	_-1
2	ABCabc2ÂÇâç	АБВГабвг2ABCabc	ABCabc2
3	DEFdef3	ДЕЄЖдеєж3DEFdef	DEFdef3
4	GHIghi4ĞİÎğıî	ЗІЇЙзіїй4GHIghi	GHIghi4
5	JKLjkl5	КЛМклм5JKLjkl	JKLjkl5
6	MNOmno6Öö	НОПноп6MNOmno	MNOmno6
7	PQRSpqrs7Şş	РСТУрсту7PQRSpqrs	PQRSpqrs7
8	TUVtuv8ÜÛüû	ФХЦфхц8TUVtuv	TUVtuv8
9	WXYZwxyz9	ШЩЮЯшщюя9WXYZwxyz	WXYZwxyz9
0	.@0!"#$%&'()*+,/:;<=>?[\]^`{|}~	.@0!"#$%&'()*+,/:;<=>?[\]^`{|}~	.@0!"#$%&'()*+,/:;<=>?[\]^`{|}~

Multitap tables for 2k10/2k11

Key	Arabic (ar)	Bulgarian (bg)	Brazilian_Portuguese (bpt)
1	.-/@1,'?!"():_;+&%*=<>€£$[]{}\~^`#|1	.-/@1,'?!"():_;+&%*=<>€£$[]{}\~^`#|1	.-/@1,'?!"():_;+&%*=<>€£$[]{}\~^`#|1
2	2CBAcba ث ة ت ب	2абвгАБВГabcABC	abc2ABC2
3	3defDEF ء ا	3дежзДЕЖЗdefDEF	def3DEF3
4	4IHGihg ض ص ش س	4ийклИЙКЛghiGHI	ghi4GHI4
5	5LKJlkj ز ر ذ د	5мнопМНОПjklJKL	jkl5JKL5
6	6ONMonm خ ح ج	6рстуРСТУmnoMNO	mno6MNO6
7	7SRQPsrqp ي و ه ن	7фхцчФХЦЧpqrsPQRS	pqrs7PQRS7
8	8VUTvut م ل ك ق ف	8шщъыШЩЪЫtuvTUV	tuv8TUV8
9	9ZYXWzyxw غ ع ظ ط	9ьэюяЬЭЮЯwxyzWXYZ	wxyz9WXYZ9
0	␣ ↵ 0	␣ ↵ 0	␣ ↵ 0

Key	Croatian (hr)	Czech (cs)	Danish (da)
1	.-/@1,'?!"():_;+&%*=<>€£$[]{}\~^`#|1	.-/@1,'?!"():_;+&%*=<>€£$[]{}\~^`#|1	.-/@1,'?!"():_;+&%*=<>€£$[]{}\~^`#|1
2	abcćč2ABCĆČ2	aábcč2AÁBCČ2	aåæbc2AÅÆBC2
3	dđef3DĐEF3	dďeéěf3DĎEÉĚF3	def3DEF3
4	ghi4GHI4	ghií4GHIÍ4	ghi4GHI4
5	jkl5JKL5	jkl5JKL5	jkl5JKL5
6	mno6MNO6	mnňoó6MNŇOÓ6	mnoø6MNOØ6
7	pqrsš7PQRSŠ7	pqřsš7PQRŘSŠ7	pqrs7PQRS7
8	tuv8TUV8	tťuúův8TŤUÚŮV8	tuv8TUV8
9	wxyzž9WXYZŽ9	wxyýzž9WXYÝZŽ9	wxyz9WXYZ9
0	␣ ↵ 0	␣ ↵ 0	␣ ↵ 0

Key	Dutch (nl)	English (en)	Estonian (et)
1	.-/@1,'?!"():_;+&%*=<>€£$[]{}\~^`#|1	.-/@1,'?!"():_;+&%*=<>€£$[]{}\~^`#|1	.-/@1,'?!"():_;+&%*=<>€£$[]{}\~^`#|1
2	abc2ABC2	abc2ABC2	aäbc2AÄBC2
3	def3DEF3	def3DEF3	def3DEF3
4	ghi4GHI4	ghi4GHI4	ghi4GHI4
5	jkl5JKL5	jkl5JKL5	jkl5JKL5
6	mno6MNO6	mno6MNO6	mnoõö6MNOÕÖ6
7	pqrs7PQRS7	pqrs7PQRS7	pqrsš7PQRSŠ7
8	tuv8TUV8	tuv8TUV8	tuüv8TUÜV8
9	wxyz9WXYZ9	wxyz9WXYZ9	wxyzž9WXYZŽ9
0	␣↵0	␣↵0	␣↵0

Key	Finnish (fi)	French (fr)	German (de)
1	.-/@1,'?!"():_;+&%*=<>€£$[]{}\~^`#|1	.-/@1,'?!"():_;+&%*=<>€£$[]{}\~^`#|1	.-/@1,'?!"():_;+&%*=<>€£$[]{}\~^`#|1
2	aääbc2AÄÅBC2	aàábcç2AÀÁBCÇ2	aäbc2AÄBC2
3	def3DEF3	deèéêf3DEÈÉÊF3	def3DEF3
4	ghi4GHI4	ghi4GHI4	ghi4GHI4
5	jkl5JKL5	jkl5JKL5	jkl5JKL5
6	mnoö6MNOÖ6	mno6MNO6	mnoö6MNOÖ6
7	pqrs7PQRS7	pqrs7PQRS7	pqrsß7PQRS7
8	tuv8TUV8	tuv8TUV8	tuüv8TUÜV8
9	wxyz9WXYZ9	wxyz9WXYZ9	wxyz9WXYZ9
0	␣↵0	␣↵0	␣↵0

Key	Greek (el)	Hungarian (hu)	Italian (it)
1	.-/@1,'?!"():_;+&%*=<>€£$[]{}\~^`#|1	.-/@1,'?!"():_;+&%*=<>€£$[]{}\~^`#|1	.-/@1,'?!"():_;+&%*=<>€£$[]{}\~^`#|1
2	αβγABΓ2abcABC2	aábc2AÁBC	aàbc2AÀBC2
3	δεζΔEZ3defDEF3	deéf3DEÉF3	deèf3DEÈF3
4	ηθιHΘI4ghiGHI4	ghií4GHIÍ4	ghìì4GHIÌ4
5	κλμKΛM5jklJKL5	jkl5JKL5	jkl5JKL5
6	νξοNΞO6mnoMNO6	mnoóöő6MNOÓÖŐ6	mnoò6MNOÒ6
7	πρσςΠΡΣ7pqrsPQRS7	pqrs7PQRS7	pqrs7PQRS7
8	τυφTΥΦ8tuvTUV8	tuúüű8TUÚÜŰV8	tuùv8TUÙV8
9	χψωXΨΩ9wxyzWXYZ9	wxyz9WXYZ9	wxyz9WXYZ9
0	␣↵ 0	␣↵ 0	␣↵ 0

Key	Kazakh (kk)	Latvian (lv)	Lithuania (lt)
1	.-/@1,'?!"():_;+&%*=<>€£$[]{}\~^`#|1	.-/@1,'?!"():_;+&%*=<>€£$[]{}\~^`#|1	.-/@1,'?!"():_;+&%*=<>€£$[]{}\~^`#|1
2	аәбвгАӘБВГ2abcABC2	aābcč2AĀBCČ2	aąbc2AĄBC2
3	ғдежзҒДЕЖ33defDEF3	deēf3DEĒF3	deęėf3DEĘĖF3
4	ийкқлИЙКҚЛ4ghiGHI4	gģhīī4GĢHIĪ4	ghiį4GHIĮ4
5	мноөпМНОӨП5jklJKL5	jkķlļ5JKĶLĻ5	jkl5JKL5
6	рстуұРСТУҰ6mnoMNO6	mnņo6MNŅO6	mno6MNO6
7	үфхцчҮФХЦЧ7pqrsPQRS7	pqrsš7PQRSŠ7	pqrsš7PQRSŠ7
8	шщъыіШЩЪЫІ8tuvTUV8	tuūv8TUŪV8	tuųūv8TUŲŪV8
9	ьэюяЬЭЮЯ9wxyzWXYZ9	wxyzž9WXYZŽ9	wxyzž9WXYZŽ9
0	␣↵ 0	␣↵ 0	␣↵ 0

Key	Norwegian (no)	Polish (pl)	Portuguese (pt)
1	.-/@1,'?!"():_;+&%*=<>€£$[]{}\~^`#|1	.-/@1,'?!"():_;+&%*=<>€£$[]{}\~^`#|1	.-/@1,'?!"():_;+&%*=<>€£$[]{}\~^`#|1
2	aåæbc2AÅÆBC2	aąbcć2AĄBCĆ2	aáãbcç2AÁÃBCÇ2
3	def3DEF3	deęf3DEĘF3	déêef3DÉÊF3
4	ghi4GHI4	ghi4GHI4	ghií4GHIÍ4
5	jkl5JKL5	jklł5JKLŁ5	jkl5JKL5
6	mnoø6MNOØ6	mnńoó6MNŃOÓ6	mnoóõ6MNOÓÕ6
7	pqrs7PQRS7	pqrsś7PQRSŚ7	pqrs7PQRS7
8	tuv8TUV8	tuv8TUV8	tuúv8TUÚV8
9	wxyz9WXYZ9	wxyzźż9WXYZŹŻ9	wxyz9WXYZ9
0	␣↵ 0	␣↵ 0	␣↵ 0

Key	Romanian (ro)	Russian (ru)	Serbian (sr)
1	.-/@1,'?!"():_;+&%*=<>€£$[]{}\~^`#|1	.-/@1,'?!"():_;+&%*=<>€£$[]{}\~^`#|1	.-/@1,'?!"():_;+&%*=<>€£$[]{}\~^`#|1
2	aăbc2AĂBC2	абвгАБВГ2abcABC2	abcćč2ABCĆČ2
3	def3DEF3	деёжзДЕЁЖ33defDEF3	dđef3DĐEF3
4	ghiî4GHIÎ4	ийклИЙКЛ4ghiGHI4	ghi4GHI4
5	jkl5JKL5	мнопМНОП5jklJKL5	jkl5JKL5
6	mno6MNO6	рстуРСТУ6mnoMNO6	mno6MNO6
7	pqrsş7PQRSŞ7	фхцчФХЦЧ7pqrsPQRS7	pqrsš7PQRSŠ7
8	tţuv8TŢUV8	шщъыШЩЪЫ8tuvTUV8	tuv8TUV8
9	wxyz9WXYZ9	ьэюяЬЭЮЯ9wxyzWXYZ9	wxyzž9WXYZŽ9
0	␣↵ 0	␣↵ 0	␣↵ 0

Key	Slovak (sk)	Slovenian (sl)	Spanish (es)
1	.-/@1,'?!"():_;+&%*=<>€£$[]{}\~^`#|1	.-/@1,'?!"():_;+&%*=<>€£$[]{}\~^`#|1	.-/@1,'?!"():_;+&%*=<>€£$[]{}\~^`#|1
2	aáäbcč2AÁÄBCČ2	abcćč2ABCĆČ2	aábc2AÁBC2
3	dďeéf3DĎEÉF3	def3DEF3	deéf3DEÉF3
4	ghií4GHIÍ4	ghi4GHI4	ghií4GHIÍ4
5	jklĺľ5JKLĹĽ5	jkl5JKL5	jkl5JKL5
6	mnňoóô6MNŇOÓÔ6	mno6MNO6	mnñoó6MNÑOÓ6
7	pqrsš7PQRSŠ7	pqrsš7PQRSŠ7	pqrs7PQRS7
8	tťuúv8TŤUÚV8	tuv8TUV8	tuúüv8TUÚÜV8
9	wxyýzž9WXYÝZŽ9	wxyzž9WXYZŽ9	wxyz9WXYZ9
0	␣↵ 0	␣↵ 0	␣↵ 0

Key	Swedish (sv)	Turkish (tr)	Ukrainian (uk)
1	.-/@1,'?!"():_;+&%*=<>€£$[]{}\~^`#|1	.-/@1,'?!"():_;+&%*=<>€£$[]{}\~^`#|1	.-/@1,'?!"():_;+&%*=<>€£$[]{}\~^`#|1
2	aäåbc2AÄÅBC2	aâbcç2AÂBCÇ2	абвгАБВГ2abcABC2
3	deéf3DEÉF3	def3DEF3	деєжДЕЄЖ3defDEF3
4	ghi4GHI4	gğhıiî4GĞHIİÎ4	зїїй3ЇЇЙ4ghiGHI4
5	jkl5JKL5	jkl5JKL5	клмКЛМ5jklJKL5
6	mnoö6MNOÖ6	mnoö6MNOÖ6	нопНОП6mnoMNO6
7	pqrs7PQRS7	pqrsş7PQRSŞ7	рстуРСТУ7pqrsPQRS7
8	tuüv8TUÜV8	tuüûv8TUÜÛV8	фхцФХЦ8tuvTUV8
9	wxyz9WXYZ9	wxyz9WXYZ9	шщюяШЩЮЯ9wxyzWXYZ9
0	␣↵ 0	␣↵ 0	␣↵ 0

Key	Argentinian Spanish (eas)
1	.-/@1,'?!"():_;+&%*=<>€£$[]{}\~^`#|1
2	aábc2AÁBC2
3	deéf3DEÉF3
4	ghií4GHIÍ4
5	jkl5JKL5
6	mnñoó6MNÑOÓ6
7	pqrs7PQRS7
8	tuúüv8TUÚÜV8
9	wxyz9WXYZ9
0	␣↵ 0

Revision History

Version	Date	Comments
1	20101215	Initial version
2	20110202	Adapted for first release
2.1	20110204	Rework after review comments
2.2	20110208	Small adaptation for 2k9 in the Remote Control command table