Technical Document

GPF

XPANEL+

DESIGN SPEC

Document Number:	8434.408.02.005
Version:	0.6
Status:	Draft
Approval Authority:	
Creation Date:	2002-Jan-28
Last changed:	2015-Mar-08 by XINTEGRA
File Name:	XPanel_plus_design_spec.doc

Change History

Date	Changed by	Approved by	Version	Status	Notes
2002-Jan-28	CKR		0.1		1
2002-Feb-06	CKR		0.2		2
2002-Feb-12	CKR		0.3		3
2002-Feb-14	CKR		0.4		4
2002-Mar-12	CKR		0.5		5
2003-May-21	XINTEGRA		0.6	Draft	

Note s:

1. Initial version
2. Update
3. Update
4. Update
5. Update

Table of Contents

List of Figures and Tables

List of References

[ISO 9000:2000]	International Organization for Standardization. Quality management systems - Fundamentals and vocabulary. December 2000

1.1 References, Abbreviations, Terms

[C_7010.801] 7010.801, References and Vocabulary, Condat AG

2 Introduction

xPanel+ is a tool allowing to control a simulated keypad and LCD on a PC. Therefor the PC needs to be connected via a HW interface – currently only the serial interface is supported. To control a phone this way is no new functionality but an enhancement of the existing tool xPanel. The tool retains backward compatibility.

The new features are:
- the keypad can be designed using customer defined keypad and display layout
- support for colour displays
- display size and bits/per pixel are dynamically adjusted to the values used by the phone

Due to its nature – its not a **simulation** of an MMI but a remote front end – the amount of data transmitted can be vast especially for high resolution colour displays. Above all the speed of the HW interface between the PC and the phone may limit the refresh rate of the external display.

3 High Level Design

3.1 Protocol Description

Prior to data transmission the capabilities of the real display are exchanged e.g. display size and bits/pixel. This is done either by an explicit request by xPanel+ or by an unsolicited indication by the Protocol Stack.

For transmission of the display data a “poll” approach is taken. xPanel+ explicitly request the display data. The Protocol Stack sends the display data only upon reception of such a request. If such a request stays unanswered for a certain time i.e. because the Protocol Stack could not send the data or the display content did not change, xPanel+ reissues the request.

The format of the data used for transmission i.e. primitives are defined in a dedicated SAP. The format is such that future extensions are easy to implement.

3.2 Display Driver

Part of the implementation needs to be done in the display driver. Here it is monitored whether the display content changed and thus needs to be sent to xPanel+. Since this code may need to be adapted or replaced by the customer i.e. to allow the use of other displays, the implementation serves as sample code.

All relevant parts of the code are marked and commented elaborately. The customer MUST not modify the upper interface of the display driver. Maintaining this interface allows changing the MMI implementation without any changes to the rest of the protocol stack.

3.3 Runtime Architecture

The functionality used for the transmission of the display data and capabilities is implemented either in a dedicated Entity (process) or as an extension to an existing Entity.

Transmission takes place using the Frame serial driver directly thus bypassing the restrictions imposed by the mechanisms used for the sending of Traces and Primitives. Display data is sent in partitions of limited size. The size can be adjusted to the restrictions imposed by the interface.

It will be assured that the transmission of the possibly fast amount of display data does not cause any misbehaviour of the Protocol Stack. If the load on the interface is too high display data can be discarded.

3.4 Configuration

1, 8 and 16 bits colour information per pixel are supported. Colour mapping for 8bit colour will be done using an (editable) colour map table. For 16 bit colour an RGB format is used, which will be described later.

The keypad and display layout can be controlled using an (editable) table.

4 Low Level Design

4.1 Protocol Description

4.1.1 Primitives

4.1.1.1 EXTDSPL_CAP_REQ

Description:

The EXTDSPL_CAP_REQ is used to initiate a xPanel+ originated capability exchange.

Definition:

short name	ID	direction
EXTDSPL_CAP_REQ	0x3F00	xPanel+ -> stack

Elements:

long name	short name	ref	type
dummy parameter	dummy		U32

4.1.1.2 EXTDSPL_CAP_IND

Description:

The EXTDSPL_CAP_IND is used to transmit the capabilities of the display.

Definition:

short name	ID	direction
EXTDSPL_CAP_IND	0x7F00	stack -> xPanel+

Elements:

long name	short name	ref	type
number of columns	width		U16
number of rows	height		U16
bits per pixel	bpp		U16

4.1.1.3 EXTDSPL_DATA_REQ

Description:

The EXTDSPL_DATA_REQ is used to initiate data transfer.

Definition:

short name	**ID**	**direction**
EXTDSPL_DATA_ REQ	0x3F01	xPanel+ -> stack

Elements:

long name	**short name**	**ref**	**type**
dummy parameter	dummy		U32

4.1.1.4 EXTDSPL_DATA_IND

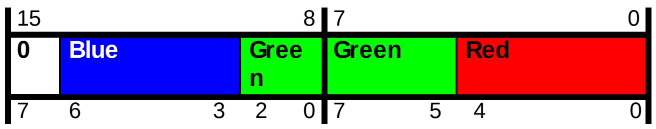
Description:

The EXTDSPL_DATA_IND is used to transmit a rectangular region of display data. The transmission starts with the upper left corner of the region and finishes with the lower right corner. The sdu contains the display data. The interpretation of the display data depends on the number of bits per pixel.

1 bpp: each transmitted byte contains display information for 8 pixel, the least significant bit describes the rightmost pixel

8 bpp: each transmitted byte contains display information for 1 pixel

16 bpp: two transmitted bytes contain display information for 1 pixel, the least significant byte will be transmitted first, the value is transmitted as an RGB colour information with 5 bits for every component, the remaining bit is always set to 0, the colour components are spread over the two bytes as follows:



Definition:

short name	**ID**	**direction**
EXTDSPL_DATA_ IND	0x7F01	stack -> xPa- nel+

Elements:

long name	**short name**	**ref**	**type**
starting column	col		U16
starting row	row		U16
width in columns	width		U16
height in rows	height		U16
service data unit	sdu		STRU CT

4.1.2 Message Sequence Charts

4.1.2.1 Stack originated capability exchange

When initialising the protocol stack issues a capability indication. Upon reception the xPanel+ enters data transfer mode and issues data request.

```
stack        xPanel+
  |             |
  |EXTDSPL_CAP_IND        |
  *==========================> *
  |EXTDSPL_DATA_REQ       |
  *<========================== *
  |             |
```

Figure 1: Stack originated capability exchange

4.1.2.2 xPanel+ originated capability exchange

When initialising the xPanel+ send a capability request to the protocol stack. Upon reception the protocol stack issues a capability indication. When this reaches the xPanel+ it enters data transfer mode and sends a data request.

```
stack        xPanel+
  |             |
  |EXTDSPL_CAP_REQ        |
  *<========================== *
  |EXTDSPL_CAP_IND        |
  *==========================> *
  |EXTDSPL_DATA_REQ       |
  *<========================== *
  |             |
```

Figure 2: xPanel+ originated capability exchange

4.1.2.3 xPanel+ originated capability exchange (with timeout)

After the xPanel+ issues a capability request it waits for a capability indication. After a certain capability timeout with no reception from the protocol stack it reissues the capability request. Only after the reception of a capability indication it enters data transfer mode.

```
stack        xPanel+
  |             |
  |EXTDSPL_CAP_REQ        |
  *<========================== *
  |             |
  |        <Timeout T_CAP>
  |EXTDSPL_CAP_REQ        |
  *<========================== *
  |EXTDSPL_CAP_IND        |
  *==========================> *
  |EXTDSPL_DATA_REQ       |
  *<========================== *
  |             |
```

Figure 3: xPanel+ originated capability exchange with timeout

4.1.2.4 Data transfer

In data transfer mode the xPanel issues a data request and waits for the reception of a data indication. Upon reception of this it again issues a data request.

```
stack        xPanel+
  |             |
  |EXTDSPL_DATA_REQ      |
  *<========================== *
  |EXTDSPL_DATA_IND      |
  *==========================> *
  |EXTDSPL_DATA_REQ      |
  *<========================== *
  |EXTDSPL_DATA_IND      |
  *==========================> *
  |      ...    |
```

Figure 4: Data transfer

4.1.2.5 Data transfer (with timeout)

When in data transfer mode a data request is not answer within a certain data timeout the xPanel reissues the data request.

```
stack        xPanel+
  |             |
  |EXTDSPL_DATA_REQ      |
  *<========================== *
  |             |
  |        <Timeout T_DATA>
  |EXTDSPL_DATA_REQ      |
  *<========================== *
  |EXTDSPL_DATA_IND      |
  *==========================> *
  |EXTDSPL_DATA_REQ      |
  *<========================== *
  |EXTDSPL_DATA_IND      |
  *==========================> *
  |             |
```

Figure 5: Data transfer (with timeout)

4.1.2.6 Data transfer (with repeated timeout)

When in data transfer mode a data request is not answer within a certain data timeout the xPanel reissues the data request. Upon no reception of a data indication the xPanel+ will reissue the data request for a maximum data retransmission number. If all these request stay unanswered. The xPanel+ issues a capability request. Upon reception of a capability indication it again enters data transfer mode.

```
stack        xPanel+
   |             |
   |EXTDSPL_DATA_REQ      |
   *<========================== *
   |             |
   |        <Timeout T_DATA>
   |EXTDSPL_DATA_REQ      |
   *<========================== *
   |             |
   |        <Timeout T_DATA>
   |EXTDSPL_DATA_REQ      |
   *<========================== *
   |             |
   |        <Timeout T_DATA>
   |EXTDSPL_DATA_REQ      |
   *<========================== *
   |             |
   |        <Timeout T_DATA>
   |EXTDSPL_DATA_REQ      |
   *<========================== *
   |             |
   |        <Timeout T_DATA>
   |EXTDSPL_CAP_REQ       |
   *<========================== *
   |EXTDSPL_CAP_IND       |
   *==========================> *
   |EXTDSPL_DATA_REQ      |
   *<========================== *
   |             |
```

Figure 6: Data transfer (with repeated timeout)

4.1.2.7 Data transfer (with no answer at all)

When in data transfer mode a data request is not answer within a certain data timeout the xPanel reissues the data request. Upon no reception of a data indication the xPanel+ will reissue the data request for a maximum data retransmission number. If all these request stay unanswered. The xPanel+ issues a capability request. Upon no reception of a capability indication it reissues a capability request after the expiry of the capability timer.

```
stack        xPanel+
  |             |
  |EXTDSPL_DATA_REQ      |
  *<=========================== *
  |             |
  |        <Timeout T_DATA>
  |EXTDSPL_DATA_REQ      |
  *<=========================== *
  |             |
  |        <Timeout T_DATA>
  |EXTDSPL_DATA_REQ      |
  *<=========================== *
  |             |
  |        <Timeout T_DATA>
  |EXTDSPL_DATA_REQ      |
  *<=========================== *
  |             |
  |        <Timeout T_DATA>
  |EXTDSPL_DATA_REQ      |
  *<=========================== *
  |             |
  |        <Timeout T_DATA>
  |EXTDSPL_CAP_REQ       |
  *<=========================== *
  |             |
  |        <Timeout T_CAP>
  |EXTDSPL_CAP_REQ       |
  *<=========================== *
  |             |
  |        <Timeout T_CAP>
  |EXTDSPL_CAP_REQ       |
  *<=========================== *
  |             |
```

Figure 7: Data transfer (with no answer at all)

4.1.2.8 Data transfer with multiple data indications

After the reception of a one data request the protocol stack can send more than one data request. The number of data requests that can be sent at once must be a fixed number.

```
stack        xPanel+
  |             |
  |EXTDSPL_DATA_REQ      |
  *<=========================== *
  |EXTDSPL_DATA_IND      |
  *===========================> *
  |EXTDSPL_DATA_IND      |
  *===========================> *
  |EXTDSPL_DATA_IND      |
  *===========================> *
  |EXTDSPL_DATA_IND      |
  *===========================> *
  |EXTDSPL_DATA_IND      |
  *===========================> *
  |EXTDSPL_DATA_REQ      |
  *<=========================== *
  |      ...    |
```

Figure 8: Data transfer with multiple fragments

4.1.3 Constants and Timer

Two timers shall be implemented in the xPanel+ - a capability (Exchange) Timer and a data (Exchange) Timer.

The data timer shall be of such value that it prevents both: slow refresh after the loss of one frame and also too much traffic in case of congestion.

The value of the capability timer shall be larger then the data timer, to allow protocol stacks that do not transmit display data to stay relatively undisturbed.

A maximum data retransmission constant shall be implemented. It controls the number of unanswered data requests after which a capability request will be send.

A constant shall be defined that controls the number of data indications that can be sent at once after the reception of one data request.

4.2 Display Driver (display.c)

The display driver provides information about the dimensions of the display e.g. height, width, bits per pixel. It processes user requests and changes the content of the display. Its interface is defined in document 8415.025.99.014.

In an implementation without support for an external display, the driver only updates the real display. To support an external display i.e. to send the display contents to the xPanel+ the driver is responsible for updating the content of the external display as well.

The driver does not send the display data to xPanel+ itself but shadows the display contents to a user supplied buffer. It also provides the information whether the content of the display has changed since the last time.

It is assumed that the display driver is only used by one task. The functionality to change the display (an thus filling the shadow buffer) as well as the piece of code responsible for sending the display content to the xPanel+ (and thus reading the shadow buffer) must be located in one and the same process. Otherwise the buffer must be protected by some means.

4.3 Implementation (mfw_extdspl.c)

The functionality to send the display data is implemented within the MMI Frame Work (MFW). It is thus available to any MMI build up on this framework.

Initially the implementation queries the display driver for the dimensions of the display. It allocates sufficient memory for the shadow buffer and registers the shadow buffer with the display driver.

Whenever an EXTDSPL primitive is received it is forwarded to the implementation. Here it is analysed and depending on whether it is a capability or a data request the corresponding indication is send.

Upon the reception of a capability request the results of the initial query to the display driver are send in a capability indication.

Upon the reception of a data request the implementation checks whether the contents of the display have changed since the last request. If so a data indication will be sent.

The display content is sent in fragments of fixed size. This is done so to overcome possible limitations of the (serial) interface. The implementation keeps track of which fragments of the screen have already been sent.

Only a fixed number of fragments are send as the result of the reception of a single data request. Upon reception of the next data request the next fragments are sent. This is done so to allow rescheduling of the process.

For the transmission (sending) of the data a direct call to the (serial) driver is used instead of the trace/primitive mechanism.

4.4 Primitive Interface (MmiMain.c, MmiMmi.c)

To allow processing of EXTDSPL primitives a primitive handler must be registered with the actual MMI. This is done as well as some basic initialisation through a call to an initialisation function. This call is located in the initialisation routine of the MMI. The same holds for an exit routine.

4.5 Modes of operation

Generally speaking two modes of operation may be imagined.

- The External Display and (real hardware display) have the same dimensions. They can then both fully functional.
- The external display has different dimensions, e.g. bigger and more colours, than the real display. Only the external display can then be fully functional.

The sense behind these modes is to allow the MMI implementation to use the display driver transparently, no matter whether an external display exists or not. The existence of two fully functional displays (extern and real) with different dimensions would require the MMI to manage both.

In the first mode the display driver stays almost unchanged. It's only modified to shadow the contents of the real display to the shadow buffer. The contents of the shadow buffer contents will then be transmitted.

For second mode the real display is switched of. The complete display driver must be replaced by an implementation that realises a display with the different dimensions. The real hardware can not be used any more.

Appendices

A. Acronyms

DS-WCDMA	Direct Sequence/Spread Wideband Code Division Multiple Access

B. Glossary

International Mobile Telecommunication 2000 (IMT-2000/ITU-2000)	Formerly referred to as FPLMTS (Future Public Land-Mobile Telephone System), this is the ITU's specification/family of standards for 3G. This initiative provides a global infrastructure through both satellite and terrestrial systems, for fixed and mobile phone users. The family of standards is a framework comprising a mix/blend of systems providing global roaming. <URL: http://www.imt-2000.org/>