PPSM

Part II
Writing PPSM Applications

Motorola Semicondu

Document Number	PPSM Version	Re
PDAPSM01U18-10	2.0	Novem
PDAPSM01U18-11	2.1	May 13
PDAPSM03SPM1-10	3.0	March 5
PDAPSM03SPM1-11	**3.1**	**Novem**
PDAPSM03SPM1-12	**3.11**	**Apirl 2(**

Produced by DragonBall Operation, WSSG, Motorola

Fax: (852) 2666-6551

Email: portable@email.sps.mot.com

Website: http://www.apspg.com/products/ppsm/ppsm

Table of Contents

Freescale Semiconductor, Inc.

Guide

Appendices

Preface

What is PPSM? Personal Portable Syste
designed specifically for
operating system enable
such as advanced pager
instruments, organizers,

Audience & Purpose The first three parts of th
ing to learn the importan
trates on a specific aspe

The forth part of this ma
for building and configuri

Part I "PPSM Architecture" des

Part II "Writing PPSM Applicatio
application development
This part is targeted part
tools. Therefore, the tool
tools for writing PPSM a

Part III "API Toolset" details the
larly to those experience
ming methodology. There
alphabetical order for qu

Part IV "System Integrator's Gui
and configure PPSM sys

Appendices At the end of the manual
information.

Part I PPSM Architec

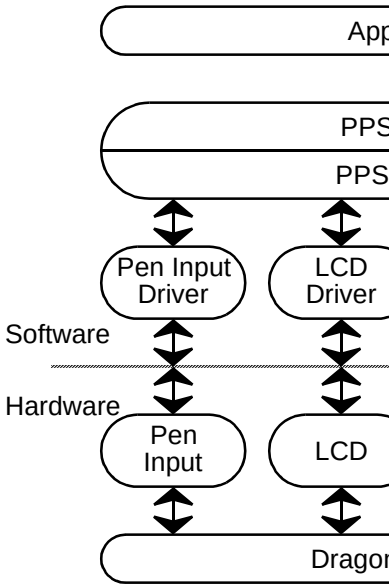


Figure 1-1 Architectu

1.2 Strengths and Feature

PPSM is modularly designed to s
shielding the developer from the
providing a good toolset for the a
their specific product design. Per
effective system which requires v

Strengths:

- SPEEDS up application
- COMPACT ROM size pe
- MODULAR Software Arc
- PEN-CENTRIC
- Third party applications s
  MULTILINGUAL HANDW
  BITMAP FONTS, Infrare
- Application code is comp
- Microsoft Windows base
  application development

Features:

- Pen / touch panel input s
- 32-bit Real Time Operati
- Real Time Interrupt Hanc
- Power Management
- 16-bit text data represen

Chapter 1 Introduction

1.1 What is PPSM?

Personal Portable Syste
designed specifically for
operating system enable
such as advanced pager
instruments, organizers,

PPSM is a real time 32-b
interrupt-driven, e.g. app
icons. Because PPSM is
configurable, and easy to
development process.

The PPSM Tools consist
System and Communica
a sophisticated User-Inte
API for LCD based produ
provides the basic contro
and the UART.

The PPSM kernel does n
devices are controlled by
supplying the appropriate
integrators greater flexib
changing the core of the
the DragonBall™ family

2.2 Error Handling

Unless otherwise specified, all P
completion. A value other than P
has occurred.

Each error code uniquely defines

Refer to *Appendix A - Error Code*

2.3 I/O Devices

This section describes the LCD d

Figure 2-2 shows the screen forr
areas:

- Pen Input Area
- Display Screen
- Panning Screen

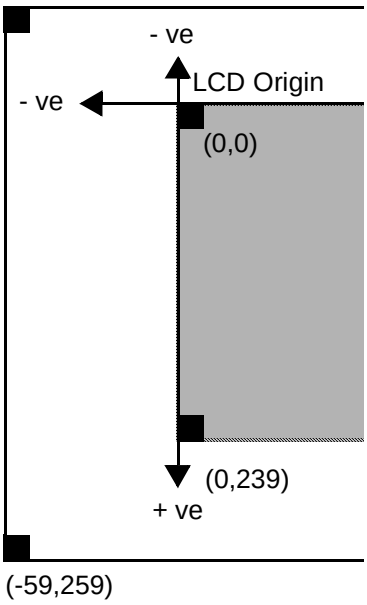


Figure 2-2 An exa

- High Level API T
- Multiple grey lev
- LCD hardware c

1.3 Software Develo

ANSI C is the main progr
implementation of low le
and hardware device driv
PC running under Windo
Debugger, from Software
level and instruction deb
simulator running on the

1.4 Hardware Develo

The M68328 and M68EZ
the reference developme
used is a 10-bit compone
direction.

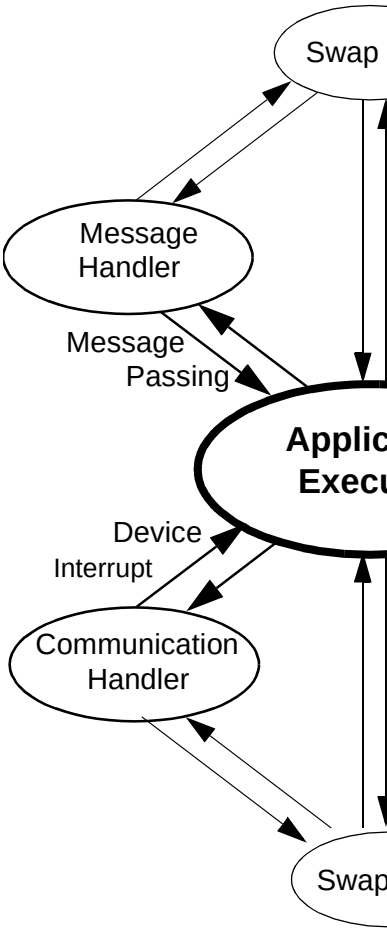
For details on the hardwa
Manual V2.0 and M68EZ

Chapter 2 PPSM Syste

2.1 Interrupt Handli

An application will alway
either be the active task,
interrupt. An application
conditions:

- Interrupt from pe
- Interrupt from a
- Interrupt from po
- Interrupt from tim
- Interrupt from m



F

choose to control the system's p
PPSM's automatic power manag

2.7.1 Direct Control

A set of tools provide the applica
during Normal mode:

- switch into any of the po
- the duty cycle of the proc

2.7.2 Automatic Control

A set of tools are available for the
management features provided b

- to switch automatically to
is idle
- to control user defined I/
saving modes

2.8 Task Management

Each application running on PPS
tasks can be actively running at a

2.8.1 PPSM Tasks

There are two types of PPSM tas
task) and tasks that are spawnec

2.8.1.1 Main Task

Most applications fall into the ma
each other. There cannot be mor
created by the system tool TaskC
created, it can be started in one

- By using the system tool
- By pressing the applicati
- By messages sent by an

2.8.1.2 Sub-task

Sub-task, on the other hand, can
that generated the sub-task. Mes
and its parent. Sub-task uses the
started with the system tool SubT

Sub-tasks are tied to the parent t

2.3.1 Pen Input

2.3.1.1 Pen Input Are

This is the touch sensitiv
touch panel is the same
(0,0) or the origin, is at th
see from *Figure 2-3*, the
have a negative value. P
allows applications to im
writing area.

Fig

If the LCD display screen
coordinates from the pen

2.3.1.2 Active area

An active area is defined
application or an action w
an icon, or an action butt

Active areas are classifie
to *Section 4.1 - Active A*

2.3.2 Screen Format

2.3.2.1 LCD Display Screen

The display screen is the LCD di
images. The LCD module can ha
graphics, giving black and white
Display data, such as graphics a
screen area.

2.3.2.2 Panning Screen

The Panning Screen is an extens
is to allow applications to write da
Although applications can write t
screen unless this area is being
areas on the panning screen will
with the LCD display screen.

2.3.3 Hardware Cursor

The maximum hardware cursor s
task swapping, the hardware cur
origin on panning screen in curre
status, position, size and the offs
used.

2.4 Data Storage

Databases are the main means o
is available in PPSM to support o

PPSM database are in global da
tasks.

When a database is created, PP
the application. Application will n
subsequent access to that partic
any limit in number of database r
limitation in creating these datab

The PPSM database tools provic
and search for particular data. Ac
record list. In particular, the tool D
DBGetPrevRecID() are meant to
sophisticated searching algorithn
example usage of each database
Management and *Chapter 21 - D*

2.5 Font Manageme

PPSM supports 8x10 En
Chinese bitmap fonts, an
are included in PPSM. T
System integrators need
these fonts.

The font bitmap lookup c
33.6 - Font Driver (font.c)
access to these fonts for

2.6 Memory Manage

PPSM provides a set of r
memory space. A heap is
allocate memory from the
provided by the compiler

Four memory manageme

- Lcalloc()
- Lmalloc ()
- Lrealloc()
- Lfree()

PPSM also provides a se
time memory size.

Three memory inquiry to

- TaskMemUsed()
- TotalMemUsed()
- TotalMemSize()
- TaskStackAvail()

The size of the largest ch
be known by calling Lma

The size of the heap me
available in the hardware
ROM? on how memory s

For SingleStep Debuggin
further description in how

2.7 Power Managem

PPSM utilizes the power
power management tools

handler assignment. It must be ca
can also perform the pen to scre
system start-up.

3.1.2 Task Registration

An application is treated as a tas
application task on the system m
application can make use of the

When integrating the individual a
integrator must first call one of th
are two types of tasks, main task
and Section 13.2 - Sub-task). Th
memory and stack required for e
memory system.

Associated with each main applic
This launch icon's position on the
tool. The application is put to the
selected by the pen input device.

When writing an application task
alone procedure as PPSM resou
implementation of tasks allow a r
independently and linked togethe

For details of the task creation to
Task, *Section 13.8 - Creating a T*
13.9 - Creating a Sub Task.

After all applications have been r
by calling the tool TaskStart(). Th
the system will be started when t
pressed.

3.2 PPSM Application Pro

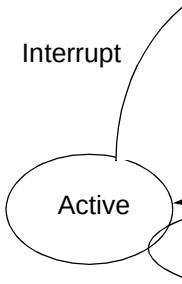
This section describes the gener
PPSM environment. The typical
Figure 3-2. After the application i
areas with PPSM, it would contin
events. When an event occurs, t
loop back to IrptGetData() to wai

terminated, the sub-task
the input pad and pannir

2.8.2 Application State T

Application tasks have th
application.

- Active
- Suspended
- Stopped



Figur

2.8.2.1 Active State

This is the state when ar
hardware resources are

2.8.2.2 Suspended St

An application task is pu
during active state execu
mode, will become the ac
the registers that are use

The suspended task will
otherwise it will be put in

2.8.2.3 Stopped State

An application changes f
application is selected. In
bitmap image are stored

A task will exit from the s

selected. It will continue executic
state.

2.8.3 Task Swapping

Task swapping is performed by F
A task swapping is executed whe
application, needs to become ac
receives an interrupt, such as tir

For example, when an applicatio
that is actively running will be put
active.

2.9 Timer Management

PPSM maintains a reprogramma
January, 1997 upon start up. This
never stops.

A set of timer tools are available
programmer to set system clock,
application.

Chapter 3 PPSM Progra

PPSM programming mai

- PPSM initializati
- Individual PPSM

3.1 PPSM Initializati

A PPSM system must firs
be assigned and used. S
individual PPSM applicat
PPSM organizer may co
scheduler and calculator

Figure 3-1 shows a typic
initialized, then applicatio
with PPSM individually. T
application which is chos

Figure 3

3.1.1 PPSM Initialization

The PPSMInit() tool perf

- OK
- UNKNOWN

3.4.3 Local Variables

All local variables start with lowe
names. There are no underscore

Example 3-3 Local variable

- xPos
- yPos
- dataLen
- temp
- rate

3.4.4 Global Variables

Same as local variables, except
case “g”.

Example 3-4 Global variabl

- gCurrentTask
- gIrptMask
- gSystemClock

3.4.5 Local Pointer Variables

All local pointer variables start wi
variable names. There are no un

Example 3-5 Local pointer

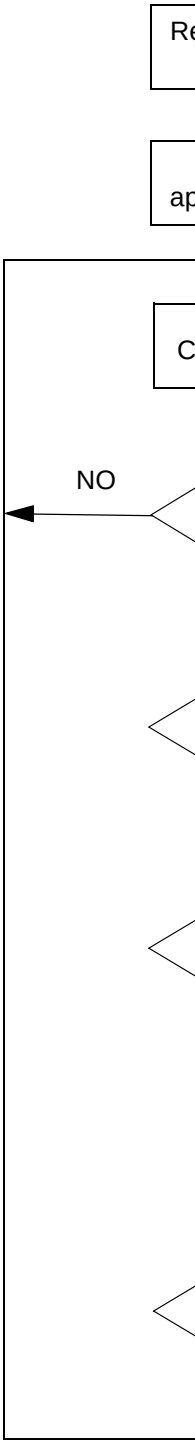
- pSourceAddr
- pDestAddr
- pTaskTable
- pReturnSize

3.4.6 Global Pointer Variables

Same as local pointer variables,
lower case “gp”

Example 3-6 Global pointer

- gpSysList
- gpPhoneBook
- gpSrcMem



Fi

3.2.1 Active Area Registr

PPSM is designed as a

regions defined in PPSM as activ
for applications to receive pen in
need to monitor the hardware co
sampling, maximizing processor

An active area is defined as a rec
messages are generated to the a
areas only generate messages t
example of an active area is an i
area.

When application needs to perfo
register each active area with PF
All remaining areas on the input
any information to the applicatio
please refer to *Chapter 4 - Pen I*
Methods.

3.2.2 Messages from PPSM

PPSM application should take a
When external events occur, suc
automatically intercepts and inte
from the application, such as an
UART, PPSM will package the da
to the waiting application's interr
Handling). However, if the event
pressing of input panel that is no
power saving mode, the event is
message sent to the application.

Upon receiving soft interrupt mes
act upon the nature of the messa
types for different types of interru
Section 15.2 - Device Interrupts)

Application tasks can receive the
IrptGetData(). The application ta
program's flow to check for any i
IrptGetData).

3.3 Data Representation

PPSM is a 32-bit system. *Table*

Table 3-1 Data

Data Type	
U8	
P_U8	ur

Table 3

Data Type
S8
P_S8
U16
P_U16
S16
P_S16
U32
P_U32
S32
P_S32
STATUS
TEXT
P_TEXT
P_VOID

3.4 Naming Convent

For consistency, a set of
software development. T

3.4.1 Procedure

All procedure names are
the name will be in uppe

Example 3-1 Proced

- PenInit()
- PagerCheck()
- DrawDot()

3.4.2 Constants and Lab

All constants and labels

Example 3-2 Consta

- DISPLAY_MODE
- DEFAULT_MOD

Upon release, either by pen-up c
touch panel, another soft interrup
event. This type of area is desigr

4.1.2 Input Area

Input area is an area where writir
will monitor the area with the give
rate, pen echoing and pen positic
Three modes of operation are av
CONFINED or CONTINUOUS.

4.1.2.1 Stroke Mode

Drawing on STROKE type of inp
coordinate integers to the applica
usually with a pen-up. This list cc
pen-input device. When the pen
then the stroke data ends.

4.1.2.2 Confined Mode

CONFINED mode is very much l
input moves out of the defined ac
outside the region are truncated t
area. This means a stroke will nc

4.1.2.3 Continuous Mode

Drawing on CONTINUOUS type
x and y coordinates to the applic
panel. With this type of input, sof
The last input point will be a set c
of area must ensure the soft inte
very significant.

4.2 Creating an Active Ar

STATUS **ActiveAreaEnable**
S16 *ySrc*, S16 *x*

This tool creates a new active are
the new area to the caller. Once
returned to the application.

The argument type is used to sp
required. Mode specifies the inpu

Example 4-1 Create an acti

```
49    U32   nextWinId;/*  NextWin'
    .
```

Part II Writing PPSM Applica

Chapter 4 Pen Input Ha

The Pen Input Tools ena

- define active are
- control ink echoi

Refer to *Section 6.2.2 - S*

4.1 Active Area

Active area provides an
samples from the touch
constantly. PPSM uses i
utilization.

An active area is defined
interrupt messages are g
An example of this is an

Active areas are only "ac
the corresponding pen u
"B", and one of its sub-ta
"C" and "D". When a use
active areas "A" and "B",
moves across them. By t
"D" will receive message

If active areas are overla
receive pen messages.

There are two types of a
have three different mod

Type	
ICON_AREA	N/A
INPUT_AREA	STRO
	CONT
	CONI

4.1.1 Icon Area

Icon area is for the purpo
either from a pen-down
generates a soft interrup

```
114   if ( CtrlIconEnable(&abortRc
115        != PPSM_OK )
116     return (PPSM_ERROR);
```

4.10 Removing a Control A

STATUS **CtrlIconDisable**(U3

Remove the control icon from PF
no longer generate icon interrupt

4.11 Push Active Area List

STATUS **ActiveListPush**(vo

PPSM maintains a stack for stori
the active areas that have been c
and a new list begins. A new list
area created will belong to the ne

This tool allow applications to cre
original active areas when the te

Example 4-7 Push active ar

```
/* Push all existing active ar
If (rv = ActiveListPush())
{     /* error */
      return (rv);
}

/* create new set of active ar
GenerateNew();

/* Call subroutine */
ScratchPad();

/* restore original active are
if (rv = ActiveListPop())
{     /* error */
      return (rv);
}
```

4.12 Pop Active Area List t

STATUS **ActiveListPop**(voic

PPSM maintains a stack for stori
least recently pushed in active ar
tool must be called after an Activ
Otherwise, an error will be return

```
    .
    .
104   /* Create an active
105    * as specified belo
106    */
107
108   if (ActiveAreaEnable
109                       NE
110       != PPSM_OK)
111     return PPSM_ERROR;
112
```

4.3 Removing an Ac

STATUS **ActiveArea**

Removes a valid active a
tool must be a valid activ
ActiveAreaEnable tool. T
area.

Once removed, the regio
identifier will no longer re

Example 4-2 Remov

```
74    U32     backId;
    .
    .
    .
262               if ( Acti
263               return (P
```

4.4 Suspending an A

STATUS **ActiveArea**

Activate or deactivate a v
it can be suspended from

To suspend, call this tool
active area no longer se
writing in the region.

To re-enable the active a

Example 4-3 Suspen

```
if ((rv = ActiveAreaSu
{     /* error */
      return (rv);
}
```

Example 4-4 Re-enal

```
if ((rv = ActiveAreaSu
{     /* error */
      return (rv);
}
```

4.5 Active Area Enquiry

STATUS **ActiveAreaRead**(U
xDest, P_S16 *yI*

Given a valid active area identifie
of the active area.

Example 4-5 Enquire coord

```
85    U32     id, size;
86    P_U16   inData;
87    S16     xSrc, ySrc, xDest, y
    .
    .
    .
180             ActiveAreaRead(id,
```

4.6 Put Active Area to Fro

STATUS **ActiveAreaToFron**

Given the active area identifier, t
area linked list and insert the ele

Once the element is at the front
input if there are other active are

4.7 Pen Echoing

STATUS **AreaEchoEnable**(U

STATUS **AreaEchoDisable**(

For input active areas, echoing c
enabled when input active areas
valid active area identifier genera

4.8 Pen Color and Pen Siz

STATUS **PenEchoParam**(U

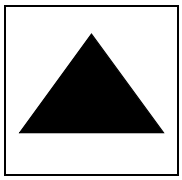
PPSM allows the application dev
color. This tool only sets the echo
other applications within the syst

4.9 Creating a Control Ac

STATUS **CtrlIconEnable**(P_

A pre-defined group of direction

purpose of scrolling. The
to the icon area. Two set



F

An identifier is returned t

Icon Type
PPSM_ICON_8_UP
PPSM_ICON_8_DOWN
PPSM_ICON_8_LEFT
PPSM_ICON_8_RIGHT
PPSM_ICON_8_DONE
PPSM_ICON_16_UP
PPSM_ICON_16_DOWN
PPSM_ICON_16_LEFT
PPSM_ICON_16_RIGHT
PPSM_ICON_16_DONE

Example 4-6 Create

```
80    U32       sendButto
    .
    .
    .
102   /* create control bu
103   if ( CtrlIconEnable(
104        != PPSM_OK )
105     return (PPSM_ERROR
106
107   if ( CtrlIconEnable(
108        != PPSM_OK )
109     return (PPSM_ERROR
110
111   if ( CtrlIconEnable(
112        != PPSM_OK )
113     return (PPSM_ERROR
```

```
                break;
            /* Close keyboard
            case IRPT_ICON:
                if (id == C
                    Clos
                    breal
            /* no more interr
            default:
                break;
        } /* switch */
    } /* while */
```

5.2 Handwriting Recognit

The handwriting recognition inpu
row by column format layout (refe

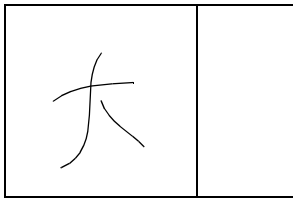


Figure 5-3 An Example I

It serves as an interface betweer
recognition engine. It captures th
handwriting input, and passes the
processing. (Refer to *Figure 5-4*
through the input pad).

Chapter 5 Character In

PPSM supports two type
input from the user. Ther
numeric input, and an inp
support any coded langu

This chapter describes th
Input Tools provided by F

5.1 Soft Keyboard

A default QWERTY soft
any position within the pa
for upper case letters an
letters and symbols (refe
layouts by pressing one

1	2	
Q	W	
A	S	
SHIFT	Z	

Figure 5

!	@	
q	w	
a	s	
SHIFT	z	

Figure 5

As an alternative, an use
column and row in numb
defined return keycode a

For both of the above keyboards,
S16) of the pen on the key is also

Only one soft keyboard can be o

5.1.1 Starting Soft Keyboard Cl

STATUS **OpenSoftKey**(U16

OpenSoftKey() opens the soft, o
keyboard (as shown in *Figure 5-*
upper-left corner position specifie
PPSM saves the display area cov
keys automatically. The soft keyb

When the user presses a key on t
be returned to the calling applica
application calls IrptGetData(). (F
IRPT_KEY interrupt message is
ASCII code returned is of type TE
byte.

Example 5-1 Open soft keyl

```
118   /* open soft keyboard for in
119   if ( OpenSoftKey(KEYBD_X, KE
120   return (PPSM_ERROR);
```

STATUS **AdvOpenSoftKey**(
keyHeight, U16
bitmap)

AdvOpenSoftKey() opens the so
configurable details.

- Location of the soft keyb
- Width and height of the k
- Number of rows and colu
- The return code of each
- The bitmap user interfac

The return codes are defined in a
array is from top left key across t
bottom right key. The contents of
AdvOpenSoftKey() has been call
or it must fit to cover the entire sc
this soft keyboard must be (keyW
number of pixels. For the NULL p
screen.

Example 5-2 Open soft keyl

```
/*  7, 8, 9, 4, 5, 6, 1, 2, 3, *, (
static const U16 keyMap[] = {55, 56
```

```
/* open user specified sof
/* with 10x10 key size and
/*  7 8 9  */
/*  4 5 6  */
/*  1 2 3  */
/*  * 0 #  */
if ( AdvOpenSoftKey(KEYBD_
    return (PPSM_ERROR);
```

5.1.2 Auto-Key-Repeat

Continually pressing a ke
first and second returned
to be 20. The time durati
AUTO_REPEAT_RATE.
tick. For example, a 32Hz
sec, then, for every 5/32

5.1.3 Terminating Soft Ke

STATUS **CloseSoftK**

CloseSoftKey() closes th
or AdvOpenSoftKey(). P
soft keyboard automatica

5.1.4 Suspend Soft Keyb

STATUS **ActiveListI**

STATUS **ActiveListI**

ActiveListPush() pushes
current task into backgro
and soft keyboard of the
that is currently being us
list. For more details, ple

Example 5-3 Display

```
if (rv = OpenSoftKey(5
{     /* error */
      return (rv);
}
while (running)
{
      switch(IrptGetD
      {
            /* Any ke
            case IRPT
                  /*
                  Di

                  /*
                  x=
                  y=
                  Di
```

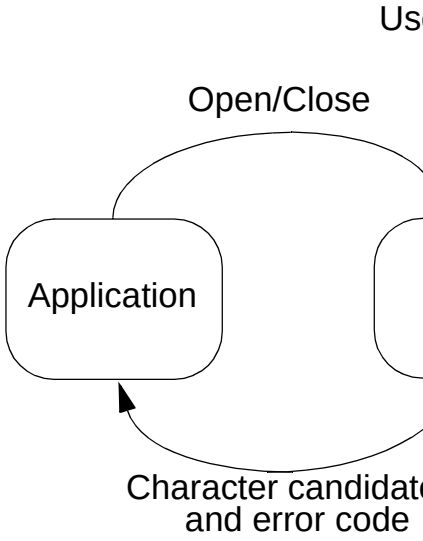


Figure 5-4 Data f

5.2.1 The Input Pad Mecl

The user writes a charac
system proceeds to reco
different box, or when a
whichever occurs first. T
entered, independent of
mechanism to determine
(e.g. the user clicks on a

The input pad is a subtas
messages, IRPT_HWR,
recognized. Each individ
IRPT_HWR interrupt me
returns the recognized c
input pad. (Refer to *Sect*

Only ONE instance of th
parent and sub-tasks). If
attempt to open the input
fail. If the user switches t
in the current task. The h
area is cleaned if the *are*

5.2.2 Starting Handwritin

STATUS **OpenInput**
U16 *are*

An application can open

application needs to specify the x
pad, the number of rows and colu
box (in units of pixel). If the input
and that the specified layout fits v
the specified location ready for u

The area of the panning screen c
function is called. Any changes t
function is called will not be reco

The default length of timeout for

STATUS **AdvOpenInputPa**
numCol, U16 are
echoWidth, U32
stackSize)

AdvOpenInputPad is similar to th
configurable details. It allows the

- position of the input pad
- number of rows and colu
- the width and the height
- the echo ink colour and
- the length of time out aft
- the sampling rate of the
- if the system should clea
  character is written
- the stack size for the inp

5.2.3 Terminating Handwriting

STATUS **CloseInputPad**(voi

CloseInputPad() closes the input
OpenInputPad() or AdvOpenInpu
recognition messages will be ger
original image covered by the inp
flag has been set to be 1(TRUE)

Example 5-4 Display charac

```
/* open an input pad at locatio
      being 64x64 pixels each
if (rv = OpenInputPad(50, 50,
{     /* error */
      return (rv);
}
while (running)
{
      switch(IrptGetData((P_U3
      {
            /* Any icon press
            case IRPT
            /* Code h
      recognized char
            DisplayKe
            break;

      /* Close keyboa
      case IRPT_ICON:
            if (id ==
                  Cl
            break;
      /* no more inte
            break;
} /* switch */
} /* while */
```

6.2.2 Screen Resolution

LCD display screen resolution is
hardware. This is normally a fixe
panel resolution must be equal t
cannot point at every pixel of the

6.3 Sample LCD Display S

The following two figures show e
hardware platform.

The LCD Display Screen area ha
and 320 pixels wide by 200 pixel

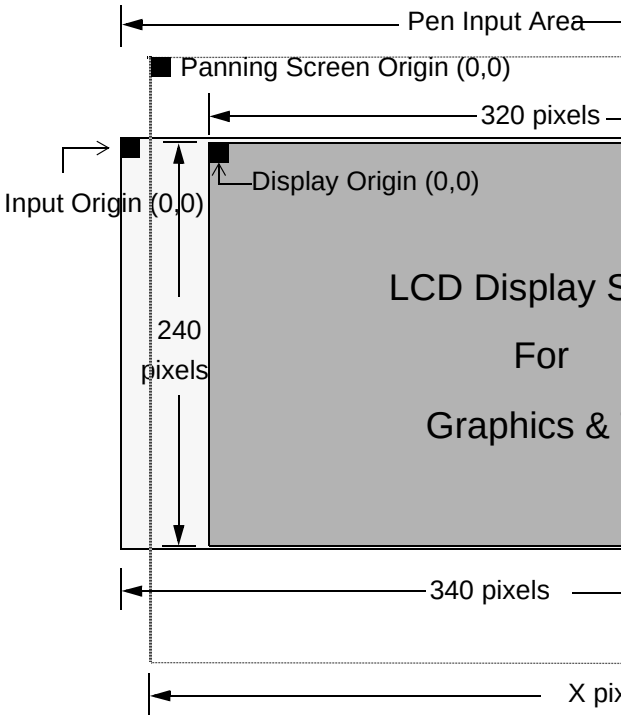


Figure 6-2 320x240 LCD Pane

Chapter 6 Using Graph

PPSM supports LCD mo
hardware cursor, hardwa

The Graphics Tools enab

- draw lines and s
- display and man
- swap bitmap ima
- control and resto
- control hardware
- set grey levels
- get information a
- change the pann
- direct all drawing area
- hardware cursor
- allocate memory
- set dot width in p arc and vector
- set pattern fill m
- draw vector by c

This chapter will describe
hardware cursor control

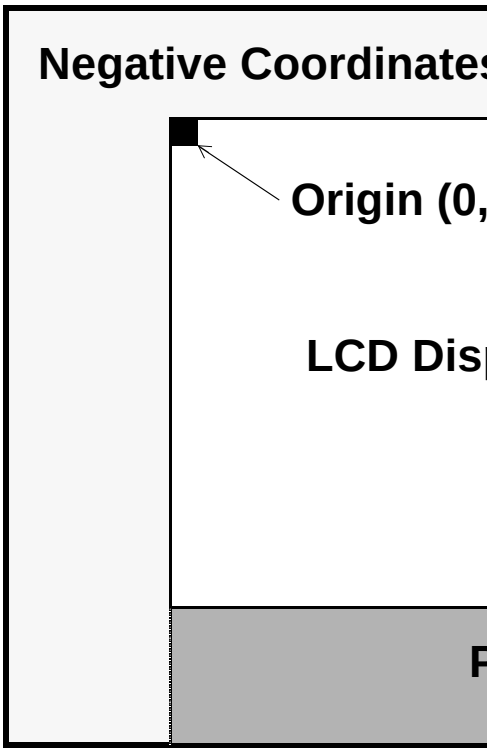


Figure 6-1 Generic S

6.1 Display Screen Forma

Figure 6-1 shows the general scr
three major areas:

- Pen Input Area
- LCD Display Screen
- Panning Display Screen

6.1.1 LCD Display Screen

The Display Screen is the region
display output data. Its size will d
200 pixels high or 320 pixels wid

The LCD module is capable of 1
grey levels or 4 grey levels respe
1 bit per pixel and WHITE, LIGHT

pixel.

The reference coordinate
corner, the Display Origi
associated with the LCD

6.1.2 Panning Display Sc

The Panning Screen is a
is to allow applications to
Although applications ca
screen unless this area i
areas on the panning scr
with the LCD display scr

Panning Screen has a si
different origin. It is confi

- The maximum m
byte.
- In 1 bit per pixel
divisible by 16 a
per pixel mode, t
maximum of 204
- The height of the
that is available

6.2 Screen Initializat

Because every hardware
procedure is required wh
calibration of the alignme
screen.

Screen initialization is im

Example 6-1 Initializ

```
57  main()
58  {
    .
    .
    .
62    /* Initialize PPSM w
63    PPSMInit(TRUE);
```

6.2.1 LCD Display Scree

The touch panel can be l
tools will only return LCD
outside the LCD display
or even greater than the

PutRec(), and ClearRec().

The following tables show the gr
is applied.

X is the existing grey level on the
and R is the final grey level on sc

AND_STYLE

Table 6-6 R = X

X	Y
00	00
01	00
10	00
11	00
00	01
01	01
10	01
11	01
00	10
01	10
10	10
11	10
00	11
01	11
10	11
11	11

OR_STYLE

Table 6-7 R =

X	Y
00	00
01	00
10	00
11	00
00	01

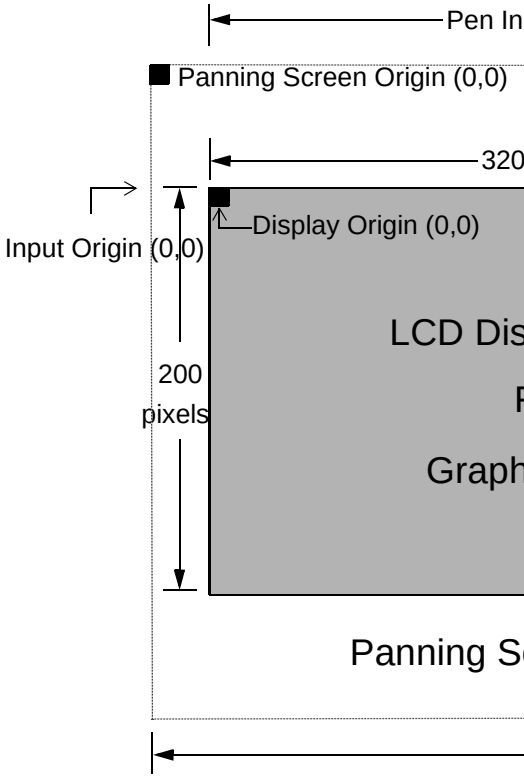


Figure 6-3 320x20

6.4 1 bit-per-pixel Gr

The graphics routines wi
panning screen.

The byte and bit within b

1 0 0

BLACK (

The above image will be
0x8A88 in hexadecimal.

6.4.1 Drawing Operators

PPSM supports drawing operato
DrawLine(), DrawCircle(), DrawE
PutRec(), and ClearRec().

The following tables show the gr
is applied.

X is the existing grey level on the
and R is the final grey level on sc

AND_STYLE

Table 6-1 R = X

X	Y
0	0
1	0
0	1
1	1

OR_STYLE

Table 6-2 R =

X	Y
0	0
1	0
0	1
1	1

EXOR_STYLE

Table 6-3 R = X

X	Y
0	0
1	0
0	1
1	1

REPLACE_STYLE

Tabl

X	
0	
1	
0	
1	

INVERT_STYLE

Table 6

X
0
1

6.5 2 bits-per-pixel G

The graphics routine will
DARK GREY and BLAC

The byte and bit within b

11 01 00 0

BLACK (1

LIGHT GR

The above image will be
in binary and 0xD08CC4

6.5.1 Drawing Operators

PPSM supports drawing
DrawLine(), DrawCircle()

6.9 Get Panning Screen W

U16 **GetLogicalX**(void)

GetLogicalX() returns to the calle
the current application.

The panning screen width is dyn
recommended to use this tool to
purposes.

6.10 Get Panning Screen H

U16 **GetLogicalY**(void)

GetLogicalY() returns to the calle
the current application.

The panning screen height is dyr
recommended to use this tool to
purposes.

6.11 Set Pattern Fill

STATUS **SetPatternFill**(U16
fillSpace)

This routine allows application pr
These settings include the patter
the background grey level, and th
called, the settings will be applie
DrawEllipse(), and DrawArc().

The pattern will be drawn with th
DrawRec(), DrawCircle(), DrawE

The argument *fillSpace* lets appli
between the pattern lines. The si

There are 8 fill patterns available

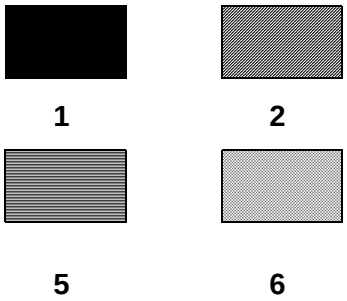


Table 6

X	
01	
10	
11	
00	
01	
10	
11	
00	
01	
10	
11	

EXOR_STYLE

Table 6-8

X	
00	
01	
10	
11	
00	
01	
10	
11	
00	
01	
10	
11	
00	
01	
10	
11	

REPLACE_STYLE

Table 6-9 R

X	Y
00	00
01	00
10	00
11	00
00	01
01	01
10	01
11	01
00	10
01	10
10	10
11	10
00	11
01	11
10	11
11	11

INVERT_STYLE

Table 6-10 R =

X	
00	
01	
10	
11	

6.6 Graphics Tools

The following sections explain ea

Top left corner of the panning scr
Top left corner of the LCD displa

origin. All co-ordinates gi
screen origin. There is n

Some of the graphics rou
argument in calling the r
drawing 5 dots with spec
Those graphics routines
DrawVert(), DrawLine() a

DrawDot(), DrawHorz(),
DrawEllipse(), DrawArc()
which is described above

For all the following exam
LCD display screen size
(50, 50) from panning sc

6.7 Get LCD Display

U16 **GetDisplayX**(v

GetDisplayX() returns to
LCD display panel being

When writing an applicat
numbers for the width of
flexible to run on differen

6.8 Get LCD Display

U16 **GetDisplayY**(v

GetDisplayY() returns to
LCD display panel being

When writing an applicat
numbers for the height o
run on different LCD pan

Example 6-2 Get LCI

```
362 STATUS DrawTextIcon(P_
363              U16 font,
364 {
365
366     U16     xDest, yDe
    .
    .
    .
375     /* Check to see if
376     if ( (xSrc < 0) ||
377        (yDest >= GetDi
378       return PPSM_ERRO
```

equals to FALSE to
screen to the task's

Example 6-5 Use ChangePa panning screen

```
#include <ppsm.h>
#include <errors.h>
#include <proto.h>

PAN_SCREEN newScreen;

STATUS TestApp()
{
  /* use the common panning sc
  ChangePanning(&newScreen, FA

  /* draw a circle with center
  DrawCircle(BLACK, 100, 100,

}

main()
 {
  U32 taskId;
  P_U8 newPanning;

  PPSMInit(FALSE);

  /* create a common panning s
  newPanning = (P_U8)GetScreen

  newScreen.panAddress = (U32)
  newScreen.displayScreenAddr
  newScreen.horzSize = 640;
  newScreen.vertSize = 400;
  newScreen.displayXOrigin = 0
  newScreen.displayYOrigin = 0
  newScreen.regPOSR = 0;
  newScreen.regPSW = 80;/* as
                          PSW

   /* destroy the system panning
       */
  ChangePanning(&newScreen, FA

  /* clear whole panning scree
  ClearScreen(WHITE);

 /* draw a circle with center a
      screen */
  DrawCircle(BLACK, 100, 100,

  /* create a new task with no
  AdvTaskCreate(&taskId, (P_VOI

  TaskStart(taskId);
 }
```

In the example above, both the s
the same panning screen. So wh
system task or task TestApp() wi

If a panning screen is shared am
screen in one of the task can als

The pattern fill mode 0 w

6.12 Set Dot Width

STATUS **SetDotWid**

After this routine is calle
DrawDot(), DrawHorz(),
DrawEllipse(), DrawArc()

If the dot width is larger t
thick arc and thick vecto

Example 6-3 Set dot

```
160   * Drawing1 - Draw a L
161   *            Both hav
162   *            drawn as
      .
      .
      .
170    SetDotWidth(6, 0);
171    SetPatternFill(2, WH
```

6.13 DisplayMove

STATUS **DisplayMo**

This function is to set the
screen. It sets the displa
function is called, the ne

6.14 Direct All Graphi

STATUS **ChangeWi**
P_U16 *

Run-time computation in
Users may see the graph
ChangeWindow() allows
routines to an off-screen
on the LCD display scre
generated, it can be disp
that the image is display

ChangeWindow() assum
values are supposed to b

Example 6-4 Use Ch

```
U32 oldScreen1, oldScr
U16 oldWidth1, oldHeig
```

```
U16 oldWidth2, oldHeight2;
U32 tempScreen = (U32)GetScree

/* Direct all graphic routine
   tempScreen with width 64 pi
ChangeWindow(tempScreen, 64, 6

/* This may be any routine for
   be drawn or drawing anythin
DoCalculation();

/* Direct all graphic routine
ChangeWindow(oldScreen1, oldWi
      &oldHeight2);
```

6.15 Change Panning Scre

STATUS **ChangePanning**(P
P_PAN_SCREE

ChangePanning() allows applica
another memory area during run
panning screen is still needed or

P_PAN_SCREEN is a pointer to
elements:

Table 12-11

Name
panAddress
horzSize
vertSize
displayXOrigin
displayYOrigin

Table

Name
displayScreenAddr
regPOSR
regPSW

Upon start-up, *panAddre*
displayXOrigin and *displ*
bits per pixel display or *h*

When DisplayMove() is c
modified so that the syst
position within the word t

When ChangeWindow()
changed so that all grapl
and image processing.

If all or several tasks in a
memory, the following st

1) In main()
   - call GetScre
     panning scre
   - call Changel
   - for each tas
     AdvTaskCre
     the task with
2) for each corresp
   screen
   - during initial

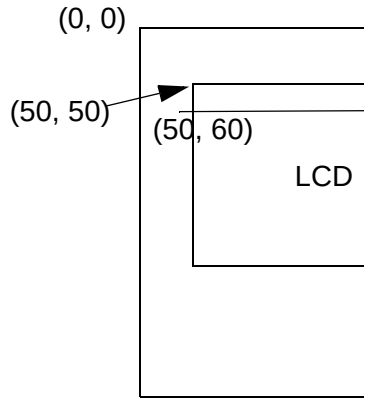


Figure 6-9 Sc

Example 6-12 Draw a thick

```
STATUS ret;

/* set dot width to 4 */
ret = SetDotWidth(4, 0);

if (ret !=PPSM_OK)
return ret;

/* draw a black horizontal lin
ret = DrawHorz(BLACK, 60, 60,

if (ret != PPSM_OK)
return ret;
```

In the above example, a thick ho

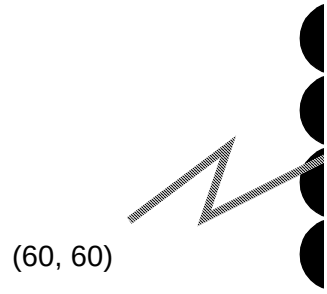


Figure 6-10 Scre

6.19 Draw a Vertical Line

STATUS **DrawVert**(U16 *grey*
dotLine, U16 *sty*

This routine will draw a vertical li

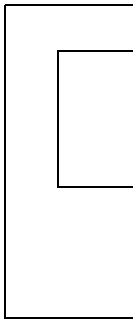
6.16 Fill the whole Pa

STATUS **ClearScree**

This routine will fill the w

Example 6-6 Fill the

```
STATUS ret;

/* fill the whole pann
ret = ClearScreen(WHI
```



Figure

6.17 Draw a Dot

STATUS **DrawDot**(U

This routine will draw a d

If dot width is 1, a pixel w
be drawn with top left pix
the dot width is greater th
value of (dot width - 1)/2
ordinate, (*xPos*, *yPos*).

Example 6-7 Draw a

```
STATUS ret;

/* draw a dot at (52,
ret = DrawDot(BLACK, 5
```

The calling of DrawDot(B
panning screen. As the L
on screen is at (2, 2) in c
have a dot which is very

Example 6-8 Draw a

```
STATUS ret;
```

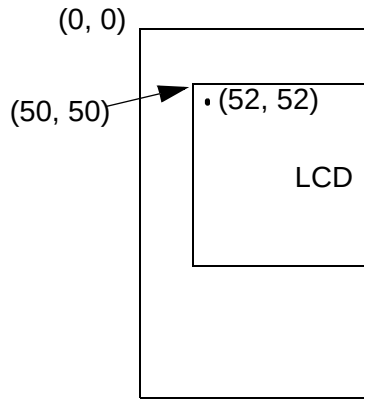


Figure 6-5 Sc

```
/* set dot width to 2 */
SetDotWidth(2, 0);

/* draw a dot at (52, 52) */
ret = DrawDot(BLACK, 52, 52, R
```

When the dot width equals 2, a s

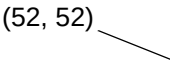


Figure 6-6 Sc

Example 6-9 Draw a dot wit

```
STATUS ret;

/* set dot width to 3 */
SetDotWidth(3, 0);

/* draw a dot at (52, 52 */
ret = DrawDot(BLACK, 52, 52, R
```

When the dot width is 3, a circula
drawn.

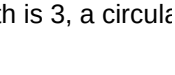


Figure 6-7 Sc

Example 6-10 Draw a dot w

```
STATUS ret;
```



```
/* set dot width to 4
SetDotWidth(4, 0);

/* draw a dot at (52,
ret = DrawDot(BLACK, 5
```

When the dot width is 4,
drawn.

(52, 5

Figure

6.18 Draw a Horizonta

STATUS **DrawHorz**(
dotLine,

This routine will draw a h

If the dot width is greater
truncated of (dot width -
lines below it. The length
2 pixels to the left of the
point.

If the width of the horizon

Example 6-11 Draw a

```
STATUS ret;

/* draw a black horizo
ret = DrawHorz(BLACK,
```

In this example, the dot w
0, REPLACE_STYLE) w
on panning screen. Only
display.

Example 6-17 Draw a recta

```
STATUS ret;

/* draw a black rectangle with t
      at (500, 400) */
ret = DrawRec(BLACK, 310, 250,
```

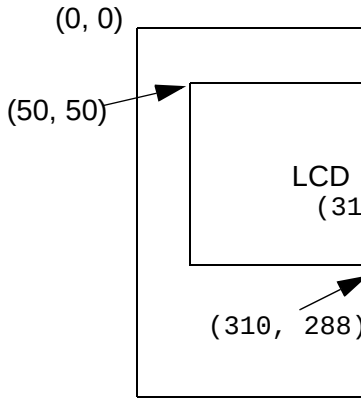


Figure 6-15 Sc

In this example, the dot width is 1
400, 0, REPLACE_STYLE) will d
and bottom right corner at (500, 4
horizontal line from (310, 250) to
(310, 289) will be seen on the LC

Example 6-18 Draw a recta fill pattern mode 1

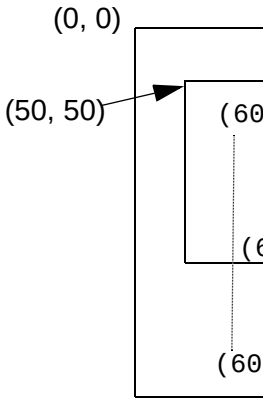
```
STATUS ret;

/* set dot width to 3 */
ret = SetDotWidth(3, 0);

if (ret !=PPSM_OK) return ret;

/* set pattern fill mode to 1
ret = SetPatternFill(1, WHITE,

if (ret !=PPSM_OK) return ret;

/* fill a rectangle from top l
ret = DrawRec(BLACK, 310, 250,
```

If the dot width is greater
truncated of (dot width -
at right. The height of ea
pixels above the source

If the height of the vertica

Example 6-13 Draw a

```
STATUS ret;

/* draw a black vertic
ret = DrawVert(BLACK,
```



Figure

In this example, the dot w
REPLACE_STYLE) will
screen. However, only th
which is (60, 60) to (60,
drawn in the form of 2 BL
pixels, and so on.

Example 6-14 Draw a

```
STATUS ret;

/* set dot width to 4
ret = SetDotWidth(4, 0

if (ret !=PPSM_OK)
return ret;

/* draw a black thick
ret = DrawVert(BLACK,

if (ret != PPSM_OK)
return ret;
```

In the above example, a thick ve

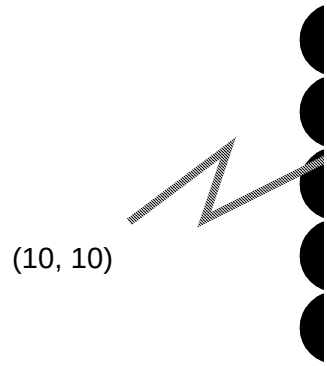


Figure 6-12 Sc

6.20 Draw a Line

STATUS **DrawLine**(U16 *gre*
yDest, U16 *dotL*

This routine will draw a line from

If dot width is greater than 1, eac
a thick square dot. Each of the th
truncated (dot width - 1)/2 pixels
pixels to the left and (dot width)/2
line is then generated by overlap

Example 6-15 Draw a black

```
STATUS ret;

/* draw a black line from (60,
ret = DrawLine(BLACK, 60, 240,
```

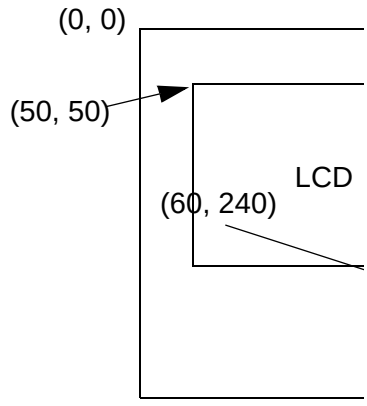


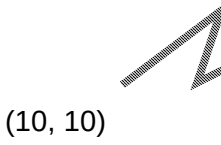
Figure 6-13 Sc

In this example, the dot w
470, 0, REPLACE_STYL
panning screen. Howeve
be seen.

Example 6-16 Draw a

```
STATUS ret;

/* set dot width to 4
ret = SetDotWidth(4, 0

if (ret !=PPSM_OK)
return ret;

/* draw a black thick
ret = DrawLine(BLACK,

if (ret != PPSM_OK)
return ret;
```

In the above example, a



Figure

6.21 Draw a Rectangl

STATUS **DrawRec**(U
yDest, U

This routine draws a rect
corner at (xDest, yDest).

If the dot width is greater
inside the rectangle and

If both fill pattern mode a
which is not covered by t

If fill pattern mode is set
rectangle border will be f

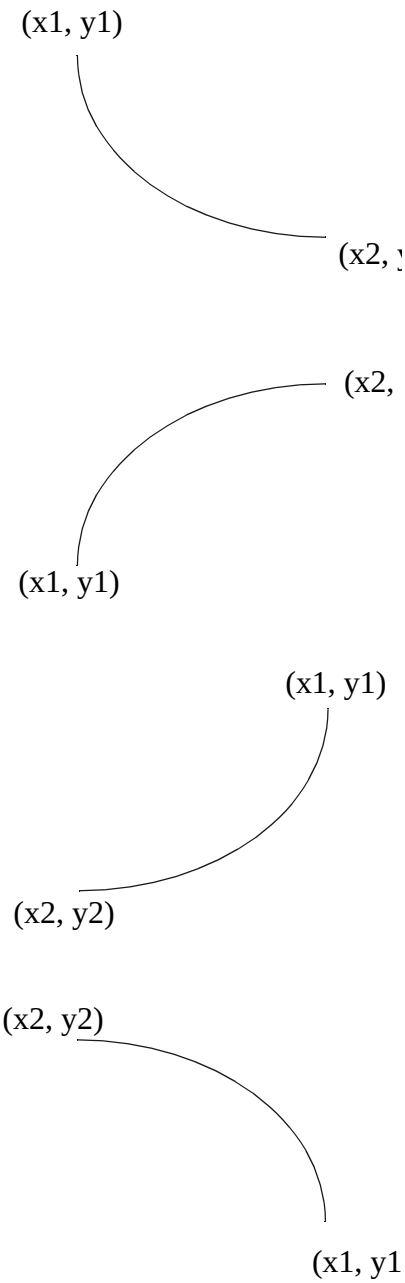
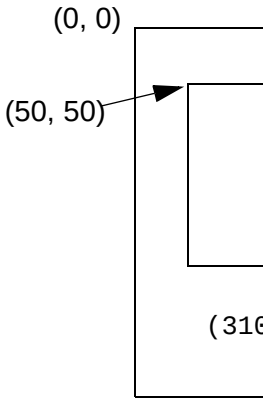


Figure 6-19 Cases

If the dot width is greater than 1, i
inside the arc and (dot width)/2 li

If both fill pattern mode and bord
not covered by the border of the

If fill pattern is set and border is o
filled.



Figure

In this example, the dot w
DrawRec(BLACK, 310, 2
with top left corner at (30
screen. However, only a
will be seen on the LCD

6.22 Draw a Circle

STATUS **DrawCircle**
U16 *styl*

This routine will draw a c
radius and grey level.

If the dot width is greater
inside the circle and (dot

If both fill pattern mode a
which is not covered by

If fill pattern mode is set
border will be filled.

Example 6-19 Draw a

```
STATUS ret;

/* draw a black outlin
ret = DrawCircle(BLACK
```

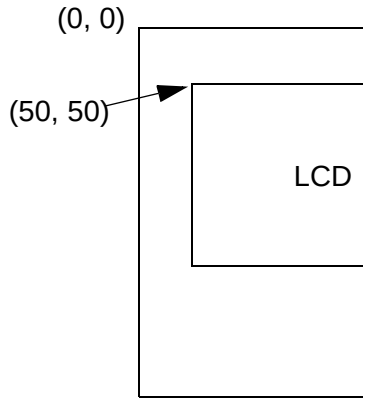


Figure 6-17 Sc

In this example, the dot width is 1
150, REPLACE_STYLE) will draw
As the circle is drawn outside the
LCD.

6.23 Draw an Ellipse

STATUS **DrawEllipse**(U16 *g*
xLength, U16 *yL*

This routine will draw a ellipse ce
size.

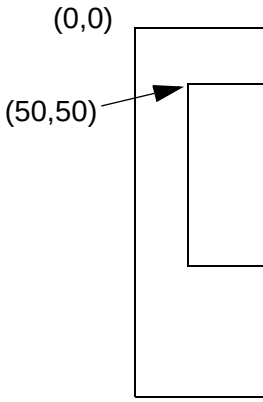
If the dot width is greater than 1, i
inside the ellipse and (dot width)/

If both fill pattern mode and bord
is not covered by the border will

If fill pattern mode is set and borc
border will be filled.

Example 6-20 Draw an ellip

```
STATUS ret;

/* draw an ellipse with center
      length 100 */
ret = DrawEllipse(BLACK, 560,
```

Figure

In this example, the dot w
150, 100, REPLACE_ST
longest distance on y ax
distance on x axis from c

6.24 Draw an Arc

STATUS **DrawArc**(U
style)

This routine will draw an

DrawArc() will draw a qu
DrawArc(BLACK, x1, y1,
will be drawn according t

The following will be seen on LCI

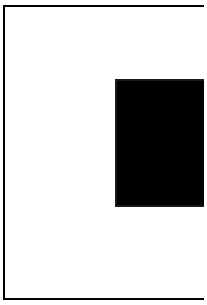


Figure 6-25 Sc

**Example 6-26 When LCD Di
ary of the panning scree**

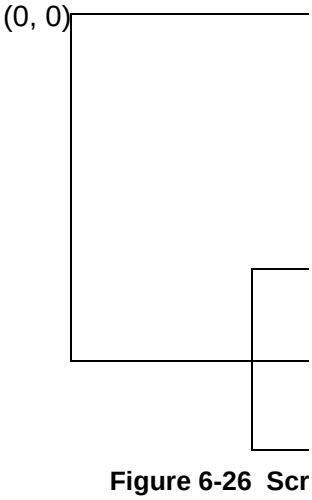


Figure 6-26 Scr

The following will be seen:

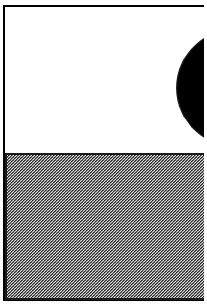
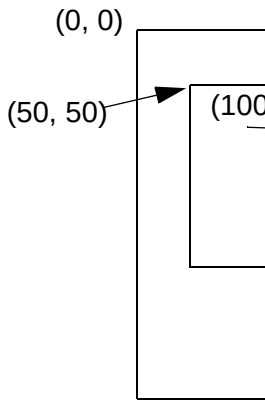


Figure 6-27 Sc

The pattern of the noise part of th
that follows the panning screen.
0, the noise will appear as a blan

Example 6-21 Draw a

```
STATUS ret;

/* draw an arc from (2
ret = DrawArc(BLACK, 2
```

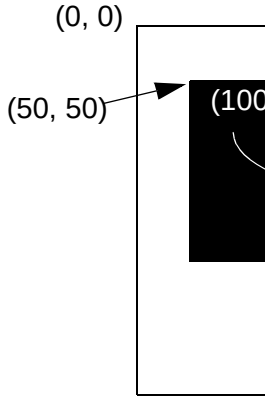


Figure

In this example, the dot v
50, OR_STYLE) will draw
The arc is actually a qua
distance of 141 pixels in
center is determined by t
of the first point which is

Example 6-22 Draw a

```
/* draw an arc from (1
ret = DrawArc(BLACK, 1
```



Figure

In this example, the dot v
the calling of DrawArc() i
a black background.

6.25 Draw a Vector from a

STATUS **DrawVector**(U16 *g*
pointPtr, U16 *sty*

This routine will draw lines to cor
whether the first point and last pc

6.26 Put a Rectangular Are

STATUS **PutRec**(P_U8 *bitm*
U16 *style*, U16 *r*

This routine puts an image from

PutRec() supports style such as
and AND_STYLE. There is error
the value of *bitmap.*

Example 6-23 Put a bitmap

```
/* put an image on panning scr
      height 480 */
ret = PutRec(bitmap, 0, 0, 640
```

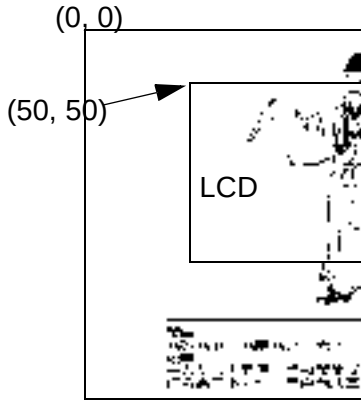


Figure 6-22 Sc

The calling of PutRec(bitmap, 0,
image from the memory area poi

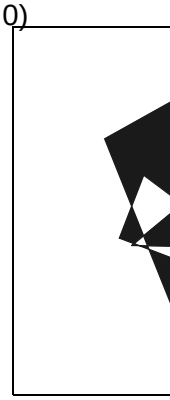
6.26.1 Special cases of Pu

The following are the few

Example 6-24 Displa setting

Two similar images will b

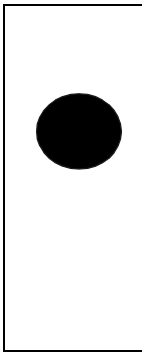
(0, 0)



Figure

Example 6-25 LCD D panning screen

(0, 0)



Figure

```
/* set hardware cursor width t
CursorInit(15, 15);

/* turn on the hardware cursor
CursorSetStatus(PPSM_CURSOR_ON
```

The above will create a cursor at
and will turn the cursor on.

6.31.2 Set Hardware Cursor Pos

STATUS **CursorSetPos**(U16

This routine will set the hardware

Example 6-32 When the har other position:

```
/* set hardware cursor positio
CursorSetPos(15, 150);
```

This will change cursor to new po

6.31.3 Set Hardware Cursor Stat

STATUS **CursorSetStatus**(U

This routine will change the hard
PPSM_CURSOR_OFF, PPSM_C
PPSM_CURSOR_REVERSED.

Table 6-1

Status name
PPSM_CURSOR_OFF
PPSM_CURSOR_ON
PPSM_CURSOR_REVERSED

Example 6-33 When hardwa it needs to be on with re

```
U16 x, y;

/* turn on hardware cursor in
CursorSetStatus(PPSM_CURSOR_RE
```

6.31.4 Get Hardware Cursor Stat

STATUS **CursorGetStatus**(

This routine will return the curren
of the following states: PPSM_C

memory is invalid, a bus

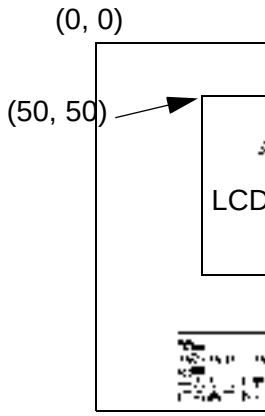
6.27 Save a Rectangu

STATUS **SaveRec**(P
U16 *res*

This routine saves an im

Example 6-27 Save a

```
/* save the portion of
      width 320 and h
ret = SaveRec(bitmap,
```



Figure

The calling of SaveRec(b
display image into memo

6.28 Exchange a Rec

STATUS **Exchangel**
rese

This routine exchanges i

Example 6-28 Save a

```
/* exchange the image
      and height 120
ret = ExchangeRec(bitm
```

This example swaps the
rectangular region from t
169). After this call, *bitm*
region (50, 50) to (369, 1
displayed on the rectang

6.29 Fill a Rectangular Are

STATUS **ClearRec**(U16 *grey*
height, U16 *style*

This routine fills an rectangular a

Example 6-29 Fill a rectang

```
STATUS ret;

/* fill a rectangular area with
      161 */
ret = ClearRec(BLACK, 300, 240
```

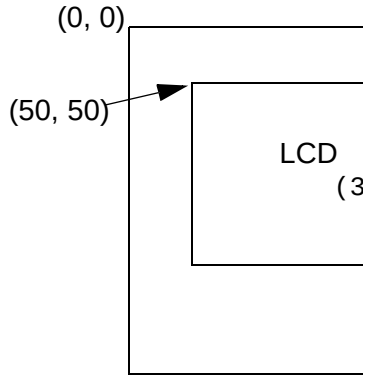


Figure 6-29 Sc

This example fills the rectangular
panning screen with width 261 pi

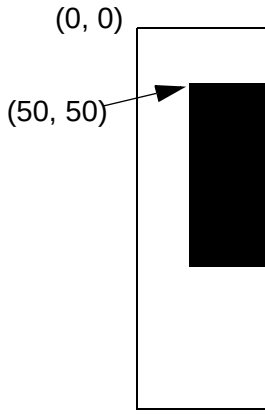
6.30 Inverse a Rectangular

STATUS **InvRec**(U16 *xSrc*, U

This routine will inverse the grey
at (xSrc, ySrc) and bottom right c

Example 6-30 Inverse a rec

```
STATUS ret;

/* inverse a rectangular area
      height 100 */
ret = InvRec(250, 200, 200, 10
```



Figure

In this example, the LCD
screen is white. After inv
200) and 200 pixels wide
screen at bottom right po

6.31 Hardware Curso

When there is no hardwa
requires to set the curso
CursorSetStatus(PPSM_

When hardware cursor is
should call CursorOff(). If
has to set the cursor cha
CursorSetStatus(PPSM_

When hardware cursor is
CursorSetStatus(PPSM_

A application can change
hardware cursor status fr
calling of functions to cha
immediate effect.

6.31.1 Set Hardware Curs

STATUS **CursorInit(**

This routine will change t
both width and height is

Example 6-31 When

```
/* set hardware cursor
CursorSetPos(150, 158)
```

Table 7-1 Predefined fie

Field Index	
DB_COMPANY	Co
1,2,3,4,5	5 a

7.1.2 Unformatted Data

Unformatted data is data that has only one unformatted data field a one of the following eight types c

- type 0 text (ASCII code
- type 1 decompressed F
- type 2 compressed PP
- type 3 mixed text and g
- type 4 reserved
- type 5 text followed by
- type 6 text followed by
- type 7 text followed by

PPSM will keep track of the size record information field.

7.2 The Database Manipu

PPSM provides a set of tools to c they can be grouped together in

1) Operate at the database
   - Add or delete a data
   - Inquire the total num environment.
   - Set or clear the data
   - Interrogate the datab
2) Operate at the individual
   - Add or delete a reco
   - Add a blank record t
   - Append a blank recc
   - Write or read formatt
   - Write or read unform
   - Get the ID of the firs
   - Get the ID of the pre
   - Search for a record u
   - Set or clear the reco
   - Interrogate the recor
   - Inquire the total num

PPSM_CURSOR_REVE

Example 6-34 When

```
U16 status;

/* get the hardware cu
CursorGetStatus(&statu
```

6.31.5 Set Hardware Curs

STATUS **CursorSet**

This routine will set the h of blinks per 10 seconds

6.31.6 Turn Hardware Cur

STATUS **CursorOff(**

This routine will turn off t hardware cursor again, t and follows by calling Cu

6.32 Display Other Re

STATUS **CursorSet**

STATUS **LCDScreen**

These two routines toget different regions of the p two routines must be use replaced by DisplayMove

Example 6-35 Displa

```
U16 x=50, y=50;

/* set the LCD display
CursorSetOrigin(x, y);

/* change the hardware
      top left corner
LCDScreenMove(x, y);
```

The LCD Display screen screen with top left corne

6.33 Get LCD Display

STATUS **CursorGet**

This routine will return th

panning screen.

Example 6-36 Get LCD disp

```
U16 x, y;

/* get the position of LCD dis
CursorGetOrigin(&x, &y);
```

6.34 Allocate memory for F

P_VOID **GetScreenMem**(U1

This routine will allocate memory
the size of the new panning scre
is available, it will return NULL.

Example 6-37 Allocate mem

```
P_U8 screenPtr;

/* allocate memory for panning
screenPtr = (P_U8)GetScreenMem
```


Chapter 7 Database Ma

PPSM supports global d
type may be formatted o

The Database Tools ena

- create and delet
- add and delete r
- store, retrieve ar
- search for partic
- get status about

This chapter gives some
getting up to speed fast.
power reset, all database
to save the database info

7.1 Data Format

PPSM database tools su

7.1.1 Formatted Data

Formatted data means te
in *Table 7-1*, will be used
predefined. Other Forma
need arise. When a new
seven by default. PPSM
record. The number of m
PPSM at compile time, c
Formatted Data, not Unf

There is a size limit of 60

Table 7-1 Predef

Field Index
DB_LAST
DB_FIRST
DB_HOME
DB_OFFICE
DB_ADDRESS
DB_FAX

database.

7.3 Creating and Ed

Whenever a record is cre
Application will need to u
particular record.

The following example ill
memory variables which
formatted data field in a

Note that user must have
calling DBAdd(). It is imp
access to the database

Note also that in the exa
creation of the next reco
because of the linked list
retrieve the record that w
ID. However, for those fr
memory variable to keep

Example 7-1 Create a
number of records

```
/* Database identifier
U32 gAddBkDBaseID;
      /* Data to be wr
TEXT grec1LName[5]={'A
TEXT grec1FName[4]={'K
TEXT grec1Phone[9]={'2
TEXT grec1Add[17]={'1'

      /* Data to be wr
TEXT grec2LName[8]={'A
TEXT grec2FName[4]={'J
TEXT grec2Phone[9]={'6
TEXT grec2Add[17]={'1'

    STATUS buildDBase(
    {
      STATUS ret;
      U32 recID;
      S32 numRec;

      ret = DBAdd(&gAd
      if(ret != PPSM_O
            /* Add Rec
       DBAddRecord(gAd
       DBChangeStdData
       DBChangeStdData
       DBChangeStdData
       DBChangeStdData

            /* Add Rec
       DBAddRecord(gAd
       DBChangeStdData
       DBChangeStdData
       DBChangeStdData
       DBChangeStdData
           /* Read bac
```

```
DBReadTotalNumberRecord
}
```

7.4 Searching and Retriev

The following example uses the
searches for a particular record u
programming practice to check tl
ensuring that a match is found be
operation.

Example 7-2 Search a databas retrieve record data

```
/* Global database identifier
U32 gAddBkDBaseID;
/* This is the search key */
TEXT grec1FName[4]={'K','e','n
    STATUS searchRec(void)
    {
      STATUS ret;
      U32 recID;
      /* Pointers to the forma
      P_TEXT tempLname, tempFn

      /* Search for a record i
         formatted data field
      ret = DBSearchData(gAddB

      if (ret != PPSM_OK) retu
      /* recID now is the reco
      /* We can now access the
      DBReadData(gAddBkDBaseID
      DBReadData(gAddBkDBaseID
      DBReadData(gAddBkDBaseID
      DBReadData(gAddBkDBaseID
      /* tempLname, tempFname,
         data field stored in
      } /* end searchRec() */
```

7.5 Navigating along a Re

The following example uses the
sequentially. It is assumed that a
already existed. Initially, the first
while loop is set up to access rec
is by using the botFlag passed to
set to 1, it signifies that the curre

Example 7-3 Use the record lis tially

```
/* Global database identifier
U32 gAddBkDBaseID;

 STATUS seqAccessofRec(void)
 {
    U32 RecId,nextRecId;
    S32 srchToken = 1;
    U16 botFlag = 0;
    P_TEXT stdDataOut;
    STATUS ret;

    /* Get the first r
    ret = DBGetFirstRe

    /* Check if the da
    if (ret == PPSM_EF

    /* Read back data
    DBReadData(gAddBkD
    .
    /* Do something he
    .
    /* Get next record

    while(srchToken)
      {
      DBGetNextRecID(
       /* The RecId p
      if(botFlag == 1
      /* Read back da
      DBReadData(gAdd
      .
       /* Do somethin
       .
       /* Continue se
       RecId = nextRe
       }   /* end whi
}    /* end of seqAcc
```

8.4.2.1 Text output style

The output style defines an opera
image at a specified display loca
bitmap can replace, OR with, AN
replace the existing image.

Table 8-1 Supported

Output Styles	
REPLACE_STYLE	Re
OR_STYLE	Or
AND_STYLE	An
EXOR_STYLE	Ex
INVERSE_STYLE	Inv

8.4.2.2 Text Grey Levels

Depending on the hardware syst
supported. For details about the
Graphics and *Section 6.5 - 2 bits*

Table 8-2 Supported

Grey Level Values	
BLACK	Bla
DARK_GREY	Da
LIGHT_GREY	Lig
WHITE	Wh

8.4.3 Setting Font Attributes

STATUS **TextSetFont**(U32 *t*

Font attributes includes font type
(e.g. outline) that's related to a p

A font attribute data structure, ca
desired values in order for these
attributes will take effect when te
these new values set. The follow
ppsm.h:

```
typedef struct
{
      U16         type;
      U16         width;
      U16         height;
      U16         attrib;
```


Chapter 8 Text Display

Applications must map th
an area on the display sc
seen. This chapter descr
the display of text on the

PPSM supports 16-bit te
coded languages. The d
Asian and English chara
scalable and bitmap font
fonts with size 8 x 10 an
ISV.

8.1 Text Representa

Both ASCII and Asian ch
TEXT) system-wide. For
Chinese, both GB and B

The most significant byte
zero-extended ASCII an
most significant byte whi
byte.

8.2 Text Display Are

Text can be displayed ar
2.3.2.2 - Panning Screen
row by column format on

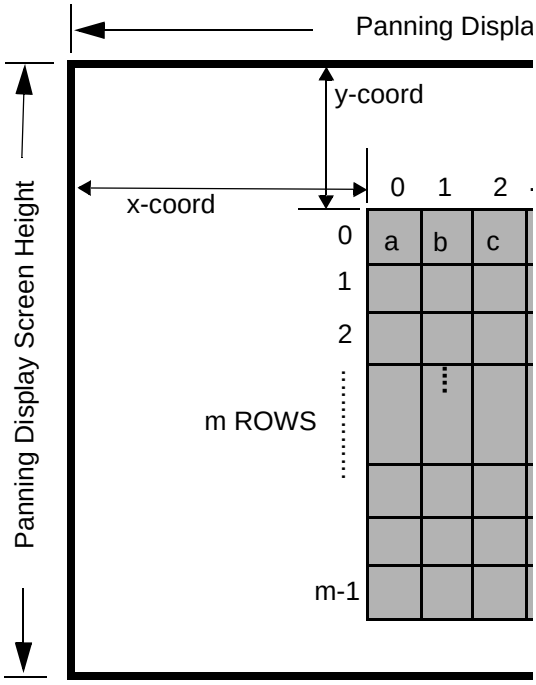


Figure 8-1 A Text Displa

8.3 Text Templates

A text template refers to a collect
displayed. These text properties
style, coordinates and size of the
cursor. These text templates are
flexibility for applications to chan
efficient manner. Applications ca
discretion on an as needed basis
indicator showing where the text

8.3.1 Creating text templates

STATUS **TextCreate**(P_U32

A text template needs to be crea
unsigned 32-bit text template ide
template created. This text templ
created text template. Refer to *T*
properties when a text template i

Example 8-1 Create a text t

```
336   U32 tId;  /* textId for the
337
338   if(TextCreate(&tId) != PPSM_
339     return PPSM_ERROR;
340
```

8.3.2 Deleting text templ

STATUS **TextDelete**

When a text template is
up space that is being us
returned by TextCreate()

Example 8-2 Delete a

```
352   /* Delete the text w
353   if(TextDelete(tId) !
354     return PPSM_ERROR;
```

8.4 Text Properties

Text properties describes
the panning display scre
template, include the pos
text characters, the font
the text display area. Re
of a text template when i

8.4.1 Setting Text Displa

STATUS **TextSetDis**
U16 *hei*

The text display layout o
boundary of the panning
the upper left corner, an
characters. The size of th
to the size of the selecte
the range of 0 and one le
can display.

In *Figure 8-1*, the text dis
columns in size. This tex
wishes.

8.4.2 Setting Text Outloo

STATUS **TextSetOu**

The text outlook of a text
to be displayed on the te
take effect on subsequen
modified.

8.6 Text character cursor

The character cursor position de
will be displayed next. This positi
specified in the given text templa
through one less than the size of

In *Figure 8-1*, the range of valid c
the current cursor position is 3 af

8.6.1 Setting the character curs

STATUS **TextSetCursor**(U3

Setting the character cursor posi
template to the given value. Subs
character cursor position.

Example 8-5 Set character

```
53  static U32 gTextId, gTmpTextId
    .
    .
    .
279           /* Clear the text on
280                TextUnmap(gText
281                TextSetCursor(g
```

8.6.2 Reading the character cu

STATUS **TextReadCursor**(U

Applications can inquire the curre
specified by a text template. The
will be displayed next.

Example 8-6 Set and read t

```
    U32   tId;         /* text tem
    TEXT  moto[] = {'M', 'o', 't',
          */
    U16   len;  /* # chars to be
    U16   curPos;/* cursor positic
    .
    .
    .
    /* create a text template */
    TextCreate(&tId);

    /* calculate # chars to be dis
    len = Strlen(moto);

    /* set up text properties. */
    TextSetup(tId, LARGE_NORMAL_FO

    /* set current character cursor
          */
    TextSetCursor(tId, len);

    /* display "Motorola" using th
```

```
                P_U8        bi
} FONTATTR, *P_FONTATT
```

8.4.3.1 Font Types

Eight font types are supp
English fonts. Large Nor
Normal is 16 x 16 Chines
as GB Normal (for backw
in BIG5 code format. BIG

Table 8-3 Suppor

Output Styles
SMALL_NORMAL_FONT
SMALL_ITALIC_FONT
LARGE_NORMAL_FONT
LARGE_ITALIC_FONT
GB_NORMAL_FONT
CHINESE_NORMAL_FO
BIG5_NORMAL_FONT
BIG5_VARIABLE_FONT

Note: Asian fonts a

8.4.3.2 Font Sizes

If a scalable font engine i
an application to specify
template. The minimum a
particular scalable font e

All the other font types a
and height of those font

8.4.3.3 Special font a

No other special font attr
modify the font attributes

reserved for future extensions ar

Table 8-4 Tex

Text Properties
(x,y)-coordinate of the origin (top lef text display area
Width of text display area in numbe
Height of text display area in numbe
Character cursor position relative to display area
Font type
Font width
Font height
Special font attributes
Text grey level value
Text output style

Example 8-3 Setting text pr

```
U32    tId;                /* te
                           /* te
TEXT   moto[] = {'M', 'o', 't',
                           /* tl
                              2
U16    len;                /* #
FONTATTR     fontAttr;     /* fe
.
.
.
/* create a text template */
TextCreate(&tId);

/* calculate # chars to be dis
len = Strlen(moto);

/* subsequent text displayed w
      length of moto character
      position zero. */
TextSetDisplay(tId, 102, 0, le

/* subsequent text displayed w
      the existing image on sc
TextSetOutlook(tId, EXOR_STYLE

/* set up font attributes */
fontAttr.type = BIG5_VARIABLE_
fontAttr.width = fontAttr.heig
fontAttr.attrib = 0;

/* subsequent text displayed w
      type of size 20x20 pixel
TextSetFont(tId, &fontAttr);

/* delete unused text template
TextDelete(tId);
.
.
.
```

8.5 Text Mapping

Mapping functions are pr
panning display screen a
template.

8.5.1 Displaying text

STATUS **TextMap**(U

The given text is displaye
character cursor position
described by the specifie

There is no word-wrap fu
characters of a word that
text display area. Text dis
end of the text display ar
mapped, or when *numC*
on panning screen, chara
available position, or (the
at the end of the templat
without returning any err

Example 8-4 Display

```
333 STATUS  DisplayString(
334 {
335   U16 len;  /* length
336   U32 tId;  /* textId
    .
    .
    .
341   /* get the string le
342   if (len = Strlen(str
343     {
344       /* Set up the te
345       if(TextSetup(tId
346        return PPSM_ERR
347
348       /* Map the text
349       if(TextMap(tId,
350       return PPSM_ERRO
```

8.5.2 Removing text

STATUS **TextUnmap**

The unmapping of text m
and size of the text displ
template.

STATUS **AlarmReadId**(U32
P_U16 *hour*, P_

Read the coming clock alarm tim
alarm time by using AlarmReadI
zero.

9.4 Setting Clock Alarm

STATUS **AlarmSet**(U16 *year*
second)

STATUS **AlarmSetId**(P_U32
hour, U16 *minut*

Set the clock alarm. When the al
interrupt to the application that ca
IRPT_RTC. If AlarmSetId() is use
IRPT_RTC when time is reached

9.5 Clearing Clock Alarm

void **AlarmClear**(void)

void **AlarmClearId**(U32 *alar*

Clear the clock alarm in current t
using AlarmClearId(). PPSM will
the application.

9.6 Setting Periodic Alarm

STATUS **SetPeriod**(U16 *per*

STATUS **SetPeriodId**(P_U3

Set or clear the periodic alarm. P
application that calls this tool. Th
MC68EZ328 only. If SetPeriodId(
be returned. The period that is al

RTC_PERI_NONE	Dis
RTC_PERI_SECOND	Se
RTC_PERI_MINUTE	Mi
RTC_PERI_HOUR	Ho
RTC_PERI_MIDNIGHT	Mi

```
TextMap(tId, (P_TEXT)m

/* read current charac
      case) */
TextReadCursor(tId, &c
.
.
.
```

Chapter 9 Timer Manag

PPSM uses one DragonB
reference timer and the s
microseconds and the sy
tools are included for app
alarm, clock alarm and ti

PPSM manages all the ti
specific time-out or alarm
appropriate alarm time o
interrupt to notify the app

The system default time

Two types of interrupt me

- IRPT_RTC
- IRPT_TIMER

IRPT_RTC message is s

IRPT_TIMER message i

If the timeout or alarm ha
alarm task, the timer or a
current task are handled

9.1 Reading System

STATUS **DateTimeR**
P_U16 *

Read the system date ar

9.2 Setting System D

STATUS **DateTimeS**
U16 *sec*

Set the system date and

9.3 Reading Clock A

STATUS **AlarmRead**
P_U16 *

RTC_PERI_NO_SECOND
RTC_PERI_NO_MINUTE
RTC_PERI_NO_HOUR
RTC_PERI_NO_MIDNIGHT

9.7 Setting Timeout

STATUS **Timeout**(U

STATUS **TimeoutId**(

General time-out tool. PR
caller once the time-out
immediately disable the
timeout set by reference
which will be returned in

IRPT_TIMER is the inter
occurs.

9.8 Setting Input Tim

STATUS **InputTimeo**

Set the repetitive time-ou
routine is an explicit time

Once this time-out is acti
immediately after a valid
next pen input occurs, PI
otherwise, the time-out p
time-out because once a
after each pen input stok

IRPT_TIMER is the inter
returned in IrptGetData()
timer set by Timeout(), T

To cancel the input time-
argument.

9.9 Continuous Referenc

PPSM provides a continuous 32-
value wraps around about every
condition, making it transparent t
either a resolution of 1 millisecon
tools allow the user to make use
stamping and time-out.

Note that there are two sets of ti
100 microsecond resolution. The
sets of tools should NOT be mixe
tools cannot be used in the 100

The millisecond resolution timer
100 microsecond resolution tools

9.10 Read The Reference T

U32 **RefTimeRead**(void)

U32 **RefFineTimeRead**(void

Read the reference timer value.
representing the current referenc
RefTimeRead(), or in 100 micros

9.11 Set Reference Timer A

STATUS **RefTimeAlarm**(U3

STATUS **RefFineTimeAlarm**

STATUS **RefTimeAlarmId**(P

STATUS **RefFineTimeAlarm**

Set the alarm time, using the refe
tool. When using this tool, the in
generate an alarm interrupt to th
obtained by calling the respective
alarmId output from this function
time is reached.

9.12 Compute Reference T

U32 **RefTimeDiff**(U32 *begin*

U32 **RefFineTimeDiff**(U32 *b*

Compute the difference i
is in millisecond resolutio
RefFineTimeDiff().

Example 9-1 Timer U

```
U32     (*TimeRead)();

STATUS  (*TimeAlarm)(U32 a
U32     (*TimeDiff)(U32 be
.
.
STATUS  RefTimer(void)
{
    .
    .
    while ( 1 )
    {

          /* Initialize al

          gOldTime    = 0;
          gNewTime    = 0;
          gDiffTime   = 0;

          /* Set up time t
           * to start the
           */
          if (gUnit == MIL
          {
            /*  Assign th
              *  timer fun
              */
            TimeRead  = R
            TimeAlarm = R
            TimeDiff  = R
          }
          else if (gUnit==
          {
            TimeRead  = R
            TimeAlarm = R
            TimeDiff  = R
          }
          SetUnit(MILLI_S
          .
          if (*inData ==
          {
                .
                .
                if (id ==
                {
                      /*
                       *
                       *
                       */

                      gOl
                      gNe
                      gDi
                      Dis
                }/*if rea
                .
                .
          }
    }/*while*/
}/*RefTimer*/
```

are driven by the system clock, p
without sacrificing peripheral res

11.2 Power Modes

Figure 11-1 shows the state diag
defined in PPSM.

System Internal Modes:

- Initialization mode
- System mode
- Wake-up mode

Application Modes:

- Normal mode
- Doze mode
- Sleep mode

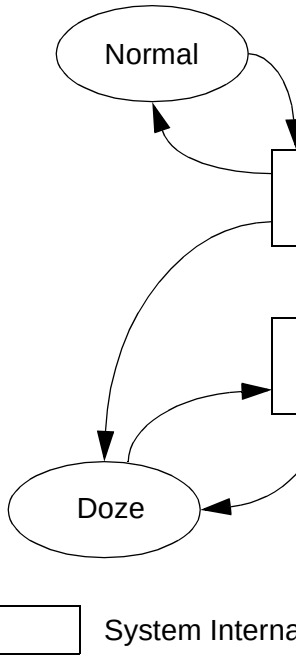


Figure 11-

11.3 System Internal Mode

Initialization, System and Wake-u

Chapter 10 Memory Man

In order for PPSM to ma
management tools provi
its own memory tools tha
allocate memory from th
PPSM is specified in the
Specification File).

*Note: For allocatin
allocation fu
Graphics To*

10.1 Allocating Memo

void ***Lmalloc**(U32 s

void ***Lcalloc**(U32 si

Memory can be allocated
a pointer to a block of av
returned to the caller is n
Lcalloc() is used. No aut
when used by the caller.
through Lmalloc() can be

If no memory is left in the

The actual size of memo
requested by user. A hea
memory management. N
use the required size of m
value is not NULL.

10.2 Freeing Memory

void **Lfree**(void *ptr)

When an application finis
memory can be recycled
into the system heap and

The pointer passed into t
Lmalloc() ,Lcalloc() or Lr

10.3 Reallocating Memory

void ***Lrealloc**(void *src, U32

Moving of memory. This routine r
system from one location to anot
content from the old location to tl
it back into the system heap. The
the system memory.

10.4 Copying Memory

STATUS **MoveBlock**(P_U32

Copying memory from one regior
area. It performs memory copy ir

10.5 Inquiring Memory

STATUS **TaskMemUsed**(U3

U32 **TotalMemUsed**(void)

U32 **TotalMemSize**(void)

S32 **TaskStackAvail**(void)

Memory allocated to the applicat
time. PPSM returns to the caller
the task with the given task ident
bytes of memory allocated to the
PPSM returns the number of byt
through Lmalloc(), Lcalloc() or Lr

PPSM returns to the caller the to
current task when calling TaskSta
has not been used up, negative v

User can inquire the size of the l
Lmalloc() with input flag LARGES

Chapter 11 Power Manag

PPSM utilizes the power
power management tools

The Power Management

- switch to one of
- control the duty
Normal mode
- switch automatic
idle
- control user defi
transition

Applications can choose

- control the syste
- use the PPSM's

By default, the system w
message waiting to be s
period and sleep period

11.1 Power Control M

PPSM makes use of the
efficiency. It allows the a
software control. System
the CPU via the PCM. B
the CPU core from a min
referred to as the CPU c

- While the CPU c
application, the
- While the CPU c
recognition appli
100% duty cycle

The PCM uses a period

- For example, wit
period of time, th
clock cycles), fol

Please refer to the MC68
PCM.

When using the PCM to
Phase Locked Loop rem

mode.

- If a Wake-up condition is
  time-out countdown for S
  countdown and return th

This automatic Sleep mode time-
mode until the Sleep mode perio
tion.

11.4.3.3 Waking up from Sle

Any one of the following internal
Sleep mode. The interrupts are:

- Pen Interrupt
- Real Time Clock Alarm,
- External Interrupts from
  and PWM

For Mid-night interrupt, system w
date and time and then go back t
system will wake up from Sleep r

11.5 Power Management T

11.5.1 Setting Duty Cycle

U16 **SetDutyCycle**(U16 *per*

This tool allows the application ta
mode. Applications within a syste
PPSM automatically changes the
becomes active.

11.5.2 Setting Doze Period

STATUS **SetDozePeriod**(U1

Sets the countdown period, in ur
mal mode to Doze mode. A value
going into Doze mode automatica
A value of zero will bring back the
setting is to go to doze mode wh
rent task to be handled.

11.5.3 Setting Sleep Period

STATUS **SetSleepPeriod**(U

Sets the countdown period, in ur

Applications need not be
are included in this chap
agement. All of these are
where PPSM takes contr
swapping, message pas

11.3.1 Initialization Mode

This is the power on or s
PPSM occur in this mode
PPSM is initialized, unles

11.3.2 System Mode

PPSM performs all of its
power module controlling
frequently invoked, only
ning, for example, to han
To minimize the actual ti
regardless of its set valu
leaves System mode, it w

11.3.3 Wake-up Mode

This is invoked either fro
Sleep mode, only interna
refer to *Section 11.4.2.3*
from Sleep for the Wake-

In Wake-up mode, the sy

- which of the inte
- where the interru
- which applicatio
- which mode the

For example, mid-night i

1) Go into Wake-up
2) Determine that t
3) Update the syste
4) Go directly back

11.4 Application Mod

Normal, Doze and Sleep
that an application sees

11.4.1 Normal Mode

In this mode, applications can ma
the CPU duty cycle value, please
Phase Locked Loop is on, all per
Application is actively executing

PPSM is designed as an event d
monitoring the calls to the systen

- When there are interrupt
with interrupt messages.
application accordingly.
- When there is no more i
message, IRPT_NONE i

11.4.2 Doze Mode

In this mode, the CPU is disable
ler, Real Time Clock, Timer and
other peripherals are disabled. S
CPU for more activities.

There are two ways for applicatic

- direct system call
- automatic time-out

11.4.2.1 Direct System Call

Application can go into Doze mo
refer to *Section 11.5.4 - Going In*
the system into Doze mode imm

11.4.2.2 Automatic Time-out

Application can set a time-out pe
Normal mode. When PPSM dete
either from the pen, timers, real t
period countdown. When this tim
mode.

PPSM uses IRPT_NONE as an i
and is ready to go into Doze mod

However, if a Wake-up condition
doze time-out countdown is rese

This automatic Doze mode time-
mode until the Doze mode perio
tion.

11.4.2.3 Waking up fr

Any one of the following
Doze mode in MC68328

- Pen Interrupt
- Real Time Clock
- Timer 1 and Tim
- External Interrup

For MC68EZ328, whatev
wake up the system.

The kind of interrupt to w
able() and restored in Po

The system can also be
sage() to send message

For Mid-night interrupt, s
and time and then go ba
tem will wake up from D

11.4.3 Sleep Mode

In this mode,

- CPU, Phase Loc
disabled.
- Only the Real Ti

This is the mode where p
power consuming parts

There are two ways for a

- direct system ca
- automatic time-o

11.4.3.1 Direct Syste

Application can go into S
refer to *Section 11.5.5 -*
the system into Sleep m

11.4.3.2 Automatic Ti

Application can set a tim
Sleep mode.

When PPSM puts the sy
to Sleep time-out countd

- When this time-o

mode to Sleep mode. A
mode.

11.5.4 Going Into Doze Mo

VOID **SetDozeMode**

System goes directly to
up condition is met.

11.5.5 Going Into Sleep M

VOID **SetSleepMod**

System goes directly to
up condition is met.

11.6 I/O Ports Contro

For those I/O ports that a
required as PPSM does
system integrator will ne
to disable and enable the
sitions.

11.6.1 Disabling I/O Port E

VOID **PortDozeDisa**

Just before PPSM goes i
defined I/O ports that are
code to disable the I/O p

11.6.2 Enabling I/O Port A

VOID **PortDozeEna**

When PPSM wakes up f
user defined I/O ports th
their own I/O initializatio

11.6.3 Disabling I/O Port E

VOID **PortSleepDis**

Just before PPSM goes i
defined I/O ports that are
code to disable the I/O p

11.6.4 Enabling I/O Port After Sl

VOID **PortSleepEnable**(VOI

When PPSM wakes up from Slee
user defined I/O ports that are nc
their own I/O initialization code ir

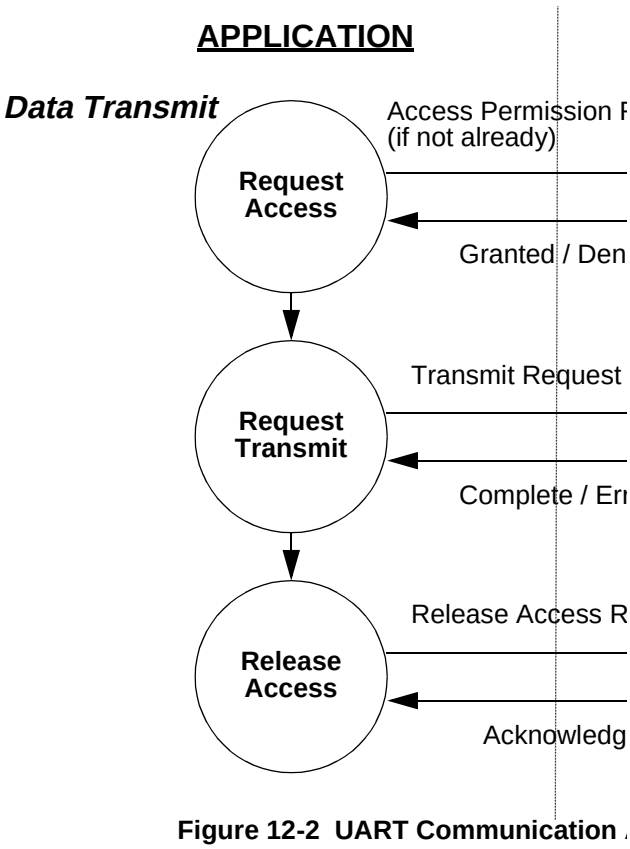


Figure 12-2 UART Communication

Chapter 12 UART Comm

PPSM supports serial co
IrDA mode. A set of inter
data through the UART.

12.1 UART Communi

The UART interface tools
receive data serially with
and *Figure 12-2* for an o
between a calling applica
transmission.

PPSM monitors the use
and IrptRelease() (refer t
tools will have effect only
to access the UART. The

Once permission is gran
or receive data through t
receive operation. The sa
UART hardware is config

12.1.1 UART hardware flo

In PPSM v3.11, data con
communication devices
RTS is asserted automat
hardware flow control is e
sender, receiver needs t
it's RTS pin. When Drag
asserting RTS pin. Thus
communication device a

Three APIs are available
UARTFlowCtrl(), UARTR
be enabled or disabled b
UARTSendCtrl(), PPSM
transmission respectively

An API, UARTSendAbor
send buffer and number
API can abort the transm

12.1.2 UART Interface Constrain

Only one task in the system can
inquiring current settings, an erro
attempts to use the UART interfa

Applications swapping is inhibite
Once a data transmission or rece
application, pen touches on appl
prevents data loss due to suspen
request.

The enforcement of these constr
initiating a request only when it is
data transmission or reception is

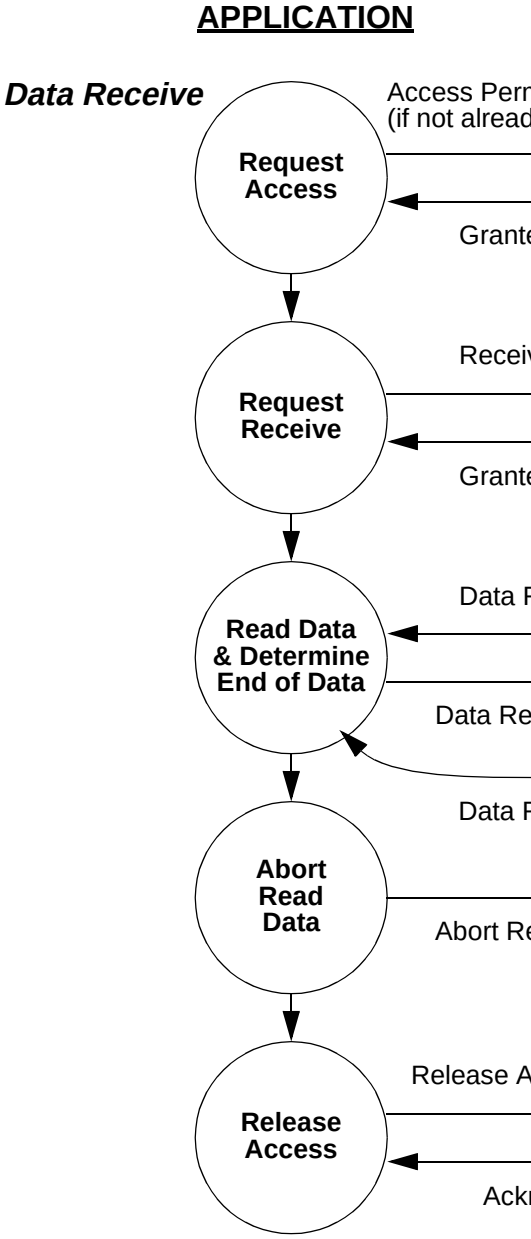


Figure 12-1 UART Commun

mode, various baud rates, parity
settings, and data transmission t
normal mode, the minimum and r
per second) and 115200 bps resp
mode, only the 115200 bps baud

There is no default UART configu
UART is configured properly by c
communciation.

Please note that the application r
before it can configure the UART
regarding the usage of IrptReque

12.2.1 Configuring the UART

STATUS **UARTConfigure**(U
U8 *charLen*)

Applications can use UARTConfi
settings. Any on-going data trans
transmission time out reset to the
approximation to the specified ba

Table 12-2 shows the list of confi
corresponding selection flag to b
26.1 - UARTConfigure for details

Table 12-2 UART Co

Configurations	
Operating Mode	
Baud Rate	
Parity	

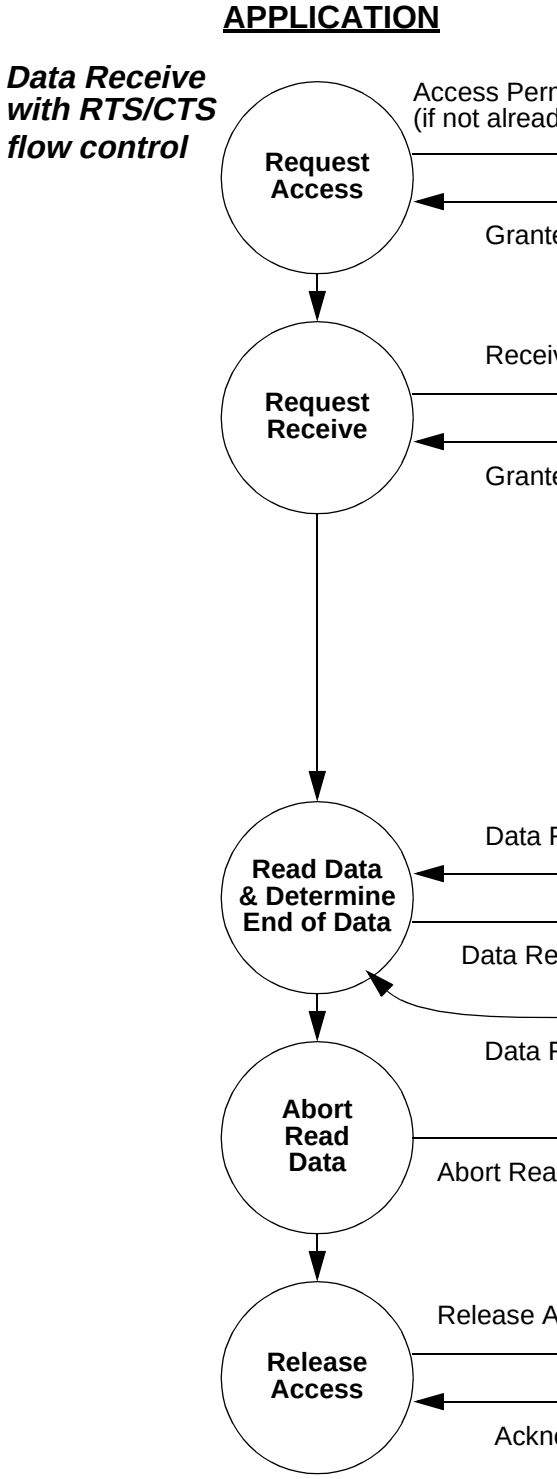


Figure 12-3 UART Communication Archit

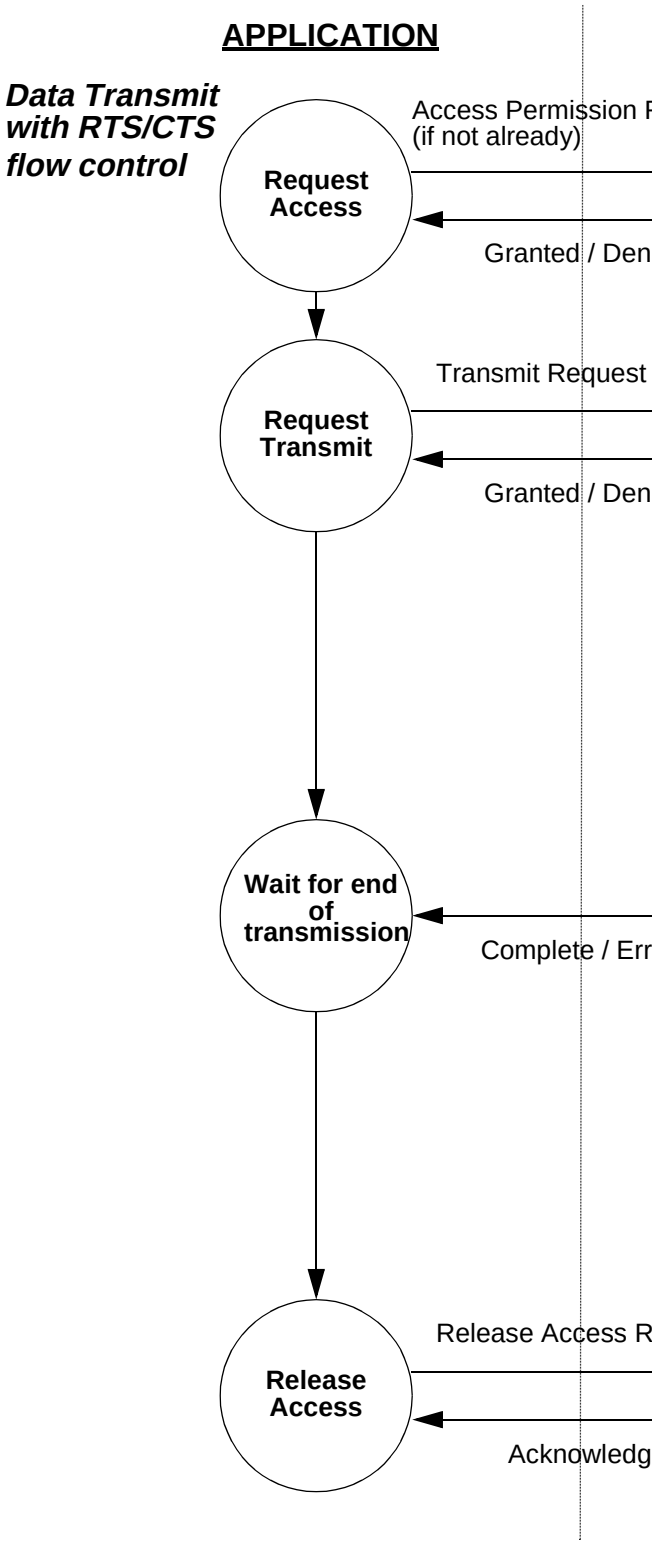


Figure 12-4 UART Communication Architecture

12.1.3 UART Interface Inte

The UART interface com
returned by IrptGetData(
Interrupt Handling for de

After an application is gr
transmission request. As
IRPT_UART interrupt me
following circumstances.

- An error conditio
  UART_ERROR,
  the calling applic
  - UART_ERR
    once the tra
  - UART_ERR
    receive.
  - UART_ERR
    receive.
  - UART_ERR
    receive.
  - UART_ERR
    data before
- Data has been r
  data, UART_DA
  application.
- Data send reque
  data, UART_DA
  application.

Table 12-1 shows the ne
with it during IrptGetData

Table 12-1 UAR

Interrupt Message
IRPT_UART

12.2 UART Configura

PPSM allows application

12.4.1 Initiating a Receive Requ

Applications can receive data
initiate receive requests. A rec
following is true:

- the application has perm
  Chapter 15 - Interrupt Ha
- there is no other on-goin

Actual data receiving does not ha
UARTReceive() returns success
interrupts and start waiting for da
to handle other interrupts (e.g. pe

For power saving reason, system
data reception, it is recommende
UARTReceive().

*Note: Application swapping
receive request.*

12.4.2 Reading Received Data

When PPSM has received data f
interrupt message with message
application. The calling applicatio
possible to read the received dat

As PPSM is receiving data from
arise:

- a frame error generated
- a parity error generated
- an overrun error when P
  behind in reading the rec

In any of the above error conditio
message with message data UAI
These error related interrupt mes
application, and does NOT stop
calling application should determ
Section 15.1.9 - IRPT_UART for

If RTS/CTS is enabled, RTS pin
UARTReadData() and asserted a

12.4.3 Terminating a Receive Re

A receive request will be termina

- If a timed out error condi
  data, PPSM will post the
  message data UART_EF

Table 12-2 UA

Configuratio
Number of Stop Bits
Character Length

12.2.2 Inquiring the UART

void **UARTInquire**(P
stopBits

UARTInquire() provides
configuration settings of
the corresponding config
rate. The actual baud rat

*Note: UARTInquir
UART acces*

12.2.3 Setting Data Transm

STATUS **UARTTime**

The data transmission ti
hardware UART interrup
deadlocking itself when t

If RTS/CTS is enabled, a
application will receive a
period. If CTS is asserte
interval between two har

The range of time out va

- Zero means disa
- 1 to 60,000 mea
  UART interrupts

12.2.4 Setting Data Transm

STATUS **UARTSetD**

In order to communicate
transmitting data in a bu
accuracy of transmission
between each transmiss
interrupts). In *Example 1*
UART to HyperTerminal
between each UART har

The range of delay values suppo

- UART_TXDELAY_CLEA transmission
- 1 to 60,000 means allow hardware UART interrup

Example 12-1 Setting delay

```
.....
      IrptRequest(IRPT_UART_FL
      /* Enable RTS/CTS flow c
      UARTFlowCtrl(UART_RCTS_E
      /* Configure UART */
      UARTConfigure( UART_NORM
      EIGHT_BIT_CHAR);
      /* Set a 600 us delay be
      _UARTSetDelay(UART_TXHAL
      /* release irpt */
      IrptRelease(IRPT_UART_FL
.....
```

12.3 Sending Data to the U

STATUS **UARTSend**(U8 *ser*

Refer to *Figure 12-2* and *Figure* architecture.

12.3.1 Initiating a Send Request

Applications can send data ou initiate send requests. A send following are true:

- the application has perm *Chapter 15 - Interrupt Ha*
- there is no other on-goin

Actual data sending does not hap UARTSend() returns success for interrupts and start sending data handle other interrupts (e.g. pen

The calling application cannot m entire course of the send reques

If RTS/CTS hardware flow contro UART when CTS pin is asserted

For power saving reason, system transmission speed is reduced. F disable the Doze mode before ca *12-2*.

Note: Application swapping data transmission.

Example 12-2 Initiati

```
.....
      IrptRequest(IRP
      SetDozePeriod(P
      UARTSend(UART_S
.....
```

12.3.2 Terminating a Send

A send request will be te

- After PPSM finis interrupt messag calling applicatio (Refer to *Sectio* interrupt messag
- If a timed out err data, PPSM will message data U This marks a fail determine the re after time out ha
- An application al UARTSend() or

The calling application sh IrptRelease() as soon as

It is recommended to for or a transmission is com

Note: Application s completed o

Example 12-3 Termi

```
.....
      /* Abort send. S
      /* .. sent in g
      UARTSendAbort(U
      IrptRelease(IRP
      SetDozePeriod(0
.....
```

12.4 Receiving Data f

STATUS **UARTRece**

STATUS **UARTRead**

Refer to *Figure 12-1* and architecture.

This marks a fail
should determin
is aborted after t

- An application al
UARTReceive()

The calling application sh
is not needed anymore.

It is recommended to for
reception or a reception

*Note: Application s
terminated.*

12.4.4 Setting Data Recep

If RTS/CTS is enabled, a
time out error depends o
application will not receiv
is off. On the other hand,
if no data arrived within t
programmer is prefer to s
UARTReceive() to avoid

Example 12-4 Settin

```
.....
IrptRequest(IRPT_UART_
UARTTimeout(1000); /*
UARTReceive(UART_RECEI
RefTimeAlarmId(&RxAlar
.....
      if(*inData==UAR
       {     DeleteTim
             UARTReadD
      .....
      }
```

12.5 UART hardware

12.5.1 Enabling RTS/CTS

Applications can enable
UARTFlowCtrl(UART_R
when calling RTS/CTS fl
returned to the applicatio

12.5.2 Disabling RTS/CTS

Applications can disable
UARTFlowCtrl(UART_R
after disabled hardware

ignored by the system. RTS is as

12.6 Data reception with ha

12.6.1 Pause data reception

After RTS/CTS hardware flow co
data reception once internal UAF
resumed after data is read out by
CTS remain negated is longer th

Applications can pause data rece
enabled. Error code PPSM_ERR
is not enabled. Applications can
UARTRcvCtrl(UART_RCTS_CO

There will be no receive timeout
Because the purpose of UARTTi
in dead loop when transmitting or
RefTimeAlarmId( ). The receive t
resumed by calling UARTRcvCtr

12.6.2 Continue data reception

Applications can continue data re
UARTRcvCtrl(UART_RCTS_PAU
Error code PPSM_ERR_RCTS_I
enabled.

12.7 Data transmission wit

12.7.1 Pause data transmission

Applications can pause data tran
enabled. Error code PPSM_ERR
is not enabled. Applications can
UARTSendCtrl(UART_RCTS_C(

There will be no transmit timeout
Because the purpose of UARTTi
in dead loop when transmitting or
RefTimeAlarmId( ). The transmit
resumed by calling UARTSendC

12.7.2 Continue data trans

Applications can continu
UARTSendCtrl(UART_R
Error code PPSM_ERR_
enabled.

needs to be swapped in later by
SWAP_TASK_BACK_LATER, th
while it's swapped out.

If the current task is parent task a
system will check whether there
If there is no more main task to b
head of subtask to see whether a
to be swapped in may due to me
the subtask is on.

If the current task is subtask and
system will check whether there i
subtask queue. If not, system will
to see whether they need to be s
more message or task to swap ir
queue to see whether it needs to

Task switching can be disabled b
function to stop task swapping w
other critical operations. If the Ap
number of times of AppSwap(TR
again.

13.4 Message Broadcastin

If the application programmer sto
a global list, message can be brc
AdvSendmessage() with or withc

13.5 Task Control

If the application programmer sto
a global list, the task swapping s
AdvSendMessage() with or witho
task swapping queue will not be
swapping queue, nothing can be
swapping queue, AdvSendMessa
queue. The earlier the task is put
the task will be swapped in.

In task swapping, PPSM will che
before checking the sub tasks of

The system will always check to
handled within the family and sw
Sometimes the message in curre
immediately to other member of t
the memory for the last message
next IrptGetData() is called. And
that task will be next IrptGetData

Chapter 13 Task Manage

Each application running
of PPSM tasks:

- main task - appli
- sub-task - task tl
  sub task.

The task management to

- Create a main ta
- Start execution
- Terminate execu

This chapter describes h
PPSM tasks.

Message passing or task
task and any other main
the panning screen para
task will affect the pannir
belonging to the same m

13.1 Main Task

Most applications fall into
each other. There canno
They are created by the
main task is created, the

- By using the sys
- By pressing the
- By messages se

13.1.1 System Task

System task is a special
terminated since creation
defined in SPC file so it
can be deleted and repla
method:

**Example 13-1 Sharin
variable**

```
/* Global variable for
U32 gPanScreen;

STATUS TaskApp()
```

```
{
        PAN_SCREEN tempScreen;

        tempScreen.panAddress =
        tempScren.horzSize = 166
        tempScreen.vertSize = 24
        tempScreen.displayXOrigi
        tempScreen.regPOSR = 0;
        tempScreen.regPSW = temp
        is set to 2 bits/pixel a

        /* Set the panning scree
        ChangePanning(tempScreer

        while(1)
        {
              ....
        }
}

main()
{
        U32 taskAppId;
        PAN_SCREEN tempScreen;

        /* PPSM Initialization *
        PPSMInit(FALSE);

        /* Get screen memory */
        gPanScreen = (U32)GetScr

        /* Assign panning screer
        tempScreen.panAddress =
        tempScren.horzSize = 166
        tempScreen.vertSize = 24
        tempScreen.displayXOrigi
        tempScreen.regPOSR = 0;
        tempScreen.regPSW = temp

        /* Delete the default sys
        to system task */
        ChangePanning(tempScreer

        /* Create a main task wi
        AdvTaskCreate(taskAppId,
        0, NULL);

        TaskStart(taskAppId);
}
```

13.2 Sub-task

Sub-task, on the other hand, can
that generated the sub-task and
task can create multiple sub-task
order they are created initially. Ho
task is swapping in or out by usir

Sub-task uses the display resour
and can only be created with the

Sub-tasks are tied to the parent t
terminated, the sub-task will be s
the input pad properties from the
input pad among the main task a

13.2.1 Sub-task Managem

When the active area of
the current task.

The IrptGetData() tool is
pending among a main t
current main or sub-task
task or its sub tasks, tas

If there are multiple sub-
most recently swapped o
system is ready to restar
of the queue.

13.3 Task Switching

When the task is started
task application. When th
current Program Counte
resume execution from w

Example 13-2 Task s


```
TaskApp1()
{
     TaskInit();

     while(TRUE)
     {
         switch(I
         ......
         ......
     }
```

At arrow 1, TaskApp1() is
also started for the first t
TaskApp1() is swapped b

If the task is swapped by
being SWAP_TASK or S

If the task is swapped by
parameter being SWAP_
immediately. It will be sw
task are handled. If the t
parent of the other family

All tasks to be swapped
swapped out will be put a

This tool launches a task that ha
AdvTaskCreate(). This routine ne
argument and begins execution
application caller from executing
main() to start the first task.

Example 13-4 Start a task

```
59    U32   SlideTask;
.
.
.
91    /* Slide is the default task
92    TaskStart(SlideTask);
```

13.11 Termination of a Task

STATUS **TaskTerminate**(U3

Termination of a task. The task i
memory associated with the task
are freed, such as stack memory
explicitly allocated by the task th
going to be freed by the system
tasks. That area can be freed by

A task cannot terminate itself. If i
either.

13.12 Task Reinitialization

STATUS **TaskReInit**(U32 *tas*

This tool will set the reinit flag in
the task is swapped in, it will star
application programmer needs to
swapping using TaskHook().

In task swapping, the PC and sta
task is not executed.

This function must be called imm
when the task is created. This fu
reinitialization at anytime but can

13.13 Task Hook

STATUS **TaskHook**(U32 *tas*

This tool will hook the entryCallba
task. When the task is swapped i
the registers. When the task is s

13.6 Task Swapping

```
TaskAppA()
{
      SubTaskCreate(&
      SubTaskCreate(&
      AdvSendMessage(
      .....
      while(1)
      {
            IrptGetDa
            ....
      }
}

TaskAppB()
{
      SubTaskCreate(&
      SubTaskCreate(&
      .....
      While(1)
      {
            IrptGetDa
            .....
      }
}

SubTaskAppA1()
{
      .....
      While(1)
      {
            IrptGetDa
            .....
      }
}

SubTaskAppA2()
{
      .....
      AdvSendMessage(
      .....
      While(1)
      {
            IrptGetDa
            .....
      }
}

SubTaskAppB1()
{
      .....
      While(1)
      {
            IrptGetDa
            .....
      }
}

SubTaskAppB2()
{
      .....
      AdvSendMessage(
      AdvSendMessage(
      AdvSendMessage(
      ......
      While(1)
      {
            IrptGetDa
            .....
      }
```

```
}
```

Task swapping sequence is A->E

When task A is created, it will cre
subtasks wouldn't be executed u
AdvSendMessage() with SWAP_
IrptGetData() so the next task is
created. In IrptGetData() of task
command is SWAP_TASK_BACI
be executed in sequence. In sub
task is B. In IrptGetData() of task
Then in IrptGetData() of subtask
SWAP_TASK_LATER is called fo
system will check for next subtas
IrptGetData() of subtask B2. In Ir
will check for the parent after che
and then A1 in IrptGetData() of tl
swapping task to A1 in subtask E
subtask in other family, the parer

13.7 Creating a Task

STATUS **TaskCreate**(P_U32
S16 *xDest*, S16

PPSM needs to know the exister
access PPSM resources. The ma
once for each application. PPSM
memory space required to run th
each application with the coordin
application is put to the foregrour
not start the execution of the app
the user does not want to have a
width or height to be zero(*xSrc* =
application icon to be created.

By default, a screen is created w
physical size as the dimension fo
specified in the Linker Specificati
Specification File.

A 2K byte of memory is allocatec

13.8 Creating a Task with S

STATUS **AdvTaskCreate**(P_
ySrc, S16 *xDest,*
screenWidth, U1

Creation of a new PPSM task. Th

same manner as the exis
allows the caller to speci
required by the applicatio
settings for the panning s

- PPSM_SCREEN
- PPSM_SCREEN
  screenHeight an
  However, if eithe
  screenHeight, is
  the linker specifi

A default of 512 byte of r
argument is negative.

Example 13-3 Create

```
57  main()
58  {
59    U32   SlideTask;
60    U32   UartDemoTask,
.
.
.
66    /*  Create the UART
67     *  and a panning sc
68     */
69    if (AdvTaskCreate(&U
70      src_y[UART_ICON],
71      PPSM_SCREEN_NEW, 0
72      return(PPSM_ERROR)
```

13.9 Creating a Sub T

STATUS **SubTaskCr**
U16 *nur*

Creating a sub-task. Any
task is itself a sub-task, t
parent(ie. the calling and
task has already created
to the head of the sub-ta
task a parent task can cr

This routine accepts varia
into the sub-task by PPS
input arguments.

Subtask will be started w
handled and there is no r
the IrptGetData() routine

13.10 Starting a Task

STATUS **TaskStart**(

In the above example, task B will
then task C will be put after task
system will swap to task B. Wher
swap to task C.

*Note: If UARTSend() or UA
can be happened in
aborted. It is applied
this chapter.*

14.1.1 With Delayed Task Swapp

```
TaskApp1()
{
     TaskInit();

     while(TRUE)
     {
         switch(IrptGetDa
         ......
       SendMessage(TaskAp
         ......
     }
```

In the diagram above, TaskApp1
TaskApp2(). However, the active
be active until it executes IrptGet
swapped back in as in arrow 2. M
IrptGetData() of TaskApp2() as ir

The task swapping happens whe

This will happen if SendMessage
SWAP_TASK_LATER are used.
task swapping queue if the target
queue otherwise.

14.1.2 With Immediate Task Swa

In AdvSendMessage(), if the flag
swapping will happen inside Adv
touching the panel.

14.1.3 With Immediate Task Swa

In AdvSendMessage(), if the flag
passing and task swapping will h
the pen is not touching the panel
the task swapping queue. So wh
current task will be swapped bac

before storing the registe
one input parameter func
task id. for the task just s
be the task id. for the tas

e.g. STATUS Entry(U32

This function can be call
the task, the entry routin
function.

Example 13-5 TaskH

```
VOID EntryR(U32 oldId)
{
......
}

VOID ExitR(U32 nextId)
{
......
}

Task1()
{
       Task1Init();
       TaskHook(task1I
       EntryR(0);

       While(1)
       ....
}
```

If TaskHook() is called ou
automatically once the ta

13.14 Stop task swapp

void **AppSwap**(U16

If *flag* is FALSE, no task
SendMessage() or AdvS
the message will be sent
executed. This function w
FALSE several times, the
before task swapping is

Chapter 14 Inter-Task M

PPSM supports asynchr
provided tools. This tool
different parent tasks.

The sender task sends o
structure using the tool S
task identifier of the task

The receiving task receiv
the same way as other in
this message by the app

The messaging tool enal

- notify other appli
- pass data betwe
- swap to the spec

The format of the data pa
receiver must have their
PPSM only performs the
application task of the ar

The AdvSendMessage()
without message passing

This can be used to send
are called to send messa
the system is in doze mo

14.1 Message Passin

Message can be sent be
SendMessage() will send
the target task once all m
AdvSendMessage() has
message sent to the targ
can be used to send mes
as a purely message pas

Example 14-1 Multip

```
TaskA()
{
        ......
        SendMessage(tas
        SendMessage(tas
        .....
}
```

14.1.4 Message Passing w

In AdvSendMessage(), it
passing only and no task

14.2 Message Structu

A pre-defined structure is

```
typedef struct _MESSAG
      {
            U16    me
            U16    me
            U32    mi
            P_VOID da
            U16    si
            U16    re
      } PPSM_MESSAGE,
```

Name
messageType
message
misc
data
size

14.3 Sending Messag

STATUS **SendMess**

This tool sends a messa

not known, this tool cannot be us

All data that the sender wants to
structure. No protocol or data for
receiver must have a mutual und
transferred.

14.4 Advanced Sending M

STATUS **AdvSendMessage**

This is similar to SendMessage()
The *flag* can control whether the
also controls whether the current
in target task are handled.

If *msg* is NULL, this is a task swa

If *flag* is NO_TASK_SWAP, this i
control.

14.5 Deleting Message for

STATUS **MessageDelete**(U1

This is for deleting all messages
parameter such as IRPT_PEN, II

14.6 Deleting Message for

STATUS **AdvMessageDelet**

This is for deleting messages in
The short data here refers to the
timer, etc. If *taskId* is 0xFFFFFFF
messages in main task queue wi
the taskId is 0xFFFFFFFF and th
matching *type* and *shortData* in
0xFFFF, all messages with matc

If *type* is 0xFFFF and *shortData* i
will be deleted.

14.7 Receiving Message

STATUS **IrptGetData**(P_U32

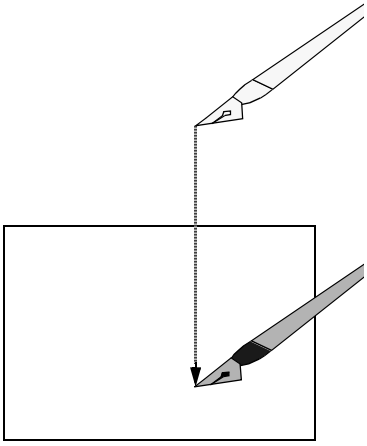
This tool is used to receive the m
receive the standard software int

The arguments returned
the MESSAGE structure

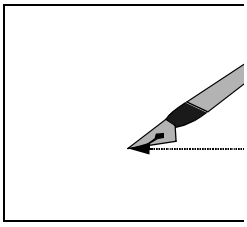
IrptGetData	Data Type
return value	STATUS
sData	P_U32
data	P_U32*
size	P_U32

Example 14-2 Receiv

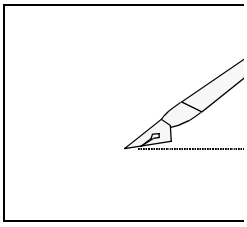
```
73  STATUS UartDemo()
74  {
75    P_U16         inData
76    U8          selected;
77    U32         size, id;
    .
    .
    .
134   while (1)
135     {
136       switch (IrptGetD
137       {
138       case IRPT_UART:
139         switch (*inDa
140           {
141           case UART_D
142
143             /* Data h
```



PPSM_ICON_TOUCH and
PPSM_INPUT_TOUCH



PPSM_ICON_D



PPSM_ICON_DRAG

Figure 15-2 ICON ar

15.1.5 IRPT_KEY

PPSM provides a soft keyboard
activated, a keyboard is displaye
keys on the soft keyboard will res
to the application. The message
was pressed. The ASCII code re
zero extended in high byte and t

Chapter 15 Interrupt Har

PPSM maintains a set of
internal hardware events
aware of the characterist
real time clock. The kern
event and send them to

PPSM maintains a uniqu
hardware interrupt occur
stored into this software
PPSM and the applicatio
if the application is slow
time interrupt of the perip

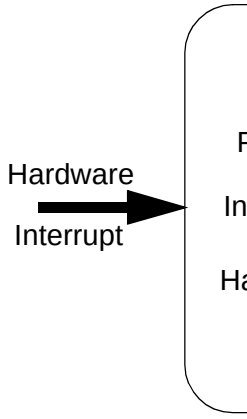


Figure 1

PPSM has two distinct ty

- System interrupt
- User defined inte

15.1 System Interrupt

These are interrupts that
developers can make us
hardware resources such

Table 15-1 shows the lis

to the application.

Table 15-1 S

Interrupt Identifier	Int
IRPT_AUDIO	PPSM A indicati
IRPT_HWR	Handwr
IRPT_ICON	Icon inp
IRPT_INPUT_STATUS	Pen inp
IRPT_KEY	Soft key
IRPT_NONE	No inter
IRPT_PEN	Pen inp
IRPT_RTC	System
IRPT_TIMER	Timer ti
IRPT_UART	UART c
IRPT_USER	User

15.1.1 IRPT_AUDIO

It is a message generated from a
the user has called AudioStopTo
stops, this interrupt is sent to the
AudioPlayWave() to indicate that

15.1.2 IRPT_PEN

It is a message generated from a
an active area as pen area on th
when pen input sequence occurs
coordinates of the pen input poin

The data message returned by IR
pair of 16-bit words in the list rep
point on the touch panel. There v
of (-1,-1) signals the end of the li

15.1.3 IRPT_INPUT_STATUS

This message is sent to the appli
beginning and the end of each p
example, when an active area cr

together with the pen sta
coordinates are sent; the
touch panel, this messag
coordinate to report the p
aware of the pen action s
area, or it is a pen down,
returned. *Figure 15-2* sh

Table 15-2 Me

Message
PPSM_INPUT_TOUCH
PPSM_INPUT_DRAG
PPSM_INPUT_PEN_UP
PPSM_INPUT_DRAG_UP

messages.

15.1.4 IRPT_ICON

Icon area is for the purpo
from the pen interrupt ha
a soft interrupt with its id
interrupts will be generat
drag in, and one interrup
for buttons, and selection

Table 15-3 shows the m

Table 15-

Message
PPSM_ICON_TOUCH
PPSM_ICON_DRAG
PPSM_ICON_PEN_UP
PPSM_ICON_DRAG_UP

Table 15-6 Interrupt I

Interrupt Handler	
SPI Slave	IRPT_S
IRQ1	IRPT_I
IRQ2	IRPT_I
IRQ3	IRPT_I
IRQ6	IRPT_I
INT0 - INT7	IRPT_I
WatchDog	IRPT_W
PWM	IRPT_P
UART	IRPT_U
User Defined	IRPT_U

The IRPT_USER is intended for i
defined interrupt messages to inf
their own meanings to this messa

Example 15-1 Request an I

```
203                /* request usag
204                if ( IrptReques
205                  return (PPSM_
     .
     .
     .
216                /* Release the
217                IrptRelease(IRP
```

15.3 Message Handling

STATUS **IrptSendData**(U16

STATUS **IrptGetData**(P_U32

User can send messages, with o
using IrptSendData(). This tool c

When an application requested a
handler, any messages sent by I
directed to that application, until

IrptSendData() appends the mes
application's software interrupt b
message via the system tool Irpt
messages.

15.1.6 IRPT_RTC

This is the clock alarm. W
message will be generat
No data is included in thi

15.1.7 IRPT_TIMER

Time-out message. This
out period is reached. No

15.1.8 IRPT_HWR

PPSM provides an input
5 - Character Input Meth
resultant characters gen
message. The data pass
character candidates an

15.1.9 IRPT_UART

When an application is g
UART Communication S
report UART data transm

A 16-bit message data is
have one of the following

Table 15-

Message
UART_DATA_SENT
UART_DATA_RECEIVED
UART_ERROR

15.2 Device Interrupts

PPSM supports another

devices. The list of device interru

Table 15-5 Interrupt I

Interrupt Source	Interru
SPI Master	IRPT_SPI
SPI Slave	IRPT_SPI
IRQ1	IRPT_IRQ
IRQ2	IRPT_IRQ
IRQ3	IRPT_IRQ
IRQ6	IRPT_IRQ
INT0 - INT7	IRPT_INT
WatchDog	IRPT_WD
PWM	IRPT_PW
UART	IRPT_UA
User Defined	IRPT_US

15.2.1 User Defined Interrupt Ha

For each of the external interrupt
their own handler. The generic, c
device library. These are:

- SPI Master
- SPI Slave(DragonBall or
- IRQ6
- IRQ3
- IRQ2
- IRQ1
- INT0 - INT7
- Watch Dog
- PWM
- UART

Each of the stub handler is assoc
_IRQ6IrptHandler is associated v
external interrupt event occurs, F
handler as part of the interrupt ha

For system that uses any of the e
handler such that integration into

For PPSM source licensee, if _U
DNO_UART_HANDLER" needs
that the internal PPSM UART int

15.2.2 Device Interrupt Ide

The device interrupt iden
handlers to send soft inte
much like the system pe
IrptSendData(), to allow
handlers to the applicatio

For example, user install
message to the applicati
event, or to pass data fro

15.2.3 Application Access

By isolating the interrupt
have access to the same
only be registered with a
application is requesting
first application making t
handler until it is release

To control access conflic
hardware concurrently, a

- IrptRequest()
- IrptRelease()

15.2.4 Request and Relea

U32 **IrptRequest**(U3

STATUS **IrptReleas**

When an application tas
it must first request for s
successfully registers wit
are directed to the regist
handler.

One application task can
handlers in the system, b
attached to one single ap
release the handler after

To request or release a h
the interrupt tools to spe
values representing indiv

Table 15-6 In

Interrupt Handler
SPI Master

16.1.1 Motorola Logo

Depending on the physical LCD s
the LCD display during pen calib

For LCD display that is larger tha
standard logo will be displayed. T
Figure 16-2.

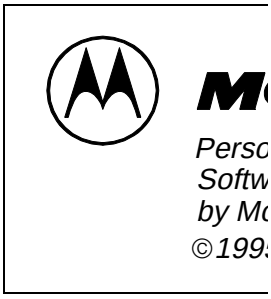


Figure 16-2

For LCD display that is larger tha
than 280 pixels by 150 pixels, a s
Motorola logo will be 104 pixels b

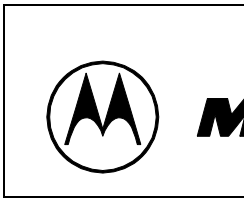


Figure 1

For LCD display that is smaller th
Motorola logo will be drawn.

15.3.1 Example

Assuming that an applica
PPSM, when the user de
sent to the application im

```
shortMessage = COMMAND
messageSize = 4;
messageData = pInData;
IrptSendData( IRPT_IRQ
```

The application retrieve t

```
switch (IrptGetData( &
.
.
.
case IRPT_IRQ6:
       /*  IRQ6 event
       if (event == CO
       .
       .
       .
       default;
```

In the above example, th
application:

Table 15-7

IrptSendDa	
Argument	
irptType	IR
sData	*sh*
data	*me*
size	*m*

Chapter 16 Using Syste

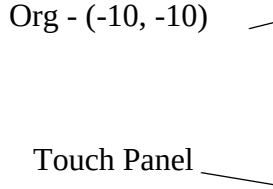
PPSM provides additiona
resources.

16.1 PPSM Initializati

STATUS **PPSMInit**(U

PPSM needs to be initial
PPSM tools can be calle
devices initialization and
display.

A device driver (PenInit.c
the touch panel origin and
This function should call
coordinate of origin is the
screen coordinate. The c
touch panel in terms of d



Figu

The default touch panel
pixels by 240 pixels (phy
An offset of 100 (in A/D c
the edge of the touch pa

The caller of CalibratePe
logoFlag is FALSE, no lo
cross-hairs (one at top ri
displayed on the LCD sc
hairs, in no particular orc

Example 16-1 Initiali

```
62    /* Initialize PPSM w
63    PPSMInit(TRUE);
```

```
    /*  Display message to indicat
    ....
    break;
case IRPT_ICON:
    /*  Click icon to stop the wav
    rv = AudioStopWave();
    ....
    break;
}
```

Chapter 17 Audio Tools

17.1 Audio Playing

PPSM supports two type
hardware limitation, the

The audio tools have the

- Only one wave f
- A wave file or to
Modulation) mod
- An interrupt "IRP
called AudioPlay
playing has finis

17.2 Tone playing

STATUS **AudioPlayTon**
autoRepeat)

PPSM supports tone pla
PWM module. Tone play
and changeable frequen
resolution, the tone frequ

Name
toneData
toneSize
toneDuration

Name
autoRepeat

Example 16-1 PPSM tone p

```
/*  100Hz, 1000Hz, 500Hz and 600Hz
U16 toneData[] = {100, 1000, 500,

/*  Play a melody with 4 different
#ifdef EZ328
    AudioPlayTone((P_U16)toneData,
#else
    AudioPlayTone((P_U16)toneData,
#endif
```

To stop the tone playing, a user
Tools are currently being used, a

Note: This is impossible to play a of duration, since the duration of duration.

17.3 Wave playing (Dragon

PPSM audio tools can play back that can be generated by many a two PPSM audio tools - AdvAudi

AdvAudioPlayWave() is provided want to have advanced configura module. For most cases, AudioP

STATUS **AudioPlayWave**(P_U8

Name
waveData
waveSize
samplingRate

Example 16-2 PPSM wave p

```
/*  Some PWM wave data */
U16 waveData[] = {...};
```

```
/*  Play a melody with 100
AudioPlayWave((P_U8)waveDa
```

STATUS **AdvAudioPlay**
repeat, U8 clksel)

Name
waveData
waveSize
prescaler(see Dragon manual)
repeat(see DragonBa manual)
clksel(see DragonBa manual)

The sampling rate can b

$$SamplingRate = ((16.58MHz)/$$

For more detail informati

Example 16-3 PPSM

```
/*  Some PWM wave data */
U16 waveData[] = {...};

/*  Play a melody with 100
/*  16kHz = 16.58Mz/(2 x 1
AdvAudioPlayWave((P_U8)wav
```

The device driver functio TRUE for proper wave p his/her own PWM interru playing tools, the _PWM

17.4 Stop the audio p

An audio stops after it ha AudioStopWave(). After called AudioPlayTone() finished.

Example 16-4 Stopp

```
/*  Some PWM wave data */
U16 waveData[] = {...};

/*  Play a melody with 100
AudioPlayWave((P_U8)waveDa

switch( IrptGetData((P_U32
{
case IRPT_AUDIO:
```

Parameter

Name
areaId
code
mode
xSrc
ySrc
xDest
yDest

Return Value

Name
PPSM_OK
PPSM_ERR_AREA_ID
PPSM_ERR_AREA_CODE
PPSM_ERR_COORDINATE
PPSM_ERR_NO_MEMORY

18.3 ActiveAreaRead

Syntax

Part III
API Toc

Chapter 18 Pen Input To

18.1 ActiveAreaDisab

Syntax

STATUS **ActiveArea**

Description

Removes an active
specifies the entry th
obtained from the Ac
creates the active ar

Parameter

Name
areaId

Return Value

Name
PPSM_OK
PPSM_ERR_AREA_

18.2 ActiveAreaEnab

Syntax

STATUS **ActiveArea**
S16 *ySr*

Description

Creates and enables
area. An active area
where a software int
the area when the
returned to the appli

Active areas can be
the boundary of the
display area are the
display area, echoin

18.8 AreaEchoDisable

Syntax

STATUS **AreaEchoDisable**(

Description

Disables active area pixel ec
selected pixels will not be
disabled for all active input a

Parameter

Name
areaId

Return Value

Name
PPSM_OK
PPSM_ERR_AREA_ID

18.9 AreaEchoEnable

Syntax

STATUS **AreaEchoEnable**(

Description

Enables active area pixel ec
selected pixels will be echoe
all active input area by defau

Parameter

Name
areaId

Return Value

Name
PPSM_OK
PPSM_ERR_AREA_ID

STATUS **ActiveArea**
xDest, P

Description

Reads the area coor
identifier *areaId* spec

Parameter

Name
areaId
xSrc
ySrc
xDest
yDest

Return Value

Name
PPSM_OK
PPSM_ERR_AREA_

18.4 ActiveAreaSusp

Syntax

STATUS **ActiveArea**

Description

Suspend or re-enabl
active area is suspen
will remain in the ac
again respond to pen

Parameter

Name
areaId

Name
flag

Return Value

Name
PPSM_OK
PPSM_ERR_AREA_ID

18.5 ActiveAreaToFront

Syntax

STATUS **ActiveAreaToFron**

Description

Given the active area identifi
area linked list and insert the

Once the element is at the fr
pen input if other active area

Parameter

Name
areaId

Return Value

Name
PPSM_OK
PPSM_ERR_AREA_ID

18.6 ActiveListPop

Syntax

STATUS **ActiveListPop**(voi

Description

Pops the top backgr
stack. The active list
top background activ

Parameter

Name
None

Return Value

Name
PPSM_OK
PPSM_ERROR

18.7 ActiveListPush

Syntax

STATUS **ActiveListP**

Description

Pushes the current
creates a new empt
these areas is disabl
to the new active list

The number of active
stored internally in a

Parameter

Name
None

Return Value

Name
PPSM_OK
PPSM_ERR_ACTIVE

Return Value

Name
None

18.14 IconScanOn

Syntax

void **IconScanOn**(void)

Description

Switches on system applicat
by default.

Parameter

Name
None

Return Value

Name
None

18.15 PenCalibration

Syntax

STATUS **PenCalibration**(U1

Description

This function performs per
peninit.c of the device driver
different calibration method
default driver will clear the sc
pen calibration are captured.

Parameter

Name
logoFlag

18.10 ActiveAreaPosit

Syntax

STATUS **ActiveArea**
S16 *yDe*

Description

This function will cha
by *areaId*.

Parameter

Name
areaId
xSrc
ySrc
xDest
yDest

Return Value

Name
PPSM_OK
PPSM_ERR_AREA_
PPSM_ERR_COORD

18.11 CtrlIconDisable

Syntax

STATUS **CtrlIconDis**

Description

Removes a predefin
must be a valid activ

Parameter

Name
iconId

Return Value

Name
PPSM_OK
PPSM_ERR_AREA_ID

18.12 CtrlIconEnable

Syntax

STATUS **CtrlIconEnable**(P_

Description

Adds a predefined icon area anywhere on the applicatio specifies the position of the i to the caller.

Parameter

Name
iconId
xSrc
ySrc

Name
iconType

Return Value

Name
PPSM_OK
PPSM_ERR_ICON_T
PPSM_ERR_COORD
PPSM_ERR_AREA_

18.13 IconScanOff

Syntax

void **IconScanOff**(v

Description

Switches off system current task.

Parameter

Name
None

For MC68EZ328 only

Sampling Period
samplingPeriod>= 250
250>samplingPeriod>=125
125>samplingPeriod>= 62
62>samplingPeriod>= 31
31>samplingPeriod>= 15
15>samplingPeriod>= 7
7>samplingPeriod>= 3
3>samplingPeriod>= 1

Return Value

Name
PPSM_OK
PPSM_ERR_PEN_RATE

18.21 ScanningOff

Syntax

void **ScanningOff**(void)

Description

Switches off touch panel sca
active areas will not respons

Parameter

Name
None

Return Value

Name
None

18.22 ScanningOn

Syntax

void **ScanningOn**(void)

Return Value

Name
PPSM_OK

18.16 PenEchoParam

Syntax

STATUS **PenEchoP**

Description

This tool allows the
system pen echoing

Parameter

Name
echoCol
echoWidth

Return Value

Name
PPSM_OK
PPSM_ERR_COLOU
PPSM_ERROR

18.17 PenGetInput

Syntax

STATUS **PenGetInp**

Description

Returns a single pair of X a
point. A set of -1 will be retur
panel range, i.e. pen up.

Parameter

Name
xPos
yPos

Return Value

Name
PPSM_OK

18.18 PenSetInputMax

Syntax

STATUS PenSetInputMax(S

Description

It allows the user to defin
coordinate in terms of scree
system is doing calibratio
CalibratePen(U16 logoFlag)

Parameter

Name
x
y

Return Value

Name
PPSM_OK

18.19 PenSetInputOrg

Syntax

STATUS **PenSetInp**

Description

It allows the user to
terms of screen dis
doing calibration/re-c
logoFlag) at the devi

Parameter

Name
x
y

Return Value

Name
PPSM_OK

18.20 PenSetRate

Syntax

STATUS **PenSetRat**

Description

To allow user to defi
period set in an task
effect after at least o

Parameter

Name
samplingPeriod

Name
timeOut
samplingTime
areaClean
stackSize

Return Value

Name
PPSM_OK
PPSM_ERR_INPUT_PAD_OF
PPSM_ERR_INPUT_PAD_WI
PPSM_ERR_INPUT_PAD_HE
PPSM_ERR_INPUT_PAD_X_
PPSM_ERR_INPUT_PAD_Y_
PPSM_ERR_PEN_RATE
PPSM_ERR_PAN_INIT
PPSM_ERR_NO_MEMORY
PPSM_ERR_TMOUT_VALUE

19.2 AdvOpenSoftKey

Syntax

STATUS **AdvOpenSoftKey**(
keyHeight, U16
bitmap)

Description

Opens a soft keyboard modu
but with advanced configurat

- location of the soft keybc

Description

Switches on touch p
all application active

Parameter

Nam
None

Return Value

Nam
None

Chapter 19 Character In

19.1 AdvOpenInputPa

Syntax

STATUS **AdvOpenI**
numCol,
echoWid
stackSiz

Description

Opens the input p
OpenInputPad() but
specify:

- position of the in
- number of rows
- the width and th
- the echo ink colo
- the length of tim
- the sampling rat
- if the system sho
character is writt
- the stack size fo

Parameter

Name
xPos
yPos
numRow
numCol
areaWidth
areaHeight
echoCol
echoWidth

Name
PPSM_ERR_INPUT_PAD_OF
PPSM_ERR_INPUT_PAD_X_
PPSM_ERR_INPUT_PAD_Y_
PPSM_ERR_INPUT_PAD_WI
PPSM_ERR_INPUT_PAD_HE
PPSM_ERR_NO_MEMORY

19.6 OpenSoftKey

Syntax

STATUS **OpenSoftKey**(U16

Description

Opens a soft keyboard modu

A soft keyboard is drawn at t
keyboard is opened, key-
generated to the application
pressed. Each individual k
message.

Only one soft keyboard is a
covered by the pseudo keyb
will be restored upon closing
system is a snap-shot of the
changes to this area by the a

Parameter

Name
xPos
yPos

Return Value

Name
PPSM_OK
PPSM_ERR_SKBD_X_POS

- width and height
- number of rows
- the return code
- the bitmap of the

Parameter

Name
xPos
yPos
keyWidth
keyHeight
numRow
numCol
keyMap
bitmap

Return Value

Name
PPSM_OK
PPSM_ERR_SKBD_
PPSM_ERR_PAN_IN
PPSM_ERR_INPUT_ EN
PPSM_ERR_SKBD_
PPSM_ERR_SKBD_
PPSM_ERR_NO_ME

19.3 CloseInputPad

Syntax

STATUS **CloseInputPad**(voi

Description

Closes the handwritten chara

The input pad image is remov
handwriting recognition mes
system to the application. T
restored by the system.

Parameter

Name
None

Return Value

Name
PPSM_OK
PPSM_ERR_PEN_INIT
PPSM_ERR_INPUT_PAD_CL

19.4 CloseSoftKey

Syntax

STATUS **CloseSoftKey**(void

Description

Closes the soft keyboard mo

The soft keyboard image is
pressed messages (IRPT_K
application. The original ima
system.

Parameter

Name
None

Return Value

Name
PPSM_OK

Name
PPSM_ERROR

19.5 OpenInputPad

Syntax

STATUS **OpenInput**
U16 *are*

Description

Opens the input pad

The input pad is draw
input pad has *numR*
box is of size *areaSi*

Once the input pad
(IRPT_HWR) are ge
recognized. Each r
message.

Only one input pad
covered by the input
restored upon closi
system is a snap-sh
changes to this area
input pad needs to b

The default length of

Parameter

Name
xPos
yPos
numRow
numCol
areaSize

Return Value

Name
PPSM_OK

Parameter

Name
newPanning
flag
oldPanning

Return Value

Name
PPSM_OK

Name
PPSM_ERR_SKBD_
PPSM_ERR_SKBD_
PPSM_ERR_NO_ME

Chapter 20 Graphics To

All coordinates mentione

- The range of val
  Screen Width - 1
- The range of val
- The range of val

20.1 ChangePanning

Syntax

STATUS **ChangePa**
P_PAN_

Description

This function chang
The panning screen
16 for 1 bit/pixel disp

Return Value

Name
PPSM_ERR_PAN_INIT
PPSM_ERROR
PPSM_OK

Hint

The LCD display origin is the
screen relative to the pannin

20.6 CursorGetPos

Syntax

STATUS **CursorGetPos**(P_U

Description

Returns the coordinate (*xP
task. If this function is called
no more information about cu

Parameter

Name
xPos
yPos

Return Value

Name
PPSM_ERR_CURSOR_INIT
PPSM_ERR_PAN_INIT
PPSM_ERROR
PPSM_OK

Name
PPSM_ERR_PAN_A
PPSM_ERR_PAN_W
PPSM_ERR_PAN_H

Hint

The panning screen
some cases, a comm
To achieve that, the
and ChangePanning

20.2 ChangeWindow

Syntax

STATUS **ChangeWi**
P_U16 *

Description

This tool will direct a
will be changed on
divisible by 8 for 2 b
The original settings

Parameter

Name
addr
width
height
oldAddr
oldWidth
oldHeight

Return Value

Name
PPSM_OK

Name
PPSM_ERR_WIDTH
PPSM_ERR_HEIGHT

Hint

This is used for displaying i
image can be plotted in othe
the image is drawn, it car
ChangeWindow() and PutRe

20.3 ClearRec

Syntax

STATUS **ClearRec**(U16 *grey*
height, U16 *style*

Description

Fills the given area with grey

Parameter

Name
greyLevel
xSrc
ySrc
width
height
style

Return Value

Name
PPSM_OK
PPSM_ERR_LCD_X
PPSM_ERR_LCD_Y
PPSM_ERR_WIDTH
PPSM_ERR_HEIGHT

Name
PPSM_ERR_LCD_S

Hint

If (*xSrc*, *ySrc*) is (5, 1
rectangle with top le
LCD with specified *g*

20.4 ClearScreen

Syntax

void **ClearScreen**(U

Description

Fills the whole panni

Parameter

Name
greyLevel

Return Value

Name
None

20.5 CursorGetOrigin

Syntax

STATUS **CursorGet**

Description

Returns the coordina

Parameter

Name
xPos
yPos

Name
PPSM_ERR_CURSOR_INIT
PPSM_ERROR

20.12 CursorSetOrigin

Syntax

STATUS **CursorSetOrigin**(U

Description

Sets the LCD display screen

Parameter

Name
xPos
yPos

Return Value

Name
PPSM_ERR_PAN_INIT
PPSM_ERR_LCD_X
PPSM_ERR_LCD_Y
PPSM_OK

Hint

This must be used with LCD

20.13 CursorSetPos

Syntax

STATUS **CursorSetPos**(U16

Description

Sets the top left corner positi

20.7 CursorGetStatus

Syntax

STATUS **CursorGet**

Description

Returns the status o

Parameter

Name
status

Return Value

Name
PPSM_ERR_CURSO
PPSM_OK

20.8 CursorInit

Syntax

STATUS **CursorInit**

Description

Changes hardware
cursorHeight. The w

Parameter

Name
cursorWidth
cursorHeight

Return Value

Name
PPSM_ERR_PAN_INIT
PPSM_ERR_CURSOR_INIT
PPSM_ERR_WIDTH
PPSM_ERR_HEIGHT
PPSM_OK

Hint

If the cursor is large, program

20.9 CursorOff

Syntax

STATUS **CursorOff**(void)

Description

Turns off the hardware curso

Parameter

Name
None

Return Value

Name
PPSM_ERR_CURSOR_INIT
PPSM_OK

20.10 CursorSet

Syntax

STATUS **CursorSet**(U16 *xP*

Description

Sets the top left co
current task must ha
panning screen. Ho
exceeds the panning

Parameter

Name
xPos
yPos

Return Value

Name
PPSM_ERR_PAN_IN
PPSM_ERR_CURSO
PPSM_ERR_LCD_X
PPSM_ERR_LCD_Y
PPSM_OK

20.11 CursorSetBlink

Syntax

STATUS **CursorSet**

Description

This will set the hard
number of blinks per
cursor is set on by c

Parameter

Name
frequency

Return Value

Name
PPSM_ERR_PAN_IN

STATUS **DrawCircle**(U16 *gr*
U16 *style*)

Description

Draws a circle centered a
specified.

If dot width is greater than 1,

If both fill pattern mode and
which is not covered by bord

If fill pattern mode is set an
circle border will be filled.

Parameter

Name
greyLevel
xCenter
yCenter
radius
style

Return Value

Name
PPSM_OK
PPSM_ERR_LCD_GREY
PPSM_ERR_LCD_X
PPSM_ERR_LCD_Y
PPSM_ERR_LCD_RADIUS
PPSM_ERR_LCD_STYLE

20.18 DrawDot

Syntax

STATUS **DrawDot**(U16 *grey*

Description

Outputs a dot with grey lev
yPos) with indicated style.

Parameter

Name
xPos
yPos

Return Value

Name
PPSM_ERR_PAN_IN
PPSM_ERR_CURSO
PPSM_ERR_LCD_X
PPSM_ERR_LCD_Y
PPSM_OK

20.14 CursorSetStatus

Syntax

STATUS **CursorSet**

Description

This will set the har
video mode, or temp
function is
PPSM_CURSOR_R

Parameter

Name
status

Return Value

Name
PPSM_ERR_PAN_INIT
PPSM_ERR_CURSOR_INIT
PPSM_OK

Hint

CursorInit() is called to set th
from 0 to 31 in pixels. Cursor
left corner of the hardware
turn the hardware cursor on
cursor is on, CursorSetPos(
hardware cursor position
CursorSetStatus() again.

20.15 DisplayMove

Syntax

STATUS **DisplayMove(**U16

Description

This function is to repl
CursorSetOrigin(). Wheneve
screen with top left corner of
DisplayMove(xPos, yPos) sh

Parameter

Name
xPos
yPos

Return Value

Name
PPSM_OK
PPSM_ERR_PAN_INIT
PPSM_ERR_COORDINATE

20.16 DrawArc

Syntax

STATUS **DrawArc**(U
style)

Description

Draws an arc from (*x*

The arc is actually a
(*x1*, *y1*) and (*x2*, *y2*)

If dot width is greate

If both fill pattern mo
is not covered by the

If fill pattern is set an
be filled.

Parameter

Name
greyLevel
x1
y1
x2
y2
style

Return Value

Name
PPSM_OK
PPSM_ERR_LCD_G
PPSM_ERR_LCD_X
PPSM_ERR_LCD_Y
PPSM_ERR_LCD_S

20.17 DrawCircle

Syntax

Parameter

Name
greyLevel
xSrc
ySrc
xDest
yDest
dotLine
style

Return Value

Name
PPSM_OK
PPSM_ERR_LCD_GREY
PPSM_ERR_LCD_X
PPSM_ERR_LCD_Y
PPSM_ERR_LCD_STYLE

20.22 DrawRec

Syntax

STATUS **DrawRec**(U16 *grey*
yDest, U16 *dotL*

Description

Draws a rectangular outline v
right corner at (*xDest*, *yDest*)

If dot width is greater than 1,

If both fill pattern mode an
rectangle which is not covere

If fill pattern mode is set and
rectangle border will be filled

If dot width is 2, a sq
is (*xPos*, *yPos*). If do
yPos) will be drawn.

Parameter

Name
greyLevel
xPos
yPos
style

Return Value

Name
PPSM_OK
PPSM_ERR_LCD_G
PPSM_ERR_LCD_X
PPSM_ERR_LCD_Y
PPSM_ERR_LCD_S

20.19 DrawEllipse

Syntax

STATUS **DrawEllips**
xLength

Description

Draws an ellipse cen
in the x-axis, and *yL*

If dot width is greate

If both fill pattern m
which is not covered

If fill pattern mode is
ellipse border will be

Parameter

Name
greyLevel
xCenter
yCenter
xLength
yLength
style

Return Value

Name
PPSM_OK
PPSM_ERR_LCD_GREY
PPSM_ERR_LCD_X
PPSM_ERR_LCD_Y
PPSM_ERR_LCD_STYLE

20.20 DrawHorz

Syntax

STATUS **DrawHorz**(U16 *gre*
dotLine, U16 *sty*

Description

Draws a horizontal line from

If dot width is greater than 1,
greater than 1 and the width
indicated by dot width will be

If *dotLine* is non-zero, *dotLin*
grey level; then, the *dotLine*
number of dots will be drawn

Parameter

Name
greyLevel

Name
xSrc
ySrc
width
dotLine
style

Return Value

Name
PPSM_OK
PPSM_ERR_LCD_G
PPSM_ERR_LCD_X
PPSM_ERR_LCD_Y
PPSM_ERR_LCD_S

20.21 DrawLine

Syntax

STATUS **DrawLine**(
yDest, U

Description

Draws a line from (*x*

If dot width is greate
than 1 and (*xSrc*, *yS*
by dot width will be d

If *dotLine* is non-zero
grey level; then, the
number of dots will b

Parameter

Name
xSrc
ySrc
width
height
bitmap

Return Value

Name
PPSM_OK
PPSM_ERR_LCD_X
PPSM_ERR_LCD_Y
PPSM_ERR_WIDTH
PPSM_ERR_HEIGHT

20.26 GetDisplayX

Syntax

U16 **GetDisplayX**(void)

Description

Returns the LCD display scr

Parameter

Name
None

Return Value

Name
N/A

Hint

Parameter

Name
greyLevel
xSrc
ySrc
xDest
yDest
dotLine
style

Return Value

Name
PPSM_OK
PPSM_ERR_LCD_G
PPSM_ERR_LCD_X
PPSM_ERR_LCD_Y
PPSM_ERR_LCD_S

20.23 DrawVector

Syntax

STATUS **DrawVecto**
pPoints,

Description

Draws lines to conn
input. No connection
specified by *mode* o

DrawVector() does n

Parameter

Name
greyLevel
numberOfPoints
pPoints
style
mode

Return Value

Name
PPSM_OK
PPSM_ERROR
PPSM_ERR_LCD_GREY
PPSM_ERR_LCD_STYLE
PPSM_ERR_LCD_X
PPSM_ERR_LCD_Y

Hint

If the output style is EXOR_
overlapped points.

20.24 DrawVert

Syntax

STATUS **DrawVert**(U16 *grey*
dotLine, U16 *sty*

Description

Draws a vertical line from (*x*

If dot width is greater than 1
greater than 1 and the heig
indicated by dot width will be

If *dotLine* is non-zero
grey level; then, the
number of dots will b

Parameter

Name
greyLevel
xSrc
ySrc
height
dotLine
style

Return Value

Name
PPSM_OK
PPSM_ERR_LCD_G
PPSM_ERR_LCD_X
PPSM_ERR_LCD_Y
PPSM_ERR_LCD_S

20.25 ExchangeRec

Syntax

STATUS **Exchangel**
bitm

Description

Swaps the image i
specified location of
now be displayed wh
stored at *bitmap*.

Note that the image
rectangle in the argu

20.32 LCDContrast

Syntax

STATUS **LCDContrast**(U8 c

Description

This tool will pass the value
The user needs to set or res

Parameter

Name
contrast

Return Value

Name
PPSM_OK

Application program
applications size ind

20.27 GetDisplayY

Syntax

U16 **GetDisplayY**(v

Description

Returns the display s

Parameter

Name
None

Return Value

Name
N/A

Hint

Application program
applications size ind

20.28 GetLogicalX

Syntax

U16 **GetLogicalX**(v

Description

Returns the current p
the size that an app
display screen (i.e. th
screen).

Parameter

Name
None

Return Value

Name
N/A

20.29 GetLogicalY

Syntax

U16 **GetLogicalY**(void)

Description

Returns the current panning gives the size that an applic LCD display screen (i.e. the display screen).

Parameter

Name
None

Return Value

Name
N/A

20.30 GetScreenMem

Syntax

P_VOID **GetScreenMem**(U1

Description

This tool will allocate approp and *height*.

Parameter

Name
width
height

Return Value

Name
N/A

20.31 InvRec

Syntax

STATUS **InvRec**(U1

Description

Inverts the grey leve

Parameter

Name
xSrc
ySrc
width
height

Return Value

Name
PPSM_OK
PPSM_ERR_LCD_X
PPSM_ERR_LCD_Y
PPSM_ERR_WIDTH
PPSM_ERR_HEIGH

Name
reserved

Return Value

Name
PPSM_OK
PPSM_ERR_LCD_X
PPSM_ERR_LCD_Y
PPSM_ERR_WIDTH
PPSM_ERR_HEIGHT
PPSM_ERR_LCD_STYLE

20.37 SaveRec

Syntax

STATUS **SaveRec**(P_U8 *bitr*
U16 *reserved*)

Description

Saves a rectangular bitmap
screen to memory.

Parameter

Name
bitmap
xSrc
ySrc
width
height
reserved

Return Value

Name
PPSM_OK

20.33 LCDRefreshRate

Syntax

STATUS **LCDRefres**

Description

This tool will set the
second. As the fran
Refresh Rate Adjus
Register and PLL Co
closest possible va
refreshRateSet will b
the calling of LCDR
called once more
refreshRateSet.

Parameter

Name
refreshRate
refreshRateSet

Return Value

Name
PPSM_OK

20.34 LCDScreenMove

Syntax

STATUS **LCDScreen**

Description

This function is repla

Maps the LCD displ
panning screen with
screen.

It is assumed that (*x*,

Parameter

Name
x
y

Return Value

Name
PPSM_OK

Hint

This must be used with Curs

20.35 PutChar

Syntax

STATUS **PutChar**(U16 greyL
font, U16 width,

Description

This tool will put a 1 bit/pixe
depending on whether ppsm

Parameter

Name
greyLevel
character
xPos
yPos
font
width
height

Name
style

Return Value

Name
PPSM_OK
PPSM_ERR_PAN_IN
PPSM_ERR_GREY
PPSM_ERR_LCD_X
PPSM_ERR_LCD_Y
PPSM_ERR_LCD_F
PPSM_ERR_LCD_S
PPSM_ERROR

20.36 PutRec

Syntax

STATUS **PutRec**(P_
U16 *styl*

Description

Puts a rectangular b
panning screen.

Parameter

Name
bitmap
xSrc
ySrc
width
height
style

Parameter

Name
dbId
recId
numFmt

Return Value

Name
PPSM_OK
PPSM_ERR_DB_DBID
PPSM_ERR_NUM_FMT

Hint

For DBAddRecord() to wor
created by calling DBAdd() p

21.3 DBAddRecToTop

Syntax

STATUS **DBAddRecToTop**(

Description

Adds a blank record at the
User has the option to specif
The valid range of *numFmt* i

This tool is meant to compler

Parameter

Name
dbId
numFmt
outRecId

Name
PPSM_ERR_LCD_X
PPSM_ERR_LCD_Y
PPSM_ERR_WIDTH
PPSM_ERR_HEIGH

20.38 SetDotWidth

Syntax

STATUS **SetDotWid**

Description

This tool sets the width f
drawn by DrawDot(), a th

This dot width is applied
DrawEllipse(), DrawArc()

If oldWidth is non-zero, t

Parameter

Name
newWidth
oldWidth

Return Value

Name
PPSM_OK
PPSM_ERR_DOT_W

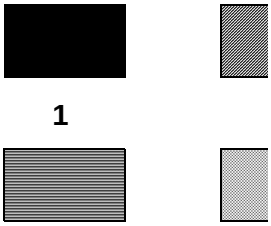
20.39 SetPatternFill

Syntax

STATUS **SetPattern**
fillSpace

Description

This tool sets the pattern fill
DrawEllipse() and DrawArc()



1

5

The pattern fill mode 0 will tu

Parameter

Name
mode
backGrey
borderMode
fillSpace

Return Value

Name
PPSM_OK
PPSM_ERR_FILL_PATTERN
PPSM_ERR_LCD_GREY
PPSM_ERR_FILL_SPACE

Chapter 21 Database Ma

21.1 DBAdd

Syntax

STATUS **DBAdd**(P_

Description

Adds a new databas
called whenever a
database.

Parameter

Name
dbId

Return Value

Name
PPSM_OK
PPSM_ERR_DB_AD

Hint

The returned *dbId* m
the database.

21.2 DBAddRecord

Syntax

STATUS **DBAddRec**

Description

Appends a blank rec
User has the option
The valid range of *n*

Name
PPSM_ERR_DB_TYPE

Hint

The record and database r
environment.

Observes the unformatted da

21.7 DBDelete

Syntax

STATUS **DBDelete**(U32 *dbId*

Description

Removes a database from P

Parameter

Name
dbId

Return Value

Name
PPSM_OK
PPSM_ERR_DB_DBID

Hint

The database referred to mu

21.8 DBDeleteRecord

Syntax

STATUS **DBDeleteRecord**(U

Description

Removes a particular record
associated memory.

Return value

Name
PPSM_OK
PPSM_ERROR
PPSM_ERR_DB_DB
PPSM_ERR_NUM_F

Hint

This tool is meant to
ordered record list.

For DBAddRecToTo
created by calling DB
recID passed must b

21.4 DBAppendReco

Syntax

STATUS **DBAppend**
outRecI

Description

Appends a blank rec
the record identifier,
option to specify ad
range of *numFmt* is f

Parameter

Name
dbId
recId
numFmt
outRecId

Return value

Name
PPSM_OK

Name
PPSM_ERROR
PPSM_ERR_DB_DBID
PPSM_ERR_DB_RECID
PPSM_ERR_NUM_FMT

Hint

This tool is meant to facilitate
ordered record list.

For DBAppendRecord() to w
created by calling DBAdd() p
recID passed must be a valid

21.5 DBChangeStdData

Syntax

STATUS **DBChangeStdData**

Description

Changes the data in a pred
database.

Parameter

Name
dbId
recId
fieldId
data

Return Value

Name
PPSM_OK
PPSM_ERROR
PPSM_ERR_DB_DB
PPSM_ERR_DB_RE
PPSM_ERR_DB_FD

Hint

Adheres to the size l

The record and dat
environment.

21.6 DBChangeUnfDa

Syntax

STATUS **DBChange**
S32 *size*

Description

Changes the *data* in

Parameter

Name
dbId
recId
type
data
size

Return Value

Name
PPSM_OK
PPSM_ERROR
PPSM_ERR_DB_DB
PPSM_ERR_DB_RE

Parameter

Name
dbId
recId
fieldId
data

Return Value

Name
PPSM_OK
PPSM_ERR_DB_DBID
PPSM_ERR_DB_RECID
PPSM_ERR_DB_FDID

Hint

The database and record r
environment. If a user define
to check the return status for
does exist before using the r

21.13 DBReadTotalNumber

Syntax

STATUS **DBReadTotalNum**

Description

Reads the total number of da

Parameter

Name
numDB

Return Value

Name
PPSM_OK

Parameter

Name
dbId
recId

Return Value

Name
PPSM_OK
PPSM_ERR_DB_DB
PPSM_ERR_DB_RE

Hint

The database and
environment.

21.9 DBGetFirstRecID

Syntax

STATUS **DBGetFirs**

Description

Gets the record ID o

Parameter

Name
dbId
recId

Return Value

Name
PPSM_OK
PPSM_ERROR
PPSM_ERR_DB_DB

Hint

This tool is useful for implem

The database referred to mu

21.10 DBGetNextRecID

Syntax

STATUS **DBGetNextRecID**(
botListFlag)

Description

Gets the identifier of the rec
record is the last record, *ne*
botListFlag will be set.

Parameter

Name
dbId
recId
nextID
botListFlag

Return Value

Name
PPSM_OK
PPSM_ERROR
PPSM_ERR_DB_DBID
PPSM_ERR_DB_RECID

Hint

This tool is for implementing

The database and record r
environment

21.11 DBGetPrevRecID

Syntax

STATUS **DBGetPrev**
topListF

Description

Gets the identifier
specified record is t
recId, and the *topLis*

Parameter

Name
dbId
recId
prevID
topListFlag

Return Value

Name
PPSM_OK
PPSM_ERROR
PPSM_ERR_DB_DB
PPSM_ERR_DB_RE

Hint

This tool is useful fo

The database and
environment.

21.12 DBReadData

Syntax

STATUS **DBReadDa**

Description

Reads the formatted

Name
PPSM_ERROR
PPSM_ERR_DB_DBID
PPSM_ERR_DB_RECID
PPSM_ERR_DB_FDID
PPSM_NO_MATCH

Hint

Current implementation of tc
There is no provision for the

For good programming practi
a valid search is found befor

21.18 DBSecretFlag

Syntax

STATUS **DBSecretFlag**(U32

Description

Checks if the secret flag of a

Parameter

Name
dbId
sFlag

Return Value

Name
PPSM_OK
PPSM_ERROR
PPSM_ERR_DB_DBID

Hint

The database referred to mu

Name
PPSM_ERR_DB_RE

21.14 DBReadTotalNu

Syntax

STATUS **DBReadTo**

Description

Reads the total num

Parameter

Name
dbId
numRec

Return Value

Name
PPSM_OK
PPSM_ERR_DB_DB

Hint

The database referre

21.15 DBReadUnfData

Syntax

STATUS **DBReadUn**
P_S32 *s*

Description

Reads the *data* in the
pointer to the unform
also pass back the *ty*

Parameter

Name
dbId
recId
type
data
size

Return Value

Name
PPSM_OK
PPSM_ERROR
PPSM_ERR_DB_DBID
PPSM_ERR_DB_RECID

Hint

The database and record r
environment.

It is good practice for user to
using the *data*.

21.16 DBRecordSecret

Syntax

STATUS **DBRecordSecret**(

Description

Checks if the secret flag of a

Parameter

Name
dbId
recId

Name
sFlag

Return Value

Name
PPSM_OK
PPSM_ERROR
PPSM_ERR_DB_DB
PPSM_ERR_DB_RE

Hint

The database and
environment.

21.17 DBSearchData

Syntax

STATUS **DBSearch**

Description

Searches though the
that matches the s
operation. If PPSM_
identifier of the recor

Parameter

Name
dbId
fieldId
data
recId

Return Value

Name
PPSM_OK

Return Value

Name
PPSM_OK
PPSM_ERR_TEXT_ID

22.3 TextMap

Syntax

STATUS **TextMap**(U32 *temp*

Description

Displays the given text onto t
text template identified by *te*

The text will be displayed sta
font type, output style and
template. Text that extends b
by the text template will be tr
automatically updated by the

Parameter

Name
templateId
buffer
numChar

Return Value

Name
PPSM_OK
PPSM_ERR_TEXT_ID
PPSM_ERR_TEXT_CUR

22.4 TextReadCursor

Syntax

STATUS **TextReadCursor**(U

Description

21.19 DBSetRecordSe

Syntax

STATUS **DBSetRec**

Description

Sets the secret flag

Parameter

Name
dbId
recId
sFlag

Return Value

Name
PPSM_OK
PPSM_ERROR
PPSM_ERR_DB_DB
PPSM_ERR_DB_RE
PPSM_ERR_DB_SF

Hint

The database and
environment.

Note that when a ne
the secret flag of the
in the database to be
after DBAdd() is call

21.20 DBSetSecretFlag

Syntax

STATUS **DBSetSecr**

Description

Sets the secret flag of a
subsequently in the specifie

Parameter

Name
dbId
sFlag

Return Value

Name
PPSM_OK
PPSM_ERROR
PPSM_ERR_DB_DBID
PPSM_ERR_DB_SFLAG

Hint

The database referred to m
User can use this flag to impl

Note that the secret flag is
wants all records in the
DBSetSecretFlag() immediat

Chapter 22 Text Tools

22.1 TextCreate

Syntax

STATUS **TextCreate**

Description

Creates and initialize
for the created text t

This is the first text
displayed on the pa
further references to

Parameter

Name
templateId

Return Value

Name
PPSM_OK
PPSM_ERR_TEXT_

22.2 TextDelete

Syntax

STATUS **TextDelete**

Description

Deletes a text templa
text template is not r

Parameter

Name
templateId

Name
pFontAttr

Return Value

Name
PPSM_OK
PPSM_ERR_TEXT_ID
PPSM_ERR_TEXT_FONT
PPSM_ERR_NO_MEMORY

22.8 TextSetOutlook

Syntax

STATUS **TextSetOutlook**(U

Description

Sets up the output style and
values. Subsequent text map
these new settings.

The output style is defined
character bitmap and the im
bitmap will be displayed. Five
replace, OR with, AND with,
existing image.

Up to four grey levels are cur
grey levels supported are wh
grey levels supported are wh

Reads the current cl
templateId. The char
text display area.

Parameter

Name
templateId
cursor

Return Value

Name
PPSM_OK
PPSM_ERR_TEXT_

22.5 TextSetCursor

Syntax

STATUS **TextSetCu**

Description

Sets the current cha
templateId to the new
positions to set to
characters - 1).

Parameter

Name
templateId
cursor

Return Value

Name
PPSM_OK
PPSM_ERR_TEXT_
PPSM_ERR_TEXT_

22.6 TextSetDisplay

Syntax

STATUS **TextSetDisplay**(U3
U16 *height*, U16

Description

Sets up the text display layou
given values. The text displa
text template. Subsequent
displayed with the new layou

The text display layout spe
panning screen. The size of t
the size of the font type sp
character cursor positions to
number of characters - 1).

Parameter

Name
templateId
xPos
yPos
width
height
cursor

Return Value

Name
PPSM_OK
PPSM_ERR_TEXT_ID
PPSM_ERR_TEXT_X
PPSM_ERR_TEXT_Y
PPSM_ERR_TEXT_WIDTH

Name
PPSM_ERR_TEXT_
PPSM_ERR_TEXT_

22.7 TextSetFont

Syntax

STATUS **TextSetFo**

Description

Sets up the font attril
using this text templa

Eight font types are
8 pixels English fonts
fonts. GB Normal is
is the same as GB N
16 Chinese font in B
BIG5 code format.

Note: Asian fonts

The specified font wi
type is specified. The
font libraries being p

The attribute field is

Parameter

Name
templateId

Parameter

Name
templateId
outputStyle
greyLevel

Return Value

Name
PPSM_OK
PPSM_ERR_TEXT_
PPSM_ERR_TEXT_
PPSM_ERR_TEXT_

22.9 TextSetup

Syntax

STATUS **TextSetup(**
greyLev

Description

Sets up the font type
template with the g
template will be disp

This tool does not support t
type and is provided for back
used, default font size of 16 :

Please refer to TextSetDispla
(*Section 22.7 - TextSetF
TextSetOutlook*) for detailed

Parameter

Name
templateId
fontType
outputStyle
greyLevel
xPos
yPos
width
height

Return Value

Name
PPSM_OK
PPSM_ERR_TEXT_ID

Name
PPSM_ERR_TEXT_
PPSM_ERR_TEXT_S
PPSM_ERR_TEXT_C
PPSM_ERR_TEXT_X
PPSM_ERR_TEXT_Y
PPSM_ERR_TEXT_W
PPSM_ERR_TEXT_

22.10 TextUnmap

Syntax

STATUS **TextUnmap**

Description

Clears the entire tex

Parameter

Name
templateId

Return Value

Name
PPSM_OK
PPSM_ERR_TEXT_

Return Value

Name
PPSM_OK
PPSM_ERR_YEAR
PPSM_ERR_MONTH
PPSM_ERR_DAY
PPSM_ERR_HOUR
PPSM_ERR_MINUTE
PPSM_ERR_SECOND

23.6 AlarmSetId

Syntax

STATUS **AlarmSetId**(P_U32
hour, U16 *minut*

Description

Set alarm at specific time an
alarm time, a message with t
task is swapped out or the s
swapped in and the system
tasks happen, the earlier the
in. If alarmId is 0, no alarm i
several alarms are set at the
task which set the alarm first
alarm will stop in the task w
messages are sent to other t

Parameter

Name
alarmId
year
month
date
hour

Chapter 23 Timer Tools

23.1 AlarmClear

Syntax

void **AlarmClear**(voi

Description

Clear all alarms set
other tasks.

Parameter

Name
None

Return Value

Name
None

23.2 AlarmClearId

Syntax

void **AlarmClearId**(U

Description

This function will cle
alarm set can be of a

Parameter

Name
alarmId

Return Value

Name
None

23.3 AlarmRead

Syntax

STATUS **AlarmRead**(P_U16
P_U16 *minute*, P

Description

Reads the up coming alarm s

Parameter

Name
year
month
day
hour
minute
second

Return Value

Name
PPSM_OK
PPSM_ERR_NO_ALARM

23.4 AlarmReadId

Syntax

STATUS **AlarmReadId**(U32
P_U16 *hour*, P_

Description

This function will read the ala

Parameter

Name
alarmId
year
month

Name
day
hour
minute
second

Return Value

Name
PPSM_OK
PPSM_ERROR
PPSM_ERR_NO_AL

23.5 AlarmSet

Syntax

STATUS **AlarmSet**(U
second)

Description

Sets the *year*, *month*
set, the application w
when the specified ti

Parameter

Name
year
month
day
hour
minute
second

For DragonBall not DragonE
pen input sampling rate. A h
accurate result.

Parameter

Name
millisecond

Return Value

Name
PPSM_OK
PPSM_ERR_TMOUT_VALUE

23.11 RefFineTimeAlarm

Syntax

STATUS **RefFineTimeAlarm**

Description

Sets up an alarm time with re
microseconds. Maximum p
0x7FFFFFFF/10 millisecond.

Parameter

Name
alarmTime

Return Value

Name
PPSM_OK

23.12 RefFineTimeAlarmId

Syntax

Name
minute
second

Return Value

Name
PPSM_OK
PPSM_ERR_YEAR
PPSM_ERR_MONTH
PPSM_ERR_DAY
PPSM_ERR_HOUR
PPSM_ERR_MINUT
PPSM_ERR_SECON

23.7 DateTimeRead

Syntax

STATUS **DateTimeR**
P_U16 *

Description

Gets the system dat

Parameter

Name
year
month
day
hour
minute
second

Return Value

Name
PPSM_OK

Name
PPSM_ERR_YEAR
PPSM_ERR_MONTH
PPSM_ERR_DAY
PPSM_ERR_HOUR
PPSM_ERR_MINUTE
PPSM_ERR_SECOND

23.8 DateTimeSet

Syntax

STATUS **DateTimeSet**(U16
U16 *second*)

Description

Sets the system date and tim

Parameter

Name
year
month
day
hour
minute
second

Return Value

Name
PPSM_OK
PPSM_ERR_YEAR
PPSM_ERR_MONTH
PPSM_ERR_DAY
PPSM_ERR_HOUR
PPSM_ERR_MINUTE
PPSM_ERR_SECOND

23.9 DeleteTimer

Syntax

STATUS **DeleteTime**

Description

Delete the timer in ti
task as far as the tim

Parameter

Name
timerId

Return Value

Name
PPSM_OK
PPSM_ERROR

23.10 InputTimeout

Syntax

STATUS **InputTime**

Description

Sets the repetitive ti
time-out routine is a
device.

Once this time-out r
argument list is set a
is detected. If the t
occurred, PPSM wi
otherwise, the time-o
This is a repetitive tim
each pen input strok

To disable the time-o
value allowed is 100

In IrptGetData(), the
timeout from normal

This routine takes in two refe
between the two. This routin
32-bit continuous reference v
in unit of milliseconds.

Parameter

Name
beginTime
endTime

Return Value

Name
N/A

23.18 RefTimeRead

Syntax

U32 **RefTimeRead**(void)

Description

Returns the current 32-bit
milliseconds. The calling of
RefFineTimeRead().

Parameter

Name
void

Return Value

Name
N/A

23.19 SetPeriod

Syntax

STATUS **RefFineTi**

Description

Sets up an alarm tim
microseconds. Maxi
0x7FFFFFFF/10 mill

Parameter

Name
alarmId
alarmTime

Return Value

Name
PPSM_OK

23.13 RefFineTimeDiff

Syntax

U32 **RefFineTimeDi**

Description

This routine takes in
between the two time
the 32-bit continuous
are in unit of 100 mic

Parameter

Name
beginTime
endTime

Return Value

Name
N/A

23.14 RefFineTimeRead

Syntax

U32 **RefFineTimeRead**(void

Description

Returns the current 32-bit r
microseconds. This is recom
it doesn't involve much calcul
will need more CPU time
millisecond.

Parameter

Name
void

Return Value

Name
N/A

23.15 RefTimeAlarm

Syntax

STATUS **RefTimeAlarm**(U3

Description

Sets up an alarm time with
milliseconds. Maximum per
0x7FFFFFFF/10 milliseconds

Parameter

Name
alarmTime

Return Value

Name
PPSM_OK

23.16 RefTimeAlarmId

Syntax

STATUS **RefTimeAl**

Description

Sets up an alarm ti
milliseconds. Maxim
0x7FFFFFFF/10 mill

Parameter

Name
alarmId
alarmTime

Return Value

Name
PPSM_OK

23.17 RefTimeDiff

Syntax

U32 **RefTimeDiff**(U3

Description

STATUS **SetPeriod(**

Description

Sets the periodic int
send a periodic in
messages, to the ca

Parameter

Name
period

Return Value

Name
PPSM_OK
PPSM_ERR_PERIO

23.20 SetPeriodId

Syntax

STATUS **SetPeriodI**

Description

This function will se
interrupt is available
going to be killed w
The returned value c

Parameter

Name
alarmId

Name
period

Return Value

Name
PPSM_OK
PPSM_ERR_PERIOD
PPSM_ERR_NO_MEMORY

23.21 Timeout

Syntax

STATUS **Timeout**(U32 *millis*

Description

General timer time-out routin

The application that calls tl
PPSM. A time interrupt is ge
expired.

Maximum allowed value for

Parameter

Name
millisecond

Return Value

Name
PPSM_OK
PPSM_ERR_TMOUT

23.22 TimeoutId

Syntax

STATUS **TimeoutId**(

Description

General timer time-o

The application that
PPSM. A time interr
expired with the time

Maximum allowed va

Parameter

Name
timerId
millisecond

Return Value

Name
PPSM_OK
PPSM_ERR_TMOUT

Return Value

Name
PPSM_OK
PPSM_ERROR

24.6 TaskMemUsed

Syntax

STATUS **TaskMemUsed**(U3

Description

Inquire memory usage of a ta
of bytes of memory allocate
Lrealloc().

Parameter

Name
taskId
pSizeUsed

Return Value

Name
PPSM_OK
PPSM_ERR_TASK_ID

24.7 TaskStackAvail

Syntax

S32 **TaskStackAvail**(void)

Description

PPSM returns to the caller the to
current task when calling TaskSta
has not been used up, negative v

Chapter 24 Memory Man

24.1 Lcalloc

Syntax

void ***Lcalloc**(U32 *s*

Description

Dynamic allocation c
caller a pointer to a
the Linker Specifica
memory, in number
memory available in

The memory pointed

Parameter

Name
size

Return Value

Name
N/A

24.2 Lfree

Syntax

void **Lfree**(void **ptr*)

Description

Returns the memor
supplied must be a
allocation tools.

Parameter

Name
ptr

Return Value

Name
None

24.3 Lmalloc

Syntax

void ***Lmalloc**(U32 *size*)

Description

Dynamic allocation of run-tim
caller a pointer to a region o
the Linker Specification File
memory, in number of bytes
memory available in the syst

The memory pointed to by th

PPSM returns the size of th
when calling Lmalloc(LARG

Parameter

Name
size

Return Value

Name
N/A

24.4 Lrealloc

Syntax

void ***Lrealloc**(void *

Description

Moving of memory.
from one location to
at the original locatio
the heap. The purpo

Parameter

Name
ptr
size

Return Value

Name
N/A

24.5 MoveBlock

Syntax

STATUS **MoveBlock**

Description

Copies a block of m
destination location.
Overlapping of mem

Parameter

Name
srcAddr
destAddr
size

Parameter

Name
milliSecond

Return Value

Name
PPSM_OK
PPSM_ERR_DOZE_TIME

25.3 SetDutyCycle

Syntax

U16 **SetDutyCycle**(U16 *per*

Description

This tool allows the applicat
within a system can hav
automatically changes the F
become active.

Parameter

Name
percentage

Parameter

Name
None

Return Value

Name
N/A

24.8 TotalMemSize

Syntax

U32 **TotalMemSize**(

Description

This routine returns
allocated through L
returned by this func
specified in the Linke

Parameter

Name
None

Return Value

Name
N/A

24.9 TotalMemUsed

Syntax

U32 **TotalMemUsed**

Description

Inquire run-time memory usa
the total number of bytes of r
through Lmalloc(), Lcalloc() c

Parameter

Name
None

Return Value

Name
N/A

Note: Value returned by To size of all memory re variables and strings function. For SDS us by the symbol lister, directory.

Chapter 25 Power Manag

25.1 SetDozeMode

Syntax

void **SetDozeMode**(

Description

Sets system to go in

Parameter

Name
None

Return Value

Name
None

25.2 SetDozePeriod

Syntax

STATUS **SetDozePe**

Description

Sets the countdown
mode. A value of ze
is no message to be
equals PPSM_NO_D
so no automatically g

Name
baudRate
parity
stopBits
charLen

Return Value

Name
PPSM_OK

Return Value

Name
U16

25.4 SetSleepMode

Syntax

void **SetSleepMode**

Description

Sets system to go in

Parameter

Name
None

Return Value

Name
None

25.5 SetSleepPeriod

Syntax

STATUS **SetSleepP**

Description

Sets the countdown
mode. A value of zer

Parameter

Name
second

Return Value

Name
PPSM_OK
PPSM_ERR_SLEEP_TIME

Chapter 26 UART Comm

26.1 UARTConfigure

Syntax

STATUS **UARTConf**
charLen

Description

Configures the UART
of *stopBits*, and *char*

The UART hardware
course of this config
aborted.

Both the normal NRZ
and maximum baud
per second correspo

Parameter

Name
mode

26.5 UARTReadData

Syntax

STATUS **UARTReadData**(P_

Description

Reads data received from th

An application can initiate a
UART by calling UARTRecei
receives data from the UAR
calling application. The calli
UARTReadData().

The calling application needs
the received data. PPSM wi
number of bytes of data reac

An error condition will be retu
permission to use the UART,
UARTReadData() is called.

If RTS/CTS is enabled, RTS
Data() and asserted after dat

Parameter

Name
pData
bufSize
sizeRead

Return Value

Name
PPSM_OK
PPSM_ERR_INVALID_ACCE
PPSM_ERR_NO_REQUEST

26.6 UARTReceive

Syntax

STATUS **UARTReceive**(U8

Name
PPSM_ERR_INVALI
PPSM_ERR_MODE
PPSM_ERR_BAUD
PPSM_ERR_PARITY
PPSM_ERR_STOPB
PPSM_ERR_CHARL

26.2 UARTFlowCtrl

Syntax

STATUS **UARTFlo**

Description

Enable or disable ha
can handle data con
hardware flow contr
UARTFlowCtrl() with

Parameter

Name
controlType

Return Value

Name
PPSM_OK
PPSM_ERR_INVALI
PPSM_ERROR

26.3 UARTInquire

Syntax

void **UARTInquire**(P_U8 *mo*
stopBits, P_U8 *c*

Description

Returns the current operating
charLen settings of the UAR

The returned values, excep
UARTConfigure() (*Section*
returned is in unit of bits per

Parameter

Name
mode
baudRate
parity
stopBits
charLen

Return Value

Name
None

26.4 UARTRcvCtrl

Syntax

STATUS **UARTRcvCtrl**(U8

Description

Pause or continue receivin
application can pause or
hardware control is enabled.
flow control is not enabled.

Parameter

Name
controlType

Return Value

Name
PPSM_OK
PPSM_ERR_INVALI
PPSM_ERR_RCTS_

Return Value

Name
PPSM_OK
PPSM_ERR_INVALID_ACCE
PPSM_ERR_RCTS_IDLE

26.10 UARTSetDelay

Syntax

STATUS **UARTSetDelay**(U

Description

In order to communicate with app
transmitting data in a burst of pul
racy of transmission. This functio
between each transmission of all
interrupts).

The range of delay values suppo

Parameter

Name
type
delay

Return Value

Name
PPSM_OK
PPSM_ERR_INVALID_ACCE
PPSM_ERR_INVALID_TXDEI

Description

Initiates or aborts a U

If *receiveFlag* is UA
initiated, and PPSM

If *receiveFlag* is UAI
be aborted.

When PPSM receiv
application notifying
read the received
UARTReadData).

Application task swa
and re-enabled after

Parameter

Name
receiveFlag

Return Value

Name
PPSM_OK
PPSM_ERR_INVALI
PPSM_ERR_BUSY

26.7 UARTSend

Syntax

STATUS **UARTSend**

Description

Initiates or aborts a U

If *sendFlag* is UART_
PPSM will start send

If *sendFlag* is UAR
aborted.

The calling application need
PPSM for sending out.

When PPSM finishes sendin
calling application notifying it

Application task swapping is
and re-enabled after the sen

Parameter

Name
sendFlag
pData
dataLen

Return Value

Name
PPSM_OK
PPSM_ERR_INVALID_ACCE
PPSM_ERR_BUSY

26.8 UARTSendAbort

Syntax

STATUS **UARTSendAbort**
sendSize)

Description

Terminate the current UART
the same as calling UART
aborting the ongoing transmi
current position of internal tra
have been sent. Caller ca
transmission by calling UAR

Parameter

Name
abortFlag
pSendData
sendSize

Return Value

Name
PPSM_OK
PPSM_ERR_INVALI

26.9 UARTSendCtrl

Syntax

STATUS **UARTSen**

Description

Pause or continue
application can pau
hardware control is e
flow control is not en

Parameter

Name
controlType

Name
newScreen
screenWidth
screenHeight
bitmap

Return Value

Name
PPSM_OK
PPSM_ERR_TASK_ID
PPSM_ERR_TASK_FLAG
PPSM_ERR_TASK_WIDTH
PPSM_ERR_TASK_HEIGHT
PPSM_ERR_COORDINATE
PPSM_ERR_NO_MEMORY

27.2 AppSwap

Syntax

void **AppSwap**(U16 *flag*)

Description

If *flag* is FALSE, no task swa
on application icon nor us
AdvSendMessage() are cal
message will still be sent to t

26.11 UARTTimeout

Syntax

STATUS **UARTTime**

Description

Sets the maximum
interrupts.

This time-out functio
data stream terminat

System do not initia
starts timeout with
UARTReceive(), UA

The valid range of ti
of milliseconds). A
function related to U

Parameter

Name
tmout

Return Value

Name
PPSM_OK
PPSM_ERR_INVALI
PPSM_ERR_INVALI

Chapter 27 Task Handlir

27.1 AdvTaskCreate

Syntax

STATUS **AdvTaskCr**
ySrc, S1
screenV

Description

Creation of a new PP
same manner as the
It allows the caller to

- The launch icon
off screen. For e
to be created.
- The stack memo
- The panning scr
associated scree

The default screen
LCDVIRTWIDTH an

Parameter

Name
taskId
procAddr
xSrc
ySrc
xDest
yDest
stackSize

Description

To set the specific task to l
swapped in, it will start fro
generally called once the tas

Parameter

Name
taskId
flag

Return Value

Name
PPSM_OK
PPSM_ERR_TASK_ID

27.7 TaskStart

Syntax

STATUS **TaskStart**(U32 *task*

Description

Begin execution of the first a
launches the first PPSM app
activating the application ico
system.

Parameter

Name
taskId

Return Value

Name
PPSM_ERROR
PPSM_OK

ignored. Moreover, i
of AppSwap(TRUE)

Parameter

Name
flag

Return Value

Name
None

27.3 SubTaskCreate

Syntax

STATUS **SubTaskC**
U16 *nur*

Description

Creating a sub-task.
task is itself a sub-ta
sub-task and the ne
sub-task has been c
the head of the sub
task a parent task c
change in run-time.

This routine accepts
passed into the sub-
can accept input arg

Parameter

Name
taskId
procAddr
stackSize
numArg

Name
...

Return Value

Name
PPSM_OK
PPSM_ERR_NO_MEMORY

27.4 TaskCreate

Syntax

STATUS **TaskCreate**(P_U32
S16 *xDest*, S16

Description

PPSM needs to know the e
stage. The main body of a P
application. PPSM will create
required to run the applicatio

Parameter

Name
taskId
procAddr
xSrc
ySrc
xDest
yDest
bitmap

Return Value

Name
PPSM_OK

Name
PPSM_ERR_TASK_
PPSM_ERR_COORD
PPSM_ERR_NO_ME

27.5 TaskHook

Syntax

STATUS **TaskHook**(

Description

Set the entry and exi
to be swapped in, en
is going to be swapp
and exitCallback sho
function should not
newApp) where oldA
and newApp will be t

Parameter

Name
taskId
entryCallback
exitCallback

Return Value

Name
PPSM_OK
PPSM_ERR_TASK_

27.6 TaskReInit

Syntax

STATUS **TaskReInit**

All data that the sender wants
structure. No protocol or data
and receiver must have a mu
the message being sent.

The data structure for the str

```
typedef struct _MESSAGE
        {
                U16     messageType;
                U16     message;
                U32     misc;
                P_VOID data;
                U16     size;
                U16     reserved;
        } PPSM_MESSAGE, *P_MESSA
```

If AppSwap(FALSE) is called
be sent but any form of task

If the system is in doze mode

Parameter

Name
taskId
msg
flag

27.8 TaskTerminate

Syntax

STATUS **TaskTermi**

Description

Termination of a tas
system memory, suc
the task and its subta

A task cannot ter
TaskTerminate() will
the task with Lmalloc

Parameter

Name
taskId

Return Value

Name
PPSM_OK
PPSM_ERR_TASK_

Chapter 28 Inter-Task M

28.1 AdvMessageDel

Syntax

STATUS **AdvMessa**

Description

This function will de
specific task, type a
0xFFFFFFFF in task
that field. SO if task
messages in all task

Parameter

Name
task
type
shortData

Return Value

Name
PPSM_ERR_TASK_
PPSM_OK

28.2 AdvSendMessag

Syntax

STATUS **AdvSendM**

Description

This tool is used wh
task. If the receiver t

If msg is 0, no me
executed.

Return Value

Name
PPSM_OK
PPSM_ERR_TASK_
PPSM_ERR_NO_ME
PPSM_ERROR

28.3 MessageDelete

Syntax

STATUS **MessageD**

Description

This function will del
specific type matche
"don't care" in that fi

Parameter

Name
type

Return Value

Name
PPSM_ERR_TASK_
PPSM_OK

28.4 SendMessage

Syntax

STATUS **SendMess**

Description

This tool is used wh
task. If the receiver t

All data that the sender wants
structure. No protocol or data
and receiver must have a mu
the message being sent. O
message received.

The data structure for the str

```
typedef struct _MESSAGE
      {
            U16    messageType
            U16    message;
            U32    misc;
            P_VOID data;
            U16    size;
            U16    reserved;
      } PPSM_MESSAGE, *P_MESSA
```

If AppSwap(FALSE) is called
be sent but any form of task

If the system is in doze mode

Parameter

Name
taskId
msg

Corresponding values betwee

SendMessage
STATUS **SendMessage**(U3 STATUS **IrptGetData**(P_U3
msg.messageType
msg.message
msg.misc

SendMes
msg.data
msg.size
msg.reserved

Return Value

Name
PPSM_OK
PPSM_ERR_TASK_
PPSM_ERR_NO_ME
PPSM_ERROR

Parameter

Name
handlerFlag

Return Value

Name
PPSM_OK
PPSM_ERR_RELEASE

29.3 IrptRequest

Syntax

U32 **IrptRequest**(U32 *handl*

Description

This tool is used by the appli
handlers. Once requested a
handlers are directed to
identifiers.

When calling this tool, the us
request. The interrupt flags
handler with a single call. If t
PPSM system will hook the
nothing.

The return value from this
handlers that have been gra
handler flag parameter. For e
set of handlers, and if all are
the same as the input hanc
handlers cannot be granted,
parameter. If none is granted

Chapter 29 Interrupt Ha

29.1 IrptGetData

Syntax

STATUS **IrptGetDat**

Description

This tool reads the
message. The interr
interrupt handler is
buffer.

The data returned fr
Different messages f
data. The pre-define
listed below. The siz
in the last argument

PPSM does not imp
from User Defined h
integrator.

Parameter

Return Value	
IRPT_AUDIO	N/
IRPT_HWR	N/
IRPT_ICON	U3
IRPT_INPUT_ STATUS	U3
IRPT_KEY	N/

Return Value	*sData
IRPT_PEN	U32 - Areal
IRPT_RTC	U32 - Timer
IRPT_TIMER	U32 - Timer
IRPT_UART	N/A
IRPT_USER	User define

Return Value

Name
IRPT_AUDIO
IRPT_ERROR
IRPT_HWR
IRPT_ICON
IRPT_INPUT_STATUS
IRPT_INT
IRPT_IRQ1
IRPT_IRQ2
IRPT_IRQ3
IRPT_IRQ6
IRPT_KEY
IRPT_NONE
IRPT_PEN
IRPT_PWM
IRPT_RTC

Name
IRPT_SPIM
IRPT_SPIS
IRPT_TIMER
IRPT_UART
IRPT_USER
IRPT_WDG

29.2 IrptRelease

Syntax

STATUS **IrptReleas**

Description

This tool is used by th
has successfully re
released must be a
the IrptRequest() too
requested for releas
released, any data o
out and removed.

Once an interrupt ha
other applications th

The application shou

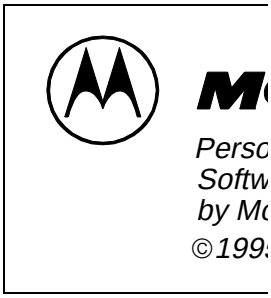


Figure 29-

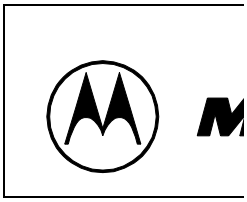


Figure 2

Parameter

Name
calibration

Return Value

Name
PPSM_OK
PPSM_ERR_NO_MEMORY
PPSM_ERROR

A request is success
granted to another ta

Parameter

Name
handlerFlag

Return Value

Name
N/A

29.4 IrptSendData

Syntax

STATUS **IrptSendData**(U16

Description

Passes the user defined in
application level. This routine
handler only. After this me
application that has requeste
via the IrptGetData() tool.

Parameter

Name
irptType
sData
data
size

Return Value

Name
PPSM_OK
PPSM_ERR_IRPT_HANDLEF
PPSM_NO_MEMORY

Chapter 30 System Tool

30.1 PPSMInit

Syntax

STATUS **PPSMInit**(U

Description

PPSM initialization r
system file before ar

The input argument
required at this time.

With the default calib
screen, one near the
The user must press
the pen input calibra

The default calculati
pixels (physical sizes
of 100 (in A/D outpu
edge of the touch p
changing CalibrateP

If logo displaying is
screen during the p
dimensions, a differe
screen, one for a sm
or the LCD height is

Width (pixe
width => 28
150 <= width <
width < 15

31.2 AudioInUse

Syntax

U8 **AudioInUse**(void)

Description

It checks if PPSM audio tools

Parameter

Name
None

Return Value

Name
AUDIO_OFF
WAVE_IN_USE
TONE_IN_USE

31.3 AudioPlayTone

Syntax

STATUS **AudioPlayTone**(P_
U8 autoRepeat)

Description

PPSM plays a sequence of di
having a fixed duration.

Parameter

Name
toneData
toneSize

30.2 ReadSMVersion

Syntax

STATUS **ReadSMVe**

Description

Returns PPSM majo
major number will be

Parameter

Name
major
minor

Return Value

Name
PPSM_OK
PPSM_ERR_NO_ME

Chapter 31 Audio Tools

31.1 AdvAudioPlayW

Syntax

STATUS **AdvAudioP**
U8 repe

Description

This tool is valid for
wave signal. This is
configuration details.

- The value of pre
- The repeat rate
- The clksel in the

This tool assumes th
module. For most ca

Parameter

Name
waveData
waveSize
prescaler
repeat
clksel

Return Value

Name
PPSM_OK
PPSM_ERR_AUDIO
PPSM_ERR_AUDIO

Name
toneDuration
autoRepeat

Return Value

Name
PPSM_OK
PPSM_ERR_AUDIO
PPSM_ERR_AUDIO

31.4 AudioPlayWave

Syntax

STATUS **AudioPlay**
sampling

Description

This tool is for Drag
signal with requested

Parameter

Name
waveData

Name
waveSize
samplingRate

Return Value

Name
PPSM_OK
PPSM_ERR_AUDIO_INUSE

31.5 AudioStopTone

Syntax

STATUS **AudioStopTone**(vc

Description

Terminates the tone playing.

Parameter

Name
None

Return Value

Name
PPSM_OK
PPSM_ERR_AUDIO

31.6 AudioStopWave (Drag

Syntax

STATUS **AudioStopWave**(v

Description

Terminates the wave

Parameter

Name
None

Return Value

Name
PPSM_OK
PPSM_ERR_AUDIO

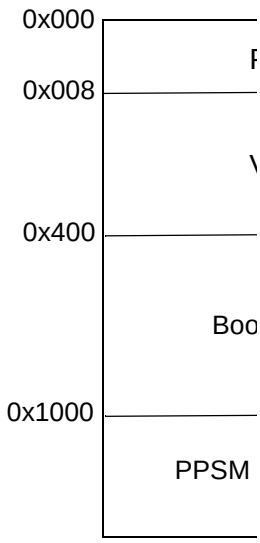


Figure 30-1 Me

32.1.1.1 ROM_RESET

This is used to map the 68K first
as:

```
SECTION     rom_reset
DC.L        MON_STACKTOP
DC.L        rom_start-ROMADDR
DCB.L       254,0
```

The labels MON_STACKTOP an
with their absolute address only
values for these locations dynam
the absolute location and size of

ROMADDR is declared in the Lir

32.1.1.2 ROM_CODE

This regions is declared to store
part of PPSM library, they are de
memory map to avoid memory c

The first line of this region MUST
the region rom_reset to work out

The last line of this region shoulc
start PPSM start-up code. The la
the startup code.

Part IV
System
Integrat
Guide

Chapter 32 How to make

To make PPSM applicati
different from running PP
items are:

- Boot Strap code
- Linker Specificat

This chapter gives some
PPSM applications.

32.1 Boot Strap Code

The boot strap code perf

- Starts the 68K c
- Map the chip-sel
  platform
- Initialization of p
- Jump into PPSM

Depending on the size a
boot.s need to be chang

32.1.1 68K Start-up

In 68K architecture, the f
0x400, are reserved for s
random values. The first
defined for the start prog
reset.

In order to make this ass
than hard-coding the add
and rom_code, are defin
mapping.

32.1.2 Chip Selects

For the M68328ADS dev
and Chip-Select group B
Integrated Processor Us
programming.

32.1.3 Peripheral Devices

Initialization of the periph
controller. Please refer to
MC68328UM/AD, for det

32.2 Linker Supplicat

The Linker Supplications
main difference being tha
address, and some regio
are Read-Only, such as
Read/Write regions, such

The listing below shows

Example 30-1 Linker

```
partition { overlay {
       region {} rom_r
       region {} rom_c
       region {} code[
       region {} const
       region {} strin
       DATA = $;

       LCDPHYSWIDTH =
       LCDPHYSHEIGHT =
       LCDVIRTWIDTH =
       LCDVIRTHEIGHT =
       UARTRCVBUF = 25
} area2;
} ROM[addr=0x400000,si
partition { overlay {
       region {} data[
       region {} ram[r
       region {} mallo
       region {} stack
       STKTOP = $;
} area1; } RAM[addr=0x
```

In this example, a systen
location 0x400000 and 1
0x0 has the following cha

- The ROM area s
- The region rom_
address, which i
- The region rom_
address, which i

- As much executable cod
byte boundary starting fr
- As much constant data s
boundary
- As much constant string
byte boundary
- DATA symbol to point to
constants to pre-initialize
- A LCD physical display s
- A panning screen of 640
- A 256 byte internal UAR
- The RAM area starts at
- As much initialized data
0x400, round to 4-byte b
- As much zeroed uninitial
byte boundary
- 512 KByte of heap space
- 128 KByte of stack spac
- A STKTOP symbol to po

32.3 Generating S-Record

After the PPSM application has l
generates an output file in a prop
ROMs.

SDS provides a tool, the loader t
file into S-Record format.

32.3.1 Loader Options

To convert .OUT file into S-Reco

Options		
	-d mot	ger
	-o <path>\<filename>.dwn	the
	-m data, DATA	Co
	-w <address>	Ge bas

32.3.2 Loader Commands

Convert .OUT file format
down -d mot <filename>.o

Example 30-2 Loade

```
down -d mot sample.out
```

This will convert the sam
be burned into ROM add

Example 30-3 Loade

```
down -d mot sample.out
```

This will convert the sam
be burned into ROM add

ADS version	Q3
M68328ADS	PJ3
M68EZ328ADS	PD3

33.2.1 Pen Initialization

Syntax

void **PenDevInit**(void)

Description

This function initializes SPI
sampling.

M68328ADS implementation:

Port J is used to control the
The default initialization value

Table

Port J	address
Direction Register	0xFFF428
Select Register	0xFFF42B

For SPI Master, the control r

Table

SPIM Control Register, 0xF
Data Rate
SPIM Enable
Exchange Bit
SPIM Interrupt Enable
Phase Shift
Polarity
Clock Count

M68EZ328ADS implementation

Chapter 33 Device Drive

PPSM supports device d
vendor dependent. Thes

- System configur
- Pen input driver
- LCD driver
- Handwriting reco
- Font driver
- UART driver

In compilation, PIXEL_1
corresponding device dri

33.1 System Configu

There are 2 system conf

- Boot Strap Drive
- Interrupt handler

33.1.1 Boot Strap Driver (

Description

The boot strap code
devices and to map c
at boot time. Differe
different boot strap
device driver library
selects for the M68
Specification File an
hardware configurati

For hardware chara
MC68328 Integrated

33.1.2 User Interrupt Hand

Description

This interrupt handle
handlers for certain

Users can replace th
driver to perform exc
these user-defined h

refer to *Chapter 15 - Interru*
Tools for details on interrupt

Please refer to the *MC*
MC68328UM/AD, for details

A single argument is passe
supplied by PPSM system
before calling the user-def
locations of the registers rela

Table 3

Argument is the address

The following functio
void _SPIMIrptHandler(P_U32
void _SPISIrptHandler(P_U32
void _IRQ6IrptHandler(P_U32
void _UARTIrptHandler(P_U3
void _WatchdogIrptHandler(P_
void _KeyboardIrptHandler(P_
void _PWMIrptHandler(P_U32
void _INTIrptHandler(P_U32 s
void _IRQ3IrptHandler(P_U32
void _IRQ2IrptHandler(P_U32
void _IRQ1IrptHandler(P_U32

M68328ADS implementation:

These interrupt handlers pe
application immediately.

In general, _UARTIrptHand
licensee, a "-DNO_UART_H
to indicate that the internal
_UARTIrptHandler() is used

33.2 Pen Input Device

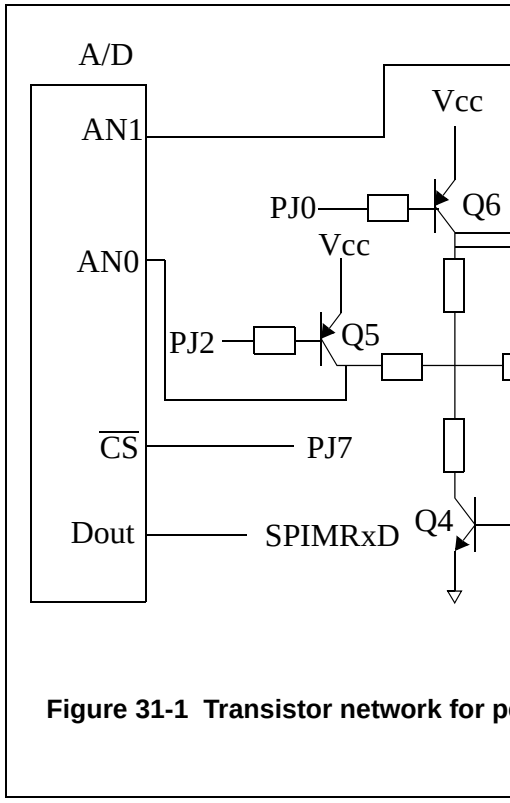


Figure 31-1 Transistor network for p

Figure 31-1 shows the c
pen input device:

- 4 I/O pins on por
network.
- I/O pin 7 on port
D convertor.
- The SPI Master
output for the sa

Please refer to the *M683*
hardware configuration.

There are 4 functions in

- Pen Initialization
- Pen Interrupt En
- Pen Interrupt Dis
- Pen Read Devic

For M68EZ328ADS, the
port D and E instead of p

For X sampling, the transistor
and Q5 are OFF, see *Figure*
the transistors are in this set
convertor for the sampling. T
stored in the SPI Data regist

For Y sampling, the proced
transistors Q3 and Q5 needs
see *Figure 31-1*, i.e. set Port

The X and Y samples thus
arguments passed in by the

33.3 Pen Calibration(PenIn

User normally needs to do calibra
a correct mapping between touch
coordination. The system needs
corner of the touch panel coordir
do this coordinate mapping.

Syntax

STATUS CalibratePen(U16

Description

When pen calibration is necessa
default pen calibration algorithm
routine, the origin and maximum
screen coordinate) should be fec
and PenSetInputOrg(x, y).

By default, the Motorola logo is d
and the bottom left corners, are u

33.4 LCD Device Drivers (l

Two functions are needed in this
LCD panel being used in the sys
functions will be called according

33.4.1 1 bit/pixel Initialization

Syntax

void **_LCDDev1**(void)

Description

This function initializes the
Application programmer ma

Port D and E are use
panel. The default in
O pins.

Port D	ad
Direction Register	0xF
Select Register	0xF
Pullup Register	0xF
Select Register	0xF
Polarity	0xF
INT Enable	0xF
INT Edge	0xF

Port E3 is initialized
initialized to SPM fur

For SPI Master, the

SPIM Control Regi
Data Rate
SPIM Enable
Exchange Bit
SPIM Interrupt Enabl
Phase Shift
Polarity
Clock Count

33.2.2 Pen Interrupt Enab

Syntax

void **PenIrptEnable(**

Description

This function enables

M68328ADS implementation:

Port M pin 6 is assigned as tl
sequence is required:

- Enable Port M pin 6 pull-
- Discharge the transistor
are off, i.e. set Port J Da
- Charge up the transistor
J Data register, 0xFFF43
- Switch Port M pin 6 to int
0xFFF448, bit 6 to 0
- Enable Interrupt Mask R

Please refer to the *M68328ADS*
switching.

M68EZ328ADS implementatior

IRQ5(port F1) is assigned
following sequence is require

- Enable Port F pin 1 pull-
- Discharge the transistor
are off, i.e. set Port D Da
- Charge up the transistor
D Data register, to 0x0D
- Switch Port F pin 1 to int
bit 1 to 0
- Enable Interrupt Mask R

Please refer to the *M68EZ328AD*
switching.

33.2.3 Pen Interrupt Disable

Syntax

void **PenIrptDisable**(void)

Description

This function disables the Pe

M68328ADS implementation:

Port M pin 6 is assigned as th
sequence is required:

- Disable Interrupt Mask R
- Switch Port M pin 6 to I/(
0xFFF448, bit 6 to 1
- Disable Port M pin 6 pull
interfere with A/D sampli
- Discharge the tra
are off, i.e. set P
- Switch transistor
0xFFF439, to 0x

Please refer to *M68328A*
switching.

M68EZ328ADS implem

IRQ5 is assigned a
sequence is required

- Disable Interrupt
- Switch Port F pir
to 1
- Disable Port F p
interfere with A/D
- Discharge the tra
are off, i.e. set P
- Switch transistor
to 0x05

Please refer to *M68EZ32*
switching.

33.2.4 Pen Read Device

Syntax

void **PenReadDevic**

Description

This function returns
transistor network ac

M68328ADS implement

For X sampling, the t
and Q5 are OFF, se
0xF9. While the trans
the A/D convertor fo
convertor is stored in

For Y sampling, the
transistors Q3 and Q
see *Figure 31-1*, i.e.

The X and Y sample
arguments passed in

M68EZ328ADS implem

33.6.1 Font Library Information

A data structure type FONTLIB is
libraries being used.

The FONTLIB type is defined as

```
typedef struct
      {
      P_U8   baseAddr;
      U16    fontType;
      U16    fontWidth;
      U16    fontHeight;
      U16    bitmapSize;
      } FONTLIB, *P_FONTLIB;
```

where:

1) baseAddr is the base ad
2) fontType is the font type
3) fontWidth is the width of
pixels
4) fontHeight is the height c
of pixels
5) bitmapSize is the amoun
font bitmap in unit of byte

Assuming there is font bitmap or
type, the default font library infor
follow:

```
FONTLIB fontLib[] =
      {
      {(P_U8)SMALL_ENG_FONT_AD
      {(P_U8)SMALL_ENG_FONT_AD
      {(P_U8)LARGE_ENG_FONT_AD
      {(P_U8)LARGE_ENG_FONT_AD
      {(P_U8)GB_FONT_ADDR, GB_
      {(P_U8)BITMAP_BIG5_FONT_
      {(P_U8)SCALABLE_BIG5_FON
      };
```

The fontLib data structure above
types.

33.6.2 Font Library or Font Gene

Syntax

void **FontInit**(void)

Description

This function initializes the
applicable. This function will

*Note: Font bitmaps librarie
whereas font genera
applicable, this drive*

module. Generally,
"move.b #$00,$21(a

33.4.2 2 bits/pixel Initializa

Syntax

void **_LCDDev2**(voi

Description

This function initiali
Application program
module. Generally,
"move.b #$00,$21(a

Please refer to the *MC68*
AD, for details on the LC

M68328ADS implement

_LCDDev1 initializes

- Panel Interface C
bus size and no
- Pixel Clock Divid
- Polarity Configur
the LCD panel u

_LCDDev2 initializes

- Panel Interface C
bus size with gra
- Pixel Clock Divid
directly
- Gray Palette Ma
0x3075
- Polarity Configur
the LCD panel u

33.5 Handwriting Rec

This driver is required fo
handwriting input while s
engines. The driver fun
handwriting recognition i

This driver consists of fo

33.5.1 Handwriting Recog

Syntax

void **ResetRecEngine**(void)

Description

This function resets the hanc

This function should call a re
recognition engine. In some
functions as InitRecEngine()
Initialization).

This function is called once w
call of OpenInputPad().

33.5.2 Handwriting Recognition

Syntax

void **InitRecEngine**(void)

Description

This function initializes or ins

This function should call ar
handwriting recognition eng
engine for the recognition pr

This function is called once a

33.5.3 Process One Stroke of Ha

Syntax

void **ProcessStroke**(U16 *nu*

Description

This function processes one
system.

Based on the input paramet
data for one stroke of input (
the corresponding handwritin
should then be sent to the ha
provided in the handwritir
optimization before the actua

This function is called for eve

Parameter

Name
numPoints
strokeData
inputAreaId

33.5.4 Initiate Character R

Syntax

void **RecognizeInpu**

Description

This function perform

This function should
handwriting recogni
processed stroke da
expected to return t
being recognized, ar

This function is calle
an input time-out.

Return Value

Name
numCandidates
candidates

*Note: Since PPSM
recognition e
default to pe*

33.6 Font Driver (font

This driver is required fo
supporting multiple font t
vendors.

The font driver consists o

Return Value

Name
None

Note: The driver function r
operation.

33.8.4 Disabling I/O ports when

Syntax

void **PortSleepDisable**(void)

Description

Just before PPSM goes into
user defined I/O ports that ar
in the code to disable the I/O

Return Value

Name
None

Note: The driver function r
operation.

routine prov

Since PPSM does r
function is default to

33.6.3 Font Accessing

Syntax

P_U8 **FontGetChar**

Description

This function returns
attributes and charac

Please refer to *Secti*
FONTATTR data stru

Font lookup or gene
supplier. This drive
corresponding font t
the font attributes.

Since PPSM include
method for mapping
font types that use th

- SMALL_NORMA
- SMALL_ITALIC_
- LARGE_NORMA
- LARGE_ITALIC_

Parameter

Name
pFont
code

Return Value

Name
N/A

33.7 UART Device Driver (u

This UART device driver is a sup
Tools as described in *Chapter 26*
driver contains the UARTDevSer
send the BREAK character. This
MC68328 UART hardware regist
restoring of appropriate system i
device driver.

33.7.1 Sending the BREAK Char

Syntax

void **UARTDevSendBreak**(U

Description

This function starts or stops t
character depending on the

If *sendBreak* flag is UART_
sending the BREAK characte

If *sendBreak* flag is UART_
sending the BREAK characte

It is assumed that *sendBrea*
value will be treated as UAR

Parameter

Name
sendBreak

Return Value

Name
None

33.8 Power Management D

4 functions are located in iodev.c
doze or sleep mode or leaving d

33.8.1 Enabling I/O ports

Syntax

void **PortDozeEnab**

Description

When PPSM wakes
user defined I/O port
in their own I/O initia

Return Value

Name
None

*Note: The driver fu
operation.*

33.8.2 Disabling I/O ports

Syntax

void **PortDozeDisab**

Description

Just before PPSM g
user defined I/O port
in the code to disabl

Return Value

Name
None

*Note: The driver fu
operation.*

33.8.3 Enabling I/O ports

Syntax

void **PortSleepEnab**

Description

When PPSM wakes
user defined I/O port
in their own I/O initia

Chapter 34 Linker Speci

PPSM makes use of the
development environmer
defines the locations and
symbols. It is written in a
files, please refer to the

PPSM required a few us
Table 32-1.

Table 32-1 Sym

Symbol Name
LCDPHYSWIDTH
LCDPHYSHEIGHT
LCDVIRTWIDTH
LCDVIRTHEIGHT
UARTRCVBUF

System integrators inform
hardware system using t

PPSM maintains a stack
The system integrator mu
and heap. In general, the
memory for application u
For details on memory m

PPSM also maintains an
storage for data received
is the optimal buffer size

34.1 .SPC File for a R

Following is an example
(for SDS use only) with 2
system has the following

- The RAM systen
- 4 KByte of reser
vectors and SDS
- As much executa
boundary
- As much constan
boundary

- As much constant strings
boundary
- As much initialized data
byte boundary
- As much zeroed memory
boundary
- 704 KByte of heap space
- 128 KByte of stack spac
- A STKTOP symbol to po
- a DATA symbol to point t
- A LCD physical display s
- A panning screen of 640
- A 256 byte internal UAR
- 2 MByte of RAM

Example 32-1 Linker Specif

```
partition { overlay {
       region {} reset[addr=0,s
       region {} code[roundsize
       region {} const[roundsiz
       region {} string[roundsi
       region {} data[roundsize
       region {} ram[roundsize=
       region {} malloc[size=0x
       region {} stack[size=0x2
       STKTOP = $;/* SP reset v
       DATA = $;/* "data" downl
       LCDPHYSWIDTH = 320;/* LC
       LCDPHYSHEIGHT = 240;/* L
       LCDVIRTWIDTH = 640;/* LC
       LCDVIRTHEIGHT = 480;/* L
       UARTRCVBUF = 256;
       #bytes) */

} example; } RAM[addr=0x0, siz
```

34.2 .SPC File for a ROM-R

The .SPC file for a ROM-RAM sy
the ROM layout and one partitior
to *Chapter 32 - How to make RC*
a system.

34.3 For SingleStep Debug

One of the methods to find out th
.SPC file will be shown below. Th
debugging system.

1) First of all, we set the RA
example,

```
partition { overlay {
    region {} reset[addr=0x0];
.....
    region {} malloc[s
......
} example; } RAM[addr=
```

2) Following error m

```
LINKER: error: actual
LINKER: error: overlay
NMAKE : fatal error U1
Stop.
```

3) Then, we can ca
exceeding bytes

```
0x
```

4) Finally, we can v

```
partition { overlay {
    region {} reset[ad
.....
    region {} malloc[s
......
} example; } RAM[addr=
```

5) This optimum ma
source code by s
found after debu

Chapter 35 Trap Usage i

PPSM tools access the P
When an application ma
is called with the correct
calling the actual functio

On the 68K, there are 16
instruction mapping for t

Tab

TRAP Number	T
1	
2	
3	
4	
5	
6	
7	

35.1 PPSM Tools Call

For the protection of the
is introduced in the PPS

The first level is a stub li
application developers. T
function body consists o

The second level is the T
in by the stub library and

With this implementation
PPSM system code can
portability of the PPSM s

35.2 TRAP Implemen

Figure 34-1 shows a blo
application makes a syst
sections are implemente

overhead induces during a syste

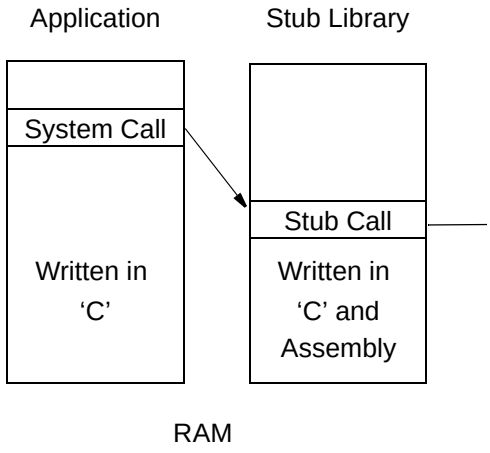


Figure 34-1 P

35.2.1 Application

Applications are written in ANSI C
calling convention. Because Sing
C source code, it is also allowed

35.2.2 Stub Library

The Stub Library provides an inte
routines. Its main function is for p
with the correct arguments. It is v
embedded format. Each function
shown below:

```
int    simplification( int a, i
       #define NO_OF_ARGUMENT2
       #define FUNC_NUMBER5
       #define TRAP_NUMBER3

       int    rv;    /*  return
       asm(
              "      LEA    8(A6
              "      MOVE.W #NO_(
              "      MOVE.W #FUN(
              "      TRAP   #TRA
              "      MOVE.L D0,{
       );
       return (rv);
       }
```

The first instruction that SingleSt
procedure is a "LINK" instruction
points to the stack that is used b
argument, an offset of 8 is added

the caller's argument list
the PPSM tools function

(+12)

(+8)

(+4)

A6

(-4)

Figure 34-2

35.2.3 TRAP Service Hand

The TRAP Service Hand
There are 7 jump tables,
The jump table consists
are used as the PPSM sy
handler when it receives
general code layout for a

```
.FREF  _Func_1,2
.FREF  _Func_2,8

JMPTABLE:
DC.L   _Func_1,

Trap_1:
LSL.W  #2,D1  ; M
ADD.L  D1,A0  ; M
LSR.W  #2,D1  ; M
loop_1:
TST.W  D1     ; A
BEQ    loop_2 ;
MOVE.L -(A0),-(A
SUBQ.W #1,D1  ; M
BRA    loop_1 ;
loop_2:
LEA    JMPTABLE,
LSL    #2,D0  ; (
ADD.L  D0,A0  ; M
MOVEA.L(A0),A1;
JSR    (A1)   ;
RTE
```

Appena

Table A-

Error Code
PPSM_ERR_DB_ADD
PPSM_ERR_DB_DBID
PPSM_ERR_DB_FDID
PPSM_ERR_DB_READNO
PPSM_ERR_DB_RECID
PPSM_ERR_DB_SFLAG
PPSM_ERR_DB_TYPE
PPSM_ERR_DOT_WIDTH
PPSM_ERR_DOZE_TIME
PPSM_ERR_DRAW_INIT
PPSM_ERR_FILL_PATTERN
PPSM_ERR_FILL_SPACE
PPSM_ERR_GREY
PPSM_ERR_HEIGHT
PPSM_ERR_HOUR
PPSM_ERR_ICON_TYPE
PPSM_ERR_INPUT_PAD_CLOSE
PPSM_ERR_INPUT_PAD_HEIGH
PPSM_ERR_INPUT_PAD_NOSCR
PPSM_ERR_INPUT_PAD_OPENE
PPSM_ERR_INPUT_PAD_WIDTH
PPSM_ERR_INPUT_PAD_X_POS
PPSM_ERR_INPUT_PAD_Y_POS
PPSM_ERR_INVALID_ACCESS
PPSM_ERR_INVALID_TMOUT
PPSM_ERR_IRPT_HDLER
PPSM_ERR_LCD_FONT
PPSM_ERR_LCD_GREY

Appendix A Error Code D

The following are the def

T

Error Cod
PPSM_OK
PPSM_ERROR
PPSM_ERR_ACTIVE_PC
PPSM_ERR_ACTIVE_PU
PPSM_ERR_AREA_COD
PPSM_ERR_AREA_ID
PPSM_ERR_AUDIO_INU
PPSM_ERR_AUDIO_RE
PPSM_ERR_AUDIO_SA
PPSM_ERR_AUDIO_TO
PPSM_ERR_BAUD
PPSM_ERR_BUSY
PPSM_ERR_CHARLEN
PPSM_ERR_COLOUR
PPSM_ERR_COORDINA
PPSM_ERR_CURSOR_I
PPSM_ERR_DAY

T

Error Cod
PPSM_ERR_LCD_RADIU
PPSM_ERR_LCD_STYLE
PPSM_ERR_LCD_X
PPSM_ERR_LCD_Y
PPSM_ERR_MINUTE
PPSM_ERR_MODE
PPSM_ERR_MONTH
PPSM_ERR_NO_ALARM
PPSM_ERR_NO_MEMO
PPSM_ERR_NO_REQUE
PPSM_ERR_NUM_FMT
PPSM_ERR_PAN_ADDR
PPSM_ERR_PAN_HEIGH
PPSM_ERR_PAN_INIT
PPSM_ERR_PAN_WIDTH
PPSM_ERR_PARITY
PPSM_ERR_PEN_GET
PPSM_ERR_PEN_INIT
PPSM_ERR_PEN_RATE
PPSM_ERR_PERIOD
PPSM_ERR_RCTS_IDLE
PPSM_ERR_REF_MAX
PPSM_ERR_RELEASE
PPSM_ERR_SECOND
PPSM_ERR_SKBD_USE
PPSM_ERR_SKBD_XSIZ
PPSM_ERR_SKBD_X_P
PPSM_ERR_SKBD_YSIZ

Table A-

Error Code
PPSM_ERR_SKBD_Y_POS
PPSM_ERR_SLEEP_TIME
PPSM_ERR_STOPBIT
PPSM_ERR_TASK_FLAG
PPSM_ERR_TASK_HEIGHT
PPSM_ERR_TASK_ID
PPSM_ERR_TASK_WIDTH
PPSM_ERR_TEXT_CR
PPSM_ERR_TEXT_CUR
PPSM_ERR_TEXT_FONT
PPSM_ERR_TEXT_GREY
PPSM_ERR_TEXT_HEIGHT
PPSM_ERR_TEXT_ID
PPSM_ERR_TEXT_STYLE
PPSM_ERR_TEXT_WIDTH
PPSM_ERR_TEXT_X
PPSM_ERR_TEXT_Y
PPSM_ERR_TMOUT_VALUE
PPSM_ERR_WIDTH
PPSM_ERR_X_POS
PPSM_ERR_YEAR
PPSM_ERR_Y_POS
PPSM_NO_MATCH

Appendix B List of Refer

1) *Addendum and*
Microprocessor
2) *Computer Graph*
Feiner and Huge
3) *CrossCode® C*
Development Sy
4) *MC68328ADS A*
5) *MC68328 Integr*
AD.Motorola, Inc
6) *SingleStep™ De*
Software Develo
7) *M68EZ328ADS*
Motorola, Inc.
8) *MC68EZ328 Int*

S

T

U

Appendix C Index of PPS

A

C

D

E

G

I

L

M

O

P

R