OSIF
API Documentation

30515ST10842A Rev. 3 – 2017-07-20

TELIT
TECHNICAL
DOCUMENTATION

[01.2017]

Mod. 0806

SPECIFICATIONS ARE SUBJECT TO CHANGE WITHOUT NOTICE

NOTICES LIST

While reasonable efforts have been made to assure the accuracy of this document, Telit assumes no liability resulting from any inaccuracies or omissions in this document, or from use of the information obtained herein. The information in this document has been carefully checked and is believed to be reliable. However, no responsibility is assumed for inaccuracies or omissions. Telit reserves the right to make changes to any products described herein and reserves the right to revise this document and to make changes from time to time in content hereof with no obligation to notify any person of revisions or changes. Telit does not assume any liability arising out of the application or use of any product, software, or circuit described herein; neither does it convey license under its patent rights or the rights of others.

It is possible that this publication may contain references to, or information about Telit products (machines and programs), programming, or services that are not announced in your country. Such references or information must not be construed to mean that Telit intends to announce such Telit products, programming, or services in your country.

COPYRIGHTS

This instruction manual and the Telit products described in this instruction manual may be, include or describe copyrighted Telit material, such as computer programs stored in semiconductor memories or other media. Laws in the Italy and other countries preserve for Telit and its licensors certain exclusive rights for copyrighted material, including the exclusive right to copy, reproduce in any form, distribute and make derivative works of the copyrighted material. Accordingly, any copyrighted material of Telit and its licensors contained herein or in the Telit products described in this instruction manual may not be copied, reproduced, distributed, merged or modified in any manner without the express written permission of Telit. Furthermore, the purchase of Telit products shall not be deemed to grant either directly or by implication, estoppel, or otherwise, any license under the copyrights, patents or patent applications of Telit, as arises by operation of law in the sale of a product.

COMPUTER SOFTWARE COPYRIGHTS

The Telit and 3rd Party supplied Software (SW) products described in this instruction manual may include copyrighted Telit and other 3rd Party supplied computer programs stored in semiconductor memories or other media. Laws in the Italy and other countries preserve for Telit and other 3rd Party supplied SW certain exclusive rights for copyrighted computer programs, including the exclusive right to copy or reproduce in any form the copyrighted computer program. Accordingly, any copyrighted Telit or other 3rd Party supplied SW computer programs contained in the Telit products described in this instruction manual may not be copied (reverse engineered) or reproduced in any manner without the express written permission of Telit or the 3rd Party SW supplier. Furthermore, the purchase of Telit products shall not be deemed to grant either directly or by implication, estoppel, or otherwise, any license under the copyrights, patents or patent applications of Telit or other 3rd Party supplied SW, except for the normal non-exclusive, royalty free license to use that arises by operation of law in the sale of a product.

USAGE AND DISCLOSURE RESTRICTIONS

I. License Agreements

The software described in this document is the property of Telit and its licensors. It is furnished by express license agreement only and may be used only in accordance with the terms of such an agreement.

II. Copyrighted Materials

Software and documentation are copyrighted materials. Making unauthorized copies is prohibited by law. No part of the software or documentation may be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any language or computer language, in any form or by any means, without prior written permission of Telit.

III. High Risk Materials

Components, units, or third-party products used in the product described herein are NOT fault-tolerant and are NOT designed, manufactured, or intended for use as on-line control equipment in the following hazardous environments requiring fail-safe controls: the operation of Nuclear Facilities, Aircraft Navigation or Aircraft Communication Systems, Air Traffic Control, Life Support, or Weapons Systems (High Risk Activities"). Telit and its supplier(s) specifically disclaim any expressed or implied warranty of fitness for such High Risk Activities.

IV. Trademarks

TELIT and the Stylized T Logo are registered in Trademark Office. All other product or service names are the property of their respective owners.

V. Third Party Rights

The software may include Third Party Right software. In this case you agree to comply with all terms and conditions imposed on you in respect of such separate software. In addition to Third Party Terms, the disclaimer of warranty and limitation of liability provisions in this License shall apply to the Third Party Right software.

TELIT HEREBY DISCLAIMS ANY AND ALL WARRANTIES EXPRESS OR IMPLIED FROM ANY THIRD PARTIES REGARDING ANY SEPARATE FILES, ANY THIRD PARTY MATERIALS INCLUDED IN THE SOFTWARE, ANY THIRD PARTY MATERIALS FROM WHICH THE SOFTWARE IS DERIVED (COLLECTIVELY “OTHER CODE”), AND THE USE OF ANY OR ALL THE OTHER CODE IN CONNECTION WITH THE SOFTWARE, INCLUDING (WITHOUT LIMITATION) ANY WARRANTIES OF SATISFACTORY QUALITY OR FITNESS FOR A PARTICULAR PURPOSE.

NO THIRD PARTY LICENSORS OF OTHER CODE SHALL HAVE ANY LIABILITY FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING WITHOUT LIMITATION LOST PROFITS), HOWEVER CAUSED AND WHETHER MADE UNDER CONTRACT, TORT OR OTHER LEGAL THEORY, ARISING IN ANY WAY OUT OF THE USE OR DISTRIBUTION OF THE OTHER CODE OR THE EXERCISE OF ANY RIGHTS GRANTED UNDER EITHER OR BOTH THIS LICENSE AND THE LEGAL TERMS APPLICABLE TO ANY SEPARATE FILES, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

APPLICABILITY TABLE

CONTENTS

1. INTRODUCTION

1.1. Scope

This document describes the interface between OSIF and a third party OS. It contains information useful to programmers who are porting the Telit Bluetooth protocol stack in a third party OS environment (FreeRTOS, Linux…).

1.2. Contact Information, Support

For general contact, technical support services, technical questions and report documentation errors contact Telit Technical Support at:

- TS-EMEA@telit.com
- TS-AMERICAS@telit.com
- TS-APAC@telit.com

or

- TS-SRD@telit.com for global Bluetooth support

Alternatively, use:

http://www.telit.com/support

For detailed information about where you can buy the Telit modules or for recommendations on accessories and components visit:

http://www.telit.com

Our aim is to make this guide as helpful as possible. Keep us informed of your comments and suggestions for improvements.

Telit appreciates feedback from the users of our information.

1.3. Text Conventions

Danger – This information MUST be followed or catastrophic equipment failure or bodily injury may occur.

Caution or Warning – Alerts the user to important points about integrating the module, if these points are not followed, the module and end user equipment may fail or malfunction.

Tip or Information – Provides advice and suggestions that may be useful when integrating the module.

All dates are in ISO 8601 format, i.e. YYYY-MM-DD.

1.4. Related Documents

- [1] BlueAPI+ API Documentation, 30515ST10833A
- [2] ComAPI+ API Documentation, 30515ST10841A

2. OVERVIEW

The operating system interface called OSIF is a small scheduler for the following purposes:

- The Telit Bluetooth protocol stack use OSIF functions as operating system calls. It simplifies the porting and integration of the protocol stacks into different operating systems by abstracting the operating system calls. Only OSIF needs to be ported to adapt the protocol stack.

- OSIF serves as the operating system interface for applications or application layers based on the API of the Telit Bluetooth protocol stack.

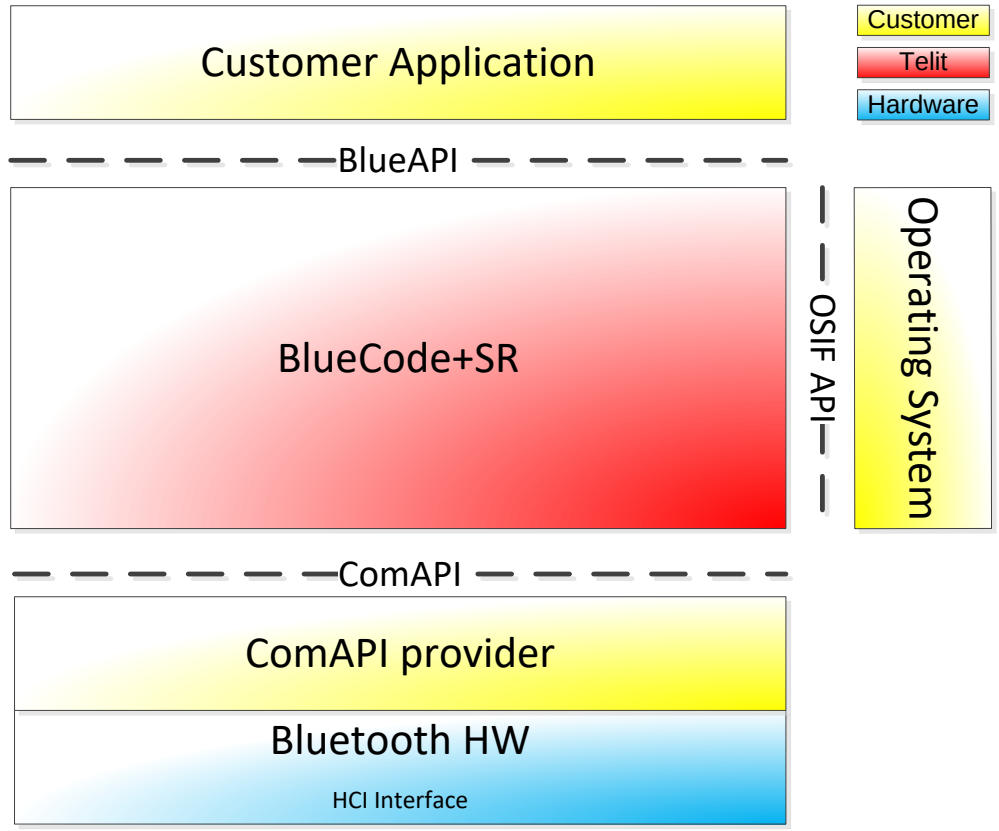

Figure 2-1 OSIF in a OS environment (FreeRTOS, Linux, others)

Since OSIF is also able to run directly on the hardware it also allows to run OSIF based protocol stacks and applications without any other operating system.

2.1. External OS Abstraction Layer

The external OS abstraction layer interface is designed as a callback interface and defined in src/osif/common/stosif.h. The init function exchange the callback function pointer.

All platform dependent implementations of the functions defined in stosif.h reside within porting/XXX, e.g. src/osif/port/stlinux.c.

For FreeRTOS and Linux exists a reference implementation. This implementations requires multitasking/threading features from underlaying target OS.

2.2. Table of CallBack Functions called from OSIF

Function	Description
OsifReady	Trigger event OSIF needs scheduling
InitCriticalSection	Init function for Mutex Semaphores…
EnterCriticalSection	Enter critical section
ExitCriticalSection	Exit critical section
InterruptEnter	Start of interrupt processing
InterruptExit	End of interrupt processing
MemoryAllocate	Allocate a block of memory
MemoryFree	Free a block of memory
FatalError	Fatal Error

Table 1: Functions called from OSIF

2.3. Table of CallBack Functions called from external abstraction OS Layer

Function	Description
OsifActive	Schedule OSIF

Table 2: Functions called from target OS

2.4. Data Structure

2.4.1. TStOSIF

This structure defines the callback functions provided by host OS and OSIF.

```
typedef struct _TStOSIF
{
  TOsifReady                    OsifReady;
  TInitCriticalSection          InitCriticalSection;
  TEnterCriticalSection         EnterCriticalSection;
  TExitCriticalSection          ExitCriticalSection;
  TInterruptEnter               InterruptEnter;
  TInterruptExit                InterruptExit;
  TMemoryAllocate               MemoryAllocate;
  TMemoryFree                   MemoryFree;
  TFatalError                   FatalError;
  TOsifActive                   OsifActive;
} TStOSIF;
```

Member	Type	Function
OsifReady	TOsifReady	OSIF need scheduling
InitCriticalSection	TInitCriticalSection	Init function for mutex, semaphores..
EnterCriticalSection	TEnterCriticalSection	Enter critical section
ExitCriticalSection	TExitCriticalSection	Exit critical section
InterruptEnter	TInterruptEnter	Indication start of interrupt processing
InterruptExit	TInterruptExit	Indication end of interrupt processing
MemoryAllocate	TMemoryAllocate	OSIF memory allocation
MemoryFree	TMemoryFree	OSIF memory release
FatalError	TFatalError	Indication of OSIF Fatal error
OsifActive	TOsifActive	OSIF scheduler

3. INTERFACE DESCRIPTION

3.1. Interface Init function stOsifInit()

The OSIFInit finction must be called first to initialize OSIF and exchange callback function pointers.

```
int stOsifInit    (
                  PStOSIF pStOsif
                  );
```

Parameter	Type	Function
pStOsif	PStOSIF	Pointer callback structure

Return value	Description
0	The function call was successful.
1	The function call failed.

Example:

```
static TStOSIF tStOSIF;
  /* setup OSIF structure with own callbacks */

  tStOSIF.OsifReady               = stLnxTriggerReschedule;
  tStOSIF.InitCriticalSection     = stLnxInitCriticalSection;
  tStOSIF.EnterCriticalSection    = stLnxEnterCriticalSection;
  tStOSIF.ExitCriticalSection     = stLnxExitCriticalSection;
  tStOSIF.TaskInitCriticalSection  = stLnxInitTaskCriticalSection;
  tStOSIF.TaskEnterCriticalSection = stLnxEnterTaskCriticalSection;
  tStOSIF.TaskExitCriticalSection  = stLnxExitTaskCriticalSection;
  tStOSIF.InterruptEnter          = stLnxInterruptEnter;
  tStOSIF.InterruptExit           = stLnxInterruptExit;
  tStOSIF.MemoryAllocate          = stLnxMemoryAlloc;
  tStOSIF.MemoryFree              = stLnxFree;
  tStOSIF.FatalError              = stLnxFatalError;


error = stOsifInit(&tStOSIF);
```

3.2. Interface callback function OsifReady()

Trigger functions from OSIF to request scheduling.

```
typedef void (* TOsifReady)                    (
                                                void
                                                );
```

Parameter	Type	Function
None		

Return value	Description
Void	

Example:

Trigger function to wakeup the main thread

```
/*****************************************************************************/
/* linux os trigger to reschedule                                            */
/*****************************************************************************/
static void stLnxTriggerReschedule(void)
{
  /* wake up kernel thread */
  /*
  printk(KERN_INFO "!!wake up kthread, state=%ld!!\n", stLnxShedThread->state);
  */
  atomic_inc( &stLnxDeferredSchedParam.iSchedPendingCnt );
  wake_up_process( stLnxShedThread );
}
```

3.3. Interface callback function InitCriticalSection()

OSIF request initializing of a critical section procedure, if required by target OS. Function return a pointer created critical section method, or zero if it isn't required by target OS.

```
typedef void *(* TInitCriticalSection)          (
                                                void
                                                );
```

Parameter	Type	Function
None		

Return value	Description
Void pointer	Context value of initializing section -1 Error

Example:

```
struct stLnxMutex
{
        struct semaphore    Mutex;
        int                 inUse;
};

static struct stLnxMutex    stLnxTaskLocks[ST_LNX_MAX_MUTEX];


void *stLnxInitCriticalSection(void)
{
  int i;
  for ( i = 0; i < ST_LNX_MAX_MUTEX; i++)
  {
    if (stLnxTaskLocks[i].inUse == FALSE)
    {
      stLnxTaskLocks[i].inUse = TRUE;
      init_MUTEX(&stLnxTaskLocks[i].Mutex);
      return((void *)&stLnxTaskLocks[i].Mutex);
    }
  }
  printk("%s-%d: no more mutex available!!\n", __FUNCTION__, __LINE__);
  return((void *)-1);
}
```

3.4. Interface callback function EnterCriticalSection()

OSIF enter a protected code sequence.

```
typedef unsigned long (* TEnterCriticalSection)(
                                                void *pParam
                                                );
```

Parameter	Type	Function
pParam	void *	Context value from init critical section

Return value	Description
unsigned long	status value before entering critical section, if required by target OS

Example:

```
static unsigned long stLnxEnterCriticalSection(void *pParam)
{
  unsigned long flags = 1;
  down(pParam); /* get mutex */
  return(flags);
}
```

3.5. Interface callback function ExitCriticalSection()

OSIF leaves a protected code sequence.

```
typedef void (* TExitCriticalSection)          (
                                               void *pParam,
                                               unsigned long psw
                                               );
```

Parameter	Type	Function
pParam	void *	Context value from init critical section
psw	unsigned long	Status from previous enter critical section

Return value	Description
Void	

Example:

```
static void stLnxExitCriticalSection(void *pParam, unsigned long flags)
{
  up(pParam); /* free mutex */
  flags = 0;  /* dummy to avoid compiler warnings */
  return;
}
```

3.6. Interface callback function InterruptEnter()

OSIF enter a code sequence in interrupt context.

```
typedef void (* TInterruptEnter)                (
                                                void
                                                );
```

Parameter	Type	Function
None		

Return value	Description
Void	

Dummy in Linux environment

3.7. Interface callback function InterruptExit()

OSIF leaves a code sequence in interrupt context.

```
typedef void (* TInterruptExit)                    (
                                                   void
                                                   );
```

Parameter	Type	Function
None		

Return value	Description
Void	

Dummy in Linux environment

3.8. Interface callback function MemoryAllocate()

OSIF request a memory block.

```
typedef void *(* TMemoryAllocate)                  (
                                                   int size
                                                   );
```

Parameter	Type	Function
Size	int	Size of memory

Return value	Description
Void pointer	Points to allocated memory block. If allocation fails or if size is 0, the function returns NULL.

Example:

```
static void * stLnxMemoryAlloc(int size )
{
  void *p;
  int i;

  p = kmalloc(size, KMALLOC_PRIO);

  if(p == NULL) return(NULL);

  /* bookkeeping about allocated memory, if necessary  */
  /* OSIF never frees memory yet  */

 return(p);
```

```
}
```

3.9. Interface callback function MemoryFree()

Release memory.

```
typedef void (* TMemoryFree)                    (
                                                void * pBlock
                                                );
```

Parameter	Type	Function
pBlock	void *	start address of freeing memory

Return value	Description
Void	

Dummy function for further use.

3.10. Interface callback function FatalError()

OSIF indicates a internal error wizh location of error.

```
typedef void (* TFatalError)                    (
                                                char * pFile,
                                                int iLine
                                                );
```

Parameter	Type	Function
pFile	char *	Filename
iLine	int	Line number

Return value	Description
Void	

Example:

Feeds a kernel message to the console

```
static void stLnxFatalError(char * File, int Line)
{
  printk("OSIF FATAL ERROR: file %s, line %d\n", File, Line);
}
```

3.11. Interface callback function OsifActive()

Target OS should call OsifActive as soon as possible after OSIF was calling OsifReady()

```
typedef int (* TOsifActive)                    (
                                               int TimeElapsedMSecond
                                               );
```

Parameter	Type	Function
TimeElapsedMSecond	int	Time value in mseconds since the last call of OsifActive(), meaning the difference between two calls of OsifActive().

Return value	Description
Int	OSIF needs rescheduling at least in mseconds

Example:

Main Linux kernel thread to schedule OSIF.

```
#if defined(__KERNEL__)
/*****************************************************************************/
/* Reschedule tasklet or thread context                                      */
/*****************************************************************************/
#if LINUX_VERSION_CODE < KERNEL_VERSION(2,5,0)
static void stLnxReschedule(void)
#else
static void stLnxReschedule(unsigned long data)
#endif
{
  unsigned long  stLnxTimerTickLast, stLnxTimeElapsedMSecond;
  /* calculate difference form last schedule in msec */
  stLnxTimeElapsedMSecond = (jiffies * 1000 / HZ) - stLnxTickCountLast;

  stLnxTimerTickLast = tStOSIF.OsifActive((int)stLnxTimeElapsedMSecond);

  /* start timer to wakeup in  msec stLnxTimerTickLast */

  if (stLnxTimerTickLast && (stLnxInitReady == TRUE))
  {
    stLnxTimerRetrigger(stLnxTimerTickLast );
  }
  stLnxTickCountLast = jiffies * 1000 / HZ;  /* update tick count */
}
```

4. DOCUMENT HISTORY

Revision	Date	Changes
1	2011-10-28	First issue
2	2011-11-04	Add description of return value of OSIFActive() Add Linux example code fragments
3	2017-07-20	First Telit issue

SUPPORT INQUIRIES

Link to **www.telit.com** and contact our technical support team for any questions related to technical issues.

www.telit.com

Telit Communications S.p.A.
Via Stazione di Prosecco, 5/B
I-34010 Sgonico (Trieste), Italy

Telit IoT Platforms LLC
5300 Broken Sound Blvd, Suite 150
Boca Raton, FL 33487, USA

Telit Wireless Solutions Inc.
3131 RDU Center Drive, Suite 135
Morrisville, NC 27560, USA

Telit Wireless Solutions Co., Ltd.
8th Fl., Shinyoung Securities Bld.
6, Gukjegeumyung-ro8-gil, Yeongdeungpo-gu
Seoul, 150-884, Korea

Telit Wireless Solutions Ltd.
10 Habarzel St.
Tel Aviv 69710, Israel

Telit Wireless Solutions
Technologia e Servicos Ltda
Avenida Paulista, 1776, Room 10.C
01310-921 São Paulo, Brazil