Application note

Porting from OS20 to OS21

1 Introduction

The majority of OS21's API is based on the OS20 operating system for the ST20 processors.

Although the APIs have a great deal in common there are a number of differences between the two operating systems. These differences are presented here in order to show how to port an application from OS20 to OS21.

The different features described in this application note, include:

- *Header files*
- *Bringing up the kernel*
- *Statically allocated memory*
- *Interrupts and caches*
- *Channels and 2D block moves*
- *Time*
- *New features in OS21*

Contents

1 Introduction 1

2 Header files 3

3 Bringing up the kernel 3

4 Statically allocated memory 4

5 Interrupts and caches 5

6 Channels and 2D block moves 5

7 Time 5

8 New features in OS21 5

9 Revision history 6

2 Header files

OS20 uses header files with 8.3 (MS-DOS style) names. OS21 is not constrained by this limitation and uses meaningful names, which do not clash with other headers. *Table 1* shows the name changes for each header file, however, it is often more convenient to replace all OS20 header files with the more general `<os21.h>`.

Table 1. Header file name changes

OS20 header file	OS21 header file
`interrup.h`	`os21/interrupt.h`
`kernel.h`	`os21/kernel.h`
`message.h`	`os21/message.h`
`ostime.h`	`os21/ostime.h`
`partitio.h`	`os21/partition.h`
`semaphor.h`	`os21/semaphore.h`
`task.h`	`os21/task.h`

3 Bringing up the kernel

OS20 provides two means to bring up the kernel: manual and automatic (through the use of the `st20cc -runtime os20` option). OS21 is normally brought up manually, although the `autostart` example shows how to bring it up automatically.

The following example demonstrates how to bring up the OS21 kernel.

```
int main(void)
{
   /*
   * Initialize the OS21 kernel, and enable timeslicing
   */
   kernel_initialize(NULL);
   kernel_start();
   kernel_timeslice(OS21_TRUE);
   ...
}
```

Note: OS21 ***does not*** *enable timeslicing by default. If timeslicing is required it must be manually enabled after the kernel has started using `kernel_timeslice()` as shown in this example.*

4 Statically allocated memory

OS21 does not support the `_init()` family of functions that are used by OS20. These functions expose the data structures used by the operating system and their general use can hinder the development of the operating system.

Normally all instances of `_init()` functions would be replaced by `_create()` functions. However, one of the advantages of the `_init()` functions is the flexibility they afford with regard to memory allocation. OS21 takes a different approach to flexible memory management. In addition to all the `_create()` functions there are `_create_p()` functions which take a partition pointer as an additional argument. This allows the application programmer to tightly control where memory is allocated from.

Every instance of the following list of APIs should be replaced with its `_create()` or `_create_p()` equivalent.

- `message_init_queue()`
- `message_init_queue_timeout()`
- `partition_init_fixed()`
- `partition_init_heap()`
- `partition_init_simple()`
- `semaphore_init_fifo()`
- `semaphore_init_fifo_timeout()`
- `semaphore_init_priority()`
- `semaphore_init_priority_timeout()`
- `task_init()`

Note: 1 *OS21 does not differentiate between timeout and non-timeout synchronization primitives. Non-timeout functions are simple macro versions of their timeout equivalents.*

2 *OS21 does not support the ST20 specific `task_create_sl()`, `task_init_sl()` and `task_onexit_set_sl()` API calls.*

5 Interrupts and caches

The OS20 interrupt and cache API is very closely tied to the ST20 interrupt and cache architectures. The rationale for this is to provide complete access to the hardware's functionality.

The cache and interrupt APIs provided by OS21 are intended to be generic, and portable between CPUs, so they differ from the APIs used in OS20.

Full details of the OS21 cache and interrupt APIs can be found in the *OS21 User Manual*.

6 Channels and 2D block moves

The channel and 2D block move API present in OS20, provide access to ST20 hardware features. These have been removed from OS21 as they are ST20 specific.

7 Time

In OS21, time is represented using a 64-bit integer type (`osclock_t`) whereas in OS20, time is represented as a 32-bit integer type (`clock_t`).

8 New features in OS21

OS21 provides features that OS20 does not provide. These offer more elegant ways to solve certain concurrent design problems and their use is to be encouraged. Applications which require their code base to be built both with OS20 and OS21 should avoid the use of events.

9 Revision history

Table 2. Document revision history

Date	Revision	Changes
14-Nov-2007	C	Minor rewording thoughout. Clarification of *Section 2: Header files on page 3* and *Section 5: Interrupts and caches on page 5*.
8-Nov-2006	B	Updated template - no changes to content.
19-May-2006	A	Initial release.

Please Read Carefully:

Information in this document is provided solely in connection with ST products. STMicroelectronics NV and its subsidiaries (“ST”) reserve the right to make changes, corrections, modifications or improvements, to this document, and the products and services described herein at any time, without notice.

All ST products are sold pursuant to ST’s terms and conditions of sale.

Purchasers are solely responsible for the choice, selection and use of the ST products and services described herein, and ST assumes no liability whatsoever relating to the choice, selection or use of the ST products and services described herein.

No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted under this document. If any part of this document refers to any third party products or services it shall not be deemed a license grant by ST for the use of such third party products or services, or any intellectual property contained therein or considered as a warranty covering the use in any manner whatsoever of such third party products or services or any intellectual property contained therein.

UNLESS OTHERWISE SET FORTH IN ST’S TERMS AND CONDITIONS OF SALE ST DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY WITH RESPECT TO THE USE AND/OR SALE OF ST PRODUCTS INCLUDING WITHOUT LIMITATION IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE (AND THEIR EQUIVALENTS UNDER THE LAWS OF ANY JURISDICTION), OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

UNLESS EXPRESSLY APPROVED IN WRITING BY AN AUTHORIZED ST REPRESENTATIVE, ST PRODUCTS ARE NOT RECOMMENDED, AUTHORIZED OR WARRANTED FOR USE IN MILITARY, AIR CRAFT, SPACE, LIFE SAVING, OR LIFE SUSTAINING APPLICATIONS, NOR IN PRODUCTS OR SYSTEMS WHERE FAILURE OR MALFUNCTION MAY RESULT IN PERSONAL INJURY, DEATH, OR SEVERE PROPERTY OR ENVIRONMENTAL DAMAGE. ST PRODUCTS WHICH ARE NOT SPECIFIED AS "AUTOMOTIVE GRADE" MAY ONLY BE USED IN AUTOMOTIVE APPLICATIONS AT USER’S OWN RISK.

Resale of ST products with provisions different from the statements and/or technical features set forth in this document shall immediately void any warranty granted by ST for the ST product or service described herein and shall not create or extend in any manner whatsoever, any liability of ST.

ST and the ST logo are trademarks or registered trademarks of ST in various countries.

Information in this document supersedes and replaces all information previously supplied.

The ST logo is a registered trademark of STMicroelectronics. All other names are the property of their respective owners.

© 2007 STMicroelectronics - All rights reserved

STMicroelectronics group of companies

Australia - Belgium - Brazil - Canada - China - Czech Republic - Finland - France - Germany - Hong Kong - India - Israel - Italy - Japan - Malaysia - Malta - Morocco - Singapore - Spain - Sweden - Switzerland - United Kingdom - United States of America

www.st.com