



SolarA²

定时任务开发指南

文档版本 01

发布日期 2024-12-20

版权所有 © 海思技术有限公司2024。保留一切权利。

非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

商标声明



HISILICON、海思和其他海思商标均为海思技术有限公司的商标。

本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受海思公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，海思公司对本文档内容不做任何明示或默示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

海思技术有限公司

地址：上海市青浦区虹桥港路2号101室 邮编：201721

网址：<https://www.hisilicon.com/cn/>

客户服务邮箱：support@hisilicon.com



前言

概述

本文档介绍定时任务相关的概念以及使用指导。

产品版本

与本文档相对应的产品版本如下。

产品名称	产品版本
SolarA ²	1.1.0

读者对象




本文档（本指南）主要适用于以下工程师：

技术支持工程师

客户开发工程师

符号约定

在本文中可能出现下列标志，它们所代表的含义如下。

符号	说明
 危险	表示如不避免则将会导致死亡或严重伤害的具有高等级风险的危害。
 警告	表示如不避免则可能导致死亡或严重伤害的具有中等级风险的危害。
 注意	表示如不避免则可能导致轻微或中度伤害的具有低等级风险的危害。



符号	说明
<div>须知</div>	用于传递设备或环境安全警示信息。如不可避免则可能会导致设备损坏、数据丢失、设备性能降低或其它不可预知的结果。 “须知”不涉及人身伤害。
<div>说明</div>	对正文中重点信息的补充说明。 “说明”不是安全警示信息，不涉及人身、设备及环境伤害信息。

修订记录

修订记录累积了每次文档更新的说明。最新版本的文档包含以前所有文档版本的更新内容。

修订日期	版本	修订说明
2024-08-23	00B01	第1次临时版本发布。
2024-12-20	01	第1次正式版本发布。



目录

前言.....	i
1 基本概念.....	1
2 特性设计.....	3
2.1 应用框图.....	3
2.2 硬件资源.....	4
2.3 API 接口.....	4
3 接口说明.....	6
3.1 创建定时任务.....	6
3.2 启动定时任务.....	7
3.3 停止定时任务.....	7
3.4 删除定时任务.....	7
4 注意事项.....	8
5 样例.....	9
5.1 样例介绍.....	9
5.2 环境准备.....	9
5.3 样例生成与运行.....	9



插图目录

图 1-1 传统 MCU 系统..... 1

图 1-2 多任务系统..... 2

图 2-1 HAL_TIMER 硬件定时器应用框图..... 3

图 2-2 NOS_TimerTask 应用框图..... 4

图 5-1 新建工程示例..... 10

图 5-2 选择样例..... 10

图 5-3 编译代码..... 11

图 5-4 输出结果开始..... 12

图 5-5 输出结果结束..... 13



表格目录

表 2-1 硬件资源对比.....4

表 2-2 API 接口对比.....4

表 3-1 timerParam 参数说明.....6

表 3-2 timerTaskId 参数说明.....6

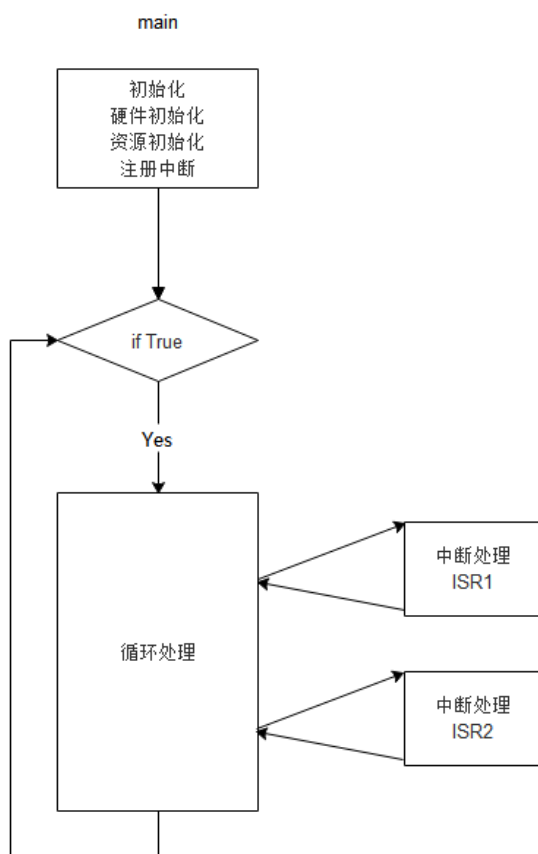
表 3-3 taskId 参数说明.....7



1 基本概念

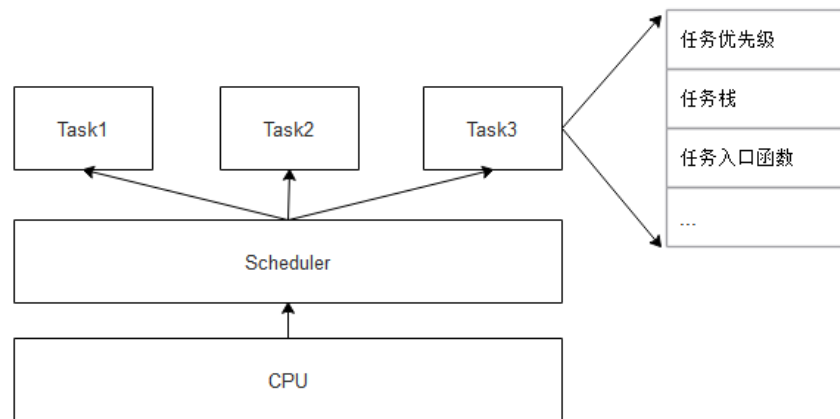
传统的MCU系统，应用程序由一个无限循环和多个中断服务程序组成，无限循环中调用模块（即函数）来执行所需的操作，中断服务程序负责处理异步事件。这种系统下只有一个无限循环的任务（即main函数）。

图 1-1 传统 MCU 系统



现代RTOS系统，引入了多任务的概念，任务是占用CPU的最小运行单元，每个任务都有自己的任务入口函数、任务栈和优先级。CPU根据调度器Scheduler的策略在多个任务之间切换执行，调度策略支持基于优先级的抢占式调度，即高优先级任务可以抢占低优先级任务。

图 1-2 多任务系统



本文的定时任务基于多任务系统实现，定时任务包含定时任务入口函数、任务栈、优先级和定时执行时间间隔等属性。启动定时任务后，定时任务入口函数按设定的定时执行时间间隔周期执行。定时任务的实现依赖系统定时器Systick，Systick按照设定的时间周期性产生中断，在Systick中断处理中判断定时任务是否需要调度执行。

系统在初始化时默认会创建一个最低优先级的main任务，main任务的入口函数就是main函数，用户可以在main函数里实现自己的业务代码，可以创建定时任务。系统中如果没有定时任务在运行，则会运行main任务。

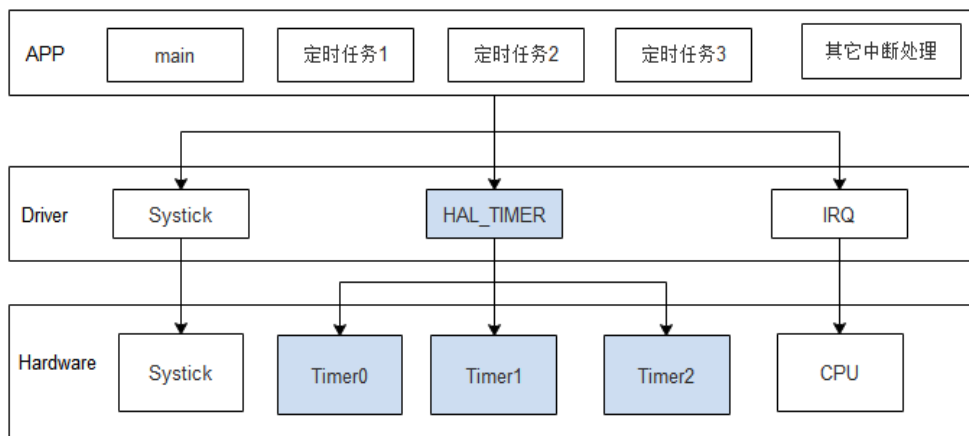
2 特性设计

本章节通过与硬件定时器驱动HAL_TIMER对比来说明NOS_TimerTask定时任务的相关设计。

2.1 应用框图

使用HAL_TIMER 实现定时任务，APP调用Driver层HAL_TIMER接口，分别创建/启动定时器。每创建一个定时器，HAL_TIMER会占用一个硬件Timer资源，在定时器周期到达时刻，触发定时任务执行。高优先级定时任务可以抢占低优先级定时任务。如图2-1所示，若创建3个定时器，则会用到硬件Timer0/1/2。

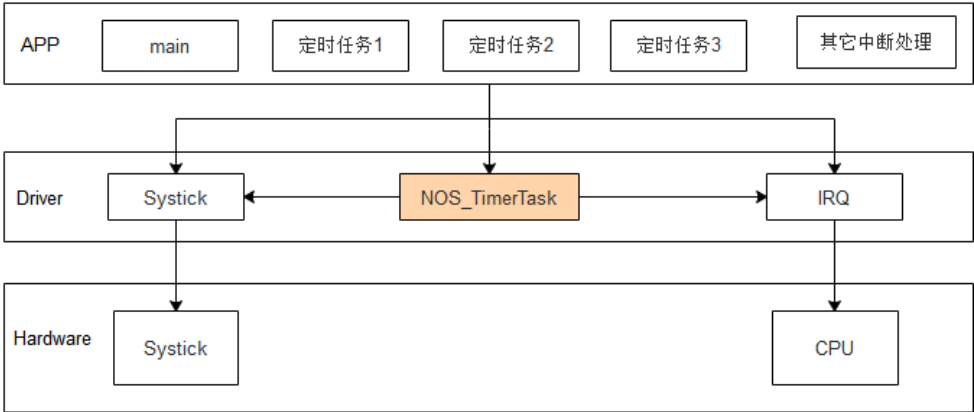
图 2-1 HAL_TIMER 硬件定时器应用框图



系统使用NOS_TimerTask实现定时任务，APP调用NOS_TimerTask接口创建/启动定时任务。在SysTick中断处理中判断定时任务是否需要调度执行。高优先级定时任务可以抢占低优先级定时任务。



图 2-2 NOS_TimerTask 应用框图



2.2 硬件资源

如表2-1所示，使用HAL_TIMER硬件定时器，一个定时器任务绑定一个硬件Timer资源。使用NOS_TimerTask，仅使用硬件TIMER3作为Systick Timer，可以同时创建多个定时任务。

表 2-1 硬件资源对比

硬件定时器编号	使用硬件定时器HAL_TIMER	使用NOS_TimerTask
Timer0	使用	不使用
Timer1	使用	不使用
Timer2	使用	不使用
Timer3	用作Systick计时	用作Systick中断

2.3 API 接口

NOS_TimerTask提供类似于硬件定时器HAL_TIMER的功能接口，接口对应关系如表2-2所示。具体用法可参考Nos Task Schedule样例。

表 2-2 API 接口对比

NOS_TimerTask	硬件定时器
NOS_CreateTimerTask	HAL_TIMER_Init。 IRQ_SetPriority。 IRQ_EnableN。
NOS_StartTimerTask	HAL_TIMER_Start。



NOS_TimerTask	硬件定时器
NOS_StopTimerTask	IRQ_DisableN。 HAL_TIMER_IrqClear。 HAL_TIMER_Stop。
NOS_TaskDelete	HAL_TIMER_DeInit。



3 接口说明

3.1 创建定时任务

接口：int NOS_CreateTimerTask(unsigned int *timerTaskId,
NOS_TimerTaskInitParam *timerParam);

表 3-1 timerParam 参数说明

参数	输入/输出	含义
const char *name	Input	表示任务名。
unsigned int timeout	Input	任务超时时长。
NOS_TimerCallBack callback	Input	超时回调函数。callback运行时长不能超过timeout。
void *callbackParam	Input	超时任务的参数，callback的入参。
unsigned int priority	Input	任务优先级，取值范围[0,4]，数值越大，优先级越低。
unsigned int stackAddr	Input	任务栈地址，该值不能为0，且需16字节对齐。详细见示例。
unsigned int stackSize	Input	任务栈stackAddr的大小。

表 3-2 timerTaskId 参数说明

参数	输入/输出	含义
unsigned int *timerTaskId	Output	创建的task的ID，用于唯一标识一个任务。



示例：

```
unsigned char __attribute__((aligned(16))) g_task0StackSpace[0x300];
int main()
{
    NOS_TimerTaskInitParam param;
    unsigned int pid;
    param.name = "task0";
    param.timeout = 200; // us
    param.callback = Sample_timer0CallbackFunc;
    param.callbackParam = 200;
    param.priority = 1;
    param.stackSize = sizeof(g_task0StackSpace);
    param.stackAddr = g_task0StackSpace;
    int ret = NOS_CreateTimerTask(&pid, &param);
    ... ..
}
```

3.2 启动定时任务

```
int NOS_StartTimerTask(unsigned int taskId);
```

表 3-3 taskId 参数说明

参数	输入/输出	含义
unsigned int taskId	Input	创建的task的ID，用于唯一标识一个任务。

启动定时任务后，会将任务添加到定时链表，等待超时调度。

3.3 停止定时任务

```
int NOS_StopTimerTask(unsigned int taskId);
```

停止定时任务后，会将任务从定时链表移除，任务将不会得到调度，但任务实例仍存在，资源未释放。

3.4 删除定时任务

```
int NOS_TaskDelete(unsigned int taskId);
```

删除定时任务后，会将任务实例删除，释放任务所占用资源。



4 注意事项

- 定时任务的代码实现通过“NOS_TASK_SUPPORT”宏隔离，使用定时任务功能必须使能“NOS_TASK_SUPPORT”宏。
- 定时任务中使用的部分指令必须要在RISCV的机器模式下才能执行，使用定时任务功能时必须把“Riscv用户模式支持”的宏配置使能去掉。
- Systick中断间隔默认为100us，可以通过在IDE的“工程配置”界面的“全局宏定义”配置栏中新建“CFG_SYSTICK_TICKINTERVAL_US”宏来配置TICK的大小；所有定时任务的周期必须是Systick中断间隔的整数倍；启动定时任务后，仅首次超时可能会有误差，误差范围小于1个Systick中断间隔。
- main中根据需要调用NOS_CreateTimerTask创建定时任务，定时任务优先级需要高于main本身（main优先级为NOS_TASK_PRIORITY_LOWEST）。定时任务创建后暂不支持动态修改优先级。
- 当前用户可创建的定时任务数最大支持4个。
- 创建任务时，任务栈空间必须由调用点提供，且栈地址必须16字节对齐。
- 系统启动时会创建任务入口函数为main的主任务MainTask，MainTask栈空间大小可通过CFG_NOS_MAINTASK_STACKSIZE宏配置。
- 使能NOS_TASK_SUPPORT后，系统中使用的栈分为中断栈和任务栈，ram划分为sram和stack。其中stack预留给中断栈，任务栈从sram中分配。用户可以根据实际需求，调整sram和stack大小。任务栈大小与函数调用深度以及局部变量大小有关，用户应合理分配。
- 定时任务仅支持创建/启动/停止/删除操作。



5 样例

5.1 样例介绍

- 样例中创建了3个定时任务，分别定时100us，200us，1ms。
- 样例在运行100ms后停止所有定时器，打印出三个Task Timer的触发执行时间，通过时间戳可以看出定时任务是否按照预定的定时间隔触发。
- 样例展示了“NOS_CreateTimerTask()”、“NOS_StartTimerTask()”、“NOS_StopTimerTask()”的使用。

5.2 环境准备

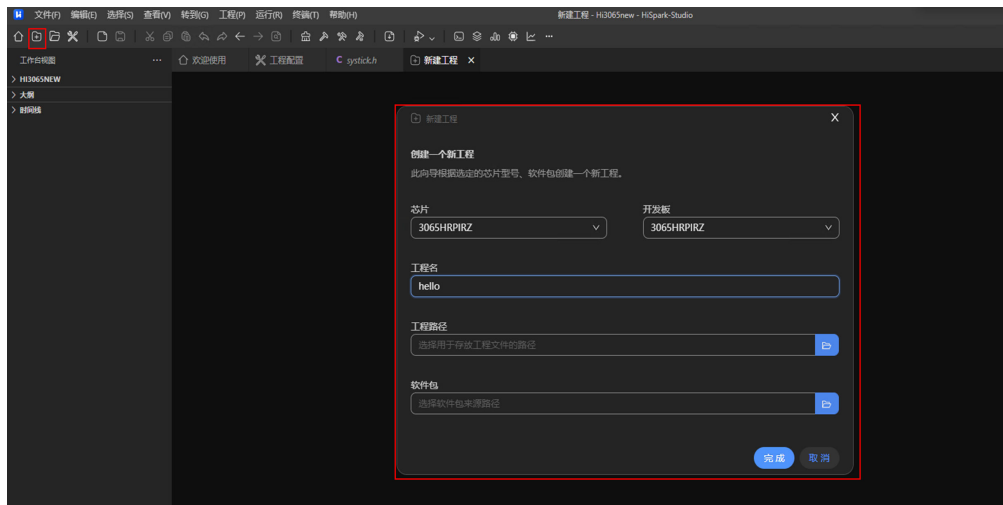
准备SolarA² SDK，安装HiSpark-Studio IDE。

5.3 样例生成与运行

步骤1 新建工程。点击“新建工程”，工程名自定义输入，工程路径自行选择，MCU选择3065HRPIRZ（以3065HRPIRZ为例），MCU驱动开发包选择指定的SDK开发包路径，如图5-1所示。

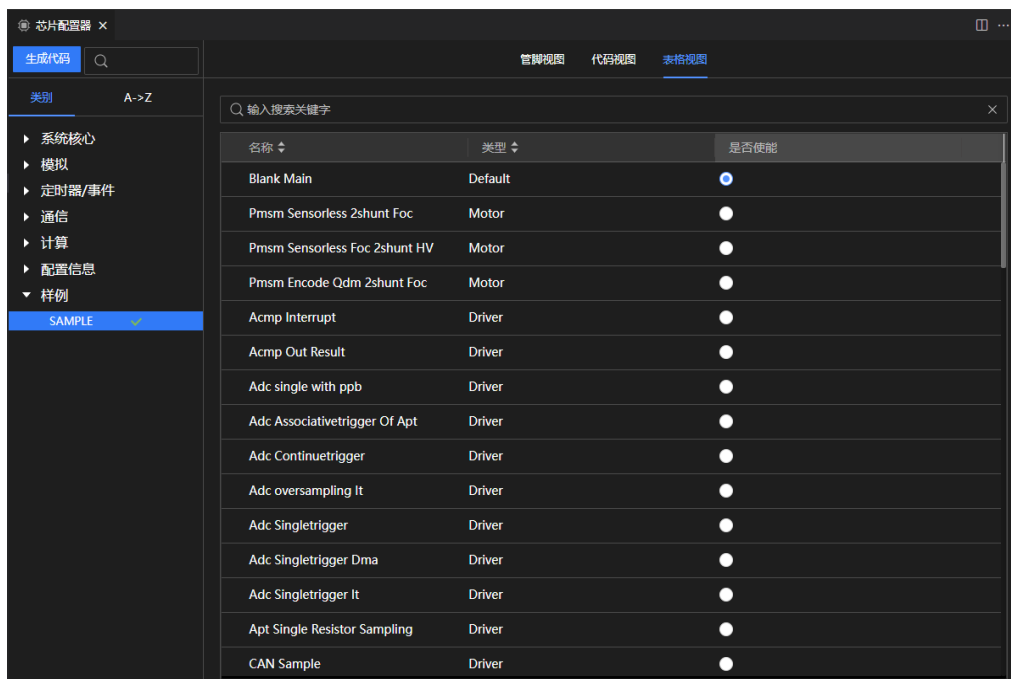


图 5-1 新建工程示例



步骤2 选择样例。打开芯片配置器，从样例中选择“Nos Task Schedule”样例。选择完成样例后点击“生成代码”，如图5-2所示。

图 5-2 选择样例



步骤3 编译代码，如图5-3所示。



图 5-3 编译代码

```
1  /**
2   * @copyright Copyright (c) 2022, Hisilicon (Shanghai) Technologies Co., Ltd. All rights reserved.
3   * Redistribution and use in source and binary forms, with or without modification, are permitted provided that the
4   * following conditions are met:
5   * 1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following
6   * disclaimer.
7   * 2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the
8   * following disclaimer in the documentation and/or other materials provided with the distribution.
9   * 3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote
10  * products derived from this software without specific prior written permission.
11  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES,
12  * INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
13  * DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
14  * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
15  * SERVICES, LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
16  * WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE
17  * USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
18  * @file    main.c
19  * @author  ... MCU Driver Team
20  * @brief   Main program body.
21  */
22
23 #include "typedefs.h"
24 #include "feature.h"
25 #include "main.h"
26 /* USER CODE BEGIN 0 */
27 /* USER CODE 区域内代码不会被覆盖，区域外会生成代码覆盖（其余USER CODE 区域同理） */
28 /* 建议用户放置头文件 */
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

步骤4 结果验证。编译好的代码烧写到指定的目标板，通过串口打印查看结果。



图 5-4 输出结果开始

```
*****
Test for: NOS_CreateTimerTask & NOS_StartTimerTask
*****
enter task timeout=100 timestamp=10421us
enter task timeout=100 timestamp=10521us
enter task timeout=200 timestamp=10524us
enter task timeout=100 timestamp=10620us
enter task timeout=100 timestamp=10721us
enter task timeout=200 timestamp=10724us
enter task timeout=100 timestamp=10820us
enter task timeout=100 timestamp=10921us
enter task timeout=200 timestamp=10924us
enter task timeout=100 timestamp=11020us
enter task timeout=100 timestamp=11121us
enter task timeout=200 timestamp=11124us
enter task timeout=100 timestamp=11220us
enter task timeout=100 timestamp=11322us
enter task timeout=200 timestamp=11324us
enter task timeout=1000 timestamp=11327us
enter task timeout=100 timestamp=11421us
enter task timeout=100 timestamp=11521us
enter task timeout=200 timestamp=11524us
enter task timeout=100 timestamp=11620us
```



图 5-5 输出结果结束

```
enter task timeout=100 timestamp=15020us
enter task timeout=100 timestamp=15121us
enter task timeout=200 timestamp=15124us
enter task timeout=100 timestamp=15220us
enter task timeout=100 timestamp=15322us
enter task timeout=200 timestamp=15324us
enter task timeout=1000 timestamp=15327us
enter task timeout=100 timestamp=15421us
enter task timeout=100 timestamp=15521us
enter task timeout=200 timestamp=15524us
enter task timeout=100 timestamp=15620us
enter task timeout=100 timestamp=15721us
enter task timeout=200 timestamp=15724us
enter task timeout=100 timestamp=15820us
enter task timeout=100 timestamp=15921us
enter task timeout=200 timestamp=15924us
enter task timeout=100 timestamp=16020us
enter task timeout=100 timestamp=16121us
enter task timeout=200 timestamp=16124us
enter task timeout=100 timestamp=16220us
enter task timeout=100 timestamp=16322us
enter task timeout=200 timestamp=16324us
enter task timeout=1000 timestamp=16327us
enter task timeout=100 timestamp=16421us
enter task timeout=100 timestamp=16521us
enter task timeout=200 timestamp=16524us
enter task timeout=100 timestamp=16620us
total cnt:100 cnt[100us]:63,cnt[200us]:31, cnt[1000us]:6
```

----结束