版本: 1.0.0 2011 年 12 月

Sword37 AKOS 接口说明

声　　明

联　系　方　式

安凯（广州）微电子技术有限公司

地址：广州科学城科学大道 182 号创新大厦 C1 区 3 楼
电话: (86)-20-3221 9000

传真: (86)-20-3221 9258

邮编: 510663

销售热线:

(86)-20-3221 9499

电子邮箱:

sales@anyka.com

主页:

http: //www. anyka.com

版 本 变 更 说 明

以下表格对于本文档的版本变更做一个简要的说明。版本变更仅限于技术内容的变更，不包括版式、格式、句法等的变更。

版本	说明	完成日期
V1.0.0	正式发布	2011 年 12 月

目录

1 简介

AKOS 多任务整体软件架构如图 1-1 所示。AKOS 整体软件架构主要包括 Thread 线程模块、App 应用程序模块、AppMgr 任务管理模块以及 AKMsgDispatch 消息管理模块和子线程（SubThread）管理模块等。本文档提供 Sword37 平台中多任务操作系统（AKOS）的相关接口说明。有关各子模块相关接口说明请用户参见《Sword37AKOS 子模块接口说明》文档。

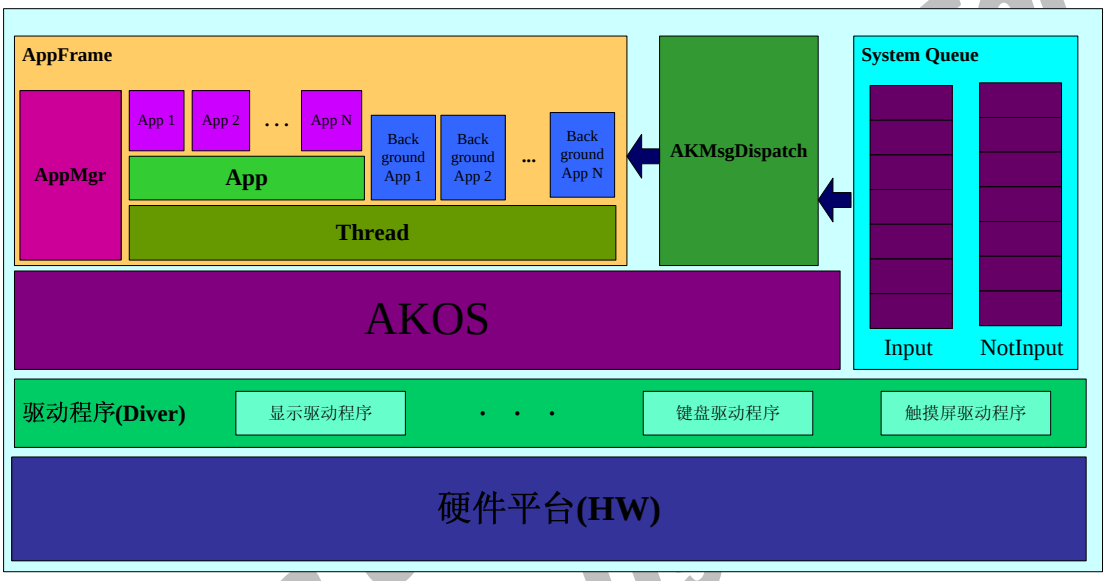

图 1-1 AKOS 多任务整体软件架构

2 数据类型

- T_OPTION 很容易操作的最小数据类型，通常为一个无符号字符
- T_hTask 任务句柄
- T_hQueue 队列句柄
- T_hSemaphore 信号量句柄
- T_hMailbox 邮箱句柄
- T_hEventGroup 事件集句柄
- T_hTimer 定时器句柄
- T_hHisr 高级中断句柄
- T_U32 32 位无符号整型

- T_S32　　32 位有符号整型
- T_U8　　8 位无符号整型
- T_S8　　8 位有符号整型

3 接口列表

3.1 任务管理接口列表

3.1.1 AK_Create_Task

原　型	T_hTask AK_Create_Task(T_VOID (*task_entry)(T_U32)(T_VOID *), T_U8 *name, T_U32 argc, T_VOID *argv, T_VOID *stack_address, T_U32 stack_size, T_OPTION priority, T_U32 time_slice, T_OPTION preempt, T_OPTION auto_start);	
功能概述	此项服务创建一个应用程序任务	
参数说明	task_entry	指定任务函数入口
	name	任务名字符串指针，最长只有 8 字节
	argc	一个 T_U32 数据类型，可以用来传递初始化信息到任务
	argv	指针，传递初始化信息到任务
	stack_address	分配任务堆栈区的起始内存地址位置
	stack_size	指定堆栈的字节数
	Priority	在 0～255 之间指定优先级值。数值越低，任务级别越高。
	time_slice	表示中止运行任务的定时器节拍最大值。0 值表示禁止任务时间片。

原　　型	T_hTask AK_Create_Task(T_VOID (*task_entry)(T_U32)(T_VOID *), T_U8 *name, T_U32 argc, T_VOID *argv, T_VOID *stack_address, T_U32 stack_size, T_OPTION priority, T_U32 time_slice, T_OPTION preempt, T_OPTION auto_start);	
	preempt	此配置的有效参数为 AK_PREEMPT 和 AK_NO_PREEMPT。 AK_PREEMPT 表示任务占先有效，AK_NO_PREEMPT 表示任务占先无效。 注：如果任务占先无效，时间片禁止
	auto_start	此配置有效参数为：AK_NO_START 和 AK_START。 AK_START 表示在任务创建后把任务放置到就绪状态。AK_NO_START 表示任务在创建之后处于休眠状态。 参数为 AK_NO_START 的任务稍后必须恢复。
返回值说明	T_hTask	任务创建成功则返回任务句柄
	AK_MEMORY_CORRUPT	表示申请内存空间失败
	AK_INVALID_ENTRY	表示入口函数指针为空
	AK_INVALID_MEMORY	表示 stack_address 指定的内存区为空
	AK_INVALID_SIZE	表示指定的堆栈尺寸不够大
	AK_INVALID_PRIORITY	表示指定的优先级无效
	AK_INVALID_PREEMPT	表示占先参数无效。这个错误发生在连同无占先配置一起时间片被指定
	AK_INVALID_START	表示 auto_start 参数无效
注意事项		

原　　型	T_hTask AK_Create_Task(T_VOID (*task_entry)(T_U32)(T_VOID *), T_U8 *name, T_U32 argc, T_VOID *argv, T_VOID *stack_address, T_U32 stack_size, T_OPTION priority, T_U32 time_slice, T_OPTION preempt, T_OPTION auto_start);
调用示例	/*假设任务控制句柄"Task"定义为全局数据结构。这是分配控制块的几种方法之一*/ T_hTask Task; /*假设 status 在本地定义*/ T_S32 status; /*创建一个入口函数为"task_entry",堆栈指针为"stack_ptr",堆栈尺寸为 2000 字节的任务。注：下列附加参数： argc and argv (0,AK_NULL)，优先级为 200，时间片为 15 个定时器节拍，占先有效，自动启动*/ Task = AK_Create_Task(task_entry, “Task1”, 0, AK_NULL, Stack_ptr, 2000, 200, 15, AK_PREEMPT, AK_START);

3.1.2 AK_Delete_Task

原　　型	T_S32 AK_Delete_Task(T_hTask Task);	
功能概述	此服务删除一个先前定义的任务。参数 Task 确定需要删除的任务。在这个任务上挂起的任务恢复时返回适当的错误状态。	
参数说明	Task	任务控制句柄
返回值说明	AK_SUCCESS	表示任务成功删除
	AK_INVALID_TASK	表示任务句柄非法
	AK_INVALID_DELETE	任务处于一个未完成或未中止状态
注意事项	只能删除处于完成或者中止的任务，如果正在运行或挂起的任务会删除失败	

原　　型	T_S32 AK_Delete_Task(T_hTask Task);
调用示例	T_S32 status; /*删除 Task 指定的任务句柄。假设"Task"在先前已经调用 AK_Create_Task 创建*/ status = AK_Delete_Task(Task);

3.1.3 AK_Suspend_Task

原　　型	T_S32 AK_Suspend_Task(T_hTask Task);	
功能概述	此项服务无条件挂起 task 指针指定的任务。如果任务已经处于挂起状态。即使在最初导致挂起的条件消失，此服务确仍然保任务保留在挂起状态。AK_Resume_Task 用于恢复这种方式的任务挂起。	
参数说明	Task	任务控制句柄
返回值说明	AK_SUCCESS	表示任务成功完成
	AK_INVALID_TASK	表示任务句柄非法
注意事项		
调用示例	T_S32 status; /*无条件挂起任务"Task"。假设"Task"在先前已经调用 AK_Create_Task 创建*/ status = AK_Suspend_Task(Task);	

3.1.4 AK_Resume_Task

原　　型	T_S32 AK_Resume_Task(T_hTask task);	
功能概述	此项服务恢复一个先前通过 AK_Suspend_Task 服务挂起的任务。	
参数说明	Task	任务控制句柄
返回值说明	AK_SUCCESS	表示服务成功完成
	AK_INVALID_TASK	表示任务句柄无效
	AK_INVALID_RESUME	表示指定的任务不处于无条件挂起状态。
注意事项		

原　　型	T_S32 AK_Resume_Task(T_hTask task);
调用示例	T_S32 status; /*恢复先前调用 AK_Suspend_Task 服务挂起的任务"Task"。假设"Task"在先前已经调用 AK_Create_Task 创建*/ status = AK_Resume_Task(Task);

3.1.5 AK_Terminate_Task

原　　型	T_S32 AK_Terminate_Task(T_hTask task);	
功能概述	此项服务终止 task 参数指定的任务。	
参数说明	Task	任务控制句柄
返回值说明	AK_SUCCESS	表示服务成功完成
	AK_INVALID_TASK	表示任务句柄无效
注意事项		
调用示例	T_S32 status; /*终止任务"Task"，假设"Task"在先前已经调用 AK_Create_Task 创建*/ status = AK_Terminate_Task(Task);	

3.1.6 AK_Reset_Task

原　　型	T_S32 AK_Reset_Task(T_hTask task, T_U32 argc, T_VOID *argv);	
功能概述	此项服务复位先前已终止或结束的任务。	
参数说明	task	任务句柄
	argc	一个 T_U32 数据类型，可以用来传输信息到任务。
	argv	一个指针，可以用来传输信息到任务。
返回值说明	AK_SUCCESS	表示服务成功调用
	AK_INVALID_TASK	表示任务指针无效

原　　型	T_S32 AK_Reset_Task(T_hTask task, T_U32 argc, T_VOID *argv);	
	AK_NOT_TERMINATED	表示指定的任务不是处于终止或完成状态。只有处于终止或完成状态的任务可以被复位。
注意事项		
调用示例	T_S32 status; /*复位先前已经终止的任务"Task"。argc 和 argv 传递任务值为 0 和 AK_NULL。假设 Task 在先前已经调用 AK_Create_Task 创建*/ status = AK_Reset_Task(Task,0,AK_NULL);	

3.1.7 AK_Change_Priority

原　　型	T_OPTION AK_Change_Priority(T_hTask Task, T_OPTION new_priority);	
功能概述	此项服务改变指定任务的优先级为包含新优先级的值。优先级为从 0 到 255 范围的数值。数字越小任务优先级越高。	
参数说明	Task	任务句柄
	new_priority	新的优先级
返回值说明	此项服务返回调用程序先前的优先级。	
注意事项		
调用示例	T_OPTION old_priority; /*更改任务"Task"的优先级为 10。假设"Task"在先前已经调用 AK_Create_Task 服务创建*/ old_priority = AK_Change_Priority (Task,10); /*恢复*/ AK_Change_Priority(Task,old_priority);	

3.1.8 AK_Sleep

原　　型	T_VOID AK_Sleep(T_U32 ticks);
功能概述	此项服务挂起正在调用任务至指定的定时器节拍数。

原　　型	T_VOID AK_Sleep(T_U32 ticks);	
参数说明	ticks	挂起的节拍数
返回值说明		
注意事项		
调用示例	/*休眠 20 个定时器节拍*/ AK_Sleep(20);	

3.1.9 AK_Relinquish

原　　型	T_VOID AK_Relinquish(T_VOID);	
功能概述	此项服务将 CPU 让位给其他同一优先级就绪任务。	
参数说明		
返回值说明	如果没有同优先级任务就绪，只有低优先级任务就绪，则任务继续运行	
注意事项		
调用示例	/*允许其他同一优先级就绪任务在调用任务再次运行之前执行。*/ AK_Relinquish();	

3.1.10 AK_Task_Status

原　　型	T_S32 AK_Task_Status(T_hTask task);	
功能概述	此项服务查询当前任务的状态	
参数说明	T_hTask	任务的句柄
返回值说明	AK_INVALID_TASK	错误的任务句柄，负数的返回值。其他的任务状态都是正数
	AK_READY	任务处于就绪状态，可是由于优先级的原因没有执行
	AK_FINISHED	任务处于完成状态
	AK_TERMINATED	任务处于被终止状态

原　　型	T_S32 AK_Task_Status(T_hTask task);	
	AK_TASK_SUSPEND	任务处于挂起状态，任务被 suspend 函数挂起、创建后没有运行、sleep 自挂起均处于这个状态。
	AK_TASK_WAITING	任务在等待挂起中（队列、信号量、事件集等挂起均在这个状态）
注意事项		
调用示例	/*假设任务 task 已经调用 AK_Create_Task 创建*/ T_S32 status status = AK_Task_Status(task);	

3.2 列队通信接口列表

3.2.1 AK_Create_Queue

原　　型	T_hQueue AK_Create_Queue(T_VOID *start_address, UNSIGHED Queue_size, T_OPTION messge_type, T_U32 message_size, T_OPTION suspend_type);	
功能概述	此服务创建一个消息队列。队列创建支持定长和变长消息的管理。队列消息由多个 T_U8 数据元素组成。	
参数说明	start_address	指定队列的起始地址。
	Queue_size	指定队列中 T_U8 数据元素的总数
	message_type	指定队列管理的消息类型。AK_FIXED_SIZE 表示队列管理定长消息。注：定长消息只能用于消息尺寸均匀分布的队列区域。 AK_VARIABLE_SIZE 表示队列管理变长尺寸消息。

原　　型	T_hQueue AK_Create_Queue(T_VOID *start_address, UNSIGHED Queue_size, T_OPTION messge_type, T_U32 message_size, T_OPTION suspend_type);	
	Message_size	如果队列支持定长消息，这个参数指定每个消息的精确长度。另外，如果队列支持变长消息，这个参数表示最大消息尺寸。所有尺寸都是字节数。注：每个变长尺寸的消息在队列内需要一个附加的 T_U8 数据类型的消耗。附加填充字节对一个消息必不可少，以区分下一个消息。此附加数据不用在这里体现
	Suspend_type	指定队列挂起类型。此参数有效配置为 AK_FIFO 和 AK_PRIORITY，分别表示先入先出和优先级顺序挂起。
返回值说明	T_hQueue	服务成功完成，返回队列句柄
	AK_MEMORY_CORRUPT	申请内存空间失败
	AK_INVALID_MESSAGE	表示消息类型参数无效
	AK_INVALID_SIZE	表示指定的消息尺寸大于队列尺寸或为 0
	AK_INVALID_SUSPEND	表示挂起类型参数无效
注意事项		

原　　型	T_hQueue AK_Create_Queue(T_VOID *start_address, UNSIGHED Queue_size, T_OPTION messge_type, T_U32 message_size, T_OPTION suspend_type);
调用示例	/*假设队列句柄"Queue"定义为全局数据。这是分配控制块的几种方法之一*/ T_hQueue Queue; /*假设状态在本地定义*/ T_S32 status; /*创建一个起始地址由"start 指定，1000 个 T_U8 数据类型的队列。支持变长消息，最大消息尺寸为 20，队列任务挂起按照 FIFO 顺序"*/ Queue = AK_Create_Queue(start, 1000, AK_VARIABLE_SIZE, 20, AK_FIFO);

3.2.2 AK_Delete_Queue

原　　型	T_S32 AK_Delete_Queue(T_hQueue queue);	
功能概述	此服务删除一个先前定义的消息队列。参数 Queue 确定需要删除的消息队列。在这个队列上挂起的任务恢复时返回适当的错误状态。	
参数说明	Queue	队列的句柄
返回值说明	AK_SUCCESS	表示成功删除队列
	AK_INVALID_QUEUE	表示队列非法
注意事项		

原　　型	T_S32 AK_Delete_Queue(T_hQueue queue);
调用示例	T_hQueue Queue; T_S32 status; /*删除 Queue 指定的队列。假设"Queue"在先前已经调用 AK_Create_Queue 创建*/ status = AK_Delete_Queue(queue); /*这里 status 表示服务请求成功与否*/

3.2.3 AK_Receive_From_Queue

原　　型	T_S32 AK_Receive_From_Queue(T_hQueue queue, T_VOID *message, T_U32 size, T_U32 *actual_size, T_U32 suspend);	
功能概述	此项服务从指定的队列中接收一个消息。如果队列包含一个或多个消息，立即从队列里移除前面的消息且拷贝到指定的位置。	
参数说明	Queue	队列的句柄
	message	消息目的指针。注：消息目标必须足够大能容纳 size 的字节数。
	size	指定消息内的 T_U8 数据类型数。这个值必须对应于在队列创建时定义的消息尺寸。只适用于定义定长字节的队列；否则忽略。
	actual_size	指向保存接收消息实际 T_U8 数据类型数的变量的指针。 注意：这里是指针，是输出项，而上一个 size 是常量，输入项

原　型	T_S32 AK_Receive_From_Queue(T_hQueue queue, T_VOID *message, T_U32 size, T_U32 *actual_size, T_U32 suspend);	
	Suspend	如果队列为空，指定是否挂起正在调用的任务。 下面是挂起类型的有效选项： AK_NO_SUSPEND 不管请求是否满足，服务立即返回。注：如果服务被非任务线程调用，这是唯一有效的配置。 AK_SUSPEND 正在调用的任务挂起直到消息有效。 时间间隔值（1~4294967293） 正在调用的服务挂起直到消息有效或直到指定的定时器节拍数到时。

返回值说明	AK_SUCCESS	表示服务成功
	AK_INVALID_QUEUE	表示队列非法
	AK_INVALID_POINTER	表示消息指针为空或者 actual_size 指针为空
	AK_INVALID_SIZE	表示 size 参数不同于队列支持的消息尺寸。只适用于定义定长字节的队列。
	AK_INVALID_SUSPEND	表示试图从非任务线程挂起
	AK_QUEUE_EMPTY	表示队列为空
	AK_TIMEOUT	表示在挂起到指定超时值之后，队列仍然为空
	AK_QUEUE_DELETE	在任务挂起期间队列被删除
注意事项	如果在中断等非任务线程中调用此接口，挂起参数设置不是 AK_NO_SUSPEND，则不论队列是否为空，都会返回错误 AK_INVALID_SUSPEND	
调用示例	T_hQueue Queue; T_U8 message[4]; T_U32 actual_size; T_S32 status; /*从"Queue"指定的队列中接收一个 4 个 T_U32 数据类型消息。如果队列为空，挂起直到请求被满足。假设"Queue"在先前已经调用务 AK_Create_Queue 创建*/ status = AK_Receive_From_Queue(queue, &message[0], 4, &actual_size, AK_NO_SUSPEND); /*这里 status 表示服务请求成功与否。如果成功，"message"包含了接收的队列消息*/	

3.2.4 AK_Send_To_Queue

原　型	T_S32 AK_Send_To_Queue(T_hQueue queue, T_VOID *message, T_U32 size, T_U32 suspend);	
功能概述	此项服务放置消息到指定的队列底端。如果队列有足够的空间保存消息，此项服务立即处理。根据队列支持的消息类型，队列消息包括定长或变长 T_U8 数据类型数。	
参数说明	Queue	队列的句柄
	message	发送消息指针
	size	指定消息的 T_U8 类型数目 如果队列支持变长消息，此项参数必须等于或小于队列支持的消息尺寸。如果队列支持定长消息，此参数必须正好等于队列支持的消息尺寸。
	Suspend	如果队列已经装满了消息，指定是否挂起调用任务下列挂起配置是有效的 AK_NO_SUSPEND 不管请求是否被满足，服务立即返回。 注：如果服务从非任务线程调用，这是唯一有效的配置。 AK_SUSPEND 调用任务挂起直到队列空间有效。 时间间隔值（1 ～ 4294967293）调用任务挂起直到一个队列消息放置，或者直到指定数量的时钟节拍到时。

返回值说明	AK_SUCCESS	表示服务成功
	AK_INVALID_QUEUE	表示队列无效
	AK_INVALID_POINTER	表示消息指针为 AK_NULL
	AK_INVALID_SIZE	表示消息尺寸与队列支持的消息尺寸不匹配
	AK_INVALID_SUSPEND	表示试图从非任务线程挂起
	AK_QUEUE_FULL	表示队列满
	AK_TIMEOUT	表示即使在挂起超时之后，队列状态仍然为满
	AK_QUEUE_DELETE	任务在挂起期间队列被删除
注意事项	如果在中断等非任务线程中调用此接口，挂起参数设置不是 AK_NO_SUSPEND，则不论队列是否为满，都会返回错误 AK_INVALID_SUSPEND	

调用示例	T_hQueue Queue;
	T_U8 message[4];
	T_S32 status;
	/*建立一个 4 个 T_U32 变量的消息用于发送。"message"内容无意义*/
	message[0] = 0x11;
	message[1] = 0x22;
	message[2] = 0x33;
	message[3] = 0x44;
	/*发送 4 字节消息至队列"Queue"。挂起调用任务直到消息可以发送。假设"Queue"在先前已经通过调用服务 AK_Create_Queue 创建。*/
	status = AK_Send_To_Queue(Queue, &message[0], 4, NO_NO_SUSPEND);
	/*这里 status 表示服务请求是否成功。如果成功，"message"发送到"Queue"*/	

3.2.5 AK_Broadcast_To_Queue

原　型	T_S32 AK_Broadcast_To_Queue(T_hQueue queue, T_VOID *message, T_U32 size, T_U32 suspend);	
功能概述	此项服务广播一个消息到从指定的队列等待消息的所有任务。如果没有任务在等待，消息只是放到队列末端。根据队列的创建，队列消息由定长或不定长字节组成。	
参数说明	Queue	队列的句柄
	message	发送消息指针
	size	指定消息的 T_U8 类型数目 如果队列支持变长消息，此项参数必须等于或小于队列支持的消息尺寸。如果队列支持定长消息，此参数必须正好等于队列支持的消息尺寸。
	Suspend	如果队列已经装满了消息，指定是否挂起调用任务 AK_NO_SUSPEND 不管请求是否满足服务立即返回。 AK_SUSPEND 调用任务挂起直到消息被拷贝入队列。 时间间隔值（1 ～ 4294967293） 调用任务挂起直到消息拷贝入队列或者直到指定的定时器节拍到时。
返回值说明	AK_SUCCESS	表示服务成功
	AK_INVALID_QUEUE	表示队列无效
	AK_INVALID_POINTER	表示消息指针为 AK_NULL
	AK_INVALID_SIZE	表示消息尺寸与队列支持的消息尺寸不匹配
	AK_INVALID_SUSPEND	表示试图从非任务线程挂起

原　　型	T_S32 AK_Broadcast_To_Queue(T_hQueue queue, T_VOID *message, T_U32 size, T_U32 suspend);	
	AK_QUEUE_FULL	表示队列满
	AK_TIMEOUT	表示即使在挂起超时之后，队列状态仍然为满
	AK_QUEUE_DELETE	任务在挂起期间队列被删除
注意事项	如果在中断等非任务线程中调用此接口，挂起参数设置不是AK_NO_SUSPEND，则不论队列是否为满，都会返回错误AK_INVALID_SUSPEND	

调用示例	T_hQueue Queue;
	T_U8 Message[4];
	T_S32 status;
	/*建立一个四字节消息发送到队列。"message"内容没有特殊意义*/
	message[0] = 0x01;
	message[1] = 0x23;
	message[2] = 0x45;
	message[3] = 0x67;
	/*发送消息至队列"Queue"。如果队列已满，挂起直到请求被满足。假设"Queue"在先前已经调用 AK_Create_Queue 服务创建*/
	status = AK_Broadcast_To_Queue(queue, &message[0], 4, AK_SUSPEND);
	/*在这里，status 表示服务请求成功与否*/	

3.2.6 AK_Send_To_Front_of_Queue

原　型	T_S32 AK_Send_To_Front_Of_Queue(T_hQueue queue, T_VOID *message, T_U32 size, T_U32 suspend);	
功能概述	此项服务放置消息到指定的队列前端。如果队列有足够的空间保存消息，此项服务立即处理。根据队列支持的消息类型，队列消息由定长或不定长字节组成。	
参数说明	Queue	队列的句柄
	message	发送消息指针
	size	指定消息的 T_U32 类型数目 如果队列支持变长消息，此项参数必须等于或小于队列支持的消息尺寸。如果队列支持定长消息，此参数必须正好等于队列支持的消息尺寸。
	Suspend	如果队列已经装满了消息，指定是否挂起调用任务 下列挂起配置是有效的 AK_NO_SUSPEND 不管请求是否被满足，服务立即返回。 注：如果服务从非任务线程调用，这是唯一有效的配置。 AK_SUSPEND 调用任务挂起直到队列空间有效。 时间间隔值（1 ～ 4294967293）调用任务挂起直到一个队列消息放置，或者直到指定数量的时钟节拍到时。
返回值说明	AK_SUCCESS	成功发送消息到队列
	AK_INVALID_QUEUE	表示队列无效
	AK_INVALID_POINTER	表示消息指针为 AK_NULL
	AK_INVALID_SIZE	表示消息尺寸与队列支持的消息尺寸不匹配

原　型	T_S32 AK_Send_To_Front_Of_Queue(T_hQueue queue, T_VOID *message, T_U32 size, T_U32 suspend);	
	AK_INVALID_SUSPEND	表示试图从非任务线程挂起
	AK_QUEUE_FULL	表示队列满
	AK_TIMEOUT	表示即使在挂起超时之后，队列状态仍然为满
	AK_QUEUE_DELETE	任务在挂起期间队列被删除
注意事项	如果在中断等非任务线程中调用此接口，挂起参数设置不是AK_NO_SUSPEND，则不论队列是否为满，都会返回错误AK_INVALID_SUSPEND	

调用示例	T_hQueue Queue;
	T_U8 message[4];
	T_S32 status;
	/*建立一个 4 字节发送消息。"message"内容没有意义*/
	message[0] = 0x11;
	message[1] = 0x22;
	message[2] = 0x33;
	message[3] = 0x44;
	/*发送一个 4 字节，定长消息到队列"Queue"。挂起调用任务直到消息可以发送。假设"Queue"在先前已经调用服务 AK_Create_Queue 创建*/
	status = AK_Send_To_Front_Of_Queue(queue, &message[0], 4, AK_SUSPEND);
	/*这里 status 表示服务请求成功与否。如果成功，"message"发送到"Queue"*/	

3.2.7 AK_Send_Unique_To_Queue

原　　型	T_S32 AK_Send_Unique_To_Queue(T_hQueue queue, T_VOID *message, T_U32 size, T_U32 suspend, CallbakCompare Function);	
功能概述	此项服务放置消息到指定的队列低端。首先对队列中已经有的消息进行比较，如果已经有了和 message 相同的消息，则不再放入。如果没有相同消息，且队列有足够的空间保存消息，此项服务立即处理。此项服务目前只支持定长队列。	
参数说明	Queue	队列的句柄
	message	发送消息指针
	size	指定消息的 T_U32 类型数目 如果队列支持变长消息，此项参数必须等于或小于队列支持的消息尺寸。如果队列支持定长消息，此参数必须正好等于队列支持的消息尺寸。
	Suspend	如果队列已经装满了消息，指定是否挂起调用任务 下列挂起配置是有效的 AK_NO_SUSPEND 不管请求是否被满足，服务立即返回。 注：如果服务从非任务线程调用，这是唯一有效的配置。 AK_SUSPEND 调用任务挂起直到队列空间有效。 时间间隔值（1 ～ 4294967293）调用任务挂起直到一个队列消息放置，或者直到指定数量的时钟节拍到时。

原　　型	T_S32 AK_Send_Unique_To_Queue(T_hQueue queue, T_VOID *message, T_U32 size, T_U32 suspend, CallbakCompare Function);	
	Function	进行比较的回调函数。预定比较结果：如果比较结果相同返回 AK_TRUE，如果比较结果不同返回 AK_FALSE。
返回值说明	AK_SUCCESS	成功发送消息到队列
	AK_INVALID_QUEUE	表示队列无效
	AK_EXIST_MESSAGE	表示已经有相同的消息存在
	AK_INVALID_POINTER	表示消息指针为 AK_NULL
	AK_INVALID_SIZE	表示消息尺寸与队列支持的消息尺寸不匹配
	AK_INVALID_SUSPEND	表示试图从非任务线程挂起
	AK_QUEUE_FULL	表示队列满
	AK_TIMEOUT	表示即使在挂起超时之后，队列状态仍然为满
	AK_QUEUE_DELETE	任务在挂起期间队列被删除
注意事项	如果在中断等非任务线程中调用此接口，挂起参数设置不是 AK_NO_SUSPEND，则不论队列是否为满，都会返回错误 AK_INVALID_SUSPEND	

调用示例	T_hQueue Queue;
	T_U8 message[4];
	T_S32 status;
	/*建立一个 4 字节发送消息。"message"内容没有意义*/
	message[0] = message[1] = message[2] = message[3] = 0x11;
	/*发送一个 4 字节，定长消息到队列"Queue"。挂起调用任务直到消息可以发送。假设"Queue"在先前已经调用服务 AK_Create_Queue 创建*/
	status = AK_Send_To_Front_Of_Queue(queue, &message[0], 4, AK_SUSPEND, Function);
	/*这里 status 表示服务请求成功与否。如果成功，"message"发送到"Queue"*/	

3.2.8 AK_Send_Unique_To_Front_of_Queue

原　型	T_S32 AK_Send_Unique_To_ Front_of_Queue(T_hQueue queue, T_VOID *message, T_U32 size, T_U32 suspend, CallbakCompare Function);	
功能概述	此项服务放置消息到指定的队列顶端。首先对队列中已经有的消息进行比较，如果已经有了和 message 相同的消息，则不再放入。如果没有相同消息，且队列有足够的空间保存消息，此项服务立即处理。此项服务的具体功能可以参考 AK_Send_To_Front_of_Queue。此项服务目前只支持定长队列。	
参数说明	Queue	队列的句柄
	message	发送消息指针
	size	指定消息的 T_U32 类型数目 如果队列支持变长消息，此项参数必须等于或小于队列支持的消息尺寸。如果队列支持定长消息，此参数必须正好等于队列支持的消息尺寸。
	Suspend	如果队列已经装满了消息，指定是否挂起调用任务 下列挂起配置是有效的 AK_NO_SUSPEND 不管请求是否被满足，服务立即返回。 注：如果服务从非任务线程调用，这是唯一有效的配置。 AK_SUSPEND 调用任务挂起直到队列空间有效。 时间间隔值（1 ～ 4294967293）调用任务挂起直到一个队列消息放置，或者直到指定数量的时钟节拍到时。
	Function	进行比较的回调函数。预定比较结果：如果比较结果相同返回 AK_TRUE，如果比较结果不同返回 AK_FALSE。

原　　型	T_S32 AK_Send_Unique_To_ Front_of_Queue(T_hQueue queue, T_VOID *message, T_U32 size,　T_U32 suspend, CallbakCompare Function);	
返回值说明	AK_SUCCESS	成功发送消息到队列
	AK_INVALID_QUEUE	表示队列无效
	AK_EXIST_MESSAGE	表示已经有相同的消息存在
	AK_INVALID_POINTER	表示消息指针为 AK_NULL
	AK_INVALID_SIZE	表示消息尺寸与队列支持的消息尺寸不匹配
	AK_INVALID_SUSPEND	表示试图从非任务线程挂起
	AK_QUEUE_FULL	表示队列满
	AK_TIMEOUT	表示即使在挂起超时之后，队列状态仍然为满
	AK_QUEUE_DELETE	任务在挂起期间队列被删除
注意事项	如果在中断等非任务线程中调用此接口，挂起参数设置不是AK_NO_SUSPEND，则不论队列是否为满，都会返回错误AK_INVALID_SUSPEND	

调用示例	T_hQueue Queue;
	T_U8 message[4];
	T_S32 status;
	/*建立一个 4 字节发送消息。"message"内容没有意义*/
	message[0] = message[1] = message[2] = message[3] = 0x11;
	/*发送一个 4 字节，定长消息到队列"Queue"。挂起调用任务直到消息可以发送。假设"Queue"在先前已经调用服务 AK_Create_Queue 创建*/
	status = AK_Send_To_Front_Of_Queue(queue, &message[0], 4, AK_SUSPEND, Function);
	/*这里 status 表示服务请求成功与否。如果成功，"message"发送到"Queue"*/	

3.2.9 AK_Queue_Information

原　　型	T_S32 AK_Queue_Information(T_hQueue Queue, VOID *start_address, T_U32 *Queue_size, T_U32 *available, T_U32 *messages, T_U32 *message_type, T_U32 *suspend_size, T_U32 *tasks_waiting);	
功能概述	此项服务返回有关指定消息通讯队列的多种信息。	
参数说明	Queue	队列的句柄
	message	发送消息指针
	start_address	指向保存队列其实地址的内存指针的指针
	Queue_size	指向保存队列总字节数的变量的指针
	available	指向保存队列可用字节数的变量的指针
	messages	指向保存队列当前消息数量的变量的指针
	message_type	指向保存队列支持的消息类型的变量的指针。有效消息类型为：AK_FIXED_SIZE 和 AK_VARIABLE_SIZE
	message_size	指向保存每个队列消息中 T_U8 数据类型数的变量的指针。如果对列支持变长消息，这个值就是最大消息尺寸。
	suspend_type	指向保存任务挂起类型的变量的指针。有效任务挂起类型为：AK_FIFO 和 AK_PRIORITY
	tasks_waiting	指向保存等待队列的任务数的变量的指针
返回值说明	AK_SUCCESS	表示服务成功
	AK_INVALID_QUEUE	表示队列无效
注意事项		

原　　型	T_S32 AK_Queue_Information(T_hQueue Queue, VOID *start_address, T_U32 *Queue_size, T_U32 *available, T_U32 *messages, T_U32 *message_type, T_U32 *suspend_size, T_U32 *tasks_waiting);
调用示例	T_hQueue Queue; T_VOID *start_address; T_U32 Queue_size; T_U32 available; T_U32 messages; T_OPTION message_type; T_U32 message_size; T_OPTION suspend_type; T_U32 tasks_suspended; STATUS status; /*获得 Queue 指定的消息队列的信息。假设"Queue"在先前已经调用 AK_Create_Queue 创建*/ status = AK_Queue_Information(Queue, &start_addres, &Queue_size, &available, &messages, &message_type, &message_size, &suspend_type, &tasks_suspended,); /*如果 status 为 AK_SUCCESS,其他的信息就是精确的*/

3.3 任务同步服务信号量接口列表

3.3.1 AK_Create_Semaphore

原　　型	T_hSemaphore AK_Create_Semaphore(T_U32 initial_count, T_OPTION suspend_type);	
功能概述	此项服务创建一个计数信号量。信号量值范围 0～4294967294。	
参数说明	initial_count	指定信号量的初始值
	supend_type	指定任务挂起类型。此参数有效配置为 AK_FIFO 和 AK_PRIORITY，分别表示先入先出和优先级顺序挂起。
返回值说明	T_hSemaphore	表示成功创建信号量，返回句柄
	AK_INVALID_SUSPEND	表示挂起类型参数无效
注意事项		
调用示例	/*假设信号量句柄"Semaphore"定义为全局数据结构。这是分配控制块的几种方法之一*/ T_hSemaphore Semaphore; /*假设 status 在本地定义*/ T_S32 status; /*创建一个信号量，初始化值为 1，优先级顺序任务挂起*/ semaphore = AK_Create_Semaphore(1, AK_PRIORITY);	

3.3.2 AK_Delete_Semaphore

原　　型	T_S32 AK_Delete_Semaphore(T_hSemaphore semaphore);	
功能概述	此服务删除一个先前创建的信号量。参数 Semaphore 确定需要删除的信号量。在这个这个上挂起的任务恢复时返回适当的错误状态。在删除期间和之后，应用程序必须防止信号量的使用。	
参数说明	semaphore	指定信号量的句柄
返回值说明	AK_SUCCESS	表示服务成功完成

原　　型	T_S32 AK_Delete_Semaphore(T_hSemaphore semaphore);	
	AK_INVALID_SEMAPHORE	表示信号量句柄非法
注意事项		
调用示例	T_S32 status; /*删除 Semaphore 指定的信号量。假设"Semaphore"在先前已经调用服务 AK_Create_Semaphore 创建*/ status = AK_Delete_Semaphore(semaphore); /*这里 status 表示服务请求成功与否*/	

3.3.3 AK_Obtain_Semaphore

原　　型	T_S32 AK_Obtain_Semaphore(T_hSemaphore semaphore, T_U32 suspend);	
功能概述	此项服务获得指定信号量的实例。一旦实例通过内部计数器实现，获得一个信号量转化为消耗该信号量，内部计数器减一。如果信号量计数器在这个调用之前为 0，服务不能满足。	
参数说明	semaphore	指定信号量的句柄
	suspend	如果信号量不能被获得（当前为 0），指定正在调用的任务是否挂起。下列挂起类型有效： AK_NO_SUSPEND 不管请求是否满足，服务立即返回。注：如果服务从一个非任务线程被调用，这是唯一有效的配置。 AK_SUSPEND 正在调用的任务挂起直到信号量可以被获得。 时间间隔值（1～4294967293）正在调用任务挂起直到信号量可以被获得或者指定的定时器节拍值到时。
返回值说明	AK_SUCCESS	表示服务成功完成
	AK_INVALID_SEMAPHORE	表示信号量句柄非法

原　　型	T_S32 AK_Obtain_Semaphore(T_hSemaphore semaphore, T_U32 suspend);	
	AK_INVALID_SUSPEND	表示试图从一个非任务线程挂起
	AK_UNAVAILABLE	表示信号量难以获得
	AK_TIMEOUT	表示甚至在挂起指定的时间间隔后信号量仍然难以获得
	AK_SEMAPHORE_DELETE	在任务挂起期间信号量被删除
注意事项	如果在中断等非任务线程中调用此接口，挂起参数设置不是AK_NO_SUSPEND，则不论能否获得信号量，都会返回错误AK_INVALID_SUSPEND	
调用示例	T_S32 status; /*获得"Semaphore"指定信号量的一个实例。如果信号量难以获得，挂起最大20 个定时器时钟节拍。注：在同一信号量上多任务挂起顺序在信号量创建时确定。假设"Semaphore"在先前已经调用服务 AK_Create_Semaphore 创建*/ status = AK_Obtain_Semaphore(semaphore,20);	

3.3.4 AK_Release_Semaphore

原　　型	T_S32 AK_Release_Semaphore(T_hSemaphore semaphore);;	
功能概述	此项服务释放由参数 semaphore 指定的信号量的一个实例。如果有很多任务等待获得同一个信号量，create 的挂起类型为 AK_FIFO 则第一个等待的获得信号量，如果为 AK_PRIORITY 则优先级最高的获得信号量。另外，如果没有任务等待这个信号量，内部计数器加一。	
参数说明	semaphore	指定信号量的句柄
返回值说明	AK_SUCCESS	表示服务成功完成
	AK_INVALID_SEMAPHORE	表示信号量句柄非法
注意事项		

原　　型	T_S32 AK_Release_Semaphore(T_hSemaphore semaphore);;
调用示例	/*释放"Semaphore"指定的信号量的一个实例。如果其他任务正在等待获得同一个信号量，这项服务的结果就是传输这个信号量的实例到第一个等待的任务。假设"Semaphore"在先前已经调用服务 AK_Create_Semaphore 创建*/ status ＝ AK_Release_Semaphore(semaphore);

3.4 邮箱通信接口

3.4.1 AK_Create_Mailbox

原　　型	T_hMailbox AK_CreateMailbox(T_OPTION suspend_type);	
功能概述	本服务创建一个任务通信邮箱。一个邮箱可以保存一条消息。邮箱消息大小等于四个 T_U32 数据类型。	
参数说明	Suspend_type	指定邮箱任务挂起模式。这个参数的有效配置为 AK_FIFO 和 AK_PRIORITY,分别表示先进先出和优先级顺序任务挂起
返回值说明	T_hMailbox	表示服务成功完成，返回邮箱句柄
	AK_MEMORY_CORRUPT	表示申请内存空间失败
	AK_INVALID_SUSPEND	表示 suspend_type 参数无效
注意事项		
调用示例	/*假设邮箱句柄"mailbox"在全局数据结构中定义。这是分配的方法之一*/ T_hMailbox mailbox; /*假设状态在本地定义*/ T_S32 status;/*邮箱创建状态*/ /*创佳一个邮箱以 FIFO 方式管理任务挂起*/ mailbox = AK_Create_Mailbox(AK_FIFO);	

3.4.2 AK_Delete_Mailbox

原　　型	T_S32 AK_Delete_Mailbox(T_hMailbox mailbox);	
功能概述	此服务删除先前创建的邮箱。参数 mailbox 识别需要删除的邮箱。在这个邮箱上的任务挂起恢复时返回相应的错误状态。在删除期间和之后，应用程序必须防止这个邮箱的使用。	
参数说明	mailbox	邮箱的句柄
返回值说明	AK_SUCCESS	表示服务成功
	AK_INVALID_MAILBOX	表示邮箱句柄非法
注意事项		
调用示例	T_S32 status; /*删除"Mailbox"指定的邮箱句柄。假设"Mailbox"已经在先前调用服务 AK_Create_Mailbox 创建*/ status = AK_Delete_Mailbox(mailbox);	

3.4.3 AK_Broadcast_To_Mailbox

原　　型	T_S32 AK_Broadcast_To_MailBox(T_hMailbox mailbox, T_U32 *message, T_U32 Suspend);	
功能概述	这项服务广播一个消息到所有正在等待指定的邮箱消息的任务。如果没有任务在等待，消息只是简单的放到邮箱中。	
参数说明	mailbox	邮箱的句柄
	message	发送消息指针

原　型	T_S32 AK_Broadcast_To_MailBox(T_hMailbox mailbox, T_U32 *message, T_U32 Suspend);	
	Suspend	如果队列已经包含了一个消息，指定是否挂起调用任务 下列挂起配置是有效的 AK_NO_SUSPEND 不管请求是否被满足，服务立即返回。 注：如果服务从非任务线程调用，这是唯一有效的配置。 AK_SUSPEND 调用任务挂起直到邮箱空间有效。 时间间隔值（1 ～ 4294967293）调用任务挂起直到一个队列消息放置，或者直到指定数量的时钟节拍到时。
返回值说明	AK_SUCCESS	表示服务成功
	AK_INVALID_MAILBOX	表示邮箱指针无效
	AK_INVALID_POINTER	表示消息指针为 AK_NULL
	AK_INVALID_SUSPEND	表示试图从非任务线程挂起
	AK_MAILBOX_FULL	表示邮箱满
	AK_TIMEOUT	表示即使在挂起超时之后，邮箱状态仍然为满
	AK_MAILBOX_DELETE	任务在挂起期间邮箱被删除
注意事项	1、邮箱消息大小为 4 个 T_U32 类型大小，如果过小可能造成内存出错 2、如果在中断等非任务线程中调用此接口，挂起参数设置不是 AK_NO_SUSPEND，则不论邮箱是否为满，都会返回错误 AK_INVALID_SUSPEND	

原　　型	T_S32 AK_Broadcast_To_MailBox(T_hMailbox mailbox, T_U32 *message, T_U32 Suspend);

调用示例	T_hMailbox Mailbox;
	T_U32 message[4];
	T_S32 status
	/*建立一个消息发送到邮箱。"消息"内容没有意义*/
	message[0] = 0x00001111;
	message[1] = 0x22223333;
	message[2] = 0x44445555;
	message[3] = 0x66667777;
	/*发送消息至邮箱句柄"Mailbox"。如果邮箱已经包含一个消息，挂起 20 个定时器节拍。假设"Mailbox"在先前通过调用 AK_Create_Mailbox 服务创建*/
	status = AK_Broadcast_To_Mailbox(mailbox, &message[0], 20);

3.4.4 AK_Send_To_Mailbox

原　　型	T_S32 AK_Send_To_MailBox(T_hMailbox mailbox, T_U32 *message, T_U32 Suspend);	
功能概述	此项服务放置消息到指定的邮箱。如果邮箱为空，消息立即拷贝到邮箱中。邮箱消息大小等于 4 个 T_U32 数据类型。	
参数说明	mailbox	邮箱的句柄
	message	发送消息指针

原　型	T_S32 AK_Send_To_MailBox(T_hMailbox mailbox, T_U32 *message, T_U32 Suspend);	
	Suspend	如果队列已经包含了一个消息，指定是否挂起调用任务 下列挂起配置是有效的 AK_NO_SUSPEND 不管请求是否被满足，服务立即返回。 注：如果服务从非任务线程调用，这是唯一有效的配置。 AK_SUSPEND 调用任务挂起直到邮箱空间有效。 时间间隔值（1 ～ 4294967293）调用任务挂起直到一个队列消息放置，或者直到指定数量的时钟节拍到时。
返回值说明	AK_SUCCESS	表示服务成功
	AK_INVALID_MAILBOX	表示邮箱指针无效
	AK_INVALID_POINTER	表示消息指针为 AK_NULL
	AK_INVALID_SUSPEND	表示试图从非任务线程挂起
	AK_MAILBOX_FULL	表示邮箱满
	AK_TIMEOUT	表示即使在挂起超时之后，邮箱状态仍然为满
	AK_MAILBOX_DELETE	任务在挂起期间邮箱被删除
注意事项	1、 邮箱消息大小为 4 个 T_U32 类型大小，如果过小可能造成内存出错 2、 如果在中断等非任务线程中调用此接口，挂起参数设置不是 AK_NO_SUSPEND，则不论队列是否为满，都会返回错误 AK_INVALID_SUSPEND	

原　型	T_S32 AK_Send_To_MailBox(T_hMailbox mailbox, T_U32 *message, T_U32 Suspend);

调用示例	T_hMailbox Mailbox;
	T_U32 message[4];
	T_S32 status;
	/*建立一个 4 个 T_U32 变量的消息用于发送。"message"内容无意义*/
	message[0] = 0x00001111;
	message[1] = 0x00002222;
	message[2] = 0x00003333;
	message[3] = 0x00004444;
	/*发送消息至邮箱句柄"Mailbox"。挂起调用任务直到消息可以发送或者 14 个定时器节拍到期。假设"Mailbox"在先前已经通过调用服务 AK_Create_Mailbox 创建。*/
	status = AK_Send_To_Mailbox(Mailbox, &message[0], 14);
	/*这里 status 表示服务请求是否成功。如果成功，"message"发送到 "Mailbox"*/

3.4.5 AK_Receive_From_Mailbox

原　型	T_S32 AK_Receive_From_Mailbox(T_hMailbox mailbox, T_U32 *message, T_U32 suspend);	
功能概述	此项服务从指定的邮箱中接收消息。如果邮箱包含一个消息，立即从邮箱里移除且拷贝到指定的位置。邮箱消息尺寸等于 4 个 T_U32 数据类型。	
参数说明	mailbox	邮箱的句柄
	message	发送消息指针

原　型	T_S32 AK_Receive_From_Mailbox(T_hMailbox mailbox, T_U32 *message, T_U32 suspend);	
	Suspend	如果队列已经包含了一个消息，指定是否挂起调用任务 下列挂起配置是有效的 AK_NO_SUSPEND 不管请求是否被满足，服务立即返回。 注：如果服务从非任务线程调用，这是唯一有效的配置。 AK_SUSPEND 调用任务挂起直到邮箱消息有效。 时间间隔值（1 ～ 4294967293）调用任务挂起直到一个队列消息放置，或者直到指定数量的时钟节拍到时。
返回值说明	AK_SUCCESS	表示服务成功
	AK_INVALID_MAILBOX	表示邮箱指针无效
	AK_INVALID_POINTER	表示消息指针为 AK_NULL
	AK_INVALID_SUSPEND	表示试图从非任务线程挂起
	AK_MAILBOX_EMPTY	表示邮箱满
	AK_TIMEOUT	表示即使在挂起超时之后，邮箱状态仍然为满
	AK_MAILBOX_DELETE	任务在挂起期间邮箱被删除
注意事项	1、 邮箱消息大小为 4 个 T_U32 类型大小，如果过小可能造成内存出错 2、 如果在中断等非任务线程中调用此接口，挂起参数设置不是 AK_NO_SUSPEND，则不论队列是否为空，都会返回错误 AK_INVALID_SUSPEND	

原　　型	T_S32 AK_Receive_From_Mailbox(T_hMailbox mailbox, T_U32 *message, T_U32 suspend);
调用示例	T_hMailbox Mailbox; T_U32 message[4]; T_S32 status; /*从"Mailbox"指定的邮箱句柄中接收一个消息。如果邮箱为空，挂起 20 个定时器节拍。注：在同一个邮箱上多任务挂起的顺序在邮箱创建时决定。假设"Mailbox"在先前已经调用服务 AK_Create_Mailbox 创建*/ status = AK_Receive_From_Mailbox(mailbox,&message[0],20);

3.5 事件集使用接口

3.5.1 AK_Create_Event_Group

原　　型	T_hEventGroup AK_Create_Event_Group (T_VOID)	
功能概述	此项服务创建一个事件标志集。每个事件标志集包含 32 个事件标志。所有事件标志都初始化为 0。	
参数说明		
返回值说明	T_hEventGroup	表示成功创建事件集，返回句柄
	AK_MEMORY_CORRUPT	内存申请失败
注意事项		
调用示例	/*假设事件集句柄"group"已经定义为全局数据结构。这是分配控制块方法之一*/ T_hEventGroup group /*假设状态在本地定义*/ T_S32 status;/*事件集创建状态 */ /*创建一个事件标志集*/ group = AK_Create_Event_Group(); /*在这里 status 表示服务成功是否成功*/	

3.5.2 AK_Delete_Event_Group

原　　型	T_S32 AK_Delete_Event_Group(T_hEventGroup group);	
功能概述	此项服务删除一个先前定义的事件标志集。参数 group 用于识别需要删除的事件标志集。在这个事件集上挂起的任务恢复时返回适当的错误状态。在删除期间和删除之后，应用程序必须防止这个事件集的使用。	
参数说明	group	事件集句柄
返回值说明	AK_SUCCESS	表示服务成功完成
	AK_INVALID_GROUP	表示事件标志集句柄无效
注意事项		
调用示例	T_hEventGroup Group; T_S32 Status; /*删除"Group"指定的事件标志集句柄。假设"Group"先前已经调用 AK_Create_Event_Group 创建*/ status = AK_Delete_Event_Group(group); /*这里 status 表示服务请求是否成功*/	

3.5.3 AK_Set_Events

原　　型	T_S32 AK_Set_Events(T_hEventGroup group T_U32 event_flags T_OPTION operation);	
功能概述	此项服务在指定的事件集中设置指定的事件标志。事件集的事件标志请求被这项服务满足时，任何等待这个事件集的任务恢复。	
参数说明	group	事件集句柄
	requested_events	被请求的事件标志。置位表示相应的事件标志被请求

原　　型	T_S32 AK_Set_Events(T_hEventGroup group T_U32 event_flags T_OPTION operation);	
	operation	有两个操作选项有效：AK_OR 和 AK_AND。AK_OR 导致指定的事件标志与当前事件集中的事件标志进行或操作。AK_AND 导致指定的事件标志与当前事件集中的事件标志进行与操作。
返回值说明	AK_SUCCESS	表示服务成功完成
	AK_INVALID_GROUP	表示事件标志集句柄无效
	AK_INVALID_OPERATION	表示 operation 参数无效
注意事项		
调用示例	T_S32 status; /*在事件集句柄"Group"中设置事件标志 7，2，1。假设"Group"在先前已经调用服务 AK_Create_Event_Group 创建*/ status = AK_Set_Events(group, 0x00000086, AK_OR); /*如果 status 为 AK_SUCCESS，事件标志设置成功。*/	

3.5.4 AK_Retrieve_Events

原　　型	T_S32 AK_Retrieve_Events(T_hEventGroup group, T_U32 requested_events, T_OPTION operation, T_U32 *retrieved_events, T_U32 suspend);	
功能概述	此项服务从指定的事件标志集中找回指定的事件标志联合(combination)。如果联合(combination)出现，服务立即完成。	
参数说明	group	事件集句柄
	requested_events	被请求的事件标志。置位表示相应的事件标志被请求

原　型	T_S32 AK_Retrieve_Events(	T_hEventGroup group, T_U32 requested_events, T_OPTION operation, T_U32 *retrieved_events, T_U32 suspend);
	operation	有四个操作配置有效： AK_AND,AK_AND_CONSUME,AK_OR,AK_OR_CONSUME。 AK_AND 和 AK_AND_CONSUME 配置表示所有的已经被请求的事件标志都是需要的。AK_OR 和 AK_OR_CONSUME 配置表示已经被请求的事件标志中的一个或几个就够了。 CONSUME 配置自动清除请求成功的事件标志。
	retrieved_events	包含真正收回的事件标志，是输出用指针
	suspend	如果被请求的事件标志联合无效，指定调用任务是否挂起。 下列挂起配置是有效的 AK_NO_SUSPEND 不管请求是否被满足，服务立即返回。 注：如果服务从非任务线程调用，这是唯一有效的配置。 AK_SUSPEND 调用任务挂起直到事件标志标志有效。 时间间隔值（1 ～ 4294967293）调用任务挂起直到一个事件标志有效，或者直到指定数量的时钟节拍到时。
返回值说明	AK_SUCCESS	表示服务成功完成
	AK_INVALID_GROUP	表示事件标志集句柄无效
	AK_INVALID_POINTER	表示收回的事件标志指针为 AK_NULL
	AK_INVALID_OPERATION	表示 operation 参数无效
	AK_INVALID_SUSPEND	表示试图从非任务线程挂起
	AK_NOT_PRESENT	表示被请求的事件标志联合当前没有出现
	AK_TIMEOUT	表示被请求的事件标志联合即使在挂起超时过后都未出现
	AK_GROUP_DELETED	表示在任务挂起时，事件标志集删除
注意事项	如果在中断等非任务线程中调用此接口，挂起参数设置不是 AK_NO_SUSPEND，则不论指定事件是否存在，都会返回错误 AK_INVALID_SUSPEND	

原　　型	T_S32 AK_Retrieve_Events(T_hEventGroup group, T_U32 requested_events, T_OPTION operation, T_U32 *retrieved_events, T_U32 suspend);
调用示例	T_U32 actual_flags; T_S32 status; /*从事件标志集句柄"Group"收回事件标志 7，2 和 1。注：所有的事件标志必须出现满足请求。如果他们没有出现，调用任务无条件挂起。当然，如果这个请求满足的时候，事件标志 7，2，1 被消灭。假设"Group"在先前已经调用服务 AK_Create_Event_Group 创建*/ status = AK_Retrieve_Events(group, 0x86, //1000 0110 AK_AND_CONSUME, &actual_flags, AK_SUSPEND);

3.6 定时器接口

3.6.1 AK_Create_Timer

原　　型	T_hTimer AK_Create_Timer(T_VOID (*expiration routine)(T_U32), T_U32 id, T_U32 initial_time, T_U32 reschedule_time, T_OPTION enable);	
功能概述	此项服务创建应用程序定时器。指定期满子程序在每个定时器每次到期时运行。应用到期子程序应该避免任务挂起配置。到期子程序挂起可以导致其他需要时钟的应用程序延时。	
参数说明	expiration_routine	在时钟到期时指定应用程序子程序运行
	id	一个由到时子程序提供的 T_U32 数据类型。参数用于帮助识别使用同一个到时子程序的定时器
	initial_time	为定时器子程序指定时钟节拍初始值
	reschedule_time	在第一次到时后指定时钟节拍到时值。如果参数为 0 的话，定时器只到时一次。

原　　型	T_hTimer AK_Create_Timer(	T_VOID (*expiration routine)(T_U32), T_U32 id, T_U32 initial_time, T_U32 reschedule_time, T_OPTION enable);
	enable	有效配置参数为：AK_ENABLE_TIMER 和 AK_DISABLE_TIMER。
返回值说明	T_hTimer	表示服务成功完成，返回定时器句柄
	AK_MEMORY_CORRUPT	表示申请内存失败
	AK_INVALID_FUNCTION	表示到时函数指针为空
	AK_INVALID_ENABLE	表示 enable 参数无效
注意事项		
调用示例	/*假设时钟控制句柄"Timer"定义为全局数据结构。这是分配控制块的方法之一*/ T_hTimer Timer; /*假设 status 在本地定义*/ T_S32 status; /*创建一个定时器，到时函数为"timer_expire",ID 为 0，初始化到时值为 23 个时钟节拍。在初始化到时值之后，定时其每 5 个时钟节拍到时。注意定时器在创建期间使能*/ Timer = AK_Create_Timer(timer_expire, 0, 23, 5, AK_ENABLE_TIMER);	

3.6.2 AK_Delete_Timer

原　　型	T_S32 AK_Delete_Timer(T_hTimer *Timer);	
功能概述	此服务删除一个先前定义的应用程序定时器。参数 Timer 确定需要删除的定时器。注：指定的定时器必须先于此项服务请求时禁止。在删除期间和之后，应用程序必须防止定时器的使用。	
参数说明	Timer	定时器的句柄

原　　型	T_S32 AK_Delete_Timer(T_hTimer *Timer);	
返回值说明	AK_SUCCESS	表示服务成功完成
	AK_NOT_DISABLED	表示指定的定时器不能禁止
注意事项		
调用示例	T_S32 status; /*删除 Timer 指定的定时器。假设“Timer”在先前已经调用 AK_Create_Timer 创建*/ status = AK_Delete_Timer(Timer);	

3.7 中断服务接口

3.7.1 AK_Create_HISR

原　　型	T_hHisr AK_Create_HISR(T_VOID (hisr_entry)(T_VOID), T_OPTION priority, T_VOID *stack_pointer, T_U32 stack_size);	
功能概述	此项服务创建一个高级中断服务子程序（HISR）。HISRs 允许调用大多数的 AKOS 服务，不像低级中断服务子程序（LISRs）。	
参数说明	hisr_entry	指定 HISR 的函数入口点
	name	HISR 名字符串指针，最长只有 8 字节
	priority	有三个 HISR 优先级（0－2）。优先级 0 为最高
	stack_pointer	HISR 的堆栈区指针。每个 HISR 有它自己的堆栈区。注意 HISR 堆栈已经被调用者分配过。
	stack_size	HISR 堆栈的字节数
返回值说明	AK_SUCCESS	表示服务成功完成
	AK_MEMORY_CORRUPT	表示内存申请失败
	AK_INVALID_ENTRY	表示 HISR 入口指针为空
	AK_INVALID_PRIORITY	表示 HISR 优先级为空
	AK_INVALID_MEMORY	表示堆栈指针为空
	AK_INVALID_SIZE	表示堆栈尺寸太小
注意事项		

原　　型	T_hHisr AK_Create_HISR(T_VOID (hisr_entry)(T_VOID), T_OPTION priority, T_VOID *stack_pointer, T_U32 stack_size);
调用示例	/*假设”HISR”定义为全局数据结构。这是分配控制块的方法之一*/ T_hHisr HISR /*假设 status 定义为局部变量*/ T_S32 status;/hisr 创建 status*/ /*创建一个 HISR。注意 HISR 入口函数为”HISR_Entry”和”stack_pointer”指向一个 400 字节预先分配内存块*/ HISR = AK_Create_HISR(HISR_Entry, 2, stack_pointer, 400);

3.7.2 AK_Delete_HISR

原　　型	T_S32 AK_Delete_HISR(T_hHisr hisr);	
功能概述	此服务删除一个先前创建的 HISR，参数 hisr 确定需要删除的 HISR。在删除期间和之后，应用程序必须防止 HISR 的使用。	
参数说明	hisr	HISR 的句柄
返回值说明	AK_SUCCESS	表示服务成功完成
	AK_INVALID_HISR	表示 HISR 句柄非法
注意事项		
调用示例	T_S32 status; /*删除指针”Hisr”指定的 HISR 句柄。假设”Hisr”再先前已经调用服务 AK_Create_HISR 创建*/ status = AK_Delete_HISR(Hisr); /*这里，status 表示服务请求是否成功*/	

3.7.3 AK_Activate_HISR

原　　型	T_S32 AK_Activeate_HISR (NT_hHisr hisr);	
功能概述	本服务激活由 hisr 指针指向的 HISR。如果指定的 HISR 正在运行，这次激活请求在当前运行结束之前不会处理。对每次激活请求，HISR 运行一次。	
参数说明	hisr	HISR 的句柄
返回值说明	AK_SUCCESS	表示服务成功完成
	AK_INVALID_HISR	表示 HISR 句柄非法
注意事项		
调用示例	T_S32 status; /*激活以前创建的 Operator_Input 输入 HISR，控制块由 Operator_Input 输入*/ status = AK_Activate_HISR(Operator_Input);	

3.7.4 AK_Register_LISR

原　　型	T_S32 AK_Register_LISR(T_S32 vector, T_VOID (*list_entry)(T_S32), T_VOID (**old_lisr)(T_S32));	
功能概述	此项服务使被 vector 指定的中断向量和被 list_entry 指向 LISR 函数联合起来。在调用指定的 LISR 之前系统上下文自动保存并且在 LISR 返回之后恢复。因此，LISR 函数可以用 C 语言编写。然而，LISRs 只允许访问少量的 AKOS 服务。如果必须和其他任务进程的交流，必须激活一个高优先级的中断服务子程序（HISR）。	
参数说明	vector	中断向量
	list_entry	中断向量指向的函数体
	old_lisr	[Out]指向旧有函数题的指针的指针
	AK_SUCCESS	表示服务成功完成
	AK_INVALID_VECTOR	表示指定的向量无效

原　型	T_S32 AK_Register_LISR(T_S32 vector, T_VOID (*list_entry)(T_S32), T_VOID (**old_lisr)(T_S32));	
	AK_NOT_REGISTERED	表示向量当前没有注册且分别注册由 lisr_entry 指定(Indicates the vector is not registered and de-registration was specified by lisr_entry.)
	AK_NO_MORE_LISRS	表示已注册 LISRs 的最大数已经超出了。最大数在启动项头文件中更改。如果更改需要重建库文件
注意事项	警告:如果一个 LISR 用汇编语言编写，它必须遵循 C 编译器关于寄存器用法和返回机制的约定。请查看关于 C-汇编交叉详细需求的编译器文档。	

调用示例	T_S32 status;
	T_VOID (*old_lisr)(T_S32);
	/*关联变量 10 到 LIST 函数"LISR_example"。假设 LISR 函数在下面定义：
	T_VOID LISR_example(T_S32 vector_number)
	{
	/*vector_number 包含实际的中断向量数*/
	/*除了 AK_Activate_HISR 和其他几个，其他的服务都不允许在这个函数中调用。*/
	}*/
	status = AK_Register_LISR(10,LISR_example,&old_lisr);
	/*如果 status 为 AK_SUCCESS,LISR_example 在中断向量 10 出现时运行。
	注："old_list"包含先前已注册的 LISR*/	

3.7.5 AK_Setup_Vector

原　　型	T_VOID *AK_Setup_Vector(T_S32 vector, T_VOID *new);	
功能概述	此项服务使用调用者（参数 new）自定义中断服务子程序代替 vector 指定的中断向量。服务返回先前中断向量内容。 警告：提供这个子程序的 ISRs 用汇编语言编写，负责存储和恢复任何使用的寄存器。一些端口有一些附加约束强加在这些 ISRs 上。请看附加指定目标信息的指定处理器端口要求。	
参数说明	vector	中断向量
	new	新的中断服务子程序
返回值说明		
注意事项		
调用示例	T_VOID *old_vector; /*放置命名为”asm_ISR”的汇编语言 ISR 到向量 5*/ old_vector = AK_Setup_Vector(5,asm_ISR);	