

Диспетчер Устройств «CH Products»

СОДЕРЖАНИЕ

Обзор

Что нового в Диспетчере версии 4

Основы Диспетчера Устройств

Термины

Основные Операции

Программный интерфейс Диспетчера Устройств

ГИП (Графический Интерфейс Пользователя) Диспетчера Устройств

Окно Графического Интерфейса

Панель инструментов

Функциональная Панель

Диалоговая Панель

Создание «Карты» (профиля) Диспетчера Устройств

Основные методы программирования Карт

Добавление Устройств

Использование мастера создания Карт

Использование мастера добавления устройств

Макросы клавиш

Специальные символы

Быстрый запуск

Запись Нажатия Клавиш

Вставка Команд

Режимы карты

Коды клавиш Диспетчера Устройств

(СМС) Файлы Команд

Командные файлы

Использование Командных файлов

Собственные Утилиты Диспетчера Устройств

Апплет Калибровки

Утилита CHDelete (Удаление)

Утилита CMStart (Старт CMS)

Throttle Quadrant (Устройство Квадрант Тяги)

Операции с Throttle Quadrant

Trackball Pro (Устройство Трекбол)

Операции с Трекбол
Функции Трекбол

(CMS) Файлы Сценариев

Введение
Основы CMS

Элементы сценариев CMS

Переменные
Имена переменных
Предопределенные переменные и константы
СМСС (Центр Управления Диспетчером), Переменные экрана и мыши
Операторы
Выражения
Директивы

Справочник Сценариев

Основы написания Сценариев
Определение Сценариев
Присваивание назначений
Арифметические сравнения
Условные операторы
Последовательности
Логические устройства
функция Масштабирования
функция Выбора
Установка Режимов в Карте

Прочее о CMS

Сценарии с трекболом
Советы по программированию
Примеры Сценариев

Редактор Диспетчера Устройств

Обзор Редактора
Панель Инструментов Редактора
Подсветка синтаксиса
Автоматическое завершение команд
Функции Ярлыков Редактора
Файлы Редактирования сценариев (CMS)
Редактирование Командных файлов (CMC)
Настройка Редактора

Установка/Удаление Диспетчера Устройств

Установка в Windows 98
Удаление в Windows 98
Установка в Windows 2K/XP
Удаление в Windows 2K/XP
Параметры установки

Дополнительно

SynEdit
Настройки в Inno

Обзор

Добро пожаловать на знакомство с Диспетчером Устройств фирмы СН Products! Диспетчер Устройств (далее просто Диспетчер) является легкой в использовании, но очень мощной системой, которая обеспечивает наиболее гибкие и широкие возможности программирования игровых Устройств СН, доступных сегодня.

Если вы новичок в Диспетчере, то Вам рекомендуется прочитать раздел Основные Операции, главы «Основы Диспетчера Устройств», чтобы получить представление о том, как различные части системы управления Диспетчера сочетаются друг с другом и как они взаимодействуют с самой ОС Windows. Если вы использовали одну из ранних версий Диспетчера и уже знакомы с основными операциями, то вы узнаете большинство объектов интерфейса и сможете сразу начать пользоваться им. Все существующие Карты (Профили) должны работать нормально. В новом Диспетчере появилось не так много новых функций, но думаю, вы захотите узнать и о них. Посмотрите раздел «Что нового?» для получения краткого обзора о новых возможностях программы.

Что нового?

Диспетчер версии 4 несколько улучшен по сравнению с более ранними версиями, а также добавлены некоторые новые функциональные возможности. Если вы знакомы с более ранними версиями, то наверное заметили, что процесс установки был переделан с применением инсталлятора InnoSetup. Новый инсталлятор включает компоненты, которые раньше были доступны только в Утилитах Диспетчера. Он также получил возможность обновления существующего ПО Диспетчера, начиная от версии 4 (и выше) без необходимости удаления программы. Эта возможность также может быть использована для добавления инструментов из Пакета Утилит Диспетчера в уже установленный Диспетчер, так, что если у Вас не были установлены Утилиты CMSC, CMPrint или CMList, а теперь вы захотите их использовать, просто запустите инсталлятор еще раз. Более подробная информация о новых возможностях установки доступна в разделе [Параметры установки](#) параграфа Установка/Удаление Диспетчера Устройств и разделе [Настройки в Inno](#) параграфа «Прочее».

Также появились некоторые дополнительные возможности программирования. Диспетчер4 включил в себя новый метод программирования списков команд для кнопок, которые будут вводиться по одному, на каждое повторное нажатие кнопки. Это бывает полезно для программирования последовательных функций и операций, для которых ранее требовалось использование блока SELECT в Сценариях CMS. Более подробная информация по программированию режимов Списков доступна в разделе [Режим «Список»](#).

В Диспетчер4 также добавлена базовая поддержка так называемых “Клавиш Windows” (RWIN и LWIN), поскольку они начали учитываться в некоторых новых играх. Поддержка является минимальной для “именных клавиш”, таких как “LSHF” (Левый Шифт), “RSHF” (Правый Шифт).

ГИП Диспетчера также получил некоторые новые дополнения. Макросы команд используемых в СМС (Файлах Команд) выделяются, если они уже были использованы, чтобы сразу дать вам понять, что определенная команда уже была назначена до этого. Этот вопрос будет более широко рассмотрен в разделе [Вставка Команд](#). Также это реализовано в новом «Поисковике Команд», в котором перечислены СМС команды, а те, которые были уже использованы «подсвечиваются». При нажатии на выделенную команду ГИП укажет, где используется данная команда. См. раздел о [Поисковике Команд](#), для получения дополнительной информации.

Редактор Диспетчера, также был переработан. Теперь в нём применяется технология «SynEdit». Это гораздо лучше, чем в редакторах более ранних версий, подсветка синтаксиса стала работать гораздо быстрее и функциональность значительно улучшилась. Для наиболее часто используемых команд доступны Ярлыки, а также появились средства для автоматического завершения команд, что будет полезно, когда будет трудно вспомнить, как именно должна быть закончена команда. Более подробная информация об использовании SynEdit в разделе [Редактор Диспетчера](#) данного руководства, а также приложении раздела [SynEdit](#), параграфа “Прочее”. Будет полезно перечитать эти разделы в связи с появлением новых функций, которые могут оказать вам большую помощь в создании CMS и СМС файлов.

Наконец, некоторые новые переменные стали доступны для использования при редактировании CMS Сценариев, если вы пользуетесь CM Control Center (Центр Управления Диспетчером Устройств, далее CMCC). Это позволит вам определять и устанавливать положение курсора на экране, определить текущее разрешение экрана средствами ваших сценариев CMS. См. раздел «переменные экрана и мыши CMCC», для получения дополнительной информации об использовании этих переменных.

Термины

Есть несколько терминов, которые используются в данном руководстве для определения различных элементов системы Диспетчера, дабы избежать двусмысленности. Они будут объяснены более подробно далее в руководстве, здесь они перечислены для справки.

Устройство

«Устройством» здесь будет называться одно из реальных (физически подключенных) устройств USB в вашей системе, такие как «Yoke LE» (Штурвал), «FighterStick» (Джойстик), «Pro Pedals» (Педали) и т.д.

Элемент Управления

Термин «управление» относится обычно к одной из кнопок, осей, «кноппелей» (переключателей обзора (англ. POV)) или «хаток» (от англ. «Hat» – шляпа, шапка – многопозиционный, обычно крестообразный, переключатель) на Устройстве.

Контроллер

«Контроллером» здесь будет называться любое устройство, которое определяется ОС Windows и просматривается в списке *апплета Игровых Устройств Windows*.

Прямо-подключенное Устройство

«Прямо-подключенное Устройство» представляет собой устройство, которое появляется в апплете *Игровых Устройств Windows* под его настоящим именем, таким как “CH Yoke LE”, “CH FighterStick” и т.д.

Устройство Диспетчера

«Устройство Диспетчера» это виртуальное устройство созданное системой управления Диспетчера Устройств. Они видны в апплете Windows как «Control Manager Device 1» (Устройство Диспетчера 1), «Control Manager Device 2» (Устройство Диспетчера 2), и так далее.

ГИП (Графический Интерфейс Пользователя)

Термин «ГИП» используется как упрощение для обозначения Основного Окна Диспетчера, который вы видите при запуске файла CHCtlMgr.exe для создания программ, и т.д.

Апплет Игровых Устройств

Это апплет Windows, который обычно используется ОС, чтобы добавить, удалить, и откалибровать стандартный контроллер под Windows.

HID Контроллер

Термин “HID” расшифровывается как “Human Interface Device” (Человеко-Машинный Интерфейс). HID Контроллеры Windows это обычно клавиатуры, мыши, джойстики.

Основные Операции

Диспетчер предоставляет вам все основные возможности для программирования осей и кнопок своих устройств, функций клавиш, управления мышью и т.д., что в общем-то и требуется от высокоорганизованных программируемых джойстиков.

Этот Диспетчер является уникальным, хотя бы потому, что он позволяет вам посредством своего *ГИП* трансформировать несколько *Устройств Диспетчера*, в одно *Виртуальное Устройство*. Это особенно необходимо в старых играх, которые могут различать лишь одно устройство, а также может быть полезно в более современных играх, так как это позволит вам настраивать множество *Устройств Диспетчера* в единой среде.

Кроме того, Диспетчером предоставлена Утилита создания и редактирования сценариев (скриптов) высокого уровня (CMS), которые могут быть использованы для назначения таких сложных функций, которые не могут быть обеспечены средствами стандартных функций программирования.

Сочетание функций программирования, умение сочетать оси и кнопки формируя устройства способные выполнять практически любые комбинации команд, а также возможность использования функций скриптов высокого уровня, когда требуются специальные функции, всё это делает Диспетчер Устройств СН самой мощной системой программирования джойстиков, которая есть на сегодняшний день.

Windows и USB-контроллеры

Чтобы получить представление о том, как работает система управления Диспетчера, полезно иметь некоторое представление о том, как Windows размещает Игровые Устройства USB. Если вы когда-либо прежде использовали устройства USB, вы могли заметить, что фактически вы получаете два устройства при подключении одного. Одним из них является устройство USB само по себе, а второе, как правило, «HID-совместимый игровой контроллер».

Устройство USB представляет собой сам контроллер аппаратных средств. При подключении устройства, он уведомляет систему, что подключен и является HID джойстиком. Система реагирует, создавая для него «HID-совместимый Игровой Контроллер» (“HID-compliant game controller”) чтобы обозначить это устройство для Windows. Это устройство HID обычно просто берет данные от устройства USB, преобразует их в вид, понятный для Windows, и передает их. Это именно те устройства, которые вы можете видеть в *апплете Игровых Устройств Windows*.

В чем же отличие Диспетчера?

Диспетчер работает почти так же, как и драйвера USB Контроллеров и большинства HID-драйверов, на самом деле это просто стандартные драйвера Windows. Основным отличием является то, что Диспетчер перехватывает контроль над созданием устройства HID вместо того, чтобы дать Windows принять их как физические устройства, тем самым позволяя себе решать, каким образом преподнести устройство для Windows, чтобы та поверила в это устройство как в физическое. Таким образом, вы можете создавать виртуальные “Джойстики” образованные комбинацией кнопок и осей

доступных на любом из устройств CH USB, к которым Диспетчер имеет доступ и Windows будет принимать их за реальные. Этот метод позволяет виртуальным устройствам становиться «реальными» и правильно понимаемыми Windows, а значит и практически всеми играми поддерживающими DirectX.

Режимы Подключения

Диспетчер всегда находится в одном из трех основных режимов работы, которые определяют, как Диспетчеру подключать (или не подключать) эти HID устройства. Ниже описаны различия между этими режимами.

Режим Прямого подключения (Прямой)

Это режим по умолчанию, в нём устройство работает, как после установки средствами Windows. В «прямом» режиме, Диспетчер создает HID устройства, которые соответствуют своим USB контроллерам, по сути, так же, как это делает Windows сама по себе. В этом случае, устройства появляются под их стандартными именами в списке *апплета Игровых Устройств* и могут использоваться непосредственно, как если бы Диспетчера не было.

Для иллюстрации, представим, что у вас есть система, к которой подключены джойстик «FighterStick», ручка управления тягой «ProThrottle», и комплект педалей «PedalPro». Схематически система может выглядеть примерно так:



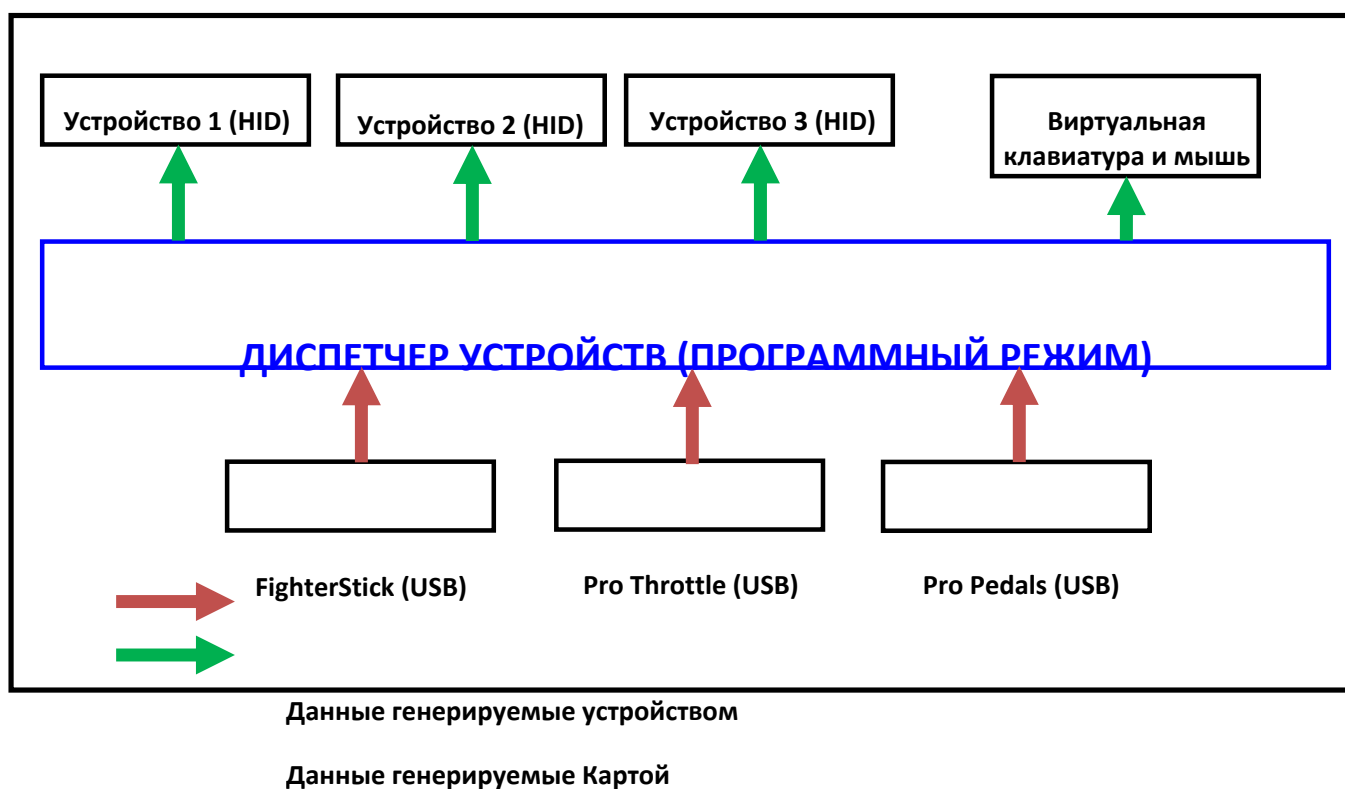
Блок синего цвета представляет собой наш Диспетчер. Ниже, USB-контроллеры и над Диспетчером, их HID коллеги. Диспетчер получает генерируемые устройствами USB

данные (показано красным) и в неизмененном виде передает их HID устройствам, которые представляют их Windows. Программирование не активно в «прямом» режиме, HID устройства в *Игровых Устройствах*, просто зеркала, которые говорят устройствам USB что они должны использовать стандартные драйвера Windows.

Режим программного подключения (Программный)

В «Программном режиме», Диспетчер получает данные от осей и кнопок устройств USB, как и в прямом режиме, разница лишь в том, что данные затем обрабатываются в соответствии с инструкциями заданными «картой». Оси и кнопки могут заменяться, формируя устройства наиболее подходящие для игры в которую вы собираетесь играть, а клавиши иметь такие назначения, которые позволяют обеспечить задачи не доступные средствами DirectX, а обработка скриптами позволяет создавать функции, не предоставляемые обычными средствами Диспетчера в **Прямом** режиме. И уже такие, обработанные данные, затем используется для создания HID устройств, которые видны Windows.

В этом режиме система управления Диспетчера выглядит примерно так:



В ГИП Диспетчера создаётся **Карта**. Она определяет, какие кнопки и оси устройства HID соответствуют каким кнопкам и осям на устройствах USB. В Программном режиме, есть также виртуальные клавиатура и мышь, для того чтобы кнопки и оси могли быть запрограммированы на ввод символов и использование функций мыши. Данные генерируемые устройствами (*показанные красным*) обрабатываются алгоритмами заданными в Диспетчере и трансформируются в данные генерируемые Картой (*показаны зелёным*).

Диспетчер предоставляет виртуальную клавиатуру и мышь, которые могут получать данные сформированные картой если это необходимо. Это применяется, для назначения нажатия клавиш или функций и команд мыши, если это задано Картой. Эти устройства будут отображаться в диспетчере устройств как дополнительные устройства «AUX1 Device» и «AUX2 Device».

Эти «виртуальные» HID устройства имеют имена, которые не соответствуют USB устройствам, если диспетчер находится в Программном режиме. Скорее всего, они будут называться «Control Manager Device 1», «Control Manager Device 2» и т.д. Это связано с тем, что полное соответствие между USB устройствами и HID устройствами или между элементами управления устройств USB и HID, вовсе не обязательно. Чтобы понять это, представьте «FighterStick», «ProThrottle», и «ProPedals», которые объединены Картой в одно устройство. Вряд ли имеет смысл называть это виртуальное устройство именем какого-либо одного из этих трех реальных устройств, для этого Диспетчером используется «кличка».

Очень важно понимать, что Устройство Диспетчера – «джойстик», что Windows будет видеть его и работать с ним, когда Карта активна, что оно ничем не будет отличаться от любого другого «реального» джойстика в системе. Кнопки и оси Устройства Диспетчера будут выглядеть и вести себя так же, как кнопки и оси нормального джойстика. Windows не должна увидеть разницу. Устройство Диспетчера, должно стать реальным «железом», что будет гарантировать его совместимость со всеми играми и симуляторами доступными сегодня.

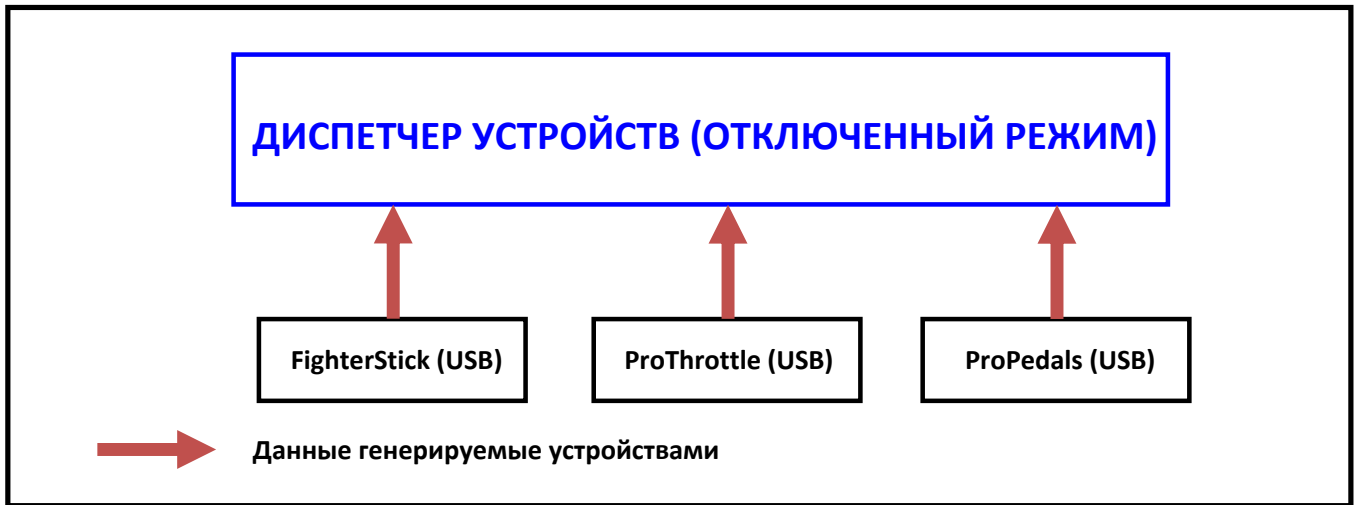
Часто количество устройств HID не совпадает с числом подключенных устройств USB. Такая ситуация наиболее вероятна, например, когда «комбинированные карты» создаются для игр, которые распознают только один джойстик, как описано выше. Все устройства USB будут иметь свои оси и кнопки запрограммированными в единое HID устройство Диспетчера. Любые оси и кнопки, которые не назначены картой на это единое устройство HID, смогут обеспечивать ввод символов или останутся неиспользуемыми. В этом случае HID устройств получится меньше, чем есть реальных.

Менее распространена, но все же возможна ситуация, когда больше Устройств Диспетчера, чем USB-устройств. Если вы используете игру или симулятор, который способен распознавать множество устройств прямого ввода (Microsofts Flight Simulator например), вы захотите сгенерировать больше кнопок, чем вам может предложить любое нормальное устройство ввода из доступных. Каждому устройству Диспетчер может назначить больше, чем 6 осей (8 с Windows XP) и 32 кнопки, не очень много, но их может быть помножено на 16 возможных Устройств Диспетчера! Так, что в играх и симуляторах, которые распознают несколько контроллеров, фактическое число управления может быть в количестве 96 осей (128 с XP) и 512 кнопок. Это, конечно, зависит от конкретного устройства, так, например, вы сможете сгенерировать до 128 кнопок и 12 осей одним лишь джойстиком «FighterStick».

В таких ситуациях, вы можете просто создать другое устройство HID (путём добавления в карту) и назначить дополнительные оси и кнопки на это устройство, таким образом, в *апплете Менеджера Игровых Устройств*, устройств станет больше, чем на самом деле подключено к компьютеру. Диспетчер может создать столько Устройств, на сколько сможет сослаться карта.

Отключенный Режим.

Последний режим «Отключенный», самый простой из всех. В этом режиме HID устройств не создаются вообще. Система выглядит следующим образом:



Диспетчер не создает никаких HID устройств, таким образом USB устройства как бы выключены, по крайней мере так это понимает Windows. Данные генерируемые устройствами не идут дальше Диспетчера самого по себе.

Запуск Windows

При запуске Windows, Диспетчер по умолчанию запускает «Прямой» режим. Перед использованием карты, Диспетчер должен быть установлен на Программный режим. Это не означает, что нужно снова скачивать программу, она по-прежнему установлена, просто выключена. Вы можете запустить ГИП Диспетчера и просто нажать «Программный режим», чтобы активировать его с последней используемой картой. Существует небольшая утилита [CMStart](#), которая может запускаться из меню «Автозагрузки» Windows и автоматически запускать карту, при каждом запуске Windows. CMStart не является резидентной утилитой, она просто включается, посылает команду на активацию карты, и выключается.

Другим вариантом является CM Control Center **CMCC** (Центр Управления Диспетчером), отдельная утилита, которая поставляется с Диспетчером. Она может запускаться из Автозагрузки и приводить Диспетчер в любой режим, который вы пожелаете. Она также имеет возможность редактирования и загрузки, «Launcher» (запускающий апплет) который будет загружать карты, а затем запускать игры, и ряд других функций, которые могут оказаться полезными. Центр Управления является резидентной утилитой, которая доступна с панели инструментов. CMCC имеет отдельное Руководство Пользователя, вы можете обратиться к нему, за подробным описанием функции, которые она предоставляет.

ГИП Диспетчера Устройств

Как описано ранее, Диспетчер работает, организуя взаимосвязь между осями и кнопками на реальном Устройстве СН и осями и кнопками Устройства Диспетчера, которые видит Windows. Кроме того, вы можете запрограммировать кнопки или оси, на ввод с клавиатуры или эмуляцию функций мыши. Такие назначения содержатся в файле «Карты». Основная задача ГИП Диспетчера, определение в карте какие функции, какие оси или кнопки есть на вашем устройстве. ГИП также дает вам возможность оперировать его функциями, и предоставляет средства для переключения режимов, калибровки и т.д., а также инструменты создания Карты. В этом разделе детально описываются ГИП и его функции.

Запуск ГИП

Запуск ГИП Диспетчера производится посредством файла CHCtrlMgr.exe, который был создан в процессе установки Диспетчера. Он доступен в меню «Пуск». Нажмите кнопку «Пуск», выберите «Программы». Если во время установки вы выбирали группу программ по умолчанию, то далее идите:

CH Products -> Control Manager -> CHCtrlMgr.exe

Вы также можете создать ярлык на EXE файл. По умолчанию, он находится в папке:

\\Program Files\\CH Products\\Control Manager

...Вашего основного жёсткого диска, но если вы выбрали другое место во время установки, то вы найдете его там. Ярлык может быть создан на рабочем столе или (по желанию) помещён на панели быстрого запуска. ГИП может быть запущена с помощью СМСС, если вы используете эту утилиту. См. Руководство Пользователя СМСС для получения дополнительной информации.

Если вы новичок в Диспетчере, экран ГИП сначала может показаться вам немного запутанным. Есть только кнопки в верхней части и большинство из них не активны. Это потому, что в карту ещё не добавлено ни одно Устройство, и ГИП пока нечего делать. Всё «оживёт» в мгновение, как только будет добавлено, по крайней мере, одно устройство. Обратитесь к разделу «[Добавление Устройств](#)» для получения дополнительных сведений о том, как это сделать.

Получение справки

Это руководство всегда доступно в ГИП, во время работы с Диспетчером. Вы можете получить доступ к нему нажав на кнопку «Справка» (Help) в верхней части экрана:



Дополнительная справка по любому пункту на главном экране ГИП, также можно получить, нажав кнопку “Что это такое?” (What`s This?):



Кнопка “прилипнет” к курсору и вы сможете щелкнуть левой кнопкой мыши на элемент, о котором вы хотите получить справку. Откроется всплывающее окно с кратким объяснением того, что делает этот пункт. В нижней части всплывающего экрана появится ссылка [Подробнее], нажатие на которую приведёт вас в тот раздел этого руководства, в котором обсуждается выбранный элемент. Нажатие в любом другом месте просто закроет всплывающее окно и вернёт кнопку “Что это такое?” на место. Для получения справки о другом элементе, вам нужно будет нажать кнопку “Что это такое?” еще раз. Для получения дополнительной информации по использованию Справки, см. раздел «[Кнопки Справки](#)».

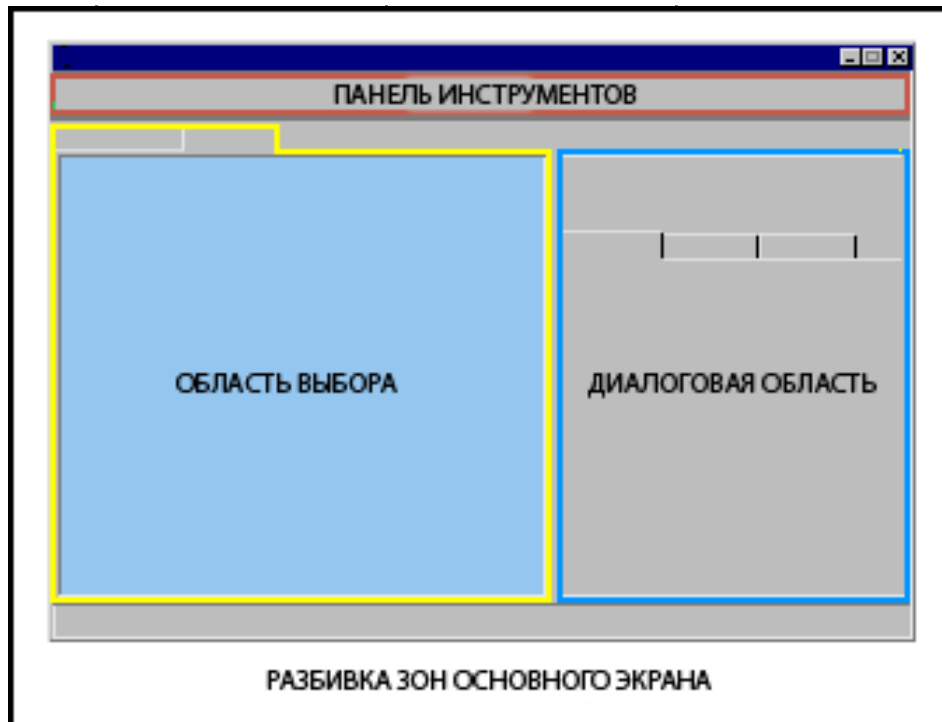
Блокировка Символов

Есть одна вещь, которая может немного озадачивать, это то что, при первом знакомстве с Диспетчером, когда ГИП активен, любые символы, которые являются результатом содержания Карты, блокируются. Это сделано для того, что бы загрузив карту, а затем, нажав кнопку, вы не могли непреднамеренно ввести символ в карту. Это упоминается здесь начиная с загрузки карт и может так же не генерировать видимых символов, что может вводить в заблуждение.

Блокировка не действует, когда запущен апплет «KeyTest» (Тест Клавиш) или если в данный момент активно любое окно Windows. Это делается для того, чтобы апплет «KeyTest» мог нормально работать и для того, чтобы если вам нужно будет перейти на другое приложение (NotePad например), символы, вводимые там не поступали одновременно в Диспетчер. Эффект блокировки действует лишь при активном ГИП самого Диспетчера.

Окно ГИП

Как только хотя бы одно Устройство добавлено, экран ГИП станет отображаться нормально. Он разделен на три основные зоны: Панель Инструментов, Область Выбора, и Диалоговую Область. Эти три области, расположены так:



На Панели Инструментов, расположены кнопки управления основными функциями интерфейса.

В Области Выбора, вы выбираете физическое устройство и элементы управления, которые вы хотите запрограммировать на этом устройстве.

В Диалоговой Панели, вы назначаете выбранный элемент управления одному из Устройств Диспетчера, которые становятся видимыми для Windows или назначаете некоторые не свойственные обычным джойстикам действия, например ввод клавиатурных символов или выполнение функций мыши, например.

Панель инструментов

Панель инструментов расположена в верхней части экрана ГИП. Она содержит кнопки для доступа к функциям различных элементов ГИП, а также “Панели Состояния”. Выглядит она примерно так:



Кнопки ГИП Панели Инструментов не имеют подписей, но при наведении на них курсора мыши, появляется всплывающая “подсказка”, которая описывает основные функции этой кнопки. Большинство из них вообще не требуют подсказок. Этот и следующий разделы предоставляют информацию о каждой из этих кнопок.

Чтобы перейти к «справке» для определенной кнопки, выберите её из списка ссылок ВНИЗУ.

Кнопка Открыть

Кнопка Сохранить

Кнопка Новая Карта

Кнопка Мастера Карт

Кнопка Добавить Устройство

Кнопка Удалить Устройство

Кнопка CMS Script Editor (Редактор Сценариев CMS)

Кнопка Загрузить

Утилита Тестирования Кнопок

Утилита Калибровки

Кнопка Поиска Команд

Кнопка Отключенный Режим

Кнопка Прямой Режим

Кнопка Программный Режим

Кнопка Что это такое?

Кнопка Справки

Панель состояния

Кнопка Выход

Область Выбора

Панель Выбора используется для выбора Устройства и элементов управления этого Устройства, которые вы хотите запрограммировать.

В процессе работы, это выглядит примерно так (в зависимости от Устройства, которое вы выбрали):



В общем, это графическое изображение самого контроллера с приложенными графическими изображениями элементов управления в более подробном виде.

Закладки Устройств

В верхней части Области Выбора есть ряд “Закладок Устройств”. Одна закладка с лева присутствует постоянно и называется “Программные Настройки”. Это будет обсуждаться позже. Кроме этой закладки, в этом ряду будут также вкладки каждого из контролёров, которые вы добавите в карту.

Выбор контроллера

Вы можете выбрать контроллер, который вы будете программировать, выбрав соответствующую вкладку. Область Выбора при этом будет изменяться, показывая графическое изображение выбранного контроллера.

Выбор элемента Управления

Как только контроллер выбран, вы должны выбрать тот определенный элемент управления на нём, который вы собираетесь запрограммировать. При перемещении мыши над картинкой контроллера, вы увидите небольшое текстовое поле - “Индикатор Элемента Управления”, всплывающий, при прохождении мыши над различными элементами управления, как это показано для “Кнопки 2” на изображении. Чтобы выбрать нужный элемент управления, кликните на нём левой кнопкой мыши, пока индикатор этого элемента управления видим. При выборе элемента управления, название в правой верхней части **Диалоговой Области** изменится и будет отображать тот элемент, который вы выбрали. Есть также небольшой желтый курсор, похожий на курсор мыши, который указывает на выбранный элемент управления на изображении.

Все изображения Области Выбора имеют “Общий Вид” в центре, который показывает полностью всё устройство. В большинстве случаев, также бывает один или более “Отдельных Видов”, которые показывают другие, отдельные области устройства. Как правило, в Диспетчере Общий Вид служит для выбора осей для программирования, а Отдельные Видов для выбора кнопок, хаток, или переключателей обзора.

Многофункциональные элементы управления

Некоторые элементы управления не могут быть выбраны полностью с помощью только Области Выбора, поскольку они сочетают в себе несколько функций и все их сложно отобразить на изображении в Области Выбора. Это относится, прежде всего к осям X/Y, хаткам и переключателям обзора. В таких случаях, вы на самом деле выбираете весь элемент управления, т. е. оси “X/Y” или переключатель обзора “POV” или хатку “Hat 1”. При этом, чтобы позволить вам выбрать какую-то определённую ось или позицию этого элемента, в правом верхнем углу диалоговой области появляется специальный блок.

Выбор по Активации

Существует еще один способ выбора элементов управления для программирования, который называется “Выбор по Активации”. Если устройство в настоящее время работает, вы можете просто нажать ту кнопку или переместить ту ось, с которой собираетесь работать и ГИП перейдёт к этому элементу и устройству в вашей карте.

Эта функция не доступна для устройства “Trackball Pro”, так как оно обычно рассматривается как мышь во время работы ГИП и может мешать нормальной работе с Диспетчером.

Закладка Программных Настроек

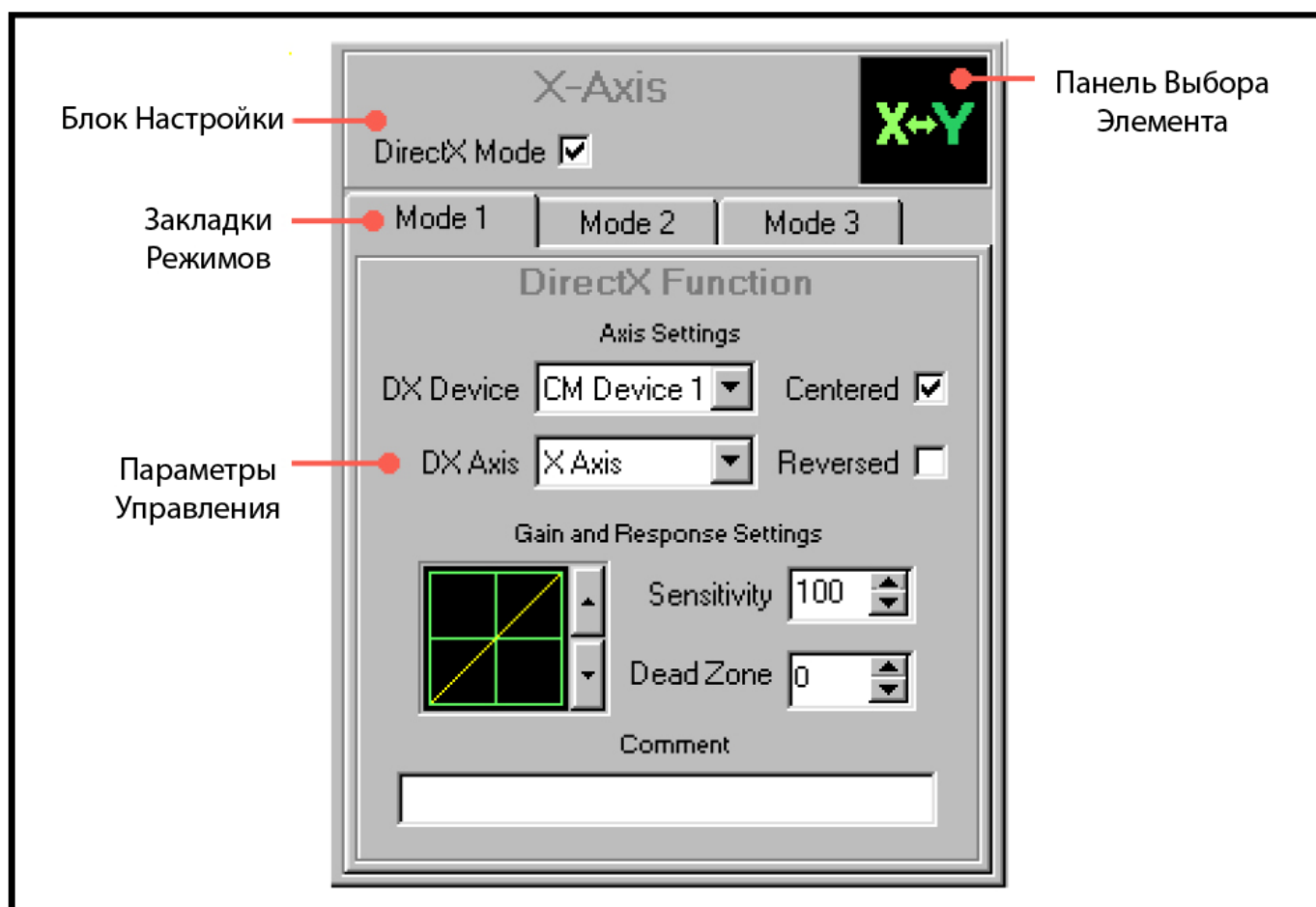
В Области Выбора дополнительно существует вкладка озаглавленная как “Program Settings” (Настройки программы). Активация этой вкладки не перейдёт к какому-либо конкретному устройству. Это вызывает в Диалоговой Области список параметров, которые влияют на всю программу. Это включают в себя такие вещи, как скорость повторения символов, чувствительность мыши и т. д. См. раздел **«Программные Настройки»** для более полного объяснения функций, доступных на этой вкладке.

Есть также флажки для включения режима «Выбора по Активации», для указания того, что переключатель обзора на «Pro Throttle» должен быть повернут на 90 градусов, и для указания «файла команд», если он имеется.

Диалоговая Область

Диалоговая Область, это то, где вы указываете тому элементу управления, который выбрали в Области Выбора, что и как делать в карте. Существует также специальная Диалоговая Область, которая используется при выборе вкладки «Программные Настройки» в Области Выбора. В ней содержатся настройки, которые относятся к карте в целом.

В зависимости от типа элемента управления, вы можете назначать их в качестве кнопки или оси, одного из Устройств Диспетчера, или вы можете запрограммировать его для ввода символов или для управления мышью. Есть несколько вариантов окон Диалоговой Области, в зависимости от типа устройства и функций, которые вы хотите программировать, но все они имеют схожую подкладку, которая выглядит примерно так:



Есть четыре основных типа Диалоговой Панели, в зависимости от типа управления которое вы пытаетесь запрограммировать. В рамках каждого типа, есть подтипы в зависимости от того, как программируется управление устройством, в режиме DirectX или ввода символов.

Вот эти четыре основных типа:

Диалоги Осей

Диалоги Кнопок

Диалоги управления обзором (POV)

И Диалог Программных Настроек

В этом разделе рассматриваются те пункты, которые являются общими для всех типов Диалогов. Для получения конкретной информации Вам следует сначала прочитать данный раздел, чтобы получить общее представление о том, как работает Диалоговая Область и лишь потом переходить к Диалогам конкретных видов управления.

Блок Настройки

В верхней части Диалоговой Области есть поле, определяемое как “Блок Настройки”, которое проходит по всей ширине панели. Оно показывает название выбранного в данный момент элемента управления и может содержать один или два флажка в зависимости от типа выбранного элемента управления.

Панель Выбора Элемента

В правой части Блока Настройки есть небольшое окно, которое называется “Панель Выбора Элемента”. Это окно активируется при выборе X/Y осей, переключателей обзора и хаток и используется для выбора конкретной оси или позиции хатки программируемой в настоящее время, если выбранный элемент управления сочетает в себе несколько компонентов.

Закладки Режимов

Под Блоком Настройки находятся “Закладки Режимов”. Карта Диспетчера может работать в четыре различных режимах, если требуется увеличить количество функций, которые могут быть запрограммированы на любой конкретный элемент управления. “Закладки Режимов” позволяют выбрать, к какому из четырех возможных режимов относится программируемая функция. См. раздел «Режимы Карты» для получения дополнительной информации об использовании Режимов.

Параметры Управления

Под вкладками Режимов – область “Параметров Управления”. Здесь вы определяете, какие действия будет выполнять выбранный элемент. Это будут в основном оси джойстика или идентификаторы кнопок джойстика, если элемент управления должен быть назначен Устройству Диспетчера. Или это будут текстовые поля того или иного рода, если элемент управления используется для ввода символов.

Блок Комментариев

Все диалоги Комментариев находятся в самом низу. Они могут использоваться для описание назначенной функции если потребуется.

Типы Диалогов

Существуют три основные группы этих диалогов, одна для использования с кнопками, одна для использования с осями, и одна для использования с переключателями обзора. Каждая из них будет подробно рассмотрена в следующих разделах.

Режим DirectX против Программного режима

Большинство диалогов может быть запрограммирован либо в режиме DirectX либо в “Программном режиме”. Хотя некоторые детали отличаются для каждого типа диалогов, принципы, по существу, одинаковы для всех. В общем, Режим DirectX будет предпочтительнее, чем Программный режим, т.к. режим DirectX, как правило, быстрее в работе и в случае осей, дает лучшую управляемость. Программный режим будет более полезным в старых симуляторах, которые не обеспечивают достаточную поддержку кнопок и управляются в основном с помощью клавиатуры. Это Также полезно, когда нужно ввести несколько команд, или когда сама команда есть текстовая строка, или что-то подобное, например, ввод сообщений для других игроков.

Режим DirectX

В “режим DirectX”, элемент управления назначается воздействующим на элемент управления одного из Устройств Диспетчера, которые появляются в апплете игровых устройств Windows. Это работает по принципу Распределительной Коробки, позволяет подключать любую кнопку или ось на любом из ваших устройств на кнопки или оси устройств, которые будет видеть Windows.

При назначении элемента управления, вы обычно устанавливаете в Диалоговой Области параметры “Устройство DirectX” и “Тип управления DirectX”. Устройство DirectX указано как “Control Manager Device (Устройство Диспетчера) + идентификатор), то есть “CM Device 1”, “CM Device 2” и т.д. Диспетчер создает отдельные Устройства Диспетчера для каждого связанного с ним Устройства с идентификатором. Исключения из этого правила могут возникать, если вы “пропустили” идентификатор. Например, если вы назначили только “CM Device 1” и “CM Device 3”, управление будет в конечном итоге определено на “CM Device 1” и “CM Device 2”. Это происходит потому, что ничего не назначено на “CM Device 2”, а Диспетчер не может пропустить идентификатор устройства, он всегда будет в первую очередь создавать “CM Device 1” и “CM Device 2”. Это может произойти, если создана карта такого типа: «FighterStick», «ProThrottle», и «ProPedals», но ProThrottle впоследствии удален.

Когда вы впервые добавляете устройства в карту, по умолчанию используется режим DirectX и управление назначается в Прямом Режиме, т.е. управление кнопкой Button 1 с первого добавленного в карту устройства (CM Device 1), будет в конечном итоге назначено на кнопку 1 на контроллере “Устройство Диспетчер 1”. Простое добавление всех устройств в карту и с их последующей загрузкой даст вам практически всё то же самое, что и в Прямом Режиме, основное отличие будет в том, что имена устройств, которые будут отображаться в списке игровых устройств Windows, будут «Устройства Диспетчера, а не фактические имена устройств (FighterStick, ProThrottle, Yoke LE и т.д.).

Программный Режим

В “Программном Режиме”, оси или кнопки запрограммированы так, чтобы эмулировать ввод символов, как вы делали бы это с помощью клавиатуры. Когда элемент управления задан в Программном Режиме, он не имеет взаимодействия с любым Устройством Диспетчера в апплете Игровых Устройств Windows. Оси или Кнопки не могут быть настроены выполнять одновременно оба действия.

Основные методы программирования Карт

В этом разделе рассматриваются основные методы создания новых Карт Диспетчера и раскрываются некоторые вещи, которые необходимо учитывать при создании карт.

Типы Карт

Прежде чем приступить к новой карте, вы должны решить, какой тип карты вам необходим. Карты обычно подпадают под одну из двух категорий, “прямую” или “Комбинированную”. Тип, который будет предпочтительнее использовать, как правило, диктуется самой игрой, для которой предназначена карта.

“Прямая” Карта.

В “Прямой” карте, каждое из устройств будет иметь соответствующее ему Устройство Диспетчера в апплете игровых устройств Windows. Устройство Диспетчер будет иметь те же оси и кнопки, что и фактическое устройство и взаимодействие между ними тоже будет один к одному. Этот тип карты подходит для игр, которые способны распознавать более одного устройства. Большинство последних игр имеют эту возможность.

Если игра поддерживает использование нескольких устройств, Прямая Карта, как правило, предпочтительнее, так как позволяет максимально использовать функции в игре, делая все оси и кнопки ваших устройств «видимыми» для игры. Это снижает объёмы необходимой работы над картой, а в некоторых играх программирование карт не требуется вовсе, кнопки и оси могут быть назначены в самой игре.

Использование Прямой Карты, очень похоже на работу Диспетчера в Прямом Режиме без карты вовсе. В простейшем виде, наиболее очевидная разница лишь в том, что устройства, которые Windows будет видеть, будут называться Устройствами Диспетчера, а не их нормальными именами (FighterStick, Yoke LE и т.д.).

Однако в этом есть и преимущества. Во-первых, это даст вам возможность определять, какое из Устройств Диспетчера будет “Джойстиком № 1”, какое “Джойстиком № 2” и т.д. В Прямом же режиме, очередность будет устанавливаться самой Windows, в зависимости от того, как устройства распределились при запуске Windows, и это может создавать проблемы в некоторых играх.

Использование Прямой Карты также дает вам доступ ко всем трем программным режимам и использованию функции Шифта (Shift), таким образом, Вы можете увеличить количество назначаемых функций в игре. Например, если вы используете «FlightSim Yoke USB», вы можете назначить одну кнопку, как «Шифт», затем назначить 11-ть

оставшихся кнопок штурвала первым одиннадцати кнопкам Диспетчера без «шифта», а затем те же 11-ть кнопок, но с «шифтом» следующим 11-ти кнопкам Устройства Диспетчера (с 12-ой по 22-ую). Это дает вам доступ к 22-ум функциям кнопок в игре, в отличие от 12-ти доступных в Прямом Режиме.

“Комбинированная” Карта

Второй тип карты, в котором несколько устройств объединены в единое виртуальное устройство в апплете Игровые Устройства. Этот тип карты, нужен для игр, которые поддерживают только джойстик № 1, а вы хотите объединить основной джойстик с Ручкой Газа и/или педалями, чтобы получить полнофункциональную систему управления.

В этом типе карт, все назначения присваиваются на управление Устройства “CM Device 1” независимо от того, с какого устройства они поступают. Кнопки, которые не назначены устройству “CM Device 1” либо программируются на ввод символов либо вообще не задействуются. Например, если у вас есть «FighterStick», «ProThrottle», и «ProPedals», вы можете объединить их в “Устройство Диспетчера 1”, которое будет включать в себя оси X и Y от джойстика, ось Z (газ) от ручки «ProThrottle», и ось R (пуль направления) от педалей «ProPedals», в результате чего основной контроллер получит 4 оси, которые будут работать практически с любой игрой.

Комбинированные карты наиболее легко создаются при помощи Мастера создания Карт, или устройства можно объединить вручную используя Мастер Устройств. Конечно можно создавать комбинированные карты вручную, просто назначая элементы управления на устройствах, но использование одного из Мастеров, сделает наиболее распространенные назначения за вас. Это также переведёт множество неиспользуемых осей и кнопок в Программный режим (не программируя их на какие-либо действия). Это избавит от многих операций с переключениями, когда нужно будет задать функции, которые не могут быть подключены непосредственно.

Сценарии в Диспетчере

Как отмечалось ранее, Диспетчер включает в себя систему поддержки сценариев (CMS), благодаря чему продвинутые пользователи могут создавать сложные функции, которые не могут быть получены средствами только ГИП. CMS требует некоторых основных пониманий концепций программирования и хорошее понимание того, как функционирует сам Диспетчер. Если вы только начинаете знакомство с созданием карт в Диспетчере, рекомендуется пропустить раздел сценариев, пока у вас не сложится хорошее понимание того, как работает вся система. В большинстве случаев Вы сможете добиться того, чего хотите без привлечения CMS.

Если вы решите, что вам нужно использовать CMS, обязательно прочитайте раздел о CMS сценариях, для получения полного описания доступных функций.

Создание Основной Карты

Как только вы решили, какой тип карты необходимо создать и нужно ли вам использовать CMS, Вы готовы приступить к процессу создания карты. Первым шагом является создание основной карты. До сих пор, самым простым способом сделать это является использование **Мастера Карт**. Это проведет вас через процесс создания карты

потребовав от вас лишь несколько кликов мыши и в результате у вас будет основная рабочая карта, с которой будет удобно начать.

Вы также можете создать карту вручную неоднократно используя кнопку «[Добавить](#)», чтобы добавить каждое из ваших устройств в карту. Если вы хотите создать комбинированную карту, вы можете использовать Мастер Устройств, как только устройства будут добавлены, чтобы сделать основные назначения автоматически. Опять же, в результате у вас будет Основная рабочая карта с которой можно начать.

Назначение Функций

После создания основной карты, всё что будут выполнять элементы управления, будет ограничиваться только вашей фантазией. Независимо от того, создана прямая или комбинированная карта, любая из осей и кнопок устройства может быть назначена или переназначена в соответствии с вашими пожеланиями. В результате первоначального создания, ничего не будет потеряно. Если вы использовали один из мастеров для создания карты и не заботились о назначениях сделанных по умолчанию, вы можете вручную поменять их, назначить их на ввод символов, даже переместить их на другие Устройства Диспетчера, если это необходимо. Всё полностью зависит от вас. Просто выберите Устройство и конкретный элемент управления не нём в Области Выбора, а затем сделайте все необходимые вам назначения в Диалоговой Области.

Добавление Устройств

Должно быть отредактировано управление по крайней мере одного устройство в карте, чтобы ГИП стал полностью активен, так что это первое, что вам нужно сделать, чтобы создать карту.

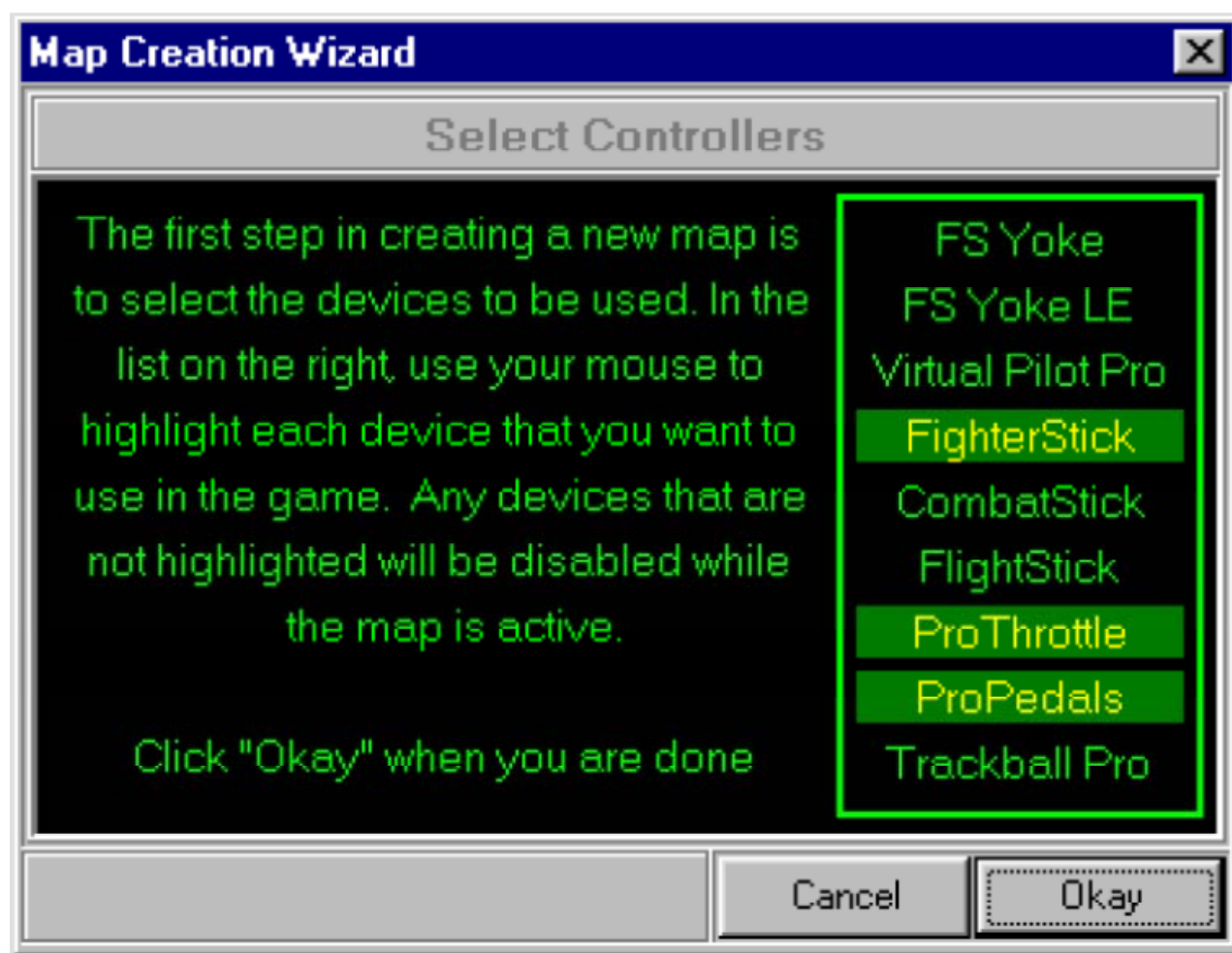
Существуют два основных способа добавления устройств в карту. Первый: используя [Мастера Карт](#). Это проведёт вас шаг за шагом через основной процесс создания карты и особенно рекомендуется для начинающих пользователей.

Второй способ требует немного больше работы, но дает несколько большую гибкость. Вы должны [Добавить устройства](#) по одному один за другим. Это также будет полезно, если у вас уже есть готовая карта и вы хотите расширить её, добавив еще одно устройство. При использовании Мастера Карт, вы всегда начинаете с пустой карты.

Использование Мастера Карт

Использование Мастера Карт это самый простой способ, чтобы начать создавать новую карту. Он проведет вас шаг за шагом через процесс создания и по пути объяснит различные опции. Если карта видима в данный момент в ГИП и она имела изменения, то лучше сохранить её перед запуском Мастера. См. раздел [Мастер Карт](#) для получения информации о том, как использовать Мастер Карт.

Когда запустится мастер, вы увидите экран, который будет выглядеть примерно так:

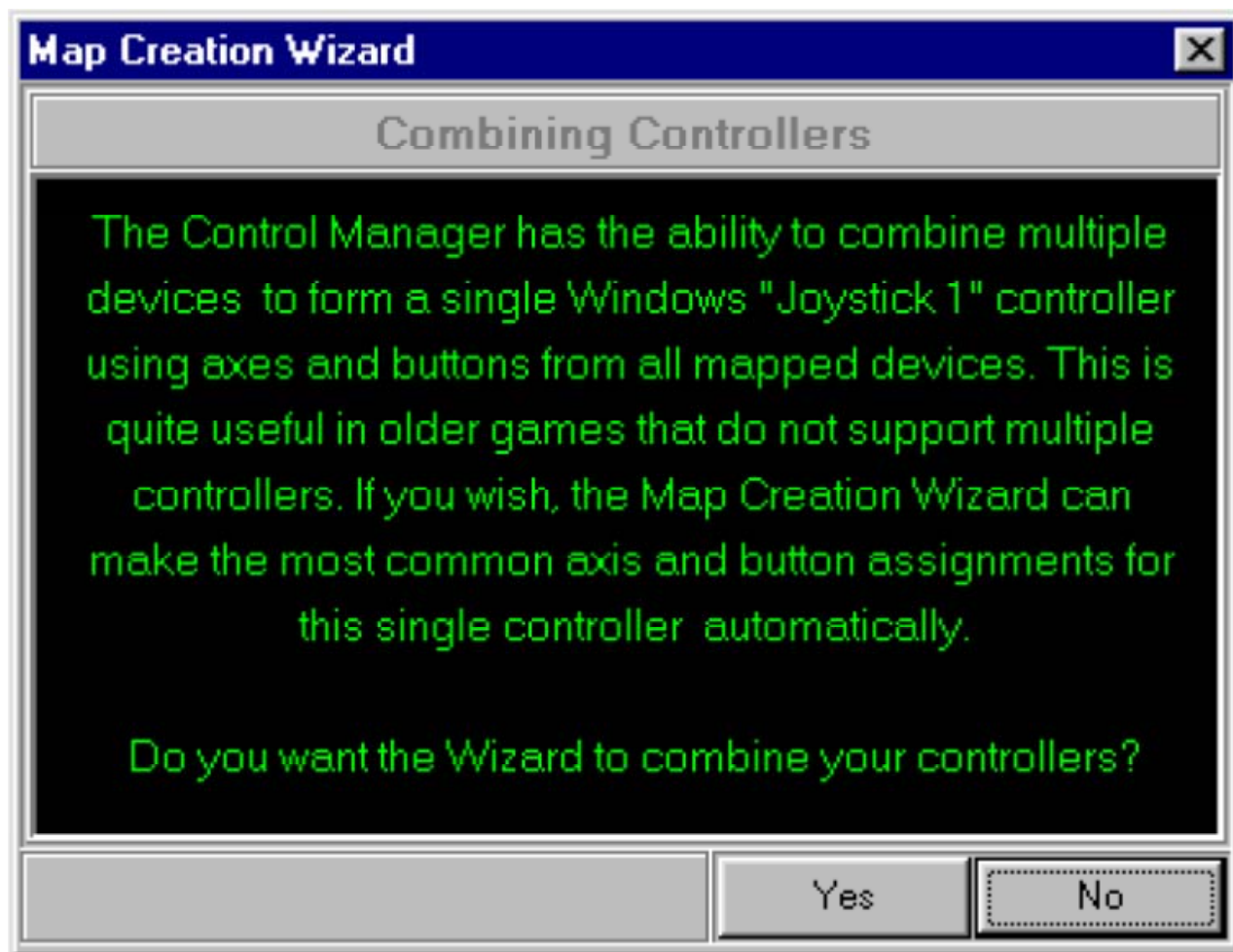


Перевод содержания информационного окна: Первым шагом в создании карты, нужно выбрать устройства которые будут использованы. Используя мышь в списке справа, выделите те устройства, которые вы хотите использовать в игре. Все НЕвыделенные устройства будут отключены при активной карте. Щелкните «Окау» по завершению.

На этом экране, вы должны выделить устройства, которые вы хотите включить в карту, щелкая по ним с помощью мыши. Если вы не выделите здесь устройство, то оно не будет активным, при работающей карте. Это означает, что вы должны добавлять все ваши устройства, даже если вы не планируете назначить им функцию. Если Вы используете «FighterStick», «ProThrottle», и «ProPedals», например, но хотите

запрограммировать ввод символа только на кнопку FighterStick, вам все равно нужно выбрать все три устройства. Любое устройство, которое не будет выбрано, будет “выключено”, при активации карты. После того, как все устройства, которые вы хотите включить были выделены, нажмите кнопку “Okay” (Готово).

Следующий экран, который Вы увидите, будет выглядеть следующим образом:



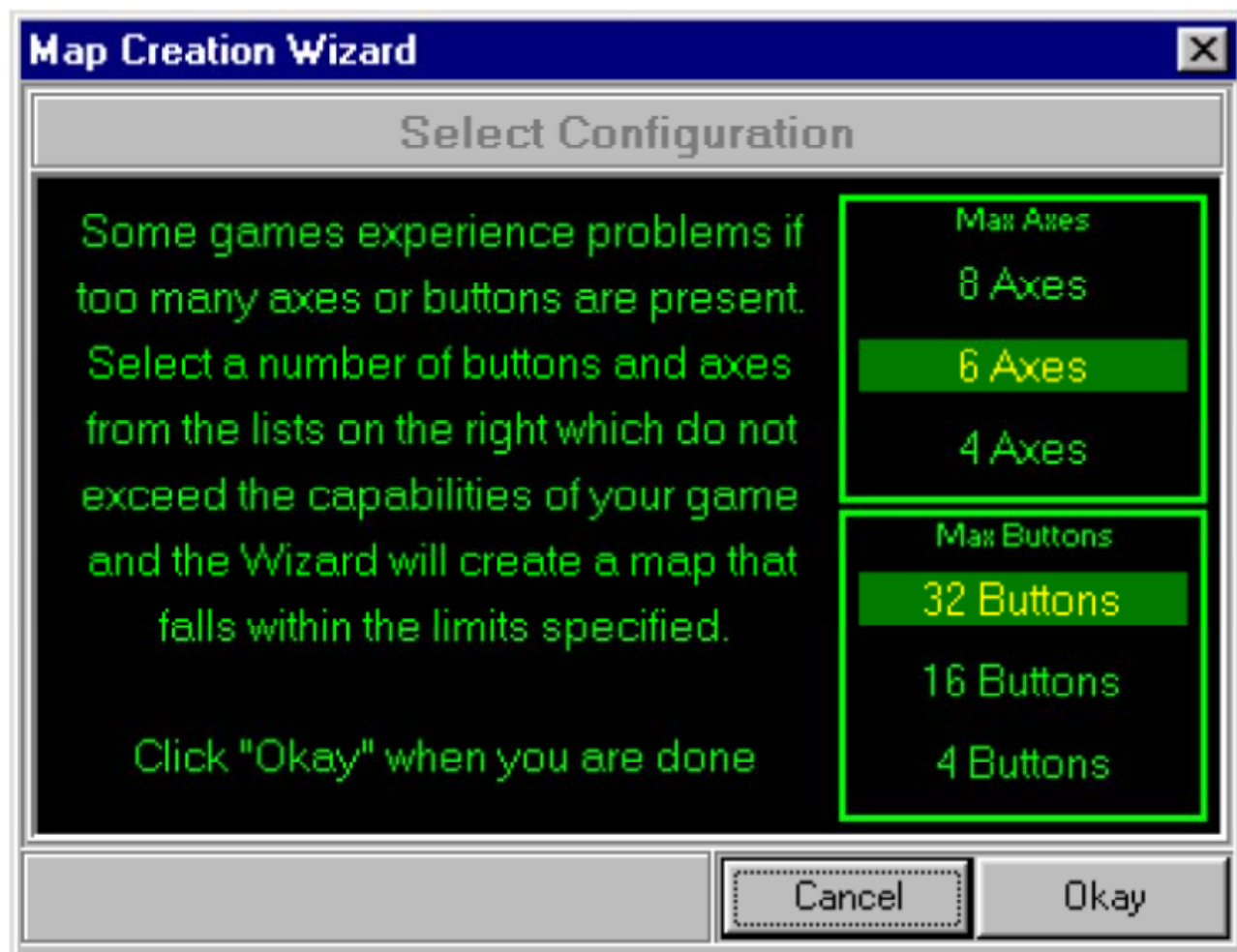
Перевод содержания информационного окна: Диспетчер имеет возможность объединить несколько устройств в единое устройство Windows “Joystick 1” которое будет использовать оси и кнопки всех подключённых устройств. Это довольно удобно использовать в старых играх, которые не поддерживают несколько устройств. Если вы желаете, Мастер создания Карт, может автоматически произвести основные назначения осей и кнопок на этом едином устройстве.
Вы хотите, чтобы Мастер объединил ваши устройства?

Это экран, где нужно указать, хотите ли вы чтобы Мастер создал “Прямую Карту” или “Комбинированную”.

Прямая Карта та, в которой, изначально, каждое устройство, которое вы добавляете, создаст для Windows особое устройство, которое будет выглядеть идентично самому устройству которое его представляет, за исключением того, что он будет иметь имя как “Устройство Диспетчера 1”. Разница между этим и просто запуском Диспетчера в Прямом режиме, в том, что вы сможете программировать кнопки и оси, на ввод символов или управление мышью. В Прямом Режиме, это не представляется возможным.

Комбинированная карта та, в которой элементы управления с нескольких устройств объединяются в одно устройство Диспетчера, которое будет видно Windows. Это полезно в играх, которые распознают только одно устройство, что позволяет, например, объединить «FighterStick», «ProThrottle», и «ProPedals» в одно устройство с 4-мя осями.

Нажмите кнопку “Да” или “Нет” в зависимости от того, что вам нужно. Если вы нажмете кнопку “Да”, вы увидите экран, который выглядит так:

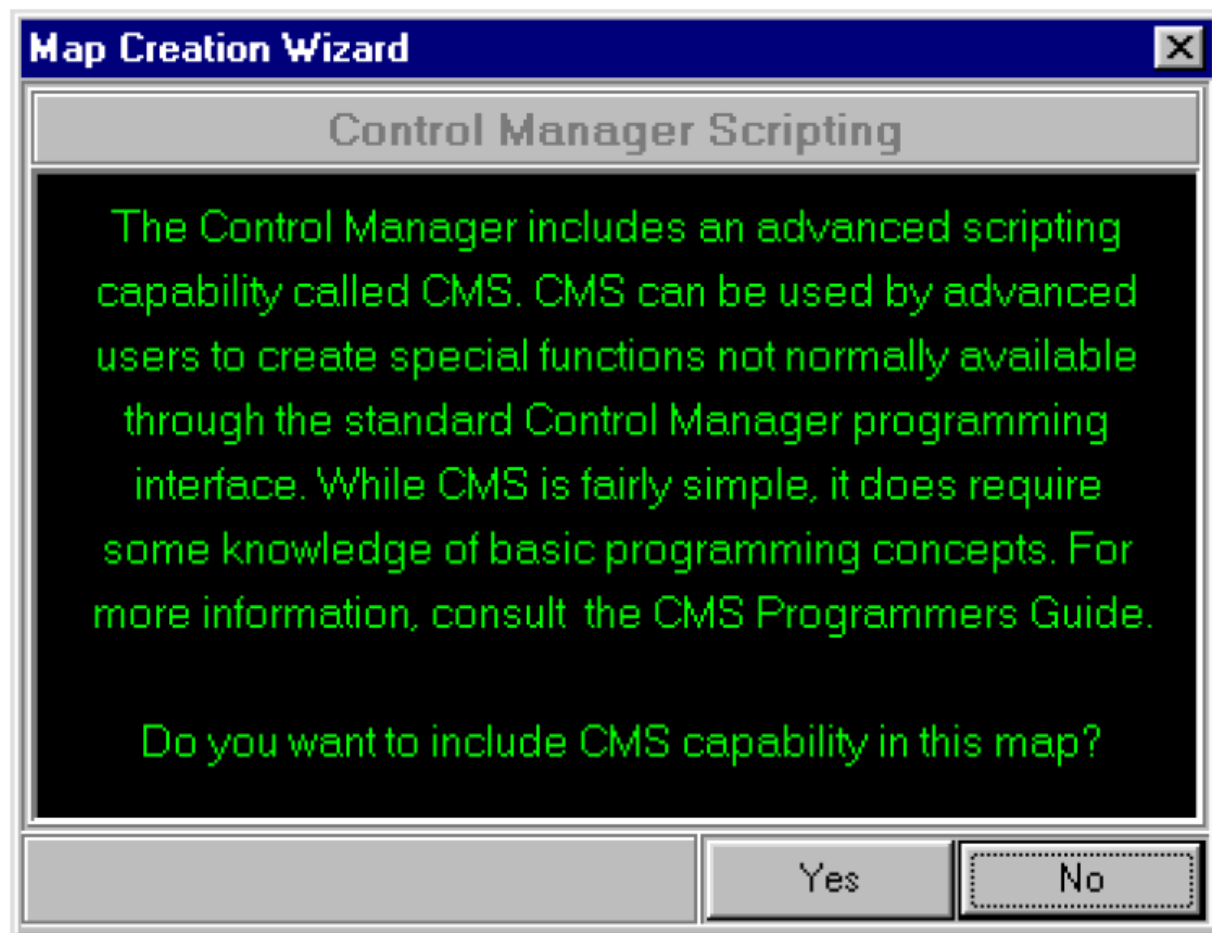


Перевод содержания информационного окна: В некоторых играх бывают проблемы со слишком большим количеством осей и кнопок. В списке справа выберите количество осей и кнопок, которые не превышают требований вашей игры и Мастер создаст карту, которая будет соответствовать указанным ограничениям. По завершению нажмите «Okay».

В этом списке перечислены различные конфигурации, которые Мастер Карты может использовать в комбинированных картах. Определяется максимальное количество осей и кнопок, которые будет иметь комбинированное устройство. Мастер Карт может создать любое количество осей и кнопок, из тех, что указаны в списке, но он не сможет создать больше осей или кнопок, чем указано в списке. Если у вас, например, есть только штурвал и комплект педалей, то у карты нет возможности создать 32-х кнопочное устройство, но установка лимита на 32-кнопки, позволяет назначить все доступные кнопки управления Устройство Диспетчера. Это не мешает вам указать больше, чем доступно, как правило, выбирается конфигурация 6-осей/32-кнопки.

Другие конфигурации в основном для старых игр и симуляторов, которые не могут правильно взаимодействовать с устройствами, которые представлены множеством осей и кнопок. Если у вас возникли проблемы с использованием конфигурации 6/32 в конкретной игре, понижение числа осей и кнопок, может прояснить проблему. Наиболее наглядный пример это, вероятно, «MechWarrior». Она не запустится с 32-кнопочным джойстиком. Используйте 16-кнопочную конфигурацию.

После того как вы сделали выбор, нажмите кнопку «Окай» (Готово). Или, если вы решили не объединять устройства в карте на предыдущем шаге, вы получите экран, который выглядит следующим образом:



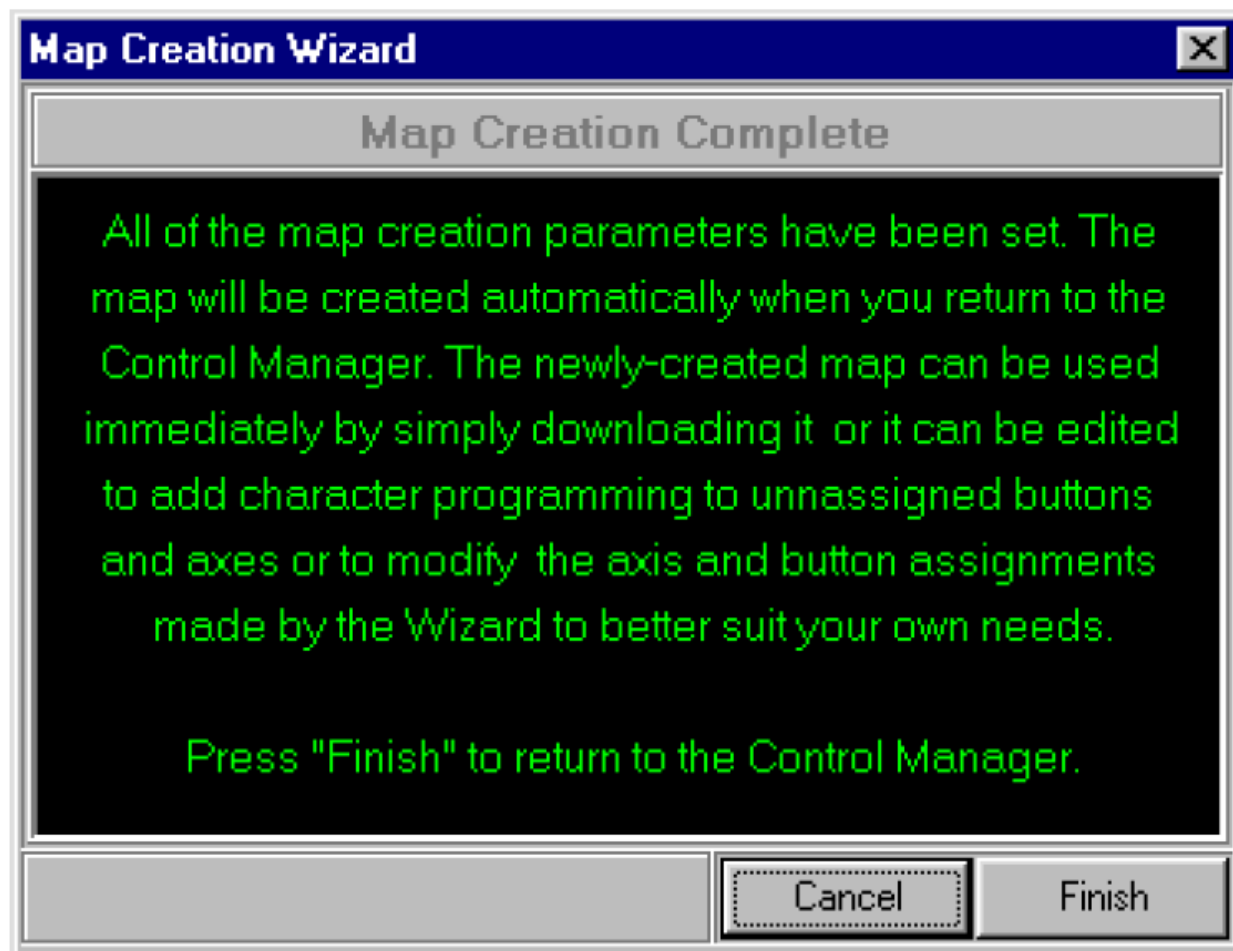
Перевод содержания информационного окна: Диспетчер включает в себя возможность углубленных сценариев, которая называется CMS. CMS может использоваться опытными пользователями для создания специальных функций, которые недоступны стандартными средствами интерфейса Диспетчера. Несмотря на то, что CMS предельно прост, он потребует некоторых знаний основных концепций программирования. Для большей информации обратитесь к Руководству по Программированию в CMS.

Хотите ли вы подключить CMS в эту карту?

CMS Сценарии это основанная на текстовом принципе система, которая позволяет опытным пользователям создавать комплексы функций, которые не могут быть реализованы с использованием обычных функций графического интерфейса пользователя. Большинство обычных функций можно назначить используя только ГИП, без использования сценариев CMS. Если вы новичок в Диспетчере, то, вероятно, лучше

просто нажать «**No**» (Нет) в этой точке. Вы сможете подключить CMS в карту позже если возникнет такая необходимость. См. [Руководство по Программированию в CMS](#) для получения дополнительной информации о CMS и том, как им пользоваться.

Любой выбор приведет к такому экрану:



Перевод содержания информационного окна: Все параметры для создания карты заданы. Карта будет создана автоматически, когда вы вернётесь в Диспетчер Устройств. Только что созданная карта может быть применена сразу же простой загрузкой её или она может быть отредактирована для символьного программирования неиспользуемых кнопок и осей или для изменения назначений осей и кнопок сделанных Мастером, в соответствии с вашими пожеланиями.

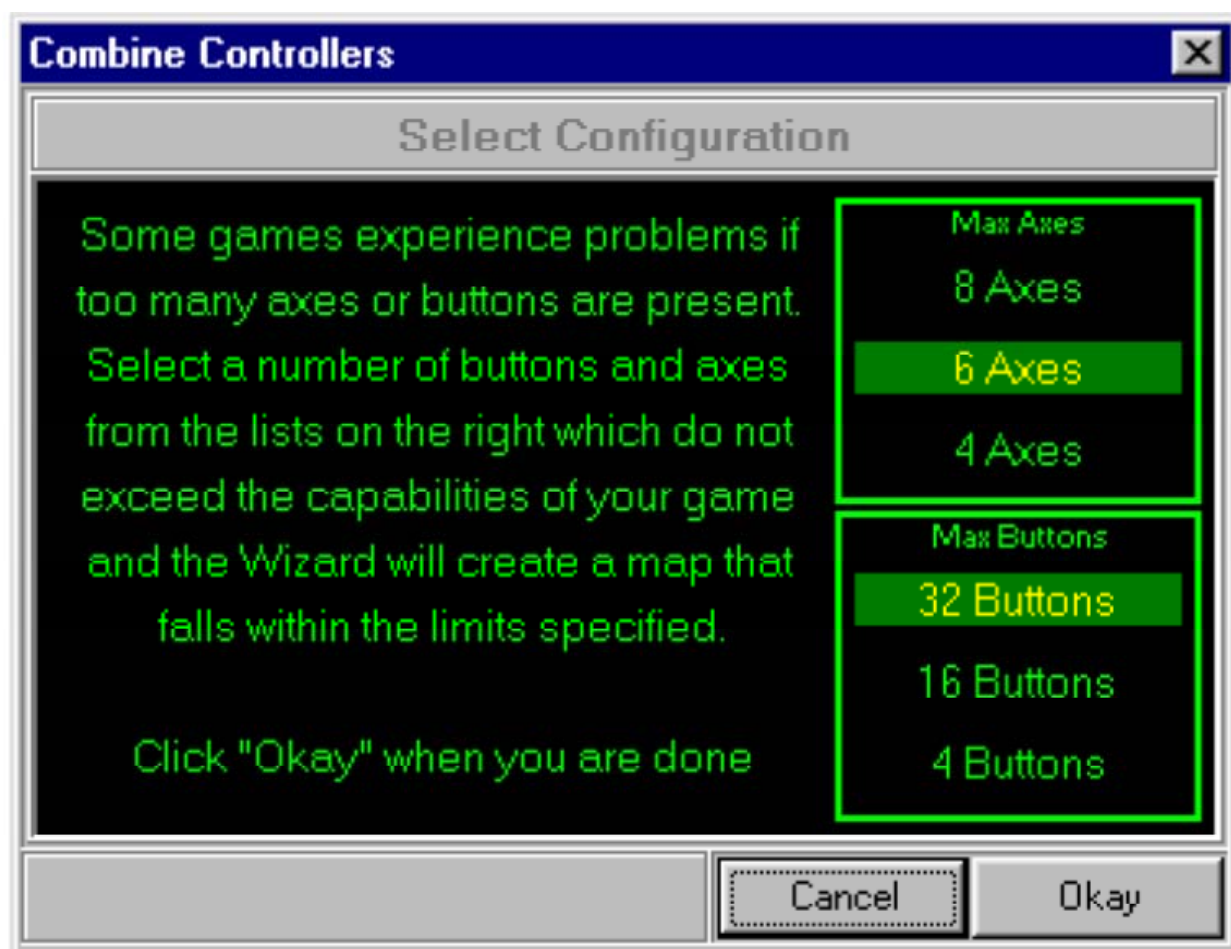
Нажмите "Finish" (Закончить), чтобы вернуться в Диспетчер Устройств.

Вы можете нажать "Окай" и Мастер Карты создаст основную карту, основанную на ваших предыдущих ответах. При нажатии кнопки "Cancel" (Отмена), Мастер Карт закроет программу без создания карты.

Стоит отметить, что в картах, созданных Мастером Карт нет ничего особенного. Мастер просто помогает автоматически задать основные назначения в карте. Вы можете изменить любое из назначений вручную, по завершению Мастера Карт, назначить кнопки, поменять оси, и т.д. В результате работы Мастера, сделаны наиболее распространенные назначения для основных осей и кнопок, но всё это всегда доступно для редактирования в будущем.

Использование Мастера Устройств

Многие игры распознавать только одно устройство, поэтому использование нескольких устройств требует “комбинирования” в единое устройство, которое будет видно Windows. Мастер Устройств автоматизирует эту задачу. При запуске мастера появится диалог, который выглядит так:



Перевод содержания информационного окна: В некоторых играх бывают проблемы со слишком большим количеством осей и кнопок. В списке справа выберите количество осей и кнопок, которые не превышают требований вашей игры и Мастер создаст карту, которая будет соответствовать указанным ограничениям. По завершению нажмите «Okay».

Этот диалог позволяет установить верхний предел количества осей и кнопок который Мастер будет использовать при создании комбинированного устройства. Причиной этого является то, что некоторые игры не могут нормально работать, если обнаруживают устройство, которое имеет слишком много кнопок или осей. MechWarrior 4, должно быть, наиболее распространенный пример игры, которая сталкивается с этой проблемой. Она «вылетает» на рабочий стол, если сталкивается с устройством с 32-мя кнопками.

С правой стороны диалога два блока, один для установки максимального количества осей и другой для установки максимального количества кнопок. Чтобы установить эти

лимиты, просто выделите один из пунктов в каждом из блоков. Имейте в виду, что это предельные значения. Во многих случаях Мастер не можете создать устройство с тем количеством осей или кнопок, которое вы укажете. Например, определить 8 осей, когда есть только «Flightstick» и «ProPedals». Тем не менее, это не мешает указать значения больше, чем может быть создано, так что, зачастую, лучше начать с выбора 8-ми осей и 32-ух кнопок, а затем уменьшить эти лимиты только в том случае, если в игре будут проблемы.

Как только выбор сделан, нажмите кнопку «Окай» (OK) и мастер объединит в себе устройства для формирования одного виртуального устройства. Если составляющие устройства будут иметь больше доступных осей или кнопок, чем определено в ограничениях, которые вы установили, то в Программном Режиме будут созданы дополнительные оси и кнопки для ввода символов, но символы не будут назначены. Стоит отметить, что нет ничего особенного в карте, созданной таким образом. Мастер просто проделает большую часть работы по назначению наиболее используемых осей и кнопок на Устройство «CM Device 1» и очистке оставшихся осей и кнопок, но вы можете отредактировать карту в соответствии со своими предпочтениями.

Макросы Программирования символов

Одна из основных функций Диспетчера, позволяет вам эмулировать нажатия клавиш клавиатуры, нажатием кнопки или установкой хатки в определённую позицию. Это позволяет выполнять несколько операций, таких как активация таких функций в игре, которые не могут быть вызваны с использованием прямого ввода кнопкой джойстика, программирование одной кнопки на выполнение нескольких операций, или на ввод «упакованных» сообщений другим игрокам в многопользовательской среде.

Общая информация

Все макросы вводятся в основном одним способом. Диспетчер предоставляет поля для их ввода. Вы устанавливаете курсор в окно и вводите символы, которые вам нужно ввести. Вы можете также использовать [Запись Ввода с Клавиатуры](#) в качестве помощи для быстрого ввода макросов.

В общем, есть четыре таких бокса для каждой кнопки или позиции хатки. Они появляются при снятии отметки DirectX в верхней части диалоговой области. Два бокса определяют макросы ввода кнопкой при свободном (отжатом) шифте (нормальное действие), а два бокса определяют ввод кнопкой с зажатым шифтом (шифтовое действие). В каждой из двух групп, по два бокса. Один определяет макрос, который будет выполнен, когда выбранная кнопка нажата, другой определяет макрос, когда выбранная кнопка отпущена.

По большей части, нажатия клавиш в макросе вводятся просто с помощью буквы/символа, который Вы хотите ввести, например, “a”. В некоторых случаях, клавиши бывают “именными”. Это такие клавиши, которые имеют имена не из единой буквы, такие как “LSHF” для левого шифта, “RALT” для правой кнопки “Alt” и т.д.

Макросы делятся на три категории. Их называют “Обычные Макросы”, “Макросы задержки”, и “Макросы Клавиш”. Какие вы будете использовать, зависит от того, что конкретно вам нужно выполнить. В этом разделе будет говориться о каждом из этих типов. Обратите внимание, что примеры макросов будут заключены в двойные кавычки, например, “XYZ”. Они будут отделять макросы от текста и не должны вводиться в реальном макросе, если, конечно, вы действительно не захотите ввести двойные кавычки.

Обычные Макросы

Обычные Макросы являются наиболее распространенными. Они используются для ввода одного или повторяющихся символов, или для отправки простых строк символов, таких как “abc”. Когда вы вводите макрос в его редакционное поле, вы должны отделять символы пробелами так, чтобы Диспетчер мог отличать буквы, именные клавиши и ключевые слова. Например, вы должны ввести “L S H F”, если вы хотели ввести четыре отдельных буквы “LSHF” одной определенной кнопкой. Если вы введёте “LSHF” в окне редактирования, вы получите функцию левой клавиши шифт.

Верхний и нижний регистр

Диспетчер понимает разницу между прописными и заглавными буквами и будет задействовать соответствующую клавишу шифт, когда это необходимо. Например, предположим, что вы вводите “A” как макрос. Когда карта активируется, Диспетчер на самом деле нажмёт левую клавишу шифт, нажмёт клавишу “a”, отпустит клавишу “a” и отпустит клавишу шифт. Вы должны иметь это в виду при программировании, так как в инструкциях по эксплуатации многих игр, назначения будут показаны в нижнем регистре, и во многих случаях ввод в верхнем регистре не приведет к адекватным действиям. Например, в игре может быть указано “закрылки: F”, когда он на самом деле закрылки: “f”. Как правило, используют нижний регистр, если в руководстве специально не указано иначе, как например “Закрылки: Shift-F”. Пользуясь этим правилом вы будете чаще правы, чем неправы.

Именные клавиши

Как упоминалось выше, каждый символ в обычных макросах должен быть разделен пробелом. Чтобы инициировать ввод самого пробела, вы должны использовать именное обозначение этой клавиши «SPC». Например, «Привет SPC друг!» даст нам “Привет друг!” при нажатии кнопки. Существует несколько таких проименованных клавиш. Полный список именных клавиш вы найдёте в разделе [Коды Клавиш Диспетчера Устройств](#).

Модальные клавиши

Хотя вы можете инициировать ввод символа с зажатым «шифтом» (Shift), просто задав его в верхнем регистре, есть также два других типа клавиш, которые похожи на Шифт тем, что они позволяют выполнять действие, в то время как сами они зажаты. Это «Alt» и «Ctrl». Они, наряду с клавишей Шифт называются “Модальными Клавишами”, потому что они не вводятся отдельно, а лишь удерживаются, в то время как нажимаются связанные с ними клавиши. Они определены тремя модальными модификаторами “SHF”, “ALT” и “CTL” и используются нажатием фактических символов. Например “CTL c” даст нам

“Контроль+С” на вводе. Модификатор «SHF» обычно не нужен, за исключением случаев с именованными клавишами, например “SHF-F4”, поскольку не бывает F4 в верхнем регистре.

Левый и правый Шифты, Альт и Контроль, могут быть также введены отдельным символом. Чтобы отличить их от модальных модификаторов, им предшествуют литеры “L” или “R”, обозначающие «левый» и «правый» соответственно. Например “LSHF” (левый Шифт), “RALT” (правый Альт) и т.д. При использовании этого способа, определённая клавиша нажимается, а затем отпускается, а не удерживается, в ожидании ввода следующей клавиши в макросе.

Повторяющиеся Клавиши

Как правило, макрос, составленный из всего одного символа (и, возможно, модального модификатора) вводится в макрос в виде повторяющегося символа. Если вы удерживаете кнопку нажатой, символ будет повторяться бесконечно. Стоит отметить, что USB система вводит повторяющиеся символы несколько иначе, чем это делали ранние клавиатуры PS2. В PS2, клавиатура действительно повторяла отправку символа, пока клавиша была нажата. В USB, символ посылается только один раз. Система сама генерирует повторения опираясь на заданную в Свойствах Клавиатуры Windows частоту повторений. Макросы, которые содержат несколько клавиш, вводят их только один раз, повторения не генерируются.

Неповторяющиеся отдельные клавиши

Во многих случаях, повторяющиеся символы могут быть надоедливой вещью. Например, если у вас есть клавиша, которая выпускает и убирает шасси, вы, вероятно, захотите выпустить их одним нажатием и убрать другим. Повторяющаяся клавиша сначала выпустит их, затем уберёт, затем снова выпустит и так далее. Чтобы обойти эту проблему, существует именной псевдо символ, который называется «нулевым». Он представлен под именем «NULL». Нулевой символ ничего не вводит на самом деле, он вообще игнорируется Диспетчером, за исключением тех случаев, когда он подсчитывается как символ, когда диспетчер определяет, следует ли повторять одну клавишу или строку неповторяющихся символов макроса. Например «а» даст нам «aaaaaaaaaaaaa» на вводе, а «NULL а» даст нам только «а».

Макросы Задержки

Макросы задержки используются там, где необходимо нажать более чем одну клавишу за раз. Это бывает полезно в определенных ситуациях, особенно там, где этого требует игра, например, «Правый Шифт - F4». Обычный модификатор SHF вызывает Левый Шифт, а «RSHF F4» активирует две клавиши по отдельности. Макрос задержки может обойти это. Для инициации макроса задержки, просто начните макрос со слова «HOLD», например, «HOLD RSHF F4», что приведет к вводу обеих клавиш одновременно.

Макросы Клавиш

Третий и вероятно самый гибкий тип макросов «Клавишный» макрос. Он в частности определяет порядок нажатий и отпускания различных клавиш, с помощью знаков + и -. Макрос клавиш начинается со слова «KEYS» (клавиши). Например «KEYS +a +b -a +c -b -c». Таким образом «а» будет нажата, «б» нажата «а» отпущена, «с» нажата «b» отпущена и «с» отпущена.

В макросах клавиш необходимо указывать каждое нажатие клавиши в правильном порядке. Можно заставить клавишу «залипнуть», например в случае «KEYS +a» клавиша «a» не будет отжата и залипнет до тех пор, пока «a» не будет нажата на клавиатуре или пока, например, другая кнопка не активирует макрос «KEYS -a». Это может быть полезным а может и нет, в зависимости от того, что вы пытаетесь сделать. Возможно вы захотите чтобы «a», «b» и «c» активировались, нажатием кнопки вниз, и деактивировались при отпускании кнопки. В этом случае вы можете назначить на нажатие кнопки макрос «KEYS +a +b +c», а на отпускание макрос «KEYS -a -b -c». [Коды Клавиш Диспетчера Устройств](#) является очень полезным подспорьем в создании такого рода макросов.

Специальные символы

В дополнение к стандартным символам, существует несколько специальных символов, которые могут быть запрограммированы. Это позволяет Диспетчеру выполнять специальные функции, управлять мышью и т. д. В этом разделе рассматриваются специальные символы и то, что они делают.

Использовать специальные символы очень просто. Примерно также как любые другие именные символы, которые вы будете использовать в программировании с отключённым DirectX, такими как «LSHF», «RCTL», и т.д. Они не вводят символы на самом деле, но иницируют их назначения. Специальные символы действуют только пока они нажаты. Они не делают ничего при отпускании. Если вы используете их в макросах клавиш, они всегда будут считаться нажатыми (+) независимо от того, каким был предыдущий символ или даже если он вообще отсутствовал. Например «+CHARDLY», «-CHARDLY» и просто «CHARDLY» в результате обеспечат одинаковое действие при использовании в макросе клавиш.

Символы функций мыши

Первые четыре специальных символа используются для выполнения функций мыши. Это символы:

Имя символа	Действие
LCLICK	Левый клик мыши
RCLICK	Правый клик мыши
MCLICK	Нажатие средней кнопки мыши
DBLCLICK	Двойной клик левой кнопкой мыши

Программирование кнопок на ввод одного из этих “символов” иницирует указанные действия мыши. Обратите внимание, что это не то же самое, что назначить кнопку DX на кнопку мыши. В этом случае, вы просто будете управлять кнопкой мыши при помощи кнопки джойстика. В случае со специальными символами, вы как бы говорите Диспетчеру вводить указанные вами клики мышью. Как только символ введен, все

происходит автоматически. Это означает, что, чтобы совершить «перетаскивание» с помощью мыши, вы должны назначить на DX кнопку функцию кнопки мыши. Не существует специального символа, который мог бы инициировать удержание кнопки мыши нажатой. LCLICK например нажмёт и сразу же отпустит левую кнопку мыши, независимо от того, как долго нажата кнопка инициировавшая ввод LCLICK.

Символ CHARDLY

Следующий особый символ, который у нас есть, это «CHARDLY». Этот символ не производит видимых действий. Единственное, что он делает, это вставляет задержку в 1 символ времени между двумя другими символами. В норме Диспетчер активирует столько символов, сколько возможно на какой-либо конкретной стадии сканирования. Аналогичным образом он деактивирует столько, сколько возможно. Как правило, это правильный алгоритм, но бывают ситуации в некоторых играх, когда это не дает ожидаемого результата, и необходима небольшая задержка, чтобы игра реагировала. Один из наиболее распространенных примеров это, когда вам нужно использовать символ с правым шифтом. Вы не можете просто использовать «SHF», потому что это левый шифт. Нормальным решением может быть что-то вроде “HOLD RSHF a”, но в некоторых играх, это не сработает, потому что “RSHF” и “a” “ вводятся точно в одно и то же время. В этом случае вам нужно иметь возможность контролировать задержки, чтобы быть уверенным, что “RSHF” включится перед “a” и что “a” выключится перед “RSHF”. В макросе клавиш этого можно реализовать функцией CHARDLY:

```
KEYS +RSHF CHARDLY +a -a CHARDLY -RSHF
```

Так RSHF будет удерживаться в течении всего ввода «a» с задержкой в один символ между временем когда был нажат RSHF и временем, когда была нажата «a», а затем будет секундная задержка между временем, когда «a» будет отпущена и временем, когда RSHF будет отпущен.

Если игра немного меньше требовательна к расстановке символов, то вы можете обойтись стандартным макросом (не макросом клавиш), который выглядит так:

```
HOLD RSHF CHARDLY a
```

Хотя это не даст такого же управления, так как “a” и RSHF также, будут отпущены в одно и то же время и в некоторых играх это может работать не правильно.

Символ DIRMODE

Второй специальный символ называется “DIRMODE”. Когда он вводится, Диспетчер переходит из Программного режима в Прямой. Это может быть полезно и само по себе, но это также может использоваться для полного осуществления функции [Быстрый Запуск](#) описанной ниже.

DIRMODE предоставляет собой удобный способ переключать Диспетчер обратно в прямой режим когда вы закончите играть и захотите закрыть карту. Делать это, как правило, желательно, т.к. если оставить Программный режим активным, то устройство может продолжить ввод символов, по случайности, если его задеть и как правило, это происходит в самый неподходящий момент. Однако нельзя допускать инициацию

DIRMODE по ошибке. Вообще можно использовать CMS сценарий для создания логического предложения, которое бы требовало сложной комбинации для инициации ввода символа DIRMODE. На «Flightstick», например, можно использовать кнопки 2 и 3, так как они обе обычно активируются большим пальцем, и трудно нажать их обе одновременно по случайности. В сценарии CMS, простое логическое предложение:

```
CMS.B1 = JS1.B2 И JS1.B3;
```

определит эту логику. Затем в ГИП программы вы назначите кнопке CMS.B1 ввод DIRMODE.

DIRMODE также может быть полезен при отладке. Иногда сценарий, который работает не правильно, делая странные вещи, например, перемещает курсор мыши в верхний левый угол, и т.д. В таких случаях, бывает трудно вернуть контроль над картой. Если вы запрограммировали кнопку на ввод DIRMODE, это, как правило, вернёт вас обратно в Прямой Режим в такой ситуации. Это конечно не гарантирует успеха, так как причины сбоя могут быть разными, но это должно помогать в большинстве случаев.

Быстрый Запуск

Начиная с Диспетчера 3-ей версии, представлена новая возможность «Быстрый Старт». Быстрый Старт позволяет Диспетчеру переходить из Прямого режима в Программный, простым нажатием кнопки 1 на первом устройстве вашей карты (устройство на левой закладке устройств рядом с закладкой «Программные Настройки»). Возможность запустить карту посредством устройства удобна во многих случаях, так как это экономит время избавляя от необходимости открывать ГИП или запускать «CMStart» и уже из них активировать карту.

Для того чтобы получить возможность Быстрого Старта, необходимо установить флажок на «ClickStart Enabled» (Быстрый Старт Задействован) в закладке «Программные Настройки». Карта должна быть загружена после установки флажка, и возможность Быстрого Запуска будет отмечена для этой отдельной карты.

Вы должны нажать кнопку, чтобы запустить карту прежде чем начать игру. Многие игры определяют при запуске, какие устройства будут использоваться. В таких случаях, Быстрый Запуск может быть не эффективным, если будет запущен после начала игры. Это запустит карту, но если игра уже увидела 3 активных устройства, а затем вы запустили карту и она скомбинирует 3 устройства в одно, игра не сможет понять этих изменений в конфигурации. В других играх, это не имеет значения, и вы можете нажать кнопку, чтобы активировать карту как до так и после запуска игры.

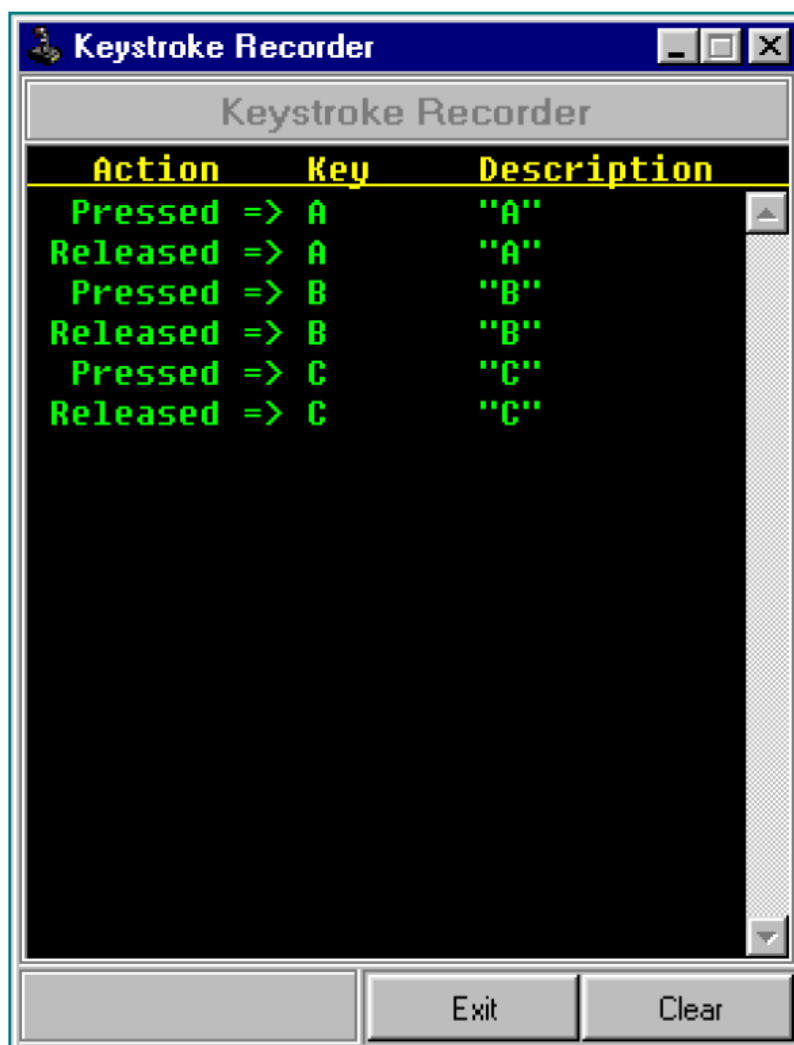
Для осуществления полной функциональности Быстрого Запуска, карта должна также быть обеспечена возможностью способа DIRMODE. С DIRMODE, вы сможете включать и выключать текущую карту без открытия графического интерфейса вовсе, за исключением необходимости редактирования карты или загрузки другой.

Заметьте, что если вы захотите запустить Диспетчер в Прямом режиме, у вас должна быть загружена карта, без Быстрого Запуска. В противном случае, Диспетчер будет

переключаться на Программный режим при первом же нажатии кнопки 1. Любая карта без поддержки Быстрого Запуска будет выполнять своё предназначение. Карта не будет работать, а лишь будет загружаться, чтобы выключить логику Быстрого Старта.

Запись Нажатия Клавиш

Запись Клавиатурного ввода может быть использована для ввода макросов символов, простым вводом клавиш и их последовательностей с клавиатуры в том порядке, который вам нужен. Это делается почти также как в утилите «Keytest» (тест клавиш). Основным отличием является то, что, когда вы закрываете запись ввода с клавиатуры, то сделанное вами анализируется, строятся правильные макросы и вводятся в поля ввода автоматически. Окно программы записи выглядит следующим образом:



Чтобы получить доступ к записи, вы должны редактировать один из макросов для кнопки или позиции хатки, которые должны будут работать в Программном Режиме (флажок DirectX снят). Щелкните правой кнопкой мыши в окне редактирования макроса, для которого требуется ввод. Внизу выпадающего меню вы увидите пункт «Record Keystrokes» (Запись нажатия клавиш). Щёлкните на нём, откроется окно программы записи клавиш.

Введите последовательности клавиш, просто набрав их на клавиатуре. После того как вы ввели комбинацию, которую хотели, нажмите кнопку «Exit» (Выход). Макрос будет создан и появится в окне редактирования, где вы щелкнули правой кнопкой мыши, чтобы начать этот процесс.

В зависимости от того, что вы наберёте, программа записи определит соответствующий тип макроса клавиш. Сначала она попытается создать “Обычный Макрос”. В противном случае, она будет пытаться использовать макрос задержки. Если и это не работает, он вернется к макросу клавиш, которые в основном представляют собой какую-либо последовательность клавиш.

При вводе символов в программу записи клавиш, порядок нажатия и отпускания клавиш будет, очевидно, влиять на тип макроса, который в конечном итоге будет применён. Обычный Макрос является наиболее эффективными с точки зрения размеров, но и для Обычных макросов, нажатия клавиш должны чередоваться в строгом порядке “нажал-отжал, нажал-отжал...”. Другими словами, если клавиша была нажата, она должна быть отжата перед нажатием следующей. Если вторая клавиша будет нажата до того как первая будет отпущена, Обычный Макрос не сможет обработать это и в конечном итоге, всё закончится созданием макроса клавиш. Не существует символа управляющего частотой ввода клавиш в программе записи клавиш. “а” нажатая и удерживаемая в течение десяти секунд, на выходе даст тот же результат, что и “а” нажатая и отпущенная немедленно, так что целесообразнее делать всё медленно, чтобы быть уверенным, что последовательность верна.

Нет ничего особенного в макросах, созданных с помощью записи клавиш. Как только макрос будет создан и появится в соответствующем поле ввода, вы можете вручную редактировать его содержимое в поле редактора, если нужно.

Обратите внимание, что вы не сможете получить доступ к записи клавиш во время работы утилиты «Keytest» (Тест Клавиш), даже если она свёрнута. Вы получите сообщение об этом, если попытаетесь сделать так, тогда вам нужно будет закрыть утилиту «Keytest» перед записью. Если утилита «Keytest» была свёрнута, просто нажмите кнопку «Keytest» на Панель Инструментов и утилита будет закрыта. Когда работает программа записи клавиш, вы не сможете получить доступ к утилите «Keytest» и любой кнопке на Панели Инструментов, так как программа записи отключает все другие функции ГИП, до тех пор, пока не будет закрыта.

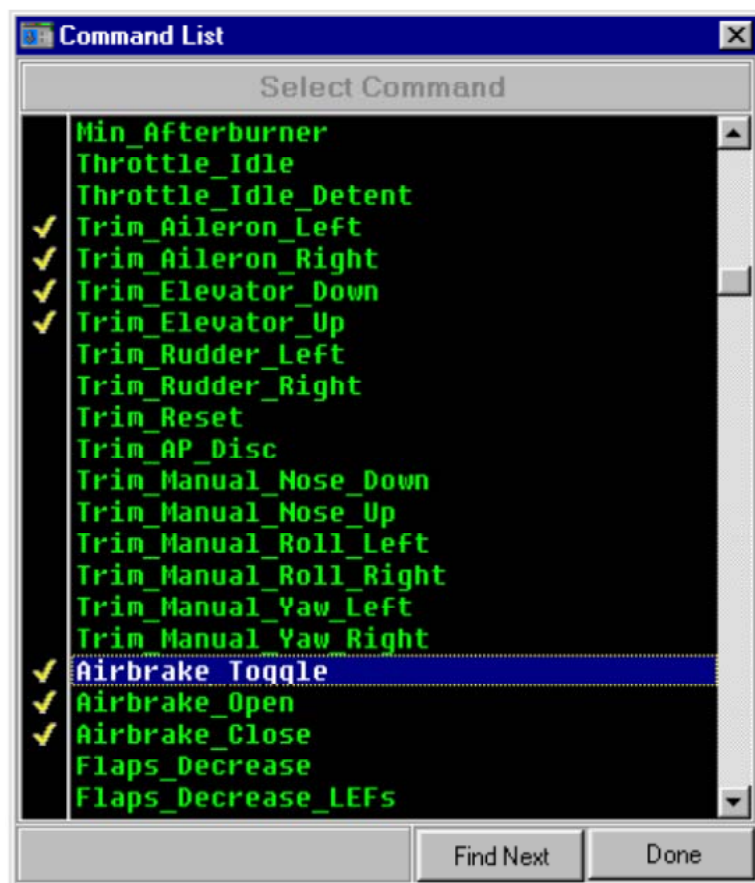
Специальные символы

При использовании записи клавиш, невозможно сгенерировать [специальные символы](#), таких как CHARDLY, LCLICK и т.п., т.к. они не имеют реальной клавиши, связанной с ними. В этих случаях, все, что вы можете сделать, это создать последовательность нажатия клавиш, без использования специальных символов, а затем вручную добавить специальные символы в поле ввода.

Это отличный пример того, где очень целесообразно применять [Командные Файлы](#) так как Макросы Клавиш можно вводить и редактировать в [Редакторе Командных Файлов СМС](#) только один раз. Чтобы помочь вам добиться этого, запись клавиш доступна из самого Редактора СМС. Команда может быть отредактирована там включая специальные символы, а затем быть вставлена в карту через пункт контекстного меню поля ввода. См. раздел «Вставка Команд» для получения дополнительной информации о том, как это делается.

Вставка Команд

Если ваша карта файл содержит Командный Файл, вы можете ввести имя команды, а не фактический символ. Хотя это можно сделать и вручную, Диспетчер позволяет упростить это действие. Нажмите правой кнопкой мыши на поле, в которое вы хотите ввести команду. Появится обычное выпадающее меню правой кнопки мыши. Выберите «Insert Command» (Вставить команду) и вы увидите окно, похожее на это:



Окно содержит все имена команд из файла СМС. Чтобы ввести определенную команду, найдите её в списке и либо дважды щелкните на её название или выделите и нажмите кнопку «Done» (Готово). Выбранная Команда будет введена в поле ввода автоматически.

Режимы Карты

Диспетчер поддерживает «Режимы» как способ увеличить число функций, доступных одному отдельному элементу управления. Текущий режим можно устанавливать с помощью «ProThrottle» или «FighterStick». В этом случае у вас будет три режима. Одна из кнопок на устройстве управляет циклом в три режима и тремя светодиодными индикаторами на корпусе, которые демонстрируют конкретный режим, в котором карта пребывает в настоящее время.

Режимами можно также управлять с помощью сценариев Диспетчера. В этом случае будут доступны четыре режима, но не будет индикации, показывающей, в каком конкретно режиме работает карта. Выбирать режим посредством CMS легче всего с помощью одной из 4-х позиционных хаток, что дает доступ к любому конкретному режиму непосредственно. Для получения дополнительной информации по управлению режимами с помощью CMS, см. раздел «CMS сценарии».

Включение режима

При первом запуске новой карты доступен только один режим MODE1. На панели вкладок Режимов в Диалоговой Области будет только одна вкладка с пометкой “Mode 1” (Режим 1).

Чтобы включить в карте возможность использовать несколько режимов вы должны определить, какое устройство будет использоваться для переключения режимов. Сегодня есть два продукта, «FighterStick» и «ProThrottle», которые имеют возможность управления режимами аппаратно. На «FighterStick», управляющая режимами кнопка, находится с правой стороны в верхней части ручки. На «ProThrottle», кнопка управления режимами активируется нажатием непосредственно на миниджойстик (утапливая его внутрь, но не перемещая вниз, право, лево или вверх). Оба этих продукта имеют цветные светодиодные индикаторы, которые показывают, какой режим активен. Они включаются и выключаются при смене режимов, так что вы всегда можете знать, какой именно режим действует и какая кнопка управляет этими переключениями.

Вы также можете управлять режимами с помощью скриптов CMS. В этом случае режимы также будут меняться, но светоиндикация меняться не будет. См. примеры в разделе «[Примеры Скриптов CMS](#)» для дополнительной информации.

Выбор способа управления режимами осуществляется с помощью диалогового окна вкладки «Программные Настройки». После того как вы выбрали способ из описанных выше, в Диалоговой Области появятся дополнительные вкладки. Если вы выберете вкладку «ProThrottle» или «FighterStick», вкладки режимов с 1-ого по 3-ий станут доступны. При выборе способа с использованием CMS, то появятся вкладки режимов с 1-ого по 4-ый.

Программирование

Программирование для нескольких режимов, не отличается от программирования для одного, с той лишь разницей, что элемент управления можно заставить делать одно из трех или четырех различных действий, отталкиваясь от текущего режима. Как упоминалось выше, Mode 1 (Режим 1), режим по умолчанию. Он должен быть отредактирован, иначе управление не будет действовать на всех остальных.

При использовании Режимов, не нужно редактировать все три режима для каждого элемента управления. Любой незапрограммированным режим будет выполнять назначения, сделанные для Режим 1. Просто запрограммируйте то, что вам надо в Режиме 1 и это автоматически будет применено к режимам 2 и 3. При этом Режим 1 будет заполнять собой только любой «пустой» Режим, а не будет применяться сразу ко всем трём режимам. Например, если Вы запрограммировали элемент управления сделать одно действие в режиме 1 и другое в режиме 2, но режим 3 остался пустым, то Режим 3 все равно получит функции, назначенные в Режиме 1.

Коды клавиш Диспетчера Устройств

Диспетчер распознает большинство стандартных символов, напрямую. Любая клавиша, которая может быть введена как один символ ("а", "В", "%", "*", и т.д.) могут быть введены напрямую. Если нужна клавиша с шифтом (SHF), это будет сделано автоматически при вводе с нажатым шифтом. Для других клавиш, используются имена клавиш. Вот полный список имен клавиш, которые распознаёт Диспетчер Устройств.

Основные клавиши клавиатуры:

Клавиша	Имя
Backspace	BKSPC
Caps Lock	CAPS
Enter	ENT
Escape	ESC
Left Alt	LALT
Left Control	LCTL
Left Shift	LSHF
NumLock	NUMLOCK
Print Screen	PRTSC
Right Alt	RALT
Right Control	RCTL
Right Shift	RSHF
Scroll Lock	SCRLK
Space	SPC
Pause	PAUSE
K102	102nd Key (не CIIA)

Функциональные клавиши:

Клавиша	Имя
F1	F1
F2	F2
F3	F3
F4	F4
F5	F5
F6	F6
F7	F7
F8	F8
F9	F9
F10	F10
F11	F11
F12	F12

Дополнительная клавиатура:

Клавиша	Имя
Del	KBDEL
Стрелка вниз	KBDOWN
End	KBEND
Home	KBHOME
Ins	KBINS
Стрелка влево	KBLEFT
PgDn	KBPGDN
PgUp	KBPGUP
Стрелка вправо	KBRIGHT
Стрелка вверх	KBUP

Нумерическая клавиатура:

Клавиша	Имя
/	KP/
+	KP+
0	KP0
1	KP1
2	KP2
3	KP3
4	KP4
5	KP5
6	KP6
7	KP7
8	KP8
9	KP9
Del	KPDEL
Стрелка вниз	KPDOWN
End	KPEND
Enter	KPENT
Home	KPHOME
Ins	KPINS
Стрелка влево	KPLEFT
PgDn	KPPGDN
PgUp	KPPGUP
Стрелка вправо	KPRIGHT
Стрелка вверх	KPUP

Специальные клавиши:

Клавиша	Имя
Левый Windows	LWIN
Правый Windows	RWIN
Клавиша программ Windows	APPS
Левый клик мышью	LCLICK
Правый клик мыши	RCLICK
Средняя кнопка мыши	MCLICK
Двойной клик мышью	DBLCLICK
Без клавиши	NULL

Командные файлы

Использование Командных Файлов призвана позволить вам программировать карты, используя названия функций вместо фактических имён символов. Командный файл, это на самом деле просто список команд, понятных игре как функция необходимая для активации этой команды. В принципе они выглядят примерно так:

LandingGearDown (Выпуск Шасси)	g
LandingGearUp (Уборка Шасси)	G
FlapsExtend (Выпуск Механизации)	f
FlapsRetract (Уборка Механизации)	F

Слева находится название команды, справа символ, который должен быть введён для активации команды. Командные файлы работают только с текстовыми командами и могут быть использованы только в графическом интерфейсе. Подобного рода возможность доступна для объектов сценариев CMS с помощью инструкции «% DEFINE», что будет рассматриваться в разделе о Программировании в CMS.

Весь этот процесс имеет ряд преимуществ и упрощает создание карты. Если вы сразу сядете с руководством пользователя и просто начнёте создавать Командный Файл к игре в которой перечислены все команды для клавиш и их имена, вам в действительности даже не нужно обращаться к руководству пользователя, чтобы создать карту, ведь со многими новыми играми идут руководства на компакт-диске и большую часть информации для Командного Файла можно получить методом «Скопировал-Вставил» (Cut-and-paste) из самого этого руководства.

Как только Командный Файл создан, редактирование карт становится легче, к тому же карта будет в значительной степени самодокументирована. Если вы соберёте согласованный набор имен команд, вы сможете создать карту по умолчанию для нового симулятора в считанные секунды. Выберите базовый набор имен для наиболее распространенных функций: GearUp (Убрать Шасси), AutopilotToggle (тумблер автопилота), FlapsDown (Закрылки выпустить), LookLeft (посмотреть влево) и т.д., и используйте эти имена, чтобы создать карту по умолчанию. Теперь, когда у вас появляется новый симулятор, вам нужно всего лишь создать файл CMS с именами команд, как в карте по умолчанию. Вы можете загрузить вашу базовую карту, определить ей использование файла CMS для нового симулятора и карта готова. Хотя кран (переключатель) шасси может быть “L” в одном симе и “G” в другом, имена команд будут одинаковыми и таким образом базовая карта получит соответствующие команды кнопкам в обоих случаях. Как правило, вы не будете применять это в 100% случаев, некоторые команды иногда бывает нужно просто изменить, но это может пригодиться для наиболее распространенных функций, и ускорит создание карт.

Еще одно преимущество - если вы сможете где-нибудь найти и скачать файл MAP/CMS, вы можете использовать файл CMS и даже не особенно заботиться о самой карте.

Следующий раздел подробнее рассматривает Командные файлы, как они создаются, и как ими пользоваться.

Использование Командных файлов

В этом разделе описывается создание и использование командных файлов.

Основы создания Командного Файла

Командный Файл сам по себе просто текстовый файл с расширением “*.СМС”. Они, как правило, редактируются с помощью Редактора СМ, но любой текстовый редактор также может быть использован для этого. Чтобы создать новый Командный Файл или использовать существующий, вы просто должны указать в ГИП файл, который хотите использовать. Если файл существует, он будет применён. Если нет, то будет создан пустой файл с определённым именем.

Командные файлы, как правило, называются именем самой игры, это не обязательно, чтобы он соответствовал названию карты, как в случае с файлами СМС. Расширение должно быть “.СМС” и файл должен находиться в той же папке, так как и любая связанная с ним карта. СМС файлы редактируются с помощью [Редактора Диспетчера Устройств](#).

После того как файл был создан, нужно ввести команды. Для этой цели можно использовать Редактор Диспетчера. См. разделы по [Редактору Диспетчера](#) и [Редактированию СМС файлов](#) для более полной информации о механизмах их использования.

Командный Файл это просто список названий команд и рядом с ними фактических символов необходимых для выполнения этих команд (назначения команд). Например, предположим, у нас есть игра, в которой “g” используется для выпуска шасси, “G” для уборки шасси, “f” для выпуска механизации, и “F” для уборки механизации. Чтобы определить эти команды, Командный файл, может выглядеть следующим образом:

LandingGearDown	g	(выпустить шасси)
LandingGearUp	G	(убрать шасси)
FlapsExtend	f	(закрылки выпустить)
FlapsRetract	F	(закрылки убрать)

Имена Команд начинаются в первом столбце и продолжаются до первого пробела или знака табуляции. Все, что идёт после, является определением команды. Вы можете вставить столько пробелов, сколько вам нужно, между именем и определением, они будут просто проигнорированы.

Многозначные Команды

Определение может состоять из более, чем одного символа, если это необходимо. Например, если вы хотите создать Команду, которая называется «ChaffAndFlares» (Тепловые и Радио Ловушки (ЛТЦ/ДО)) и команда для ДО (Дипольных Отражателей/Радио Ловушек) будет “c”, а для ЛТЦ (Ложных Тепловых Целей/Тепловых ловушек) будет “f”, то определение команд может выглядеть следующим образом:

ChaffAndFlares c f c f c f

Что определит автоматический выброс трёх наборов ДО/ЛТЦ.

Командные Аномалии

Еще один случай, когда полезно использовать Командные файлы это учёт некоторых аномалий, которые случаются с некоторыми командами. Например, предположим, есть команда “LandingGearToggle” (кран шасси), где нажатием один раз, шасси выпускается, а следующим нажатием убирается. Определение может выглядеть следующим образом:

LandingGearToggle g

Это будет вполне рабочая запись, но так как “g” является общим символом для двух действий, она может повторяться автоматически. Это может стать проблемой. Если вы случайно передержите клавишу, то шасси после выпуска, а затем снова уберётся. Что вам нужно сделать в этом случае, так это, обозначить, что “g” должна быть введена только один раз. Если вы определите команду, как:

LandingGearToggle NULL g

Тогда NULL будет автоматически применяться, когда вы будете использовать эту команду и вам не нужно будет каждый раз беспокоиться о том, чтобы не забыть ввести его в ГИП, как «NULL LandingGearToggle». В принципе, вы можете добавить в Команду всё, что вы моли добавлять в поле ввода ранее.

Вставка имен команд в Карту

Чтобы ввести имя команды, вы можете просто набрать его с клавиатуры, но так как имена иногда бывают длинными и их не всегда легко вспомнить, то есть более простой способ. Щелкните правой кнопкой мыши на поле ввода, в которое вы хотите вставить команду (Нормальное Действие, Действие с шифтом и т.д.). Выпадет меню правой кнопки мыши, найдите там «Insert Command» (Вставить команду). Выберите этот пункт и вы получите окно, похожее на программу записи клавиш, в котором перечислены все имена Команд из текущего Командного Файла. Дважды щелкните на команде, которая вам нужна, или выделите её и нажмите кнопку «Окау» (Готово). Окно закроется, а название команды будет отображаться в поле ввода, в котором вы начали этот процесс. Команды, которые уже были использованы, будут иметь желтый флажок рядом с ними. Так, что вы можете сразу знать, какие команды уже были задействованы.

Калибровка

Диспетчер имеет свой собственный специализированный калибровочный инструмент. Он активируется нажатием кнопки «Test/Calibration» (Тест/Калибровка) на панели инструментов или при выборе одного из устройств сопряжённых с Диспетчером в апплете Игровых Устройств Windows нажатием кнопки «Свойства». Появляется экран Калибровки реальных устройств, если Диспетчер в настоящее время в Прямом режиме и эти устройства могут быть откалиброваны и протестированы. Если Диспетчер в Программном режиме, то экран калибровки покажет только устройства Диспетчера доступные для тестирования. Устройства Диспетчера никогда не калибруются. **К**

Как правило, нет никакой разницы, проводите ли вы калибровку из графического интерфейса или из апплета Windows, но есть несколько исключений. Во-первых, если есть несколько однотипных устройств, например, два штурвала «Yoke USB». В этом случае, калибровку необходимо производить из ГИП, поскольку апплет Windows не сможет определить, какое из устройств выбирается для калибровки.

Windows XP и новые устройства

В Windows XP, если вы подключаете новое устройство после того, как Диспетчер был установлен, оно сначала будет зарегистрировано как стандартный «HID-совместимый игровой контроллер». Нужно будет обновить драйвер для того, чтобы Диспетчер мог увидеть новое устройство. Для обновления его, просто откалибруйте устройство сразу в ГИП Диспетчера, а не в апплете Windows. Калибровка в ГИП заставит Диспетчер просканировать систему на наличие новых устройств «CH» и обновить их, чтобы они появились в списке устройств. Подключая новое устройство после установки XP, убедитесь, что оно изначально откалибровано из ГИП.

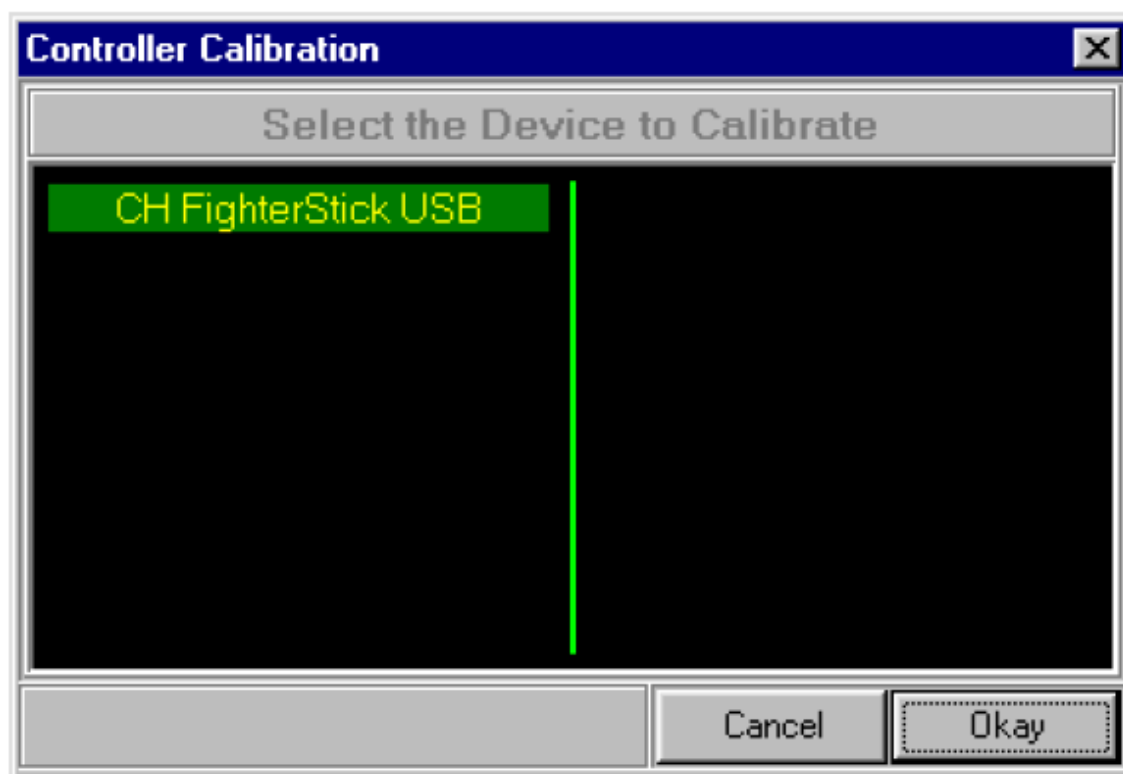
Калибровка в Диспетчере

Для начала калибровки в Диспетчере, нажмите кнопку калибровки на панели инструментов. Если вы действительно хотите откалибровать устройства, убедитесь, что Диспетчер находится в прямом режиме. В противном случае, доступным будет только тестирование. Кнопка калибровки выглядит следующим образом:

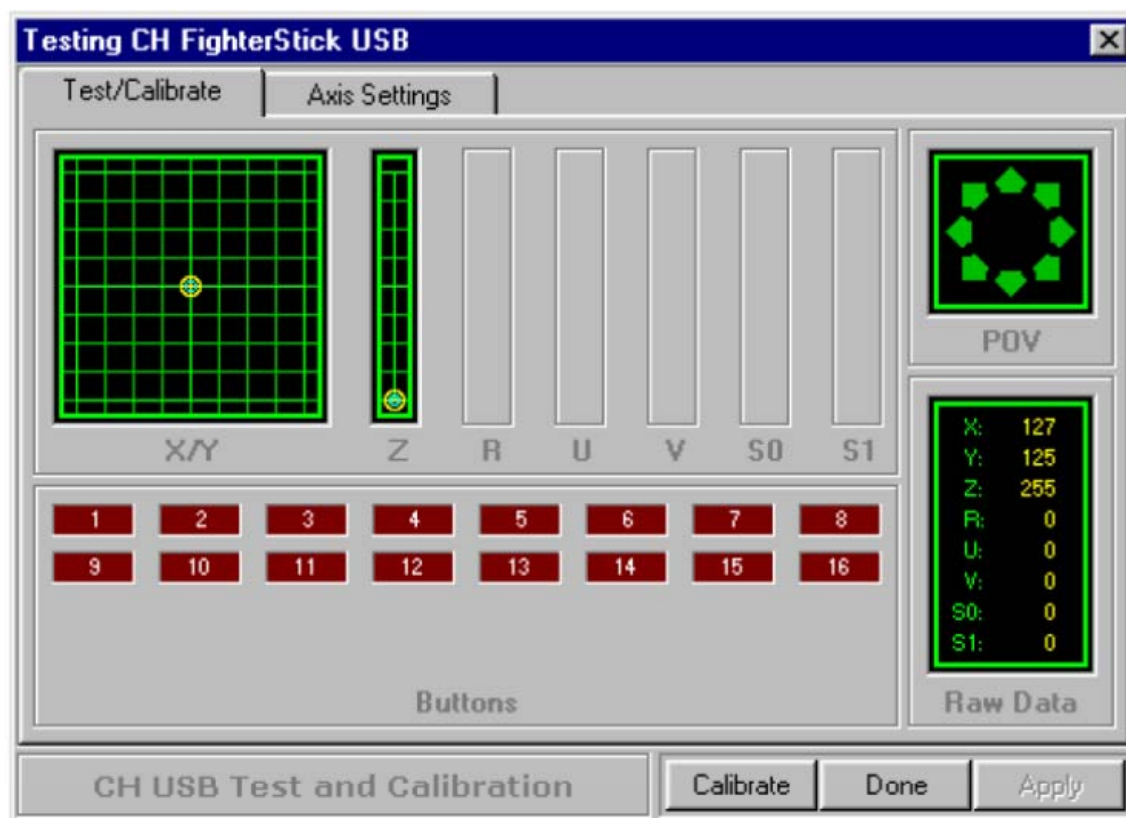


При нажатии на эту кнопку, вы перейдете к диалогу со списком доступных устройств из которого вы сможете выбрать тот, который вы хотите протестировать или откалибровать. Если диспетчер находится в Прямом режиме, будут перечислены фактические устройства. Если Диспетчер в Программном режиме, то появится список «Устройств Диспетчера», которые созданы активной картой.

Диалоговое окно выглядит следующим образом:



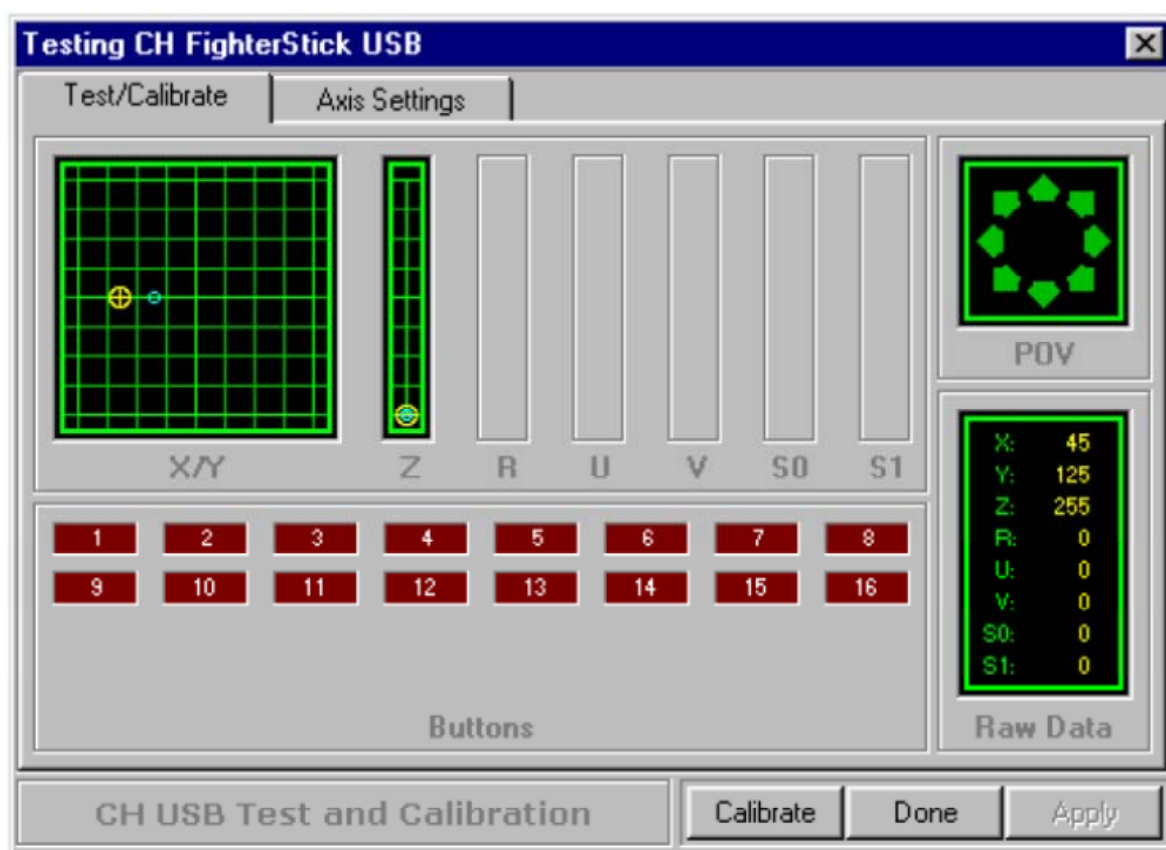
Выделите устройство, которое вы хотите проверить или откалибровать и нажмите кнопку «Okay» (Готово). Это вызовет окно экрана Тестера:



Экран тестирования имеет несколько областей. Сверху представлена область осей и позиций переключателя обзора, которая показаны в своих текущих позициях. Ниже область “Кнопок”, в которой показаны текущие состояния любой кнопки устройства. Справа область “значений”. В ней отображаются, точные значения, которые генерирует устройство.

Курсоры Осей

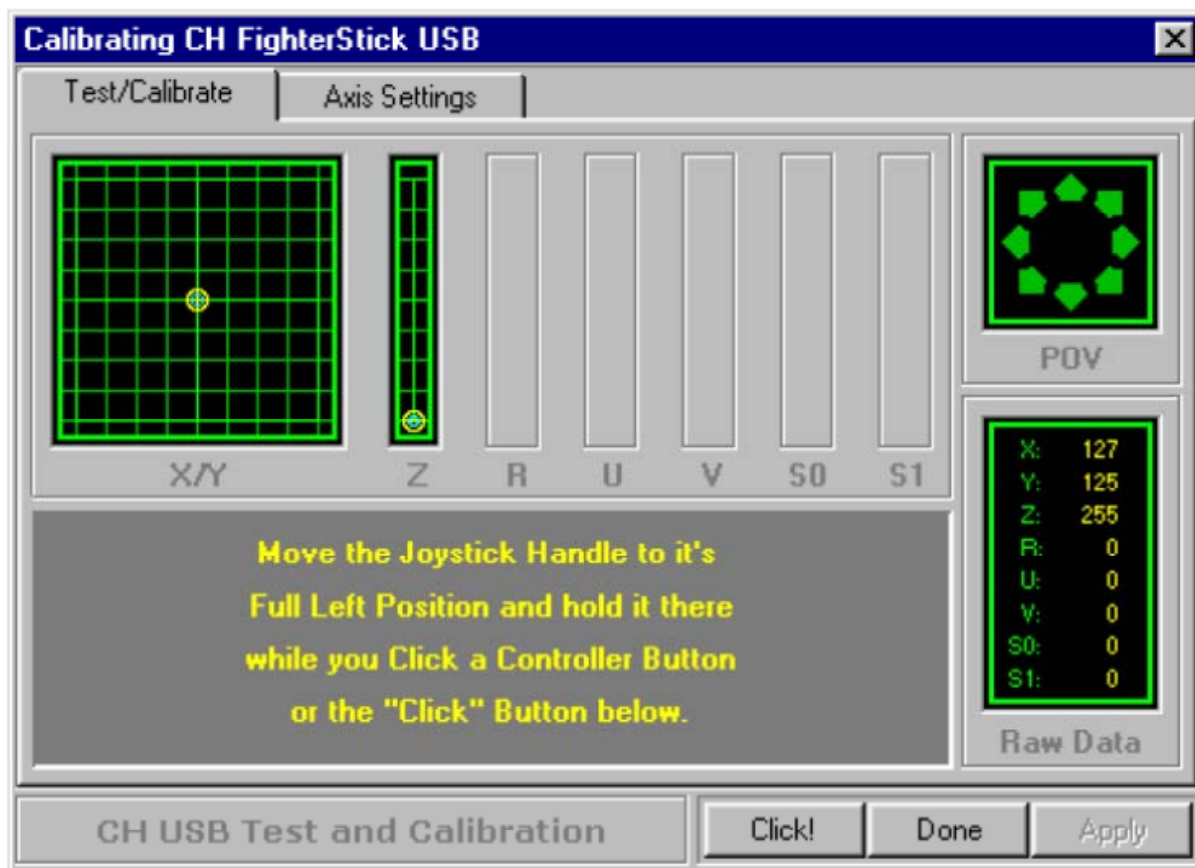
Экраны в области Осей имеют два отдельных курсора, один желтый и один светло голубой. Желтый курсор отслеживает движение самого устройства, например ручки джойстика. Синий курсор отслеживает данные, которые получает Windows после обработки настройками мертвых зон, усилий, чувствительности и т.д., которые могли быть сделаны либо на вкладке Настройки Осей (см. ниже), если Диспетчер в Прямом режиме или в настройках оси в Диалогах Диспетчера если Диспетчер в Программном режиме. Если чувствительность установлена на 100%, а коэффициент усилия имеет линейное значение, оба курсора передвигаются точно вместе. Когда параметры настраиваются в других значениях, разница можно будет увидеть не вооружённым глазом. Например:



Синий курсор показывает, что значение, которое видит Windows после произведённой настройки, составляет лишь около половины значения, которое производит само устройство. Это может быть результатом убавленной чувствительности или отклонения кривой отклика от центра. Различные параметры, которые могут быть заданы, описаны ниже и в разделе «Диалог Оси».

Проведение калибровки

Чтобы начать процедуру калибровки, нажмите кнопку «Calibrate» (Калибровка) в нижней части экрана. Вы сможете сделать это, только если диспетчер находится в Прямом режиме. «Кнопка» с дисплея исчезнет и экран будет выглядеть следующим образом:

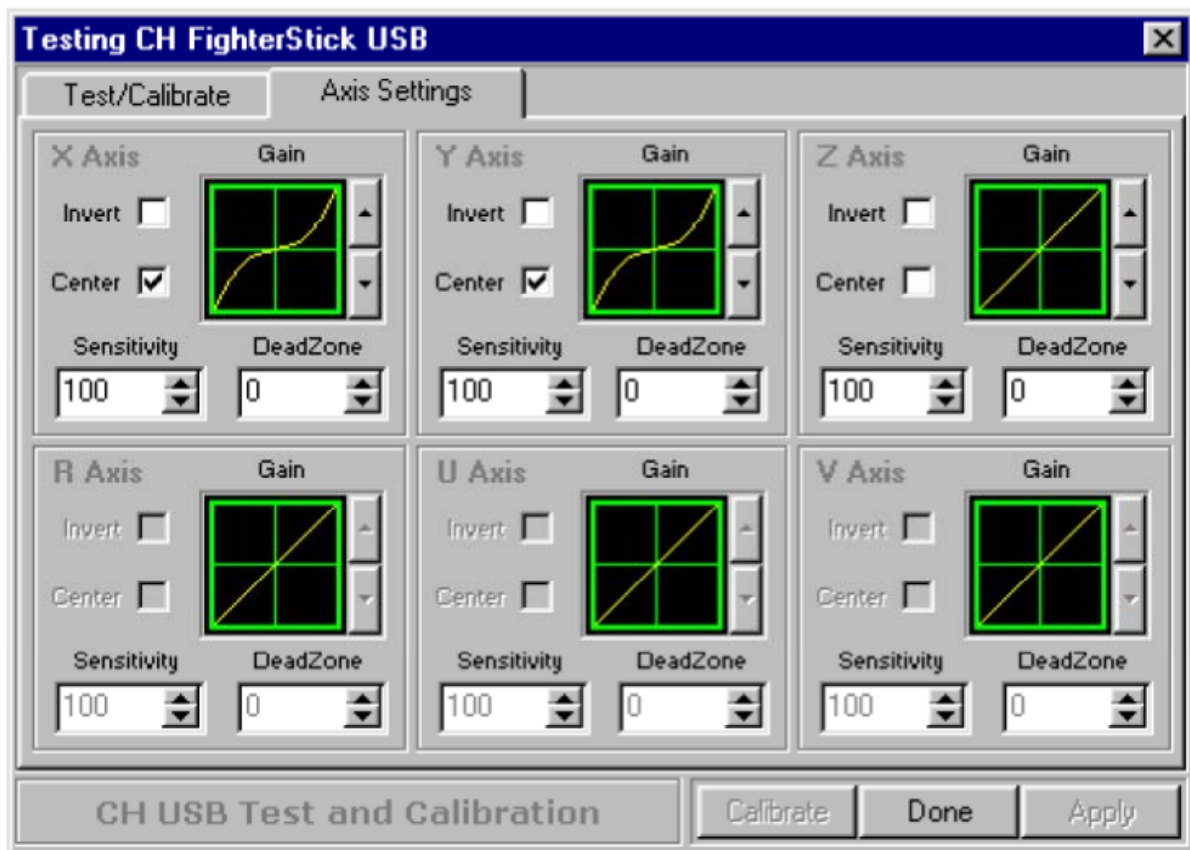


Перевод содержания информационного окна: Отклоните Ручку Джойстика в его крайнюю левую позицию и удерживая её нажмите кнопку на джойстике или кликните на кнопку «Click» (Щелчок) ниже.

Следуйте инструкциям, выделенным жёлтым цветом (перевод под картинкой) для завершения калибровки. При нажатии на кнопку, отобразится следующий запрос, о том, что теперь надо проделать аналогичные действия, но до конца вправо и нажать..., затем от себя..., на себя и в центральном (свободном) положении нажать кнопку на джойстике. Затем колесо газа, сначала до конца к себе и нажатие на кнопку, затем до упора от себя и также кнопку.... Если калибровка была завершена, появится сообщение зеленым текстом, которое сообщит вам, что калибровка завершена успешно. Нажмите кнопку «Apply» (применить) или «Cancel» (Отмена) в зависимости от того, хотите ли вы сохранить результат калибровки или нет. В любом случае вы будете возвращены к окну тестирования.

Настройки Осей

В верхней части окна Тестирования есть две вкладки, одна называется “Тест/Калибровка” и она открыта по умолчанию. Другая вкладка называется “Настройки Управления”. Опять же, это только если Диспетчер в Прямом режиме. Выбор этой вкладки открывает второй экран, который выглядит так:



Этот экран используется для установки некоторых параметров какой-либо оси контроллера. При всех своих отличиях, эта закладка регулирует тот же набор параметров, которые могут быть установлены в диалоговой области Оси, когда создаётся карта. Разница лишь в том, что установки сделанные здесь, применяются к устройству, когда Диспетчер работает в Прямом Режиме в то время как те параметры, которые заданы в диалоговой области Оси активируются, только при работе в Программном режиме. Каждый из этих параметров описывается ниже.

Флажок «Center» (Центр).

Если этот флажок установлен, то ось рассматривается как та, которая имеет центральное положение, такое, как сама ручка джойстика. Если флажок не установлен, то ось рассматривается, как если бы она не имела такого центрального положения. Колесо Газа, будет характерным примером такой оси. Флажок будет устанавливаться автоматически в соответствии с тем, что на самом деле представляет собой ось и обычно не нуждается в изменении.

Основное различие между “Центрованной” и “Не-центрированной” осью здесь то, что центрованные оси рассматриваются как имеющие центральную мертвую зону и может иметь назначенную кривую отклика. Не-центрированные оси не имеют центральной мертвой зоны и всегда откликаются линейно.

Существуют обстоятельства, когда изменение этих характеристик бывает полезным. Если в игре есть функция газа с отключенным центральным положением, то перемещение оси газа вперед вызовет движение вперед, а перемещение оси назад вызовет задний ход, а если ось “Центрована”, то можно задать широкий диапазон мертвой зоны в центре оси газа, что позволит вам найти положение “нейтрально” гораздо более легко.

Флажок «Invert» (Инвертирование)

Флажок Инвертирования меняет направление оси для Windows. Например, если ось обычно обеспечивает максимальное значение, в полностью заднем положении, а минимальное в полностью переднем, то установка флажка «Invert» даст минимальное значение в заднем положении, а максимальное в переднем.

Настройки Усилия и Отклика (Gain)

Под Настройками Осей область для настройки “Усилия и Отклика”. Эти настройки регулируют, как будет вести себя ось для Windows по отношению к фактической позиции элемента управления. Как они будут откликаться, зависит от того, установлен флажок «Center» (по центру) или нет.

Наиболее заметный пункт в этом разделе, это панель настройки усилия “Gain” (усилие). Это небольшое графическое поле и оно активно только когда ось помечена флажком «Center». Желтая линия представляет собой то, как для Windows ось будет реагировать на физическое перемещение позиции управления. Есть 11-ть кривых, которые могут быть выбраны. По умолчанию это “линейный” отклик, который представляет собой прямую желтую линию, которая проходит через всё поле, от левого нижнего угла в правый верхний.

Другие возможные кривые выбираются с помощью двух маркеров «вверх/вниз» справа от графического поля. Нажатие кнопки вверх приведет к изменению кривизны желтой линии - более вертикально ближе к центру и более горизонтально к краям. В результате управление будет более резким вблизи от центра, то есть ось Windows будет перемещаться быстрее, чем сам элемент управления в результате чего более чувствительный отклик будет ближе к центру. Каждое нажатие маркера вверх, будет приводить к ещё более крутой кривизне и соответственно ещё более резкому отклику вблизи центра.

Нажатие кнопки вниз, имеет противоположный эффект. Кривая станет более пологой ближе к центру и круче к краям. Это приводит к более точному управлению осью вблизи от центра и более резкому отклику на крайних позициях. Как и в случае с кнопкой «вверх», каждый следующий клик на маркер даст ещё больший эффект.

Чувствительность

Поле «Sensitivity» (Чувствительность) активно для центрованных и не-центрованных осей. Оно в основном определяет, насколько переместится ось Устройства Диспетчера, за полный проход элемента управления через весь диапазон. Это значение может быть от 0% до 100%.

Для центрованной оси, управление по сути отсчитывает масштаб в обоих направлениях от центра, т.е. если он установлен на 80% то управление оси Устройства Диспетчера передвинется на отметку 80% того, что Windows будет считать, полным ходом, в то время как сам элемент управления переместится на его полный ход.

Для не-центрованной оси логика будет та же, но масштабирование будет производиться с одного конца. Например, настройка колеса газа на 80%, позволит управлению оси Устройства Диспетчера, перейти от своего минимального значения к значению 80% от своего максимального значения, в то время как колесо, переместится через весь свой диапазон.

В некоторых «не-центрованных» случаях бывает нужно, отсчитывать масштаб чувствительности оси с другого края, например, если вам нужно перейти от 20% хода до 100%, а не от 0% до 80%. Чтобы добиться этого, не-центрованной оси, может быть присвоено отрицательное значение масштабирования. Установив « - 80%» мы получим ход от 20% до 100%, в то время как установка значения «80%» даёт перемещение от 0% до 80%.

«Мёртвая зона» (DeadZone)

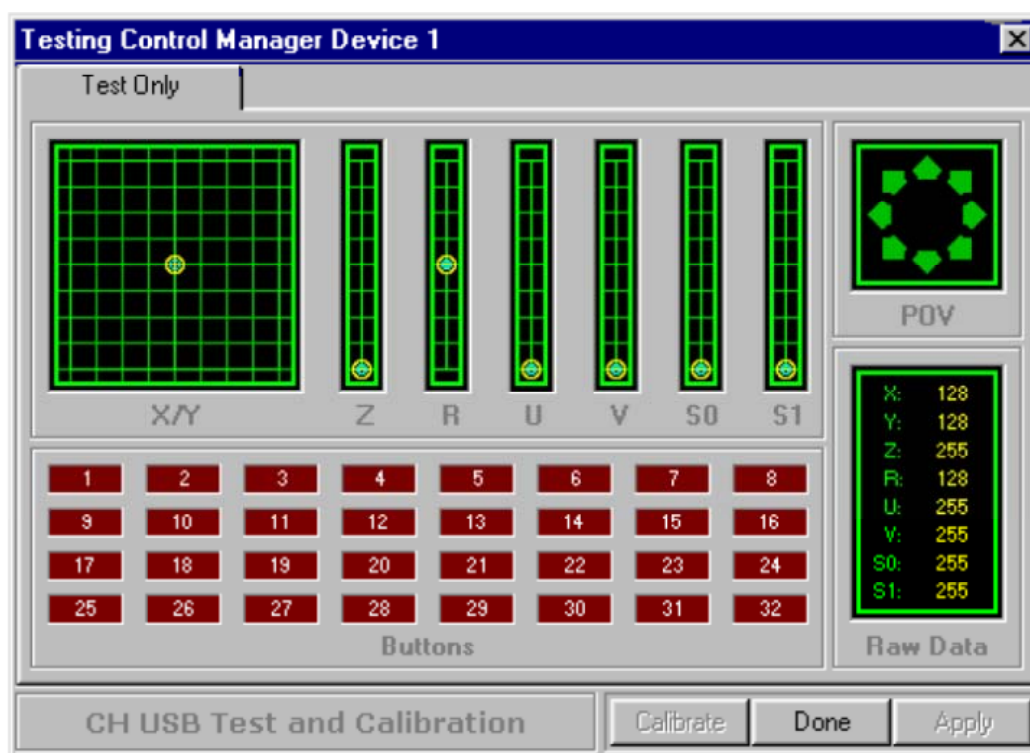
Мертвая зона применяется только к центрованным осям и определяет область вокруг откалиброванного центрального положения оси, где ось должна рассматриваться, как центральное положение. Это позволяет центральному положению быть стабильнее, даже если есть небольшой люфт вокруг центра или когда калибровка была смещена от центра на небольшое значение. Значения в 2% или около того, как правило, достаточно, но более высокие значения могут быть нужны в зависимости от того, как будет использоваться управление.

Калибровка в апплете Игровых Устройств Windows

Калибровка в апплете Игровых Устройств, по существу выполняется также, с некоторыми исключениями, указанными в начале этого раздела. Если вы в Прямом режиме и нет дублирующих устройств, то практически никакой разницы нет, за исключением того, что вы выбираете устройство для калибровки в апплете, а не в окне ГИП Диспетчера.

Как отмечалось ранее, вы можете калибровать в Windows, только если вы в Прямом режиме. Если вы находитесь в Программном Режиме, вы можете тестировать, но у вас не будет вкладки «Настройка Осе».

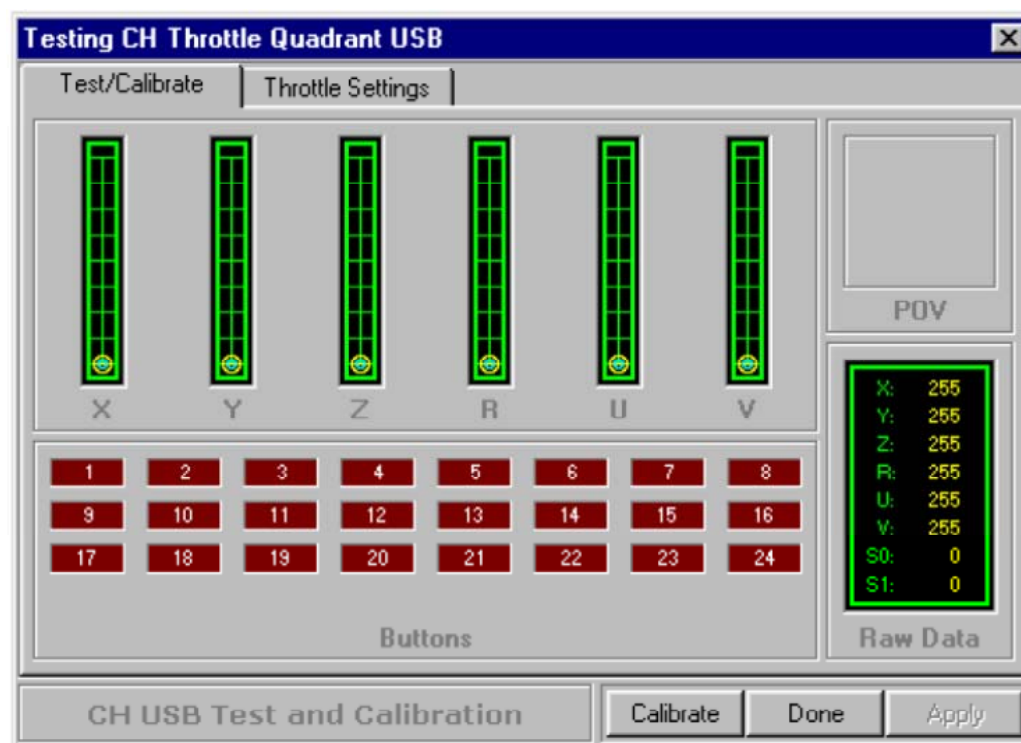
Экран будет выглядеть так в этом случае:



Где показан только выход из Диспетчера.

Калибровка «Throttle Quadrant» (Устройство «Квадрант Тяги»).

Из-за своих уникальных характеристик, апплет калибровки «Throttle Quadrant» несколько отличается от других устройств. Основное окно Теста/Калибровки выглядит следующим образом:



Очевидная разница здесь в отсутствии привычного поля «X/Y» отображаемого с другими устройствами. Здесь, отображены шесть обычных осей, по одной для каждой из ручек на «Throttle Quadrant». В остальном окно, по сути, такое же, как и стандартное.

Сама процедура калибровки также несколько отличается. Хотя все еще присутствует логика простого следования инструкциям на экране, калибровка делается для всех шести осей одновременно, упрощая процедуру до трёх шагов. Потяните все шесть ручек до конца на себя и нажмите кнопку, отдайте все шесть ручек до фиксаторов и нажмите кнопку, затем отдайте все шесть ручек до упора вперед и нажмите на кнопку. Это намного быстрее, чем калибровать каждую ось отдельно и приводит к наименьшей возможности ошибок.

Настройка «Throttle Quadrant».

Когда вы запустите апплет Тест/Калибровка для «Throttle Quadrant», вы заметите, что появилась закладка «Throttle Settings» (Настройки Тяги) в верхней части диалогового окна, в отличие от «Настройки Осей», в случае с другими устройствами. Если вы выберете эту вкладку, экран будет выглядеть следующим образом:

The screenshot shows a software window titled "Testing CH Throttle Quadrant USB" with a close button (X) in the top right corner. The window has two tabs: "Test/Calibrate" and "Throttle Settings", with the latter being the active tab. The main area is divided into six sections, each representing an axis: X Axis, Y Axis, Z Axis, R Axis, U Axis, and V Axis. Each section contains the following controls:

- Invert Data:** A checkbox, currently unchecked.
- Min Value:** A numeric input field with a spinner, set to 0.
- Max Value:** A numeric input field with a spinner, set to 255.
- Detent Value:** A numeric input field with a spinner, set to 255.
- DeadZone:** A numeric input field with a spinner, set to 0.

At the bottom of the window, there is a large button labeled "CH USB Test and Calibration" and three smaller buttons: "Calibrate", "Done", and "Apply".

Так же, как с другими устройствами, эта страница определяет характеристики осей при использовании в Прямом Режиме. Есть настройки «минимального значения», «максимального значения», «значения на упоре», и «мертвые зоны», а также флажок «инвертировать». Эти параметры имеют те же функции, что и аналогичные в Настройках Осей, для обычных устройств, когда они используются в Программном режиме. Пожалуйста, обратитесь к разделу о Параметрах DX Осей для «Throttle Quadrant» за более подробным описанием этих настроек и того как их использовать.

Утилита «CHDelete» (Удаление)

Утилита CHDelete используется для полного удаления ссылок USB устройств «CH» из реестра и позволяет проводить чистую установку, если это будет необходимо. Перед запуском утилиты CHDelete, отключите все ваши «CH» USB устройства.

Для использования CHDelete, зайдите в меню Пуск, раздел «CH Products» пункт «Control Manager». Когда вы запустите CHDelete, появится небольшой диалог с двумя основными кнопками, одна из которых удаляет все CH USB устройства, которые распознает, вторая, удаляет только «Pro Pedals USB». Опция «Delete All» (Удалить Все) удаляет и «Pro Pedals» и все другие устройства «CH». Хорошая идея, отключить любые устройства, которые вы собираетесь удалить, прежде чем запускать утилиту CHDelete.

Второй вариант, удаление только «Pro Pedals», бывает полезно, если вы позже приобретаете новое устройство «CH» (Штурвал/Джойстик) и хотели бы добавить его. Просто удаляет «Pro Pedals», прежде чем подключить Новое устройство, которое станет одним из устройств, которое Windows увидит как Джойстик№1 после перезагрузки. Это избавляет от необходимости переустанавливать и перекалибровывать весь набор устройств, просто для добавления одного устройства в набор. Для выполнения этой процедуры, сначала запустите CHDelete и выберите кнопку «Delete CH Pro Pedals...» (Удалить CH Pro Pedals...). После того, как «Pro Pedals» будут удалены, подключите новое устройство и позвольте ему пройти через процедуру Мастера Подключения нового оборудования, описанного выше. Как только новое устройство установлено, продолжите и подключите «Pro pedals» снова. Это должно вызвать Мастер нового оборудования во второй раз.

Утилита «CMStart»

Утилита CMStart это небольшая утилита, которую можно использовать для перезапуска вашей карты после перезагрузки Windows. Нормально, Диспетчер запускается в «Прямом» режиме и чтобы активировать карту вам нужно запустить Диспетчер и запустить карту. CMStart будет делать это автоматически, без необходимости каждый раз запускать Диспетчер. Для использования CMStart, необходимо скопировать ярлык CMStart.exe в папку «Автозагрузка» Windows. Сам файл CMStart.exe находится в папке, где установлен Диспетчера. Метод добавления программы в папку «Автозагрузка» различается в разных версиях Windows. См. справку Windows для получения дополнительной информации о том, как это сделать.

Операции с «Throttle Quadrant»

«CH Throttle Quadrant» это много-ручное устройство, которое обеспечивает функциональность 24-ёх кнопок и шести осей при использовании с Диспетчером. Каждая ось имеет откалиброванную фиксируемую позицию (упор малого газа) немного впереди крайне заднего положения. В принципе, это устройство работает, как и другие CH USB устройства, поддерживаемые Диспетчером, однако, есть некоторые заметные различия в том, как он функционирует и в том, как он программируется. В этом разделе летально рассматриваются эти различия. Если вы используете «Throttle Quadrant», вы также должны прочитать раздел «[Калибровка Throttle Quadrant](#)» в разделе «Калибровка» данного руководства.

Кнопки «Throttle Quadrant» (Устройства Квадранта Тяги)

Кнопки можно разделить на две группы. Во-первых, есть 12 кнопок активируемых с помощью 2-х позиционных переключателей на передней панели «Throttle Quadrant». Еще 12 “псевдо-кнопок”, по 2 на каждую ось, которые определяют когда ручка газа находится за фиксатором (Упор МГ) и когда ручка газа на фиксаторе. Кнопки назначены в пары, на оси X - кнопка 13 за упором 14 на упоре, на оси Y - кнопка 15 (за) и 16 (на), и т.д. Все 24 кнопки используют стандартный диалог для кнопок и могут быть использованы для любой из нормальных функций кнопок, которые предоставляет Диспетчер. Вы можете наблюдать за действиями этих специальных кнопок в Тестовом окне Диспетчера.

Есть 12 вспомогательных кнопок-точек в нижней части диалогового окна квадранта тяги. Они похожи на те, которые есть в Устройстве CMS и используются, чтобы назначить 12 псевдо-кнопок, поскольку нет физических кнопок на «Throttle Quadrant», которые могут быть нажаты. Функция «Выбор по Активации» недоступна для этих кнопок, так как выбор любой кнопки потребует задействовать ось для движения в соответствующую зону и само это движение оси, переключает выбор кнопки и выберет вместо неё ось. Вы должны использовать мышь, для выбора «псевдо-оси» для программирования. Функция «Выбор Активацией» будет работать нормально для переключателей (и осей).

Оси «Throttle Quadrant»

Как уже упоминалось выше, в ГИП есть 6 ссылок на оси X, Y, Z, R, U, и V. Все они в основном идентичны. Существует фиксируемая позиция немного впереди крайне заднего положения. Обычно она используется в качестве фактически минимальной позиции, область за упором используется для управления другими функциями, такими как реверс или тормоза. В связи с наличием фикс упоров и использованием осей газа, диалоги настроек, доступные для осей используемых «Throttle Quadrant» отличаются от тех, которые используются другими устройствами «CH» USB. В этом разделе говорится об этих различиях.

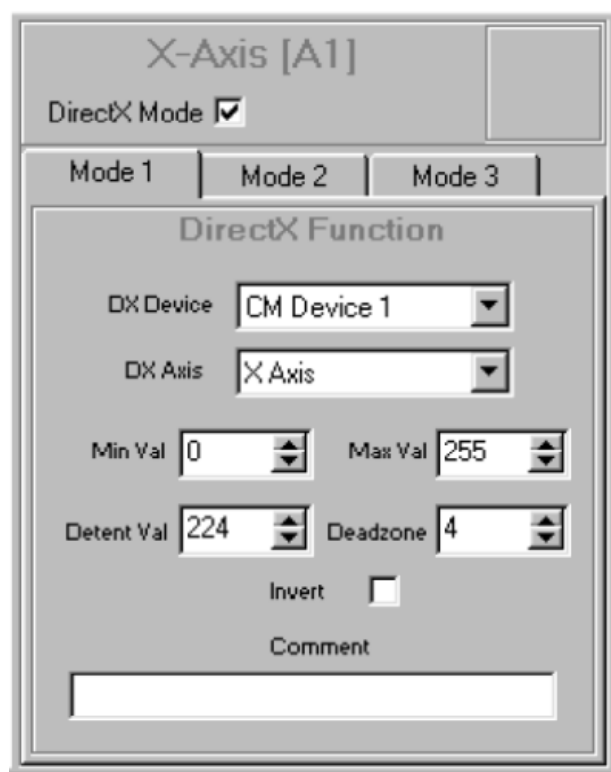
Общие Разделы

Диалог «Throttle Quadrant» разделяется на два основных элемента, так же, как диалоги для других устройств. Это флажок, расположенный в верхней части диалогового окна с пометкой «DirectX Mode» (Режим DirectX). По умолчанию, изначально установлен, чем

представляет оси для Windows, как оси обычного джойстика. Если флажок не установлен, оси переходят в “Программный режим” и используются для ввода символов. Это будет приводить к различному отображению диалогов, что будет обсуждается ниже. Второй общий элемент, это поле для комментариев. Оно используется для ввода текста, описывающего функцию в карте. Оба этих пункта будут обсуждаться более подробно в разделе, посвященном [диалогам осей](#). Вы должны ознакомиться с ним для получения дополнительной информации по этим вопросам.

Режим DirectX

Когда флажок DirectX установлен, диалог оси «Throttle Quadrant» выглядит следующим образом:



Поле «DX Device» (Устройство DX)

Первые два элемента, которые должны быть установлены, это поля «DX Device» (устройство DX) и «DX Axis» (ось DX). Поле «DX Device» определяет, какому из Устройств Диспетчера, предназначаются оси. Установка этого параметра для «Throttle Quadrant» даёт тот же эффект, что и при настройке других USB устройств «CH». Смотрите раздел «[Диалоги Осей](#)» для дополнительной информации.

Поле «DX Axis» (Ось DX) объясняет Диспетчеру, какая ось Устройства Диспетчера, указанного в поле «DX Device» должна получить назначения. Если бы вы выставили для оси X - “CM Device 1” и “X Axis”, например, то X ось будет контролировать Ось X Устройство Диспетчера №1. Опять же, это более подробно освещено в разделе, посвященном «[Диалогам осей](#)».

Параметры DX Осей

Из-за наличия «упора», Direct X параметры, которые могут быть установлены для «Throttle Quadrant» немного отличаются от тех, которые предназначены для осей других устройств. Есть четыре отдельных параметра, которые контролируют то как будет восприниматься ось и то, какими будут значения. Это «Min Val» (Минимальное Значение), «Max Val» (Максимальное Значение), «Detent Val» (Значение оси на Упоре) и «Deadzone» (Мёртвая Зона). Первые три параметра определяют, какие значения будет иметь ручка, когда она будет именно в указанном месте. Значения между этими позициями будут изменяться линейно. Например, если вы зададите «Min Val» (минимальное) - 0, «Detent Val» (упор МГ) на 240, и «Max Val» (максимальное) на 255, вы увидите, что значения будут 0, когда ручка в крайне переднем положении, 240 когда она на упоре, и 255, когда она в крайне заднем положении. Как и другие оси, оси «Throttle Quadrant» имеют значения от 0 до 255.

Один из распространенных видов настроек это установка «Max Val» (макс.) и «Detent Val» (упор) на 255 и «Min Val» (мин.) на 0. Это обеспечивает управление полным ходом газа (0%-100%) между упором и крайне переднем положением, зона за упором остаётся свободной. Эта зона может быть использована для активации таких функций, как тормоза или реверс тяги, используя специальные “кнопки”. Кроме того, значение может быть обработано скриптом CMS. Для обеспечения постоянства, значения для CMS масштабируются так, как будто газ всегда имеет значение 240 на фикс упоре. Скрипт, который должен знать, когда рычаг газа на фикс упоре, можете просто проверить это значение. Минимальные и максимальные значения видимые сценарием 0 и 255 соответственно.

Еще одно замечание по использованию этих параметров. Они «интерблокированы» для предотвращения беспорядков в настройке. В Диспетчер заложена логика, которая проверяет, чтобы максимальное значение было больше или равно значению на упоре, а значение на упоре было больше или равно минимальному значению. Если вы попытаетесь ввести максимальное значение, которое меньше значения на упоре, то оно будет установлено обратно значению упора. Подобная вещь произойдет и если вы пытаетесь установить минимальное значение больше значения на упоре. Если у вас возникли проблемы, установите сначала значение упора МГ, после чего у вас будет установить минимальное и максимальное по желанию.

DeadZone (Мёртвая Зона)

Четвертый параметр это “мертвая зона”. Он на самом деле определяет диапазон действия фикс упора. Это пространство для физических колебаний небольших значений, когда ручка находится в позиции «на упоре» и параметр позволяет сделать это пространство немного шире, чтобы компенсировать неточности.

Флажок «Invert» (Инвертировать)

Последний пункт для режима DirectX, это флажок – «Invert» (Инвертировать). Когда этот флажок установлен, исходные значения тяги инвертируются, на выходе они будут от 255 до 0, а не от 0 до 255. Это может быть полезным, например, в симуляторах вертолетов, где инвертированное управление, может более полно имитировать ручку общего шага.

Программный режим

Если флажок DirectX не установлен, диалог изменится в один из двух возможных видов. Одним из них предназначен для программирования осей на ввод символов в режиме «Up/Down» (вверх/вниз), другой в режиме «Position» (Позиция). Какой конкретно диалог появится, зависит от выбора переключателя (так называемой «радиокнопки») рядом с соответствующим пунктом с пометкой «Up/Down» (вверх/вниз) или «Position» (Позиция). По умолчанию это «Up/Down». Эти два режима работают так же, как и у других более традиционных устройств, но с отличиями обусловленными использованием фикс упора. Следующий раздел охватывает эти два режима и то, как с ними работать. Оба они также раскрыты несколько более подробно в разделе, посвященном [Диалогам Осей](#) и эти разделы должны быть изучены для получения дополнительной информации.

Режим Up/Down (Вверх/Вниз)

По умолчанию используется режим «Up/Down». Когда выбран этот режим, диалог будет выглядеть следующим образом:

X-Axis [A1]

DirectX Mode ☐

Mode 1 | Mode 2 | Mode 3

Programmed Function

Control Type

Up/Down ☒ Position ☐

Min Zone Normal Zone

Steps 2 Steps 2

Inc Key -> Enter Det -> Exit Det -> Inc Key ->

<- Dec Key <- Exit Det <- Enter Det <- Dec Key

Comment

Режим «Вверх/Вниз» наиболее сложный из этих двух режимов. Как и у его более традиционных партнеров, задача режима «вверх/вниз», вводить один символ определенное количество раз, когда ось перемещается в одном направлении и вводить второй столько же раз, когда ось перемещается в другом направлении. «Throttle Quadrant» использует этот режим немного шире с помощью отдельных пар символов, которые будут использоваться в зонах позади и перед упором и более того, предоставляя пары символов, которые будут вводиться, когда ручка входит и выходит из зоны упора перемещаясь вверх или вниз.

Счёт шагов

Ниже переключателей («кнопок радио») выбора режима есть два поля для ввода цифр. Они обозначены как «Min Zone» (минимальная зона) и «Normal Zone» (нормальная зона) и определяют количество символов, которое будет введено, пока ручка перемещается через диапазон между полностью задним положением и упором и между упором и крайне передним положением соответственно. Символы, которые будут вводиться назначаются ниже.

Назначение символов

Диалог имеет две линии полей ввода для осуществления этого действия. Верхняя линия имеет четыре поля, определяющих назначения, когда ручка движется вперед, подобно этому:

Inc Key (символ увеличивающий значение) -> Enter Det (вход в зону упора) -> Exit Det (выход из зоны упора) -> Inc Key (символ увеличивающий значение) ->

и нижняя линия, когда ручка движется назад:

<- Dec Key (символ уменьшающий значение) <- Exit Det (выход из зоны упора) <- Enter Det (вход в зону упора) <- Dec Key (символ уменьшающий значение).

Верхняя строка, читается слева направо, и имеет четыре поля. Первое, это «Inc Key» (символ увеличивающий значение), которое определяет символ, который будет вводиться в то время как ручка перемещается от крайне заднего положения к позиции упора. Следующее поле «Enter Det» определяет вход в зону упора и задаёт символ, который будет введён единожды, когда ручка достигнет начала зоны упора. Далее «Exit Det» (выход из зоны упора), определяет символ, который будет введён единожды, когда ручка покинет зону упора при движении вперед, и, наконец, второй «Inc Key» (символ увеличивающий значение), определяет символ, который будет вводиться, пока ручка перемещается от конца зоны упора к крайне переднему положению. В нижней строке всё происходит аналогичным образом, но применимо к перемещению ручки в обратном направлении. Учтите, что на практике, вы не всегда сможете назначить больше, чем несколько символов в области «Min Zone» (минимальная зона), т.к. эта зона физически очень узкая и иногда просто не может вместить много символов.

Диалог «Position» (Позиция)

Второй тип диалога из доступных в программном режиме, это диалог Позиции. Как и диалог «Вверх/Вниз», этот диалог эмулирует стандартные настройки оси, но и обеспечивает некоторую дополнительную функциональность из-за наличия упора.

При выборе режима «Позиция», диалог будет выглядеть следующим образом:

X-Axis [A1]

DirectX Mode ☐

Mode 1 | Mode 2 | Mode 3

Programmed Function

Control Type

Up/Down ☐ Position ☒

Min Zone Keys

Normal Zone Keys

Comment

Zone Keys (Клавиши Зон)

Есть два поля ввода. Они отмечены как «Min Zone Keys» (Клавиши Минимальной Зоны) и «Normal Zone Keys» (Клавиши Нормальной Зоны). Они определяют символы, которые вводятся в зоне между упором и крайне передним положением и символы вводимые между упором и крайне задним положением соответственно. Во всех случаях символы вводятся ровно один раз, какой символ должен быть введён, целиком зависит от положения ручки, а не от направления её движения. Отметим, что, как и в режиме «вверх/вниз», число шагов доступных в Минимальной зоне, вероятно, будет весьма ограничено.

Другие замечания

Функция SCALE (масштабирование) в настоящее время игнорируются в отношении осей квадранта тяги. Только настройки сделанные в диалоге осей ГИП будут иметь эффект. Вы можете вручную изменять масштабирование осей в сценарии, или вы можете скопировать их в ось CMS и применить масштабирование к этой оси.

Операции с Трекболом «Trackball Pro»

Перевод отсутствует по причине маловероятности его полезности 😊

Введение в CMS

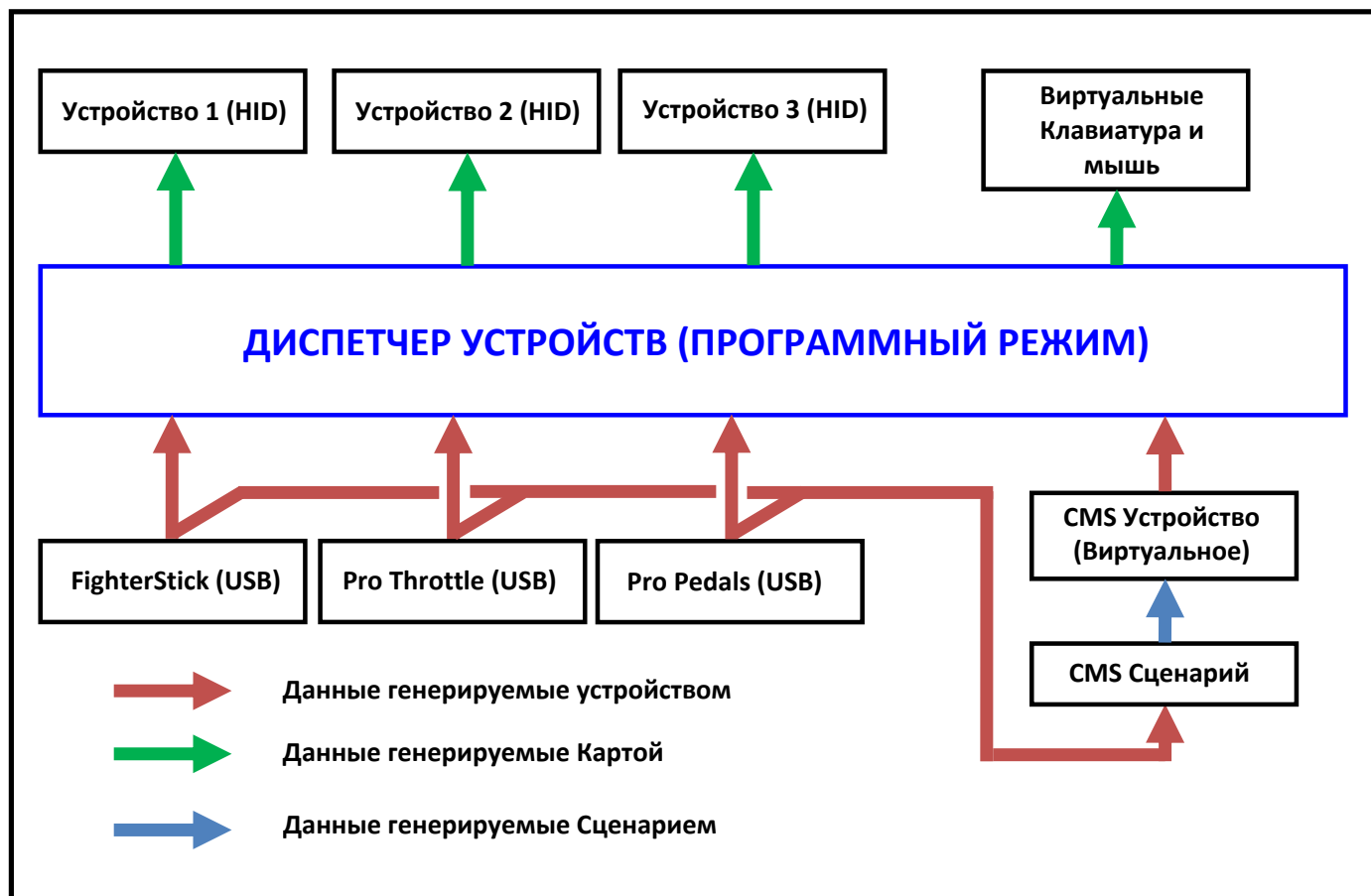
В большинстве случаев для программирования Устройств достаточно использовать только ГИП, но бывают моменты, когда нужно создать такую функцию, которая недоступна средствами одного лишь ГИП. Диспетчер включает в себя мощную Утилиту текстового редактирования сценариев, которая называется CMS (Control Manager Scripting/Язык Сценариев Диспетчера), которая может быть использована для создания таких функций, если вы в них нуждаетесь.

Следующие разделы посвящены основам CMS, использованию Редактора Сценариев CMS, различным элементам языка сценариев CMS, и рассматривают некоторые простые примеры того, что может быть сделано этим инструментом. Вы должны подробно ознакомиться с этими разделами, прежде чем начать создавать свои собственные сценарии. CMS сценарии, как правило, просты в освоении, но это не значит, что вы будете запросто понимать их “интуитивно”. Изучите различные функции и примеры, чтобы получить лучшее представление о том, что вам нужно делать.

Основы CMS

CMS работает путём непосредственного доступа к различным элементам управления ваших Устройств. Он обрабатывает их в соответствии со списком команд записанных в скрипте, затем направляет результаты обработки в особый набор псевдо элементов управления, называемый «CMS Устройством» (CMS Controls). Вы можете добавить такое «Скриптовое Устройство» в карту при помощи кнопки «Add» (Добавить) так же, как и любое другое реальное устройство. Устройство CMS, также автоматически добавляется в ГИП, при использовании Мастера Создания Карт, если вы предварительно соглашаетесь подключить поддержку сценариев.

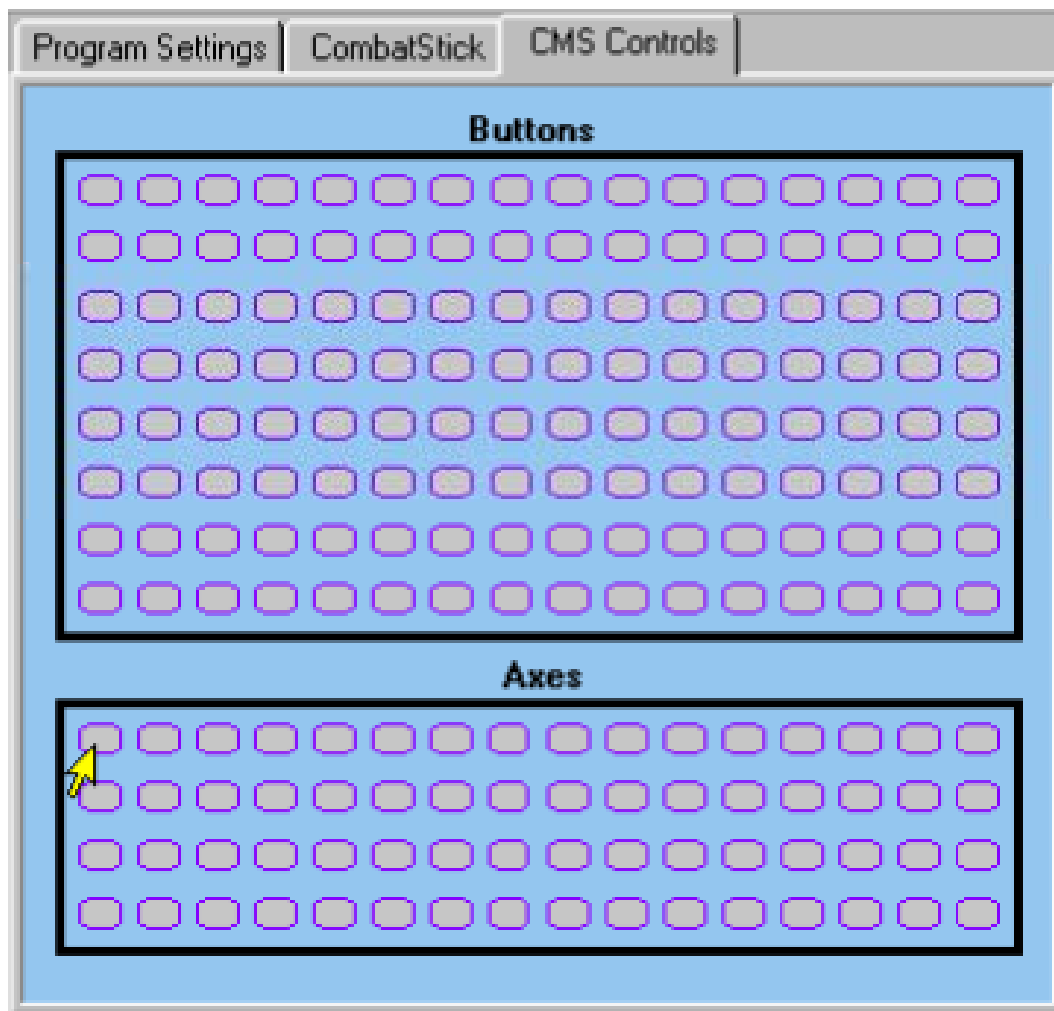
Конфигурация системы, во время использования сценариев CMS выглядит так же, как в Программном Режиме:



Разница заключается лишь в наличии CMS сценария и CMS Устройства, изображенных в правом нижнем углу. Принцип следующий, данные генерируемые устройствами (**показаны красным цветом**) поступают от устройств USB и передаются на обработку сценарию CMS. Данные обрабатываются в соответствии со списком команд, которые содержит сценарий и на выходе формируются данные генерируемые сценарием (**показаны синим цветом**). Созданные сценарием данные, передаются псевдо-устройству называемому “CMS Устройство”. CMS Устройство генерирует данные в виде элементов управления (кнопок и осей), как у реальных устройств USB. Эти данные доступны для программирования в карте, как оси и кнопки реальных USB-устройств.

CMS Устройство

CMS Устройство содержит 64 оси и 128 кнопок. При добавлении его в карту, у вас появится закладка этого Устройства на Панели Выбора, которая работает так же, как закладка любого другого Устройства. Область Выбора будет выглядеть следующим образом:



Работает это абсолютно так же как у других Устройств находящихся на панели Выбора. Обозначены все 128 кнопок и 64 оси. При прохождении курсора мыши над ними, будут всплывать метки, идентифицирующие кнопки и оси находящиеся под курсором. При нажатии на эти значки будет производиться выбор той или иной оси или кнопки для дальнейшего программирования.

Программирование CMS Устройства

Программируются эти оси и кнопки точно так же, как оси и кнопки реального устройства. Они могут быть настроены на работу в режиме DirectX и иметь назначенные функции осей или кнопок, или они могут быть установлены в Программный Режим и вводить символы или их комбинации. Диалоговая Область будет нести в себе те же диалоги, что и для других устройств в Диспетчере.

Основным различием между CMS Устройством и реальными Устройствами является то, что вы сами можете определять, когда активизировать кнопки, и какие значения имеют оси CMS Устройства, посредством файлов сценариев CMS. CMS так же может обращаться к любой оси или кнопке на любом из ваших реальных устройств. Он получает эти данные и обрабатывает их в соответствии со сценарием CMS и полученный результат направляет в CMS Устройство. Затем можно использовать ГИП для назначения функций CMS Устройства на оси и кнопки устройств Диспетчера, так, что бы они стали понятны Windows или чтобы они могли выводить символы и комбинации.

CMS Устройство, по существу, один большой «самодельный» джойстик, который вы программируете так же, как любое реальное устройство СН.

Переменные

Все, что делает CMS представляется «переменными» того или иного рода. Переменные используются для взаимосвязи кнопок и осей реальных Устройств с CMS Устройствами, а также в качестве промежуточных данных, которые хранятся в сценарии, во время их обработки. Существуют два основных типа переменных: аналоговые и битовые. Этот раздел объясняет эти основные типы переменных и их связь.

Аналоговые переменные

Первый основной тип переменной - «Аналоговый». Аналоговые переменные используются для предоставления числовых значений. Это может быть текущее значение оси на реальном устройстве, значение одной оси CMS Устройства, или просто результат какого-нибудь вычисления, определенного в сценарии. Вообще аналоговые переменные могут иметь очень широкий диапазон значений, чтобы иметь больше ресурсов для вычислений. Тем не менее, значения осей реальных устройств, всегда будут варьироваться от 0 до 255. Диспетчер имеет именно такие калибровочные мощности и вы сможете рассчитывать именно на этот диапазон значений, ссылая переменные на реальные устройства. Значения, которые относятся к осям CMS Устройства, также будут иметь диапазон от 0 до 255, так как эти данные в конечном итоге будут направлены в Windows под видом оси Устройства Диспетчера или заданы в режимах «Вверх/Вниз» или «Позиция» (в ГИП). В любом случае стоит знать, что диапазон полного хода всегда будет от 0 до 255.

CMS контролирует, чтобы значения находились в правильном диапазоне перед назначением их CMS Устройству. Если оси присвоить значение меньше нуля, то CMS Устройство будет рассматривать его как нулевое, если больше 255, то оно будет рассматриваться как 255. Заметьте, что это справедливо только в случае если значения окончательно назначены CMS Устройству и готовы к использованию картой. Значения часто могут превышать этот диапазон во время различных вычислений и т.п., конфликтов с CMS не будет.

Скрипт при обращении к фактической оси устройства, напр. JS1.A1, всегда сначала получает значения диапазона и величины отклика, а затем применяет их, даже если уже применены масштабирование и центровка (если применимо). Это означает, что сценарий может рассчитывать на стабильные, полноразмерные, линейные значения осей Устройства. Любые назначенные в скрипте параметры диапазона и кривых отклика, не применяются до тех пор, пока не будут переданы на ось CMS и далее назначены Устройству Диспетчера. Параметры диапазона и кривой отклика, не применяются вовсе в случаях, когда CMS ось используется в режимах «Вверх/Вниз» или «Позиция».

Битовые переменные

Второй основной тип переменной «Битовый». Битовые переменные могут иметь только одно из двух значений, TRUE или FALSE («верно»/«ложно»). Они могут

представлять кнопки реального устройства или CMS Устройства, где значение TRUE будет означать, что кнопка была нажата, а FALSE, что кнопка была отпущена. Они могут быть также результатом некоторых логических операций определённых файлом сценария, арифметического сравнения, вывода одного из логических устройств, и в других ситуациях, где применима логика TRUE/FALSE.

Константы

Константы это значения, заданные сценарию буквально. Они могут иметь аналоговые десятичные значения, такие как “241”, или битовые значения, в которых они могут иметь лишь значения “TRUE” или “FALSE”. Константы могут быть использованы везде, где и переменные за исключением тех случаев, когда скрипт не может присвоить значение константе. Это, если требуется, определяется в сценарии и не даёт возможности производить «запись» в скрипт (режим «read only» (только для чтения)). Константы, как и другие аналоговые значения, не могут быть назначены реальным устройствам.

Есть также несколько предопределённых констант, которые отвечают за установки в текущем режиме. Их мы обсудим в следующем разделе, посвящённом именам переменных.

Все переменные, используемые в CMS имеют предопределённые имена. Имена основываются на типе переменной (аналоговый или битовой) и на том, что представляет переменная. Последнее может быть управлением реального устройства, одним из элементов управления CMS Устройства, или одной из внутренних переменных, которые предоставляет CMS.

Основные принципы именования.

Все имена для аналоговых переменных начинаются с “A”, после которой следует цифра, обозначающая, какая аналоговая переменная на неё ссылается, например “A4”. Битовые переменные начинаются с буквы “B”, с последующим номером, указывающим, какая битовая переменная на неё ссылается, например, “B14”. Это относится ко всем переменным независимо от их источника.

Группы Переменных

Переменные обычно попадают в одну из нескольких групп. В некоторых случаях имени самой переменной достаточно, чтобы поместить его в определённую группу, в других, имя переменной формируется по усмотрению устройства и разделяется точкой (“.”) с идентификатором кнопки или оси.

Реальные Устройства

Реальные устройства это USB устройства «CH», которые есть в карте, они обозначаются «JS1», «JS2» и так до «JS16» (в зависимости от того, конечно, сколько у вас действительно устройств). «JS1» устройство в левой закладке устройств на панели Выбора (исключая закладку «Program Settings» (Программные Настройки)), “JS2” следующее устройство направо, и так далее. Для обозначения конкретного элемента управления этого устройства, сначала указывается идентификатор устройства, затем “.”

и битовая или аналоговая переменная. Например, если устройством на вашей левой вкладке в карте является «FighterStick», вы можете обращаться к его оси X так:

JS1.A1

Кнопка 7 того же джойстика будет:

JS1.B7

Если устройство на следующей вкладке «Pro Throttle», вы можете обратиться к его кнопке мини-джойстика, так:

JS2.B1

В случае с осями и хатками, не сразу понятно, к какой оси или кнопке вы обращаетесь. В этом случае ГИП в диалоговой области подскажет вам, как называется конкретный элемент управления на устройстве, и вы сможете увидеть его на экране. На джойстике «CombatStick» перемещение хатки №1 в верхнюю позицию, например, это Кнопка 7. Дисплей ГИП покажет «Hat 1 Up [B7]», тогда в сценарии вам нужно обращаться к этому положению хатки, как к кнопке B7. Таким же образом, ось газа будет отображаться как «Throttle [A3]».

Распознаваемыми ссылками осей устройств JSx будут от «A1» до «A6», т.е от X до V соответственно (X, Y, Z, R, U, V). Распознаваемые ссылки кнопок от «B1» до «B40». Предел «B40» определён для расширения функций в будущем. Фактически устройства не имеют кнопок после B16, так как они физически отсутствуют.

Существует, правда, одно исключение из вышесказанного. Даже если в тестовом окне вам не отображаются кнопки, вы можете ссылаться на позиции кноппеля (переключателя обзора), обращаясь к ним как к кнопкам от «B25» до «B32». «B25» соответствует верхнему положению кноппеля, после чего номера чередуются по часовой стрелке до «B32» соответствующей верхне-левой позиции кноппеля. Например, в сценарии CMS вы можете обращаться к верхней позиции как к «JS1.B25», а к верхне-правой как к «JS1.B26» и т.д. Эти ссылки будут работать независимо от режима, для работы в котором запрограммирован кноппель. Он может быть запрограммирован как обычно, для обзора, или выполнять функции клавиш. B25...B32 все равно будут актуальны в рамках CMS сценария.

Обратите внимание, что все позиции, соответствуют отображаемым в тестовом окне, если кноппель настроен на обзор. В «ProThrottle» же это необязательно. В этом случае, верхнее положение кноппеля в режиме обзора, достигается путем физического перемещения элемента управления вправо. Далее от этого положения по часовой стрелке.

Вы не можете присвоить значение оси или кнопке реального устройства. Это строго контролируется самим устройством. Эти значения имеют статус «Read Only» (только для чтения). Аналоговые значения, поступающие от устройства управления будут иметь диапазон от 0 до 255, если устройство правильно откалибровано в ГИП.

CMS Устройство

CMS Устройство, которое вы программируете в среде ГИП, при обращении к нему будет именоваться аналогично JSx устройствам, но вместо префикса «JSx» будут иметь префикс: «CMS». Например:

CMS.B1
CMS.A13

Так как в карте может присутствовать только одно устройство CMS, нет необходимости в цифровом идентификаторе. CMS включает в себя 64 оси которые именуются от «A1» до «A64», и 128 кнопок от «B1» до «B128». Они соответствуют осям и кнопкам, которые доступны для программирования в ГИП, когда выбрана вкладка CMS Устройства. Переменные Устройства CMS могут быть как считаны так и заданы. Задавая значения переменных в CMS вы будете управлять выводом CMS Устройства.

Как и переменные JSx, аналоговые переменные CMS могут иметь значения от 0 до 255. Вы можете превысить этот диапазон, но всё что вне этого диапазона в результате будет воспринято Диспетчером как максимальное и минимальное значение. Ось назначенная CMS Устройству посредством ГИП, будет действовать между минимумом и максимумом в соответствии со значениями переменной заданными в скрипте, переменная будет варьироваться от 0 до 255.

Внутренние переменные

CMS предоставляет переменные внутренней памяти для использования скриптом. Эти переменные не могут быть перепрограммированы непосредственно из ГИП, только CMS Устройство имеет такую возможность. Эти переменные используются для хранения рабочих данных и т.п. Их удобно использовать, если переменной не нужно реально воздействовать на ось, кнопку и т.п.

Существует 256 внутренних аналоговых переменных и 1024 внутренних битовых переменных. Так как нет никакого «устройства» связанного с этими переменными, префикс с ними не применяется. Правильными будут ссылки от «A1» до «A256» и от «B1» до «B1024».

Переменные Устройств

Есть также несколько переменных, которые называются «Переменными Устройств». Это битовые значения, используемые для определения текущего состояния некоторых функций, используемых в языке сценариев, в частности, таймеры. Позже о них будет рассказано подробнее, а пока достаточно отметить, что они существуют. Их 256, ссылки на них будут от «D1» до «D256» без префикса.

Специальные переменные

Существует ещё одна дополнительная внутренняя переменная, к которой обращаются именем CURRENTMODE. Она указывает на текущий Режим Диспетчера, который обычно устанавливается посредством устройств «FighterStick» или «ProThrottle». Переменной CURRENTMODE может быть присвоено значение и все карты будут менять режимы как если бы это было сделано при помощи одного из вышеупомянутых устройств. К сожалению скрипт не может влиять на светодиодные

индикаторы на устройствах, поэтому если вы установите CURRENTMODE через скрипт, то взаимодействия режимов Диспетчера со светоиндикаторами не будет.

Предопределенные константы

Есть также три предопределенные константы, которые могут быть использованы для задания или проверки переменной CURRENTMODE. Эти значения MODE1, MODE2, MODE3 и MODE4 и имеют значения 0, 1, 2 и 3 соответственно.

Предопределенные переменные и константы

Есть несколько предопределенных переменных и констант, которые можно использовать в написании скриптов. Этот раздел опишет их все.

Аналоговые переменные

CURRENTMODE

Эта переменная может быть использована для получения или установки текущего Режимы Диспетчера. Обычно он устанавливается специальной кнопкой на устройствах «FighterStick» или «ProThrottle», которая управляет режимами. На «FighterStick», это Кнопка 3 на рукояти джойстика сверху справа. На «ProThrottle», это Кнопка 1, которая активируется нажатием на мини-джойстик. При переключении режимов с помощью этого метода, нажатие кнопки будет циклически зажигать три светодиода на основании Устройства, демонстрируя, в каком режиме система находится в настоящее время. Обратите внимание, что, вероятно, вам придется переключать режим, по крайней мере, один раз, в начале игры, что бы синхронизировать светодиоды с фактическим режимом.

Режим может быть установлен и с помощью скрипта. Переменной CURRENTMODE может быть присвоено значение и все карты будут менять режимы как если бы это было сделано при помощи одного из вышеупомянутых устройств. К сожалению скрипт не может влиять на светодиодные индикаторы на устройствах, поэтому если вы установите CURRENTMODE через скрипт, то взаимодействия режимов Диспетчера со светоиндикаторами не будет.

Есть также три предопределенные константы, которые могут быть использованы для задания или проверки переменной CURRENTMODE. Эти значения MODE1, MODE2, MODE3 и MODE4 и имеют значения 0, 1, 2 и 3 соответственно. Следовательно, что бы заставить режим включиться, можно написать:

```
CURRENTMODE = MODE3;
```

Что объяснит карте, что нужно активировать Режим 3

PROTHRMODE и FTRSTKMODE

Диспетчер не может переключать светодиоды на «FighterStick» или «ProThrottle», но он всегда знает, какой из них в настоящее время активен. Вы можете использовать

эту информацию для настройки “под-режимов”, которые будут реагировать на Переключения Режимов с «FighterStick» и «ProThrottle» вовсе без использования их для установки первичного Режимы Карты. Две переменные, которые содержат эту информацию, именуют FTRSTKMODE и PROTHRMODE. Обратите внимание, что эти значения имеют статус Read Only (Только для чтения), вы не можете установить их в качестве отвечающих за переключение светодиодов.

Эти значения 0, 1 или 2 в зависимости от того какой индикатор был включён последним. Так же обратите внимание, что значения, могут потребовать синхронизации, когда скрипт запускается в первый раз после смены режима. Карта первоначально рассматривает устройство в Режиме 0, независимо от фактического состояния светоиндикатора. Если в этот момент горит первый светодиод, то все нормально. Если нет, то соответствия не будет до тех пор, пока вы не нажмете кнопку, чтобы изменить положение светоиндикации и тем самым «сообщите» об этом Диспетчеру.

Битовые переменные

Несколько битовых переменных также являются предопределенными. Они могут быть использованы в сценарии, чтобы определять и управлять некоторыми функциями, которые иначе были бы недоступны.

FIRSTSCAN

FIRSTSCAN имеет значение TRUE в первом цикле скрипта CMS. Он в основном используется для сброса переменных, когда диспетчер переключается из Прямого Режим в Программный Режим и сценарий начинает выполняться. Это избавляет от необходимости устанавливать такое значение для запуска первоначальной синхронизации.

CLOCKTICK

CLOCKTICK имеет значение TRUE, когда сценарий запущен и начался отсчёт времени. Возможно это потребует объяснения. Сценарий может запускаться по разным причинам. Чтобы отслеживать ход таймеров, которые может содержать скрипт, существуют внутренние часы, которые проверяются, при запуске сценария. Если часы «тикают» во время такой проверки, то значения таймера уменьшаются. Метка CLOCKTICK показывает, должны ли в ходе этой проверки, часы применяться к таймеру или нет.

XRELATIVE, YRELATIVE, и ZRELATIVE

Эти переменные используются для указания сценарию, каким образом данные значений CMS переходят в значения X, Y, Z мыши, относительно или абсолютно. Это необходимо при использовании Трекбола. Если Карта не содержит трекбола, то эти переменные могут быть проигнорированы.

Данные Трекбола существенно отличаются от данных джойстиков. Данные Трекбола указывают, насколько переместился шар с момента последней проверки, в то время как данные джойстика показывают, как далеко ручка находится от центра. Когда эти данные, предназначены для мыши, становится необходимым уведомлять систему о том, какой тип данных несёт это обращение. Когда вы назначаете оси устройства непосредственно мыши посредством ГИП, Диспетчер на основании источника данных,

может определить, следует ли рассматривать их как относительные или как абсолютные. Когда данные передаются через скрипт, они теряют свой идентификатор типа источника и тогда сценарию приходится «объяснять», какой тип данных используется.

Чтобы использовать эти переменные, просто назначите им значения TRUE для каждой из осей мыши, которые получают данные с Трекбола. Если мышь получает данные, которые представлены как оси джойстика, то эти переменные можно просто проигнорировать, поскольку по умолчанию они имеют значения FALSE. Опять же, если у вас нет Трекбола, то вам вообще не нужно беспокоиться об этом.

Обратите внимание, что эти значения, возможно, необходимо будет менять динамически во время выполнения сценария, если источник команд мыши изменится. В некоторых случаях, например, может оказаться предпочтительным использование трекбола в качестве мыши, в других использование джойстика. Вы не можете назначить более одного устройства в качестве мыши, но вы можете определить такое устройство с помощью сценария, преобразовать соответствующие данные в переменные CMS, а затем присвоить переменные CMS самой мыши как устройству. Если изменится источник оси, то значение также должно поменяться. См. раздел [«примеры сценариев»](#) для примера реализации такого рода логики.

TIMESTAMP

Хотя функций отсчёта времени встроенных в Диспетчер, как правило, достаточно для большинства потребностей программирования, они всё же могут расходиться с собственными алгоритмами отсчёта времени ОС «Windows», переключением потоков и прерываний, и т.д. В этих случаях, может быть полезно, иметь более «абсолютный» инструмент отсчёта. Переменной призванной служить этой функции является TIMESTAMP. Она обрабатывается как любая другая аналоговая переменная, но имеет статус Read Only (только для чтения) и выводит значение приблизительно равное нескольким миллисекундам, прошедшим с момента запуска Windows. Это число берётся из системного времени и поэтому не расходится с функциями нормального отсчёта времени.

Отметим, что, несмотря на то, что таймер считает в миллисекундах, это не означает, что вы можете задавать отсчёт в пределах одной миллисекунды. Вы можете определить, что какая-либо операция заняла больше времени, чем некоторое предопределённое время, записанное перед этим в переменную TIMESTAMP, а затем периодически проверять её, чтобы узнать, сколько прошло времени, но вы можете просматривать TIMESTAMP лишь периодически, поэтому время может варьироваться, хотя бы потому, что вы не можете мгновенно ответить таймеру.

Средние позиции Хаток и Кнюппелей.

Существуют четыре дополнительные кнопки, которые указывают на положение кнюппелей и хаток в их центральном положении. Все джойстики и штурвалы имеют кнюппели. Кроме того, некоторые штурвалы и джойстики имеют от одной до трёх 4-х позиционных хатки. С ними связаны четыре кнопки от JSx.B21 до JSx.B24 и назначаются таким образом:

B21 - Центр Хатки 1
B22 - Центр Хатки 2
B23 - Центр Хатки 3
B24 - Центр Кньюппеля (4)

Все устройства «CH», кроме педалей «ProPedals» имеют позицию B24 т.к. все они имеют кньюппель. «FighterStick» и «ProThrottle» имеют плюс ещё по 3 дополнительные хатки и поэтому поддерживают все 4 кнопки. «CombatStick» и «VPPro» имеют +1 дополнительную хатку и поэтому поддерживают только кнопки B21 и B24.

Это может пригодиться, например, при программировании системы быстрого обзора. Центральная позиция хатки может быть использованы для вывода функции “взгляд в центр”, таким образом, взгляд будет центроваться впереди при отпускании хатки.

При этом, обратите внимание, что эта кнопка находится в активном состоянии, когда хатка находится по центру. Если вы просто назначите на неё функцию кнопки, то эта функция будет непрерывно повторяться, всё время, пока хатка находится в центральном положении. Обязательно используйте значение NULL, чтобы предотвращать это, например, используйте “NULL а” вместо “а”.

Переменные CMCC экрана и мыши

Если Центр Управления Диспетчера Устройств (CM Control Center (CMCC)) запущен в вашей системе, CMS файлы получают доступ ещё к четырём дополнительным предопределенным переменным. Это SCREENWIDTH, SCREENHEIGHT, SCREENX, и SCREENY. Они дают доступ CMS файлам к размеру экрана и позиции курсора мыши во время работы скрипта. Если CMCC не активен, то эти переменные выводят нулевые значения.

SCREENWIDTH и SCREENHEIGHT

SCREENWIDTH и SCREENHEIGHT, переменные доступные только для чтения и содержат параметры разрешения экрана, которые в настоящее время используются системой. Например, если ваша система в настоящее время обеспечивает дисплей разрешением 1024 x 768, то SCREENWIDTH будет выводить 1024, а SCREENHEIGHT 768. В первую очередь это полезно при работе с переменными SCREENX и SCREENY, что будет описано ниже.

SCREENX и SCREENY

SCREENX и SCREENY, это переменные доступные для чтения и для записи, используются, непосредственно для определения или установки текущего положения курсора мыши. Их значения находятся в пределах координат экранного пикселя и соответствуют заданным ОС Windows, которые имеют положение 0,0 для точки в верхнем левом углу экрана. Максимальное значение будет на единицу меньше, чем разрешение экрана по X и Y. Если текущее разрешение экрана 1024 x 768, то доступный диапазон позиций мыши по оси X будет от 0 до 1023, а по оси Y от 0 до 767. Если вы

получите значения переменных SCREENX и SCREENY, вы узнаете текущую позицию курсора мыши на экране. Если вы запишите значение в переменную SCREENX и SCREENY, то курсор мыши “прыгнет” в точку, которую вы задали.

Примечание для кнопок мыши

Не смотря на то, что так легко получить информацию о курсоре мыши, информацию о левом или правом щелчках мыши получить из системы практически невозможно. Следовательно, клики мыши не доступны для программирования. Вы можете запрограммировать любую из кнопок устройств СН на выполнение функций щелчков мыши или активировать режим DX для кнопок мыши 1 или 2, ограничение касается только кликов реальной мыши.

Предостережение

Как упоминалось ранее, эти переменные обнуляются, если СМСС не работает. Это может создать проблемы, особенно если вы передадите карту тому, кто обычно не пользуется СМСС. Вы должны проверять в начале сценария, являются ли переменные активными и убедиться, что скрипт работает нормально. Если это не так, то вы должны проверить карту при отключенном СМСС, что бы знать, что может произойти. Ничего «катастрофического» конечно не произойдёт, наиболее вероятным, скорее всего, будет то, что курсор мыши «прилипнет» в одном из углов экрана, как только вы активируете карту. Единственным выходом из этой ситуации является отключение разъёма устройства, что переключит Диспетчер обратно в Прямой Режим и прекратит выполнение сценария.

Возможное применение

Есть несколько примеров возможного применения этих переменных.

Активация Функции

Многие игры и симуляторы позволяют использовать экранные переключатели и т.п., которые можно щелкать мышью. Вы можете назначить кнопку, чтобы она перемещала курсор мыши в конкретную позицию, применив SCREENX и SCREENY, затем имитировать щелчок мыши в этой точке, таким образом, эффективно управлять этой функцией нажатием одной кнопки.

Наряду с этим, вы можете активировать мышью практически любую функцию в игре, даже если игра не обеспечивает этого в какой-то области экрана, которая являлась бы “активной зоной” для функции, которую может отследить сценарий для щелчка мышью. При нажатии кнопки, вы сможете проверить позиции SCREENX и SCREENY, и если они в нужном диапазоне, активировать кнопку или ввести команду вызова нужной функции.

Например, предположим, что ваш симулятор имеет на панели индикатор, который показывает текущее положение шасси. Если вы зададите прямоугольник вокруг этого индикатора, вы сможете определить, находится ли курсор мыши в заданной области, ограниченной прямоугольником, считав значения SCREENX и SCREENY. Если это так, то вы можете задать сценарию активировать кнопку, отвечающую за выпуск/уборку шасси, таким образом, поместив курсор на индикатор положения шасси и нажав на

кнопку, переключить положение шасси. Не очень полезно для одной функции, но если есть несколько функций на экране, то это может сэкономить много кнопок, т.к. одна кнопка будет выполнять множество различных функций в зависимости от положения курсора мыши на экране.

Прямая установка позиции курсора мыши

Установка координат SCREENX и SCREENY приводит к почти мгновенному движению курсора мыши на новое место. Это может быть очень полезно в ситуациях, которые требуют точного управления мышью. Легко может быть определена последовательность серии точных команд, которые будут иметь довольно точные сроки между собой.

Простой пример

Это простой демонстрационный скрипт, который демонстрирует все четыре рассмотренные функции и может быть настроен довольно быстро. По сути, он превращает вашу системную мышь в “джойстик”. Во-первых, создайте новую карту. Добавьте какое-нибудь устройство, а затем добавьте CMS Устройство. Очистите оси X и Y устройства, задав им обеим значения «NONE», затем назначьте CMS.A1 и CMS.A2 соответственно на оси X и Y Устройства 1 Диспетчера Устройств. И наконец, создайте следующий скрипт используя Редактор:

```
SCRIPT
CMS.A1 = (SCREENX * 256) / SCREENWIDTH;
CMS.A2 = (SCREENY * 256) / SCREENHEIGHT;
ENDSCRIPT
```

Загрузите скрипт, а затем посмотрите на созданное Устройство Диспетчера с помощью экрана тестирования в ГИП (Убедитесь, что CMCC запущен). Дисплей осей X/Y должен отслеживать позиции вашей обычной мыши и достигать верха, низа и сторон окна осей X/Y, тогда же, когда и курсор мыши достигает верха, низа и сторон системного экрана.

Операторы

CMS поддерживает «операторы» как для аналоговых так и для битовых переменных. В этом разделе будут рассмотрены эти операторы и то, как они используются в сценариях CMS.

Аналоговые операторы

Для аналоговых переменных, поддерживаемые операторы это стандартные знаки “+”, “-”, “*”, и “/”, которые обеспечивают стандартные арифметические функции:

```
JS1.A1 + 100 // Сложение
CMS.A1 - A27 // Вычитание
```


A2 * 5 // Умножение
JS1.A1 / 2 // Деление

все они производят аналоговый результат, который может быть определён любой аналоговой переменной (за исключением переменных JSx).

Логические операторы

Для логических операций, CMS предоставляет функции AND, OR и NOT (и, или, не). Для тех, кто незнаком с логическими операторами, действия на самом деле довольно просты.

Например:

JS1.B1 AND JS1.B4

Значение будет верно (TRUE), когда и b1 и b4 на JS1 будут иметь значение TRUE, т.е. будут нажаты.

Оператор OR (или) аналогичен в использовании:

JS1.B1 OR JS1.B4

Будет «верным» (TRUE), если либо B1 либо B4 на JS1 будет имеет «верное» значение (TRUE).

Оператор NOT (не), просто инвертирует любое значение, с которым он применяется.

Например:

NOT JS1.B1

Будет «верным» (TRUE), если JS1.B1 будет иметь «ложное» значение (FALSE), и наоборот.

Сравнительные Операторы

Операторы сравнения используются для сравнения значений аналоговых величин. Есть шесть таких операторов. Каждый двух форм, одни используются привычным буквенным написанием, другие сокращениями, которые могут быть более знакомы пользователям, имеющим некоторый опыт в программировании. Обе формы производят одинаковый результат и могут быть использованы как синонимы.

Функция	Сокращение	Русский перевод
Less Than	LT	<i>Менее Чем</i>
Less Than or Equal To	LE	<i>Меньше или Равно</i>
Equal To	EQ	<i>Равно</i>
Greater Than	GT	<i>Больше Чем</i>
Not Equal	NE	<i>Не Равно</i>

Об использовании этих операторов будет подробнее описано в разделах, посвященных созданию файлов сценариев.

Оператор присвоения

Оператор присваивания, это знак “=”. Он позволяет вам задать переменной некоторое определённое значение. Это может быть константа, значение оси, или другое значение. Об этом будет рассказано позже в разделе, посвященном Предложениям.

Выражения

Выражение это по сути все, чему присвоено значение. Это может быть аналоговое или битовое значение. Любая Переменная становится выражением, если применяется в более сложных операциях созданных при помощи операторов, описанных выше. Выражения могут также заключаться в скобки, там, где необходима группировка операций. Вот примеры допустимых выражений:

```
JS1.B1 OR JS1.B2
(JS1.A1 + JS1.A2) / 2
(JS1.B1 AND JS1.B2) OR B2 OR (CMS.B1 AND CMS.B2)
```

Выражения являются основным строительным блоком для сценариев CMS, так как они используются для создания значений, которые передаются Устройствам CMS для определения их текущего состояния в карте.

Директивы

CMS позволяет также использовать «Директивы». Директивы это что-то вроде основных инструкций для сценария, но сами они при этом, частью сценария не являются. Они дают указания компилятору, который следует им во время составления сценария, но не вызывает ничего во время выполнения самого сценария. Эта версия поддерживает только одну директиву, «%DEFINE». Она работает очень схоже с определёнными командами, которые хранятся в файле CMC, но используется в сценарии CMS, а не ГИП. Директива задаётся примерно так:

```
%DEFINE ИмяНазначения ТекстНазначения
```

После того, как назначения были сделаны, компилятор заменит любое появление “ИмениНазначения” на “ТекстНазначения”. Эта возможность используется в основном, чтобы упростить написание и читаемость сценариев.

Есть несколько правил.:

1. Всё предложение %DEFINE должно быть в одну линию.
2. Оно должно задаваться до назначения ссылки “ИмениНазначения”.
3. “ТекстНазначения” может быть более, чем одно слово. Как и Файлы Команд, это в своей основе, просто текст для замещения, так что вы можете вставить туда почти всё, что можно поставить в то место, куда ссылается Имя. Единственное, на что стоит обратить внимание, это то, что если предложение %DEFINE имеет ошибку, тест компиляции сценария будет останавливаться на имени, упоминаемом в сценарии. Если он это делает, проблема вероятно в самом предложении %DEFINE.

Кроме этого, вы можете сделать очень многое, если потребуется. На практике, лучше ставить все предложения %DEFINE в самом начале программы перед словом SCRIPT. Это избавляет сам скрипт от захламления, позволяет иметь ссылки в любом месте сценария, и позволяет легко найти их, если вам нужно изменить их значения.

Предложение %DEFINE динамически изменяет цвет, так что ИмяНазначения будет иметь цвет как у аналоговых или битовых значений. Цвет при вводе данных должен меняться так же, как это делается для обычных переменных (js1. b1, a3, и т.д.). Редактор, определяет это на основе “ТекстаНазначения”, а из-за широкого спектра того, что там может быть, иногда трудно точно определить, будет ли это работать, особенно если предложение %DEFINE сложное. Во всяком случае, можно написать:

```
%DEFINE TimerOutput D5
```

```
%DEFINE TimeDuration 100
```

```
Script
```

```
TIMER( ONDELAY, TimerOutput, TimeDuration ) = JS1.B2;
```

```
endScript
```

Когда сценарий будет загружен, “TimerOutput” будет заменяться на “D5”, “TimeDuration” на “100”. Тоже самое что:

```
TIMER (ONDELAY, D5, 100) = JS1.B1
```

Есть много мест, где это можно применить. Для более сложного примера того, как использовать предложение %DEFINE, взгляните на Пример № 4 в разделе Примеры Сценариев.

Основы Скриптинга

Прежде чем углубиться в детали скриптового языка, проведём краткий обзор того, как все это работает. Первое, что нужно понять, это то, что сценарий просто “список дел, которые нужно сделать” для CMS процессов во время выполнения карты в Программном Режиме. Сценарий управляет перемещением данных, выполнением определенных операции, всем, что должно быть сделано. Каждый пункт списка (называется «Предложением») определяет некоторые действия, которые должны быть выполнены с использованием функций, которые предоставляет CMS.

Создание сценария, по сути, просто изложение множества задач для исполнения программой. Это упрощает многое, но создание списка по вашим потребностям, иногда может быть сложным делом, но такова суть.

Скриптовый Цикл

Работа Диспетчера это просто повторяющиеся «петли» операций, которые выглядят примерно так:

1. Получить новые значения осей и статусы кнопок с реальных устройств.
2. Обработать новые значения в порядке, установленном сценарием, для генерирования новых статусов кнопок и значений осей в Устройство CMS.
3. Обработать новое состояние Устройства CMS с помощью созданной в ГИП карты для получения осей, кнопок, клавиш или движений мыши для Диспетчера Устройств.
4. Вернуться к шагу 1 и повторить всё снова.

Этот цикл выполняется периодически в зависимости от нескольких факторов, таких как изменение реальных данных, сколько времени прошло со времени последнего выполнения цикла и т. п. Все это в значительной степени понятно для пользователя, но лучше разобраться в том, что происходит.

Когда выполняется Шаг 2, скрипт (если он есть) обрабатывается, начиная с первой строки, и следует по сценарию до последней строки. Хотя это занимает очень немного времени, следует отметить, что всё происходит в порядке, указанном в сценарии, а не одновременно или в случайном порядке. Ни одно предложение скрипта не может привести к чему-то, что находится перед ним по сценарию в тот же проход. Например, если что-то изменилось, 10-я строка в сценарии, не сможет повлиять на то, что произошло в 9-ой строке до следующего повторения скрипта. Иногда такая последовательность операций может быть очень удобной.

Следует также отметить, что, кроме изменений в значениях исходящих от реальных устройств, которые были получены во время Шага 1, ни одна из переменных не меняет своего значения в Шаге 1, 3 или 4. Они изменяются только в результате обработки сценарием в ходе Шага 2.

Простой скрипт

Простой пример законченного скрипта, может дать немного лучшее представление о том, как все это komponуется. Предположим, что по какой-то причине вы хотите создать карту с одним джойстиком в ней. В этой карте, вы хотите, чтобы кнопка№2 Устройства№1 Диспетчера, была в положении «нажата» в то время, когда кнопка№2 на самом деле свободна и наоборот. Это тривиальный пример, он не для практического использования, но он объясняет, как данные проходят через систему. Такой сценарий может выглядеть следующим образом:

```
SCRIPT
CMS.B1 = NOT JS1.B2;
ENDSCRIPT
```

Сценарий состоит только из одной строки между предложений SCRIPT и ENDSCRIPT. Эта строка говорит: «Установить виртуальную кнопку CMS.B1 (первая кнопка Устройства CMS), в состояние противоположное состоянию JS1.B2 (кнопка№2 реального джойстика). Каждый раз, когда цикл выполняется, состояние JS1.B2 проверяется и в CMS.B1 отправляется противоположное значение.

CMS.B1 не является устройством, которое Windows может увидеть самостоятельно, однако, вы можете перейти в ГИП и выбрать кнопку 1 в «Устройстве CMS», установить её в режим DX (DirectX) и назначить кнопке№2 на Устройстве№1. Это поможет ей связаться с «внешним миром». Находясь в ГИП, вы должны также очистить все назначения кнопки2 Устройства1, которые, возможно, уже были назначены ранее, так чтобы не получилось, что несколько устройств будут пытаться управлять одной кнопкой.

Вот и все. Если вы создали карту и загрузили её, вы обнаружите, что кнопка 2 в Диспетчере будет в нажатом состоянии, когда на самом деле Кнопка 2 на джойстике будет свободна, и наоборот.

Описание Сценария

Каждый раз, когда вы начинаете новую карту, диспетчер создает файл сценария, чтобы дальше работать по нему. Файл Сценария, это просто текстовый файл. Он имеет то же имя, что и карта, но расширение “CMS” вместо “MAP”. Этот сценарий изначально пуст и содержит только два предложения, которые отмечают начало и конец сценария. Пустой сценарий выглядит следующим образом:

```
SCRIPT  
ENDSCRIPT
```

Пустой скрипт ничего не делает. Вы должны вставить скриптовое предложение между предложениями `SCRIPT` и `ENDSCRIPT`, чтобы определить сценарию действие. Если скрипт будет пуст, то действия, которые будет выполнять карта, будут полностью определяться реальными устройствами и тем, что вы определите посредством ГИП. Функции, которые идут между предложениями `SCRIPT` и `ENDSCRIPT` являются предметом ближайших нескольких разделов данного руководства. Однако, перед тем, как перейти к ним, есть несколько моментов, которые следует отметить.

Чувствительность к регистру

Язык сценариев CMS не чувствителен к регистру, «SCRIPT» это тоже, что и «script» или даже «ScRiPt». Аналогичным образом “JS1.A1” тоже, что и “js1.a1”. Смешанный Регистр допускается, и даже может улучшить читаемость. Например, «exitCase», а не «EXITCASE» или «exitcase». Все, что есть в файле может быть в верхнем или нижнем регистре, и это не влияет на конечный результат.

Комментарии

Комментарии просто объясняют, что вы вносите в сценарий. Они не являются обязательными и не влияют на работу скрипта. Они могут появляться в любом месте сценария. Комментарий это текст следующий после “//”. Это означает, что любой текст, начинающийся с “//” и распространяется до конца этой строки не будет рассматриваться сценарием, как если бы его не было вовсе. Например:

```
// Это комментарий.
```

Комментарии не обязательны, но они являются хорошим помощником. Вещи, которые свежи в уме во время написания сценария, могут стать весьма затруднительными, если вы вернетесь к ним через месяц и попытаетесь выяснить, что вы делали. Даже самый короткий комментарий может оказать большую помощь в напоминании, зачем и что вы сделали.

Точка с запятой

Большинство предложений сценария требуют точки с запятой, чтобы обозначить конец функциональной части строки. Есть исключения из этого правила, в случае предложений управления данными и некоторых других функций, которые будут рассмотрены в

соответствующих разделах. Точка с запятой должна быть перед комментариями в строке иначе этот элемент будет рассматриваться как часть комментария.

Пример правильного написания:

```
CMS.A1 = JS1.A1; // Скопировать ось X джойстика1 в CMS ось1
```

Помните, что всё, что начинается с “//” означает комментарий и не рассматривается сценарием.

Назначающие Предложения

Простейшая и, вероятно, наиболее часто используемая операция, которую может выполнить сценарий, это присвоение значения переменной. Это делается с помощью оператора “=” и пишется, как математический знак равенства.

Для примера:

```
CMS.A1 = JS1.A2;
```

Это предложение назначает значение Оси 1 Устройства CMS, равным текущему значению Оси 2 (Ось Y) Устройства JS1.

Аналогичным образом (для битовых значений):

```
CMS.B1 = JS1.B5;
```

Назначает кнопке 1 Устройства CMS, то же состояние, что у кнопки 5 Устройства JS1.

Переменная слева от “=” должна быть переменной, которая может быть задана, включая внутренние переменные и CMS переменные. Нельзя поставить переменную JSx слева от “=”, поскольку сценарий не может изменить значения реальных устройств.

Справа от “=”, можно использовать любое допустимое выражение. Это могут быть ссылки на любую переменную, а могут быть арифметические и логические операции, скобки и т.д. по необходимости. Например:

```
CMS.A1 = JS1.A1 + JS1.A2 + 10;  
CMS.B1 = (JS1.B5 OR JS1.B6) AND NOT (JS1.B6 OR JS2.B7);
```

Отметим, что предложения заканчиваются “;”. Это необходимо для любого присваивающего типа предложения. Для других видов предложений, которые будут обсуждаться позже, символ “;” в некоторых случаях нужен, а в некоторых нет. Все они будут рассмотрены вместе с отдельными функциями для них. По большому счёту, знак “;” не вызывает проблем, если используется там где не нужен, поэтому, если есть сомнения, поставьте его в конце предложения.

Арифметические Сравнения

Арифметические сравнения могут быть использованы везде, где допустимы битовые выражения. Они рассматриваются, как верные (TRUE) если сравнение работает и как ложные (FALSE), если наоборот. Есть шесть арифметических сравнений. Как и Логические Операторы, они также могут быть представлены как в текстовом, так и в сокращённом видах. К ним относятся:

Функция	Символ	Русский перевод
Less Than	LT	<i>Менее Чем</i>
Less Than or Equal To	LE	<i>Меньше или Равно</i>
Equal To	EQ	<i>Равно</i>
Greater Than	GT	<i>Больше Чем</i>
Not Equal	NE	<i>Не Равно</i>

Чтобы обозначить, что это сравнения, эти обозначения заключаются в квадратные скобки. Результат может быть использован везде, где может быть связь с битовой переменной. Сравнение может выглядеть следующим образом:

```
[ JS1.A1 > JS2.A1 ]
```

Что означает, что если JS1.A1 больше чем JS2.A1, то такое значение верно (TRUE). Следует отметить, что квадратные скобки, ДОЛЖНЫ заключать все сравнения, даже если они уже в скобках, как может быть в случае с последовательностью IF/THEN (будет обсуждаться ниже). Сравнения могут быть использованы для нескольких вещей. Например, активация Форсажа, когда РУД будет выше определенной позиции. Имея в виду, что диапазон значений оси от 0 до 255, а максимум РУДа, в действительности имеет минимальное значение, и если предположить, что РУД это JS1.A3, вы можете написать так:

```
SCRIPT  
CMS.B1 = [JS1.A3 < 25];  
ENDSCRIPT
```

В этом случае, кнопка CMS.B1 будет начинать функционировать всякий раз, когда JS1.A3 будет достигать значения выше 90% ($255 / 25 = \sim 10\%$; $100\% - 10\% = 90\%$). Затем кнопку CMS.B1 запрограммировать в ГИП на функцию нужной клавиши или задействовать кнопку в режиме DX для активации Форсажа.

Условные операторы

Есть несколько элементов, которые позволяют управлять обменом данными в сценарии на основе текущих значений оси и переменных кнопок, позволяющих одному блоку кода выполняться если встречается определённый набор условий, а другому блоку кода выполняться, если таких условий не встречается.

IF/THEN (Если/То)

Самый основной из них блок IF/THEN. Он обеспечивает управление типа:

Если некоторое условие верно, то сделать то-то...

Предложению IF/THEN не требуется точка с запятой.

Например, чтобы сделать что-то только в том случае, когда JS.B4 нажата, вы могли бы написать такой фрагмент сценария:

```
IF( JS1.B4 ) THEN
// Сделать что-то
ENDIF
```

Если JS1.B4 имеет значение TRUE (нажата), то оператор(-ы) между THEN и ENDIF выполняются. Если JS1.B4 в значении FALSE (отпущена), то такой сегмент сценария будет пропущен.

Значение в предложении IF, может также быть представлено более сложными выражениями. Например, чтобы выполнить блок предложений, если обе кнопки 1 и 2 на JS1 одновременно нажаты, сценарий может выглядеть следующим образом:

```
IF( JS1.B1 AND JS1.B2 ) THEN
// Сделать что-то
ENDIF
```

IF/THEN/ELSE (Если/Тогда/Иначе)

Существует также конструкция IF/THEN/ELSE, которая позволяет одной из двух групп операторов выполняться в зависимости от состояния выражения. Она функционирует во многом так же, как IF/THEN, что было описано выше, но предоставляет сценарию сегмент, который будет выполняться, в случае, когда результатом IF (ЕСЛИ) станет значение FALSE. Как и предложение IF/THEN, IF/THEN/ELSE не требует точки с запятой в конце. Например, чтобы выполнить один сегмент сценария при нажатии JS1.B4 и другой сегмент сценария, если нет, вы можете написать так:

```
IF( JS1.B4 ) THEN
// Сделать что-то если JS1.B4 имеет значение TRUE
ELSE
// Сделать что-то если JS1.B4 имеет значение FALSE
ENDIF
```

Компановка IF/THEN и IF/THEN/ELSE

Разделы IF/THEN и IF/THEN/ELSE могут быть “взаимовложенными”, т.е. пункт IF или ELSE может содержать другой:

```
IF( JS1.B4 ) THEN
    IF( JS1.B5 ) THEN
        // Действие #1
    ELSE
        // Действие #2
    ENDIF
ELSE
    IF( JS1.B6 ) THEN
        // Действие #3
    ELSE
        // Действие #4
    ENDIF
ENDIF
```

Последовательности

Существует еще один тип операторов управляющих обменом данными, который понимает CMS, это «последовательности». Последовательность начинается со слова «SEQUENCE» и заканчивается словом «ENDSEQUENCE». Уникален этот оператор тем, что позволяет выполнять последовательность событий, являясь, по сути, независимой частью сценария, которая выполняется основным скриптом. Последовательность будет выполняться при каждом цикле скрипта и работать до конца последовательности, пока не встретит определённые условия, которые ей заданы.

Когда последовательность достигает того места, где она не может продолжать выполняться в этом цикле, например, ей нужно ожидать нажатия кнопки, тогда она перепрыгивает в конец предыдущей последовательности и возобновляет выполнение сценария. В следующий проход, последовательность начнётся с места, где она была прервана. Если встретятся условия, при которых она должна будет остановиться, скажем, кнопка, которую она ожидала, была нажата, тогда последовательность продолжится с этого места до следующего, где она сможет продолжиться снова. Если во время второго прохода, не было встречено условий для прерывания, последовательность будет просто пропущена и начнётся заново в следующем цикле.

Последовательность может содержать большинство основных предложений, как и основной скрипт, но существует несколько исключений. Блок SELECT в последовательности не допускается.

В дополнение к стандартным функциям, существуют ещё три специальные функции, которые могут быть использованы только в последовательности SEQUENCE. Это предложения WAIT, WHILE и DELAY (ОЖИДАТЬ, ПОКА и ЗАДЕРЖКА).

WAIT and WHILE (Ожидать и Пока продолжается действие)

Оба предложения WAIT и WHILE работают примерно, так, как будто они принуждают последовательность остановиться, пока они имеют значение TRUE. Разница между ними лишь в том, что WAIT требует, чтобы в выражении определялся переход от значения TRUE к FALSE, в то время как WHILE просто проверяет, верно ли (TRUE) утверждение во время выполнения. Чтобы проиллюстрировать это немного яснее, рассмотрим следующие две последовательности:

```
// Последовательность 1
//
SEQUENCE
    WHILE( JS1.B1 );
    B1 = TRUE;
    DELAY( 10 );
    B1 = FALSE;
    DELAY( 10 );
ENDSEQUENCE
```

```
// Последовательность 2
//
SEQUENCE
    WAIT( JS1.B1 );
    B1 = TRUE;
    DELAY( 10 );
    B1 = FALSE;
    DELAY( 10 );
ENDSEQUENCE
```

В последовательности 1, предложение WHILE указывает последовательности выполняться непрерывно, пока JS1.B1 нажата. Бит B1 будет включаться и выключаться до тех пор, пока JS1.B1 нажата.

В последовательности 2, предложение WAIT указывает последовательности выполняться единожды при первом нажатии JS1.B1. Чтобы такой последовательности продолжиться, JS1.B1 должна быть отпущена, а затем нажата снова. При каждом нажатии JS1.B1, B1 будет получать значение TRUE, затем FALSE, только один раз.

DELAY

Предложение DELAY указывает последовательности встать на паузу на определенное время. Значение его указывает, количество знаков за единицу времени, которые необходимы процедуре для задержки. Знак за единицу времени примерно эквивалентен частоте повторения символов ("Character Rate") установленной в закладке Program Settings (Настройки Программы) в ГИП Диспетчера. Значение 50, установленное по умолчанию, означает, что один разряд в значении задержки, увеличивает время задержки до 50 миллисекунд.

Образец последовательности с использованием задержки DELAY может выглядеть следующим образом:

```
// Ждать пока кнопка 4 на JS1 нажата, затем
// дважды активировать Кнопку 1 на Устройстве CMS.
//
SEQUENCE
    WAIT( JS1.B4 ); // Ждать нажатия кнопки на JS1
    CMS.B1 = TRUE; // Нажать Кнопку CMS
    DELAY( 10 ); // Ждать пол секунды
    CMS.B1 = FALSE; // Отжать кнопку CMS
    DELAY( 10 ); // Ждать пол секунды
    CMS.B1 = TRUE; // Нажать Кнопку CMS
    DELAY( 10 ); // Ждать пол секунды
    CMS.B1 = FALSE; // Отжать кнопку CMS
ENDSEQUENCE
```

Каждый раз, при нажатии JS1.B4, переменная CMS.B1, включится и выключится дважды, каждый раз на период примерно по половины секунды в зависимости от установленной у вас частоты повторений знаков.

Компоновка

Блок SEQUENCE/ENDSEQUENCE может иметь вложения и может быть вложенным в блоки IF/THEN и IF/THEN/ELSE. Имейте в виду, что последовательность, в некоторых из них не выполняется вообще, если блок пропущен в результате работы команды IF, даже если последовательность уже запущена. Она будет просто «висеть» в том месте, где находилась, когда команда IF прервала её выполнение. Это может быть полезно, а может и проблематично, в зависимости от того, чего вы пытаетесь добиться.

Логические Устройства

Логические устройства используются для таймеров и т.п., того, из чего остальная часть сценария может узнать о том, в каком она состоянии. Доступны, в общей сложности, 256 таких устройств. Они обозначаются от “D1” до “D256”. Идентификатор присваивается при назначении функции Устройства. Этот идентификатор затем используется остальной частью сценария, для определения состояния этого устройства.

В настоящее время существует шесть типов таких устройств. Четыре из них таймеры, пятый используется для генерации короткого импульса, когда заданное условие приобретает значение TRUE, и шестая используется для генерации «переключения», когда вводимая функция меняет своё состояние. В этом разделе рассматривается работа этих устройств.

Таймеры (устройства отсчёта времени)

Таймеры контролируются однобитными значениями (которыми, конечно, могут быть и выражения). В настоящее время существует четыре типа таймеров, которые распознаются CMS. Они обеспечивают наиболее часто применяющиеся в сценариях функции отсчёта, и могут быть объединены с другими устройствами, чтобы производить более сложные последовательности отсчёта времени, если это необходимо.

Таймер задержки по включению (On-Delay timer)

Первый таймер, это Таймер с Задержкой при активации. Он задаёт связанным с ним переменным значение TRUE на указанный промежуток времени, после того, как выражение управляющее таймером (его ввод) приобретает верное значение (TRUE). После того как переменная задана, она будет продолжать оставаться верной (TRUE), до тех пор, пока данные ввода не изменятся на противоположное, ложное значение – (FALSE). В это время, переменные Устройств получают значение FALSE и таймер обнуляется. Любые последующие активации будут перезапускать эту задержку.

Например:

```
TIMER( ONDELAY, D5, 10 ) = JS1.B2;
```

Первый параметр (ONDELAY) указывает тип таймера. Второй параметр (D5) обозначает Переменную Устройства, которую остальной сценарий использует для связи с этим таймером. Третий параметр (10) задает период времени в единицах на основе «Частоты Повторения Знаков», выставленной на вкладке «Program Settings» (Настройки Программы) в ГИП. JS1.B2 это ввод. Когда он приобретает значение - верно (TRUE), отсчёт начинается. Если JS1.B2 задействован в течении указанного периода времени, то переменная D5 становится верной (TRUE) и остаётся верной до тех пор, пока JS1.B2 не будет отпущена. Если JS1.B2 будет отпущена до истечения установленного периода, то D5 не станет «верной» вовсе.

Чтобы воздействовать на таймер, D5 должна быть обозначена в другом месте в скрипте.

Например скрипт:

```
SCRIPT
    TIMER( ONDELAY, D5, 10 ) = JS1.B2;
    CMS.B1 = D5;
ENDSCRIPT
```

будет включать CMS.B1 на время в 10 единиц или около половины секунды, после нажатия и последующего удержания кнопки JS1.B2. Если JS1.B2 будет отпущена до истечения установленного периода, таймер сбросится и CMS.B1 не будет запущена.

Таймер задержки по отключению (Off-Delay Timer)

Второй тип таймера – с задержкой по отключению. Этот Таймер, задаёт Переменной Устройства значение TRUE сразу же после её ввода, и остается с этим значением, до тех пор, пока ввод имеет значение (TRUE), т.е. активирован. Когда ввод снова становится «ложным» (FALSE), запустится отсчёт заданной задержки. По истечению задержки, переменная Устройства получает значение FALSE. Если ввод снова становится верным (TRUE), прежде чем время задержки вышло, таймер сбрасывается, а если ввод снова становится FALSE, этот период времени будет начат сначала.

Таймер с Задержкой по отключению, задаётся почти, как и Таймер с Задержкой по включению:

```
TIMER( OFFDELAY, D3, 20 ) = JS2.B3;
```

Первый параметр определяет его тип - «OFFDELAY», второй параметр определяет переменную, и третий параметр задаёт продолжительность задержки. В приведенном выше примере, переменная D3 станет верной (TRUE) сразу после нажатия JS2.B3 и будет оставаться таковой пока JS2.B3 не будет отпущена, после чего пойдёт отсчёт времени продолжительностью примерно в секунду. По окончании этого периода времени, переменная D3 снова будет установлена в значение FALSE.

Это бывает полезно в самых различных случаях. В симуляторах Боевой Авиации, например, вам, может понадобится активировать камеру оружия, после нажатия на Боевую Кнопку и удерживать камеру включённой некоторый период времени после того, как курок отпущен.

Скрипт для этого, может выглядеть следующим образом:

```
SCRIPT
    TIMER( OFFDELAY, D1, 200 ) = JS1.B1;
    CMS.B1 = D1;
ENDSCRIPT
```

Здесь переменная D1, и связанная с ней виртуальная кнопка CMS.B1, будет иметь верное значение (TRUE), как только вы нажмёте на гашетку, и будет оставаться в этом состоянии примерно около 10 секунд после отпускания гашетки. В ГИП, можно запрограммировать CMS.B1 на ввод символа отвечающего за «Включение Камеры Оружия» при нажатии и «Выключение Камеры» при отпускании, используя соответствующие поля для ввода.

Таймер периода

Третий тип таймера - таймер периода. Этот таймер присваивает переменной значение TRUE, сразу после того, как ввод инициирует TRUE (был нажат, например). Он остается в этом состоянии на протяжении указанного периода времени, независимо от того, изменились ли данные на вводе. Если Таймер Периода установлен, например, на 2 секунды, его вывод будет в значении TRUE в течение 2-х секунд, вне зависимости от того, сколько длится ввод со значением TRUE, 0.1 секунды или 10 секунд. Синтаксис похож на другие типы таймеров. Выглядит это примерно так:

```
SCRIPT
    TIMER( PERIOD, D3, 20 ) = JS1.B1;
    CMS.B1 = D3;
ENDSCRIPT
```

Параметры, по существу те же, что у Таймеров ONDELAY и OFFDELAY. «PERIOD» используется для определения типа таймера, «D3» идентифицирует устройство, а «20» это длительность периода.

Таймер Интервалов

Последний тип таймера - таймер интервалов. Он работает подобно мигалке, назначая своей переменной, значения TRUE и FALSE, поочерёдно и непрерывно, пока на вводе значение TRUE. Объявляется этот таймер аналогично таймерам ONDELAY и OFFDELAY, но требует указания второго периода времени. Это может выглядеть следующим образом:

```
TIMER( INTERVAL, D7, 30, 40 ) = JS2.B4;
```

Первые два параметра как обычно определяют тип таймера и переменную. Вторые два параметра определяют время для статусов в значениях TRUE и FALSE соответственно. Таймер принимает значение TRUE, когда ввод инициирует TRUE, затем начинает отсчёт задержки для первого периода времени, после чего назначает значение FALSE, отсчитывает задержку для второго периода времени, и снова принимает значение TRUE, повторяя цикл. Это продолжается, пока нажата JS2.B4.

Этот тип таймера полезен для реализации целого ряда различных функций. Одним из примеров может быть функция «Плавного Триммирования»:

```
SCRIPT
    TIMER( INTERVAL, D1, 20, 100 ) = JS1.B2;
    TIMER( INTERVAL, D2, 20, 100 ) = JS1.B3;
    CMS.B1 = D1;
    CMS.B2 = D2;
ENDSCRIPT
```

Такой скрипт генерирует импульс на D1 примерно каждые 5 секунд, пока нажата JS1.B2 и примерно каждые 5 секунд импульс на D2, пока нажата JS1.B3. Если после этого CMS.B1 назначена как «Триммер Руля Высоты вверх» (на кабрирование), а CMS.B2 как «Триммер РВ вниз» (на пикирование) в ГИП, то, удерживание одной из этих кнопок приведет к довольно медленному триммированию самолета в том или ином направлении по тангажу.

Переключатели

Переключатель меняет свое состояние с TRUE на FALSE или с FALSE на TRUE, если ввод меняет своё значение с FALSE на TRUE. Это полезно при создании функций вида: «Нажал – Команда1, Отжал-Команда2». Объявляется примерно так:

```
TOGGLE( D3 ) = JS1.B2;
```

При каждом нажатии JS1.B2, переменная D3 будет менять свое состояние.

Импульсные Устройства

Шестой тип устройства - Импульсное Устройство. Оно устанавливает своей переменной значение TRUE, когда его ввод меняется с FALSE на TRUE. Переменная Устройства остается в состоянии TRUE на протяжении одного полного цикла по ходу сценария, во время чего снова устанавливает себе значение FALSE, независимо от того, какое значение на вводе. Ввод должен перейти из FALSE снова в TRUE, чтобы сгенерировать другой импульс. Таким образом, импульс длится ровно один цикл по CMS сценарию. Импульсные Устройства полезны, когда другим частям сценария необходимо знать, какое событие произошло, но выдерживание переменной устройства в значении TRUE на протяжении нескольких сканирований, может создать проблемы.

Объявить импульсное устройство, можно следующим образом:

```
PULSE( D2 ) = B1;
```

В этом примере, D2 (переменная устройства) приобретает значение TRUE в том цикле скрипта, где «B1» впервые была замечена в значении TRUE. В следующем цикле сценария, она снова приобретает значение FALSE и остаётся таковой, пока «B1» сама не приобретёт значение FALSE, а затем снова TRUE. Импульс генерируется только при переходе переменной (B1) от FALSE к TRUE.

Импульсные Устройства не предназначены непосредственно для ввода символов. Они, как правило, слишком коротки, чтобы их мог правильно обработать символьный процессор.

Чтобы растянуть импульс так, чтобы он стал виден, вы можете использовать таймер OFFDELAY:

```
SCRIPT
    PULSE( D2 ) = B1;
    TIMER( OFFDELAY, D3, 10 ) = D2;
    CMS.B1 = D3;
ENDSCRIPT
```

В этом случае, короткий импульс от D2 запускает таймер OFFDELAY для переменной D3. D3 сразу же приобретает значение TRUE, и когда при последующем сканировании D2 переключится в FALSE, D3 практически мгновенно начинает отсчёт. Таймер OFFDELAY привязан к «частоте повторения символов», таким образом, продолжается достаточно долго, чтобы быть распознаваемым символьным процессором. CMS.B1 может быть запрограммирована в ГИП выводить необходимые символы. Если действия пропускаются, попробуйте увеличить «Частоту Повторения Символов» в Программных Настройках (вкладка «Program Settings») или увеличить значение задержки в самом предложении Таймера OFFDELAY.

SCALE (Масштабирование)

функция SCALE (Масштаб) используется, чтобы установить параметры масштабирования «на лету» для Устройства CMS и Фактических Устройств. Она заменяет любой масштабный параметр, установленный в ГИП, кроме параметра «centered» (отцентрировано). Это включает в себя мертвые зоны, чувствительность, настройку кривых отклика и т.д., обычно устанавливается в разделе «Axis» (Оси) устройства. Результат функции SCALE остается в силе до тех пор, пока не начинает выполняться другая функция SCALE.

Объявить функцию SCALE можно так:

```
SCALE( CMS.A1, 100, 5, GAIN6 );
```

Первый параметр указывает на то, к какой оси применить параметр масштабирования, в данном случае это CMS.A1. Второй параметр - значение чувствительности, он может принимать значения от -100 до 100. Это процентные значения. Третий параметр является значением «Мёртвой Зоны», также в процентном выражении. Последний параметр - значение кривой отклика (усилие) и может быть каким угодно в интервале от GAIN1 до GAIN11. Что соответствуют 11-ти настройкам кривых, которые могут быть установлены в ГИП. GAIN6 «прямая», это та линия, которая проходит от нижнего левого в правый верхний углы через поле настройки кривых, отображаемое в графическом интерфейсе. С GAIN1 по GAIN5, имеют меньшую чувствительность к центру и большую к краям. С GAIN7 по GAIN11 напротив, более чувствительны к центру и менее к краям. Масштабирование влияет на отклик во всех трёх режимах в Программном (многорежимном) Режиме. Если необходимо получить особые, независимые от действий

устройства (наиболее распространенный случай), настройки реагирования органов управления, параметры могут быть просто выставлены в ГИП отдельно.

Применять SCALE нужно, только если вы хотите применить изменения управления джойстиком «на лету», например, вам нужно нормальное управление на маршруте, а на посадке более чувствительное. Пример этого, смотрите в разделе [Примеры Сценариев](#).

SELECT (Выбор)

Функция SELECT (выбор) используется для выполнения одной конкретной из нескольких различных функций, основанных на текущем значении аналоговой переменной. Это самая сложная функция в наборе команд, но при этом очень мощная. В основном она принимает значение аналогового ввода, определяет в каком диапазоне это значение, и затем выполняет инструкции на основании этой зоны. Те, у кого есть опыт в программировании, узнают в ней модифицированную форму операторов «case» (ситуация) или «switch» (переключатель), которые применяются в большинстве языков программирования. Обратите внимание, что функция SELECT, не может быть использована в пределах блока SEQUENCE (последовательность).

```
SELECT( INPUTVALUE, TYPE ) OF
// В определённой ситуации переместиться сюда
ENDSELECT
```

«INPUTVALUE» это аналоговое значение, которое управляет блоком и может быть единой переменной или математическим выражением. «TYPE» (тип) может быть положением (POSITION) или диапазоном (RANGE) и управляет работой предложения SELECT.

В общем, если установлен тип «позиция» (POSITION), предложение SELECT будет реагировать на изменения в пределах текущей зоны, просто делая то, что требуется для выхода из текущей зоны, а затем делать все, что требуется, чтобы войти в новую зону.

Если установлен тип «диапазон» (RANGE), SELECT будет реагировать иначе, в той же ситуации, когда происходит перемещение из одной зоны в другую, он будет проходить все зоны между ними. Например, если ввод был внезапно изменён с зоны 1 на зону 3, тип POSITION просто выйдет из зоны 1 и войдёт в зону 3, а тип RANGE выйдет из зоны 1, войдёт в зону 2, выйдет из зоны 2 и войдёт в зону 3.

Конечно, тип POSITION будет протекать относительно медленно, выполняя точно то же действие, поскольку он будет проходить через зону 3, бывают моменты в игре, где управление перемещается так быстро, что зона 2 может быть не замечена управлением «Позиции».

В конце предложения слово “OF”. За которым следует серия предложений “CASE” (ситуация), каждое из которых определяет пределы зоны и меры, которые необходимо принять, при входе в эту зону или выходе из неё. Эти предложения начинаются со слова “CASE”, после чего указывается положительная цифра константы и двоеточие. Начало действия всегда объявляется. Окончание действия также может включаться путем отделения начала и конца действия словом «EXITCASE», хотя это совершенно не

обязательно. В конце каждая ситуация, отмечается предложением «BREAK» (прекратить).

Блок SELECT может выглядеть так:

```
SELECT( JS1.A1, RANGE ) OF
    CASE 0:
        B1 = TRUE;
    EXITCASE:
        B1 = FALSE;
    BREAK;

    CASE 64:
        B2 = TRUE;
    EXITCASE:
        B2 = FALSE;
    BREAK;

    CASE 128:
        B3 = TRUE;
    EXITCASE:
        B3 = FALSE;
    BREAK;

    CASE 192:
        B4 = TRUE;
    EXITCASE:
        B4 = FALSE;
    BREAK;

ENDSELECT
```

Здесь ось JS1.A1 будет поделена на 4 зоны. Ось JS1.A1 имеет стандартный диапазон (0 ... 255) поэтому будет разделена на равные четверти, разделённые значениями 64, 128 и 192. Так как управление будет проходить через этот диапазон, B1 включится в нижней четверти, затем включится B2, B3, и B4 наконец, когда управление будет перемещено из его нижней точки в высочайшую, в последней четверти диапазона, где и находится B4.

Обратите внимание, что обозначенные зоны проходят от одного значения к следующему, т.е., значение, следующее после «CASE», на самом деле - нижний предел зоны, который идёт к следующему значению минус 1. Значения в верхней зоне могут быть любыми выше своего минимального значения. Как в рассмотренном выше примере, любое значение JS1.A1 выше 191 считается принадлежащим четвертой зоне.

Если значение в нижней зоне CASE, больше нуля, то существует еще одна «зона бездействия» которая отсчитывается от нуля на единицу меньше чем минимальное значение CASE. Если входное значение меньше нижней зоны, то ничего не происходит

или производится выход из этой зоны. Каждой ситуации CASE последует EXITCASE:. Код между предложением CASE и EXITCASE: будет выполняться, когда значение попадает в эту зону. Код между EXITCASE: и BREAK будет выполняться, когда аналоговое значение покинет эту обозначенную зону. CMS обеспечивает синхронизацию с символьным процессором во время смены зон, так, что символ будет передан оператору EXITCASE, прежде оператора CASE для новой зоны.

Как упоминалось выше, EXITCASE не является обязательным. Ситуация CASE может быть заявлена так:

```
CASE 100:
B1 = TRUE;
BREAK
```

В этом случае, когда входная переменная в первый раз превысит 100, B1 будет задана. Некоторые условия необходимо будет обеспечить, чтобы B1 в итоге стала FALSE, например, определить кнопку. В противном случае она просто останется заданной постоянно.

Предложения с ситуациями (CASE)

Только несколько видов предложений допускаются в пределах CASE, во избежание некоторых проблем, которые могут произойти. Допустимые предложения включают IF/THEN, IF/THEN/ELSE, а также SCALE, а как основной оператор (“=”). Последовательности не допускают назначений устройствам, хотя ссылки на Устройства вполне приемлемы.

Однако можно контролировать это посредством блока SELECT. Например, следующим образом, можно позволить последовательности выполняться, если js1.a1 в значении больше, чем 128 (центр):

```
SCRIPT
// Сначала, задаём B1 статус TRUE в любом месте выше центра.
//
SELECT( JS1.A1, POSITION ) OF
CASE 0:
BREAK;
CASE 128:
B1 = TRUE;
EXITCASE:
B1 = FALSE;
BREAK;
ENDSELECT

// Теперь запускаем последовательность на основе B1
// (сейчас мы вне блока SELECT, поэтому делаем так)
//
```

```
SEQUENCE
WAIT( B1 ); // Ждём B1
CMS.B1 = TRUE; // Включаем CMS.B1
DELAY( 20 ); // Оставляем примерно на секунду
CMS.B1 = FALSE; // Отключаем
ENDSEQUENCE
ENDSCRIPT
```

Это лишь малая толика того, что можно сделать с помощью этой функции, но это должно дать вам понятие того, что можно делать и как это строится.

Установка Режимов в Карте

Вы можете установить текущий режим посредством сценария. Для этого нужно установить переменную CURRENTMODE в нужный режим:

```
CURRENTMODE = MODE2;
```

Это позволяет сценарию управлять режимами, когда сам контроллер не имеет возможности переключения Режимов, или тогда, когда требуется использовать кнопку, отличную от predetermined кнопки переключения Режимов на устройствах, которые имеют возможность переключения Режимов. Обратите внимание, что индикаторы на устройствах с поддержкой режимов не будут отражать текущий режим, если он был установлен с помощью CMS. Индикаторы работают только при смене режимов специальной кнопкой на устройствах.

Допустимы режимы: MODE1, MODE2, MODE3, и MODE4. Вы также можете использовать числовые значения 0, 1, 2, и 3. Это позволяет сделать программирование циклического переключения режимов немного легче. Пример того, как это делается см. в разделе «Примеры Сценариев».

При создании карты, в которой для управления режимами применяется CMS, нужно зайти на вкладку “Программные Настройки” в ГИП Диспетчера и установить в “Mode Control” (Управление Режимом) - “CMS”. Это добавит вкладки режимов 2, 3, и 4 в диалоговой области, но не даст устройствам FighterStick или ProThrottle устанавливать режимы.

Для разъяснения того, что каждый режим из себя представляет, см. раздел Режимы.

Сценарии с Трекболом

Трекбол ассоциируется сценарием CMS, как Джойстик. Это с JS1.A1 по JS1.A4 и с JS1.B1 по JS1.B4 (если это левая вкладка после вкладки Программные Настройки). JS1.A3 и JS1.A4 являются псевдоосями, которые доступны только посредством CMS. Они предоставляют относительные значения движений трекбола. Если вы ссылаетесь на JS1.A1 и JS1.A2, вы запрашиваете значения джойстика. При назначении трекбола в

сценарии CMS, нельзя определить, какую ось вы хотите использовать, это позволяет сделать двойной набор осей.

Оси настроены так, чтобы пройти по CMS без изменения значений.

Если вы напишите что-то вроде:

```
CMS.A1 = JS1.A3;
```

```
CMS.A2 = JS1.A4;
```

а затем и в ГИП назначите CMS.A1 и CMS.A2 как оси мыши X и Y, это будет работать так же, как если бы вы назначили трекболу оси мышь X и Y, непосредственно в ГИП.

Значения для трекбола задаются так же, как и всем другим осям, поскольку они используются в CMS. В частности, значение 128 рассматривается как “центр” и поэтому не вызывает движение курсора мыши. Значения ниже 128 - налево и вверх, значения выше 128 - направо и вниз. Максимальное значение равно 255.

Используя CMS, вы можете использовать любой из этих типов данных при ссылке на трекбол. Это будет зависеть от того, куда, в конечном счете, будут направлены данные, в Устройство Диспетчера или модуль мыши. CMS позволяет делать это, фактически распознавая все 4 оси. Оси A1 и A2, действуют как оси джойстика X и Y. Оси A3 и A4 передают данные перемещения, такие как от мыши. Вы можете использовать любой набор значений (или сразу оба, если придумаете для этого причину). Обычно, используют A1 и A2, направляя данные в Устройство Диспетчера и A3 и A4, если обращаются к мыши. Если вы назначаете оси трекбола непосредственно в графическом интерфейсе, Диспетчер определяет типы данных (перемещение или положение), основываясь на том, что вы назначили, Устройство Диспетчера или мышь.

Когда данные отправляются в Устройство CMS, которое в итоге отправляет их на модуль мыши, бывает случаются некоторые проблемы. CMS обращается с вещами по разному в зависимости от типа данных, относительных (как у мыши) или абсолютных (как у джойстика). К сожалению, CMS не может понять это самостоятельно, потому что он не может знать, откуда пришли данные и куда они отправятся дальше.

Следовательно, есть три маркера, которые необходимо задать или удалить (например, динамически), если вы генерируете данные мыши в CMS. После будут определены специальные имена для этих маркеров в «CM3 Final» (Финальная версия), но сейчас используются маркеры B1022 (Z), B1023 (Y), и B1024 (X). Вы должны установить их значения как «верные» (TRUE), если вы передаёте относительные данные мыши посредством Устройства CMS. Если вы передаёте данные Джойстика, то маркеры должны быть установлены как FALSE. Как бы то ни было, это данные по умолчанию, так что если у вас в карте нет Трекбола, который можно было бы промаркировать как отправляющий относительные данные, вы можете просто проигнорировать B1022..B1024. Ваша карта должна работать нормально.

Советы при Программировании

Этот раздел предоставляет несколько советов при работе над сценариями. Их не так много, но они могут сделать создание и тестирование карт проще и помогут вам справиться с некоторыми проблемами, которые могут возникать по ходу.

Совет № 1 - Старайтесь обходиться одной функцией за раз.

Старайтесь создавать сценарий одной функцией за раз и работать в минимальной карте, чтобы видеть как работает ваша Функция. Можно даже создать временную “рабочую карту”, которая будет просто вспомогательной, где вы будете испытывать все особенности функции над которой работаете в данный момент и в которой будут только заданные в ГИП назначения, которые нужны для этой функции. Это сэкономит много времени при загрузке, если в карте не все строки и назначения, которые будут использоваться в конечном варианте, а так же меньший код сценария легче отлаживается. Как только функция будет работать так, как вам надо, вы сможете вырезать и вставить её в реальную карту для окончательного тестирования.

Совет № 2 - Не забывайте использовать кнопку «Проверить Сценарий» (Script Check)

Возьмите за привычку использовать кнопку **Script Check** («Проверить Сценарий») в Редакторе CMS, прежде чем загружать карту. Это быстро обнаружит опечатки и упущенные операторы, и поможет сохранить много времени и избежать проблем при переходах туда-сюда между редактированием и загрузкой карт.

Совет № 3 – Сохраняйте Карту перед запуском

Это крайне важно во время отладки. Когда вы запускаете карту и начинают генерироваться символы или случается сбой, это может в конечном итоге изменить карту. Сначала сохраните её на диск. Тогда, если что-то пойдет не так, вы не потеряете результаты вашей работы.

Совет № 4 - Старайтесь использовать «видимые» символы.

Когда вы изначально создаёте функцию, старайтесь избегать использования “невидимых” символов, таких как Функциональные Клавиши (F1, F2 и т.д.), и заменять их чем-то видимым, как например “a” или “b”. Если сценарий работает не так, как вам надо, стандартные клавиши, менее вероятно, будут активировать некоторые меню или функции Windows. Это также очень удобно, чтобы иметь возможность просмотра символов в «Блокноте» или других текстовых редакторах. Хотя утилита «Keytest» (Проверка Клавиш) покажет вам, нажатие и отжатие любой клавиши, она может и не учесть того, что на самом деле видит сама Windows. Просмотр результатов в «Блокноте» может позволить вам видеть такие вещи, как системные автоповторы и дать вам более полное представление о том, как вывод влияет на игру. Как только, покажется, что всё протекает правильно, вы сможете вернуться назад и заменить символы на те, которые там действительно должны быть.

Совет № 5 – Используйте утилиту Тестирования и Калибровки «Test/Calibrate» для начального тестирования.

Проводите начальную проверку используя экран тестирования/калибровки и утилиту Keytest (тест клавиш). Так вы сможете обнаружить проблемы, просто наблюдая за Осями, Кнопками и символами, которые генерирует сценарий. Это также намного быстрее, чем выходить из Диспетчера и запускать игру, что бы проверить работу сценария. В конце концов, всё, конечно, будет испытано в игре, но вы можете свести к минимуму, переходы “туда - обратно” между Диспетчером и игрой, наблюдая за происходящим в ГИП и видя, на что похоже, то что он делает, прежде чем вы на самом деле попробуете это. Если повезет, вам останется только настроить некоторые вещи, прежде чем приступить к игре.

Совет № 6 - Отлаживайте Сценарии

Если при запуске сценария, выполняется не то, что вы планировали, имейте в виду первое правило отладки - “делать лишь то, что вы сказали делать”. CMS на самом деле не может “думать”, он просто следует инструкциям, которые вы задали в сценарии. Понаблюдайте за тем, что он делает и посмотрите на сценарий, постарайтесь найти, что могло стать причиной наблюдаемого поведения. Обычно не трудно найти часть скрипта, которая подскажет, где находится то, что вам нужно, и откуда начинать искать.

Совет № 7 - Решение проблемных-скриптов, которые уже запущены.

В CMS, возможны случаи сценариев, которые могут вызвать проблемы при активации карты. Обычно в этом нет ничего серьезнее залипания клавиши из-за сбоя логики, которая бы её отключила, но и это иногда может доставлять хлопоты. Вот несколько советов, которые могут пригодиться, если такое происходит:

1. Если вы используете отладчик в ГИП, обычно достаточно просто нажать на кнопку «Dirrect Mode» (Прямой Режим) и отключить карту.
2. Если вы знаете, какая клавиша залипла, можно попробовать остановить её, нажав ту же клавишу на клавиатуре. Это сгенерирует “прерывание” клавиши и остановит застрявшую клавишу в сценарии.
3. Если клавиша является той, которую перехватывает и обрабатывает Windows, то это может быть немного более сложным. Если можно использовать мышь, то закрытие карты и переход в Прямой Режим должно сработать. Если клавиша оказывается той, что используется системой для выполнения конкретных функций, это может быть проблемой. Залипшая клавиша TAB, например, может заставить курсор мыши непрерывно бегать по рабочему столу, что делает мышь довольно малоприспособной для использования. В таких случаях важно помнить, что отключение любого из устройств USB CH, переводит Диспетчер обратно в прямой режим. Просто отключите то устройство, которое нужно исправить. Вы сможете подключить его обратно после очередной смены режима.

С такими ситуациями не трудно справляться. Основная идея состоит в том, чтобы тем или иным способом перевести Диспетчера в прямой режим.

Совет № 8 - Как избежать проблем возникающих при запуске.

Если при запуске Windows у вас запускается утилита «CMStart», будьте осторожны, чтобы не оставить проблемную карту загруженной в конце сеанса отладки. Когда Windows перезагрузится, карта будет включена и начнёт работать. Вместе с этим начнут возникать все те проблемы, которые могли быть в карте. Вы можете отключить CMStart, или просто загрузить правильно работающую карту, прежде чем завершить работу.

Примеры сценариев

Этот раздел включает в себя несколько примеров сценариев, которые иллюстрируют, что можно делать с помощью Диспетчера и CMS, а так же методы, которые при этом используются. Эта информация отнюдь не исчерпывающая, что в принципе невозможно, но надеемся, даст вам некоторые представления о том, как создавать ваши собственные CMS функции.

Некоторые из примеров в данном разделе, очень длинные из-за комментариев. Для максимального удобства чтения Вам, вероятно, необходимо расширить окно справки (данного руководства), чтобы предотвратить перескакивания строк на следующие, что может запутать чтение.

Пример # 1 - Объединение Осей

Этот пример показывает, как можно использовать CMS для объединения осей и формирования их в новые оси, которые будут более полезными в конкретной ситуации. В этом случае проблема будет заключаться в объединении тормозов в одну Y-Ось. Эта функция иногда полезна в симуляторах автогонок, которые не предусматривают отдельных педалей газа и тормоза, а полагаются на то, что педаль будет использовать половину оси для газа, а другую половину для тормоза, а в центральной позиции, по существу, не имея ни газа не тормоза.

Перед началом сценария, мы должны обдумать проблему, рассмотреть факты, а затем выяснить, как мы собираемся эту проблему решить.

Первый полезный факт в том, что все оси имеют от 0 до 255 отсчётов для полного хода. Это относится к осям, которые генерируют реальные устройства, а также значениям, которые должны быть отправлены в Windows с помощью Устройства CMS.

Второе, что мы должны знать, это то, что педаль левого тормоза это ось A1 на педалях ProPedals и правого тормоза – ось A2. С помощью утилиты «Test/Calibrate» легко определить значения и то, в каком направлении эти оси работают, а ГИП предоставит вам имена устройств, если это что-то другое, а не просто кнопка.

Наконец, мы должны знать, что нужно в этом случае симулятору. Обычно это 0 для полного газа, 255 для полного тормоза, и 128 для нейтральной позиции, но может быть и по-другому.

Когда собрано всё необходимое, у нас есть две оси педалей, обе из которых имеют значения 0, когда они в свободном положении (отжаты) и 255, когда они полностью нажаты. Идея заключается в том, чтобы сгенерировать одну ось, которая работает от

128 (эффективное значение центра) до 0, когда одна педаль нажата, и от 128 до 255, когда нажата вторая. В действительности это всего лишь простая арифметика, ей и займёмся.

Сценарий.

Для этого примера, мы будем полагать, что на первой вкладке устройств в ГИП у нас есть штурвал, который мы будем использовать в качестве рулевого колеса, это будет JS1. Также мы предполагаем, что у нас есть набор педалей «ProPedals», который находится на второй вкладке устройств в ГИП и это JS2. Если у вас устройства расположены по-другому, то эти ссылки нужно поменять. Скрипт будет выглядеть следующим образом:

```
SCRIPT
CMS.A1 = 128 + (JS2.A2/2) - (JS2.A1/2);
ENDSCRIPT
```

всего лишь одна строка кода. Она начинается с центра в значении 128, прибавляет половину значений Y-оси и вычитает половину значений X-оси. Таким образом, нажав правую педаль (A2), начальное значение увеличится со 128-ми до 255, при нажатии левой педали (A1) уменьшится к начальному значению – «0», что как раз то, что мы хотели. Для завершения карты, оси должны быть назначены в ГИП. Ось X Штурвала, будет назначена как ось X на Устройстве Диспетчера 1. Ось Y Штурвала, будет назначена «None», поскольку она не будет использоваться. Ось 1 на Устройстве CMS будет назначен на ось Y на Устройства Диспетчера 1.

Пример # 2 - Управление Режимami Карты

Как было отмечено в разделе о Настройке Режимов, можно использовать CMS для управления Режимami карты используя переменную CURRENTMODE. В этом примере мы будем реализовывать циклическое переключение режимов, подобное тому, которое выполняется аппаратно, переключателем режимов на устройствах «FighterStick» и «ProThrottle».

Единственный интересующий нас реальный факт в том, что для переключения между тремя режимами, нам нужно установить переменную CURRENTMODE со значения 0 в значение 1, затем 2, а затем обратно в 0 (ноль). В этом примере мы используем одну из внутренних аналоговых переменных, A1, чтобы отслеживать, какой режим мы собираемся установить. Эта переменная будет увеличиваться на единицу каждый раз, когда нажата кнопка, а когда она превысит 3, мы сбросим её на ноль. Последовательность, вероятно, самый простой способ для осуществления этого, поскольку мы можем использовать функцию WAIT () для остановки последовательности, пока не будет нажата наша кнопка.

Сценарий

Для этой карты, мы будем считать, что у нас есть только джойстик «CombatStick» в карте, и мы хотим, что бы кнопкой переключающей режимы, была Кнопка 2 на этом устройстве. Сценарий будет выглядеть примерно так:

```

SCRIPT
SEQUENCE
WAIT( JS1.B2 ); // Ждать нажатия JS1.B2
A1 = A1 + 1; // Прибавить 1 к счётчику режимов
IF( [ A1 > 3 ] ) THEN // Если значение больше 3-х, то
A1 = 0; // Сбросить на ноль
ENDIF
CURRENTMODE = A1; // Теперь скопировать результат в
                // CURRENTMODE

ENDSEQUENCE
ENDSCRIPT

```

JS1.B2 будет увеличивать A1 на единицу каждый раз, когда она нажата. Когда A1 увеличится выше трех, она сбросится на ноль. Значения будут цикла 0, 1, 2, 3, 0, 1, 2, 3 и т. д. Таким образом, циклически перебирая четыре режима, как мы и хотели. Так как это не управляет осями или кнопками Устройств Диспетчера, то нет ничего, что должно быть назначено в ГИП, за исключением того, что кнопку 2 на джойстике «CombatStick», нужно установить на «None» (ничего), чтобы она не мешала работать остальной карте.

Другой метод будет управлять четырьмя режимами четырёх позиционной хаткой. Он имеет преимущество, в том, что все четыре режима доступны независимо, без необходимости переключения между другими 3-мя режимами, и мы можем знать, в каком режиме находимся по направлению, в которое мы передвинули хатку.

В этом примере мы предполагаем, что у нас есть «FighterStick» и мы хотим использовать «Hat 3» (Хатка3) для управления четырьмя режимами. Hat 3 генерирует кнопки с 13-ой по 16-ую. Сам скрипт прост, всего лишь набор вложенных IF/THEN/ELSE блоков, которые устанавливают переменную CURRENTMODE на основании текущей позиции «Hat 3». Полезно помнить при использовании хатки, что она может иметь только одну позицию активной одновременно, и нам не нужно делать какие-либо условия для ситуации, когда B13 и B14 были бы закрыты одновременно.

Сценарий может выглядеть следующим образом:

```

SCRIPT
IF( JS1.B13 ) THEN
CURRENTMODE = MODE1;
ELSE
IF( JS1.B14 ) THEN
CURRENTMODE = MODE2;
ELSE
IF( JS1.B15 ) THEN
CURRENTMODE = MODE3;
ELSE
IF( JS1.B16 ) THEN
CURRENTMODE = MODE4;

```

```
ENDIF  
ENDIF  
ENDIF  
ENDIF  
ENDSCRIPT
```

В ГИП, положения «Hat 3» должны быть выставлены на “None”, чтобы не выполнять никаких команд непосредственно.

Пример # 3 - Автоматическое Триммирование

В этом примере реализуется функция автоматического триммирования. На практике это будет выглядеть так - вы перемещаете джойстик, пока не полетите прямо и в горизонте, в этот момент вы нажимаете на кнопку, которая блокирует оси в этом положении, при нажатой кнопке вы возвращаете джойстик в нейтральное положение и отпускаете кнопку. Настройки триммеров будут удерживать органы управления в оттриммированном положении, пока джойстик в центральном положении и джойстик будет управлять органами управления начиная с текущей позиции, если джойстик будет смещаться от центра. Для сброса триммирования, вы оставляете джойстик в центральном положении и нажимаете кнопку еще раз.

Опять же, всё это всего лишь арифметика. Мы должны выяснить, какие значения в оттриммированном положении, чтобы использовать их при расчёте разницы между ними и центральным положением джойстика. Как только мы получим эту разницу, мы добавим её к фактическим значениям джойстика, чтобы получить значения оттриммированного положения.

Сценарий

Для этого примера, мы будем предполагать, что у нас есть единственный «CombatStick» в карте, это будет JS1, и мы хотим использовать Кнопку 2 на «CombatStick» как нашу кнопку «Триммера». В действительности сценарий - дело двух шагов. Для начала, вычисляем значения триммера, когда нажата Кнопка 2, затем применяем значения триммера, когда Кнопка 2 отпущена. Мы будем использовать внутренние переменные A1 и A2, чтобы проследить смещения относительно осей X и Y.

Смещение значений джойстика должно продолжаться постоянно, пока кнопка отпущена что бы, таким образом, каждое полученное значение обновлялось. С другой стороны, вычисление значений смещения нужно производить, только когда кнопка впервые будет нажата. IF/ENDIF блок является наиболее подходящим для первого случая, последовательность SEQUENCE для последнего. Сценарий будет выглядеть следующим образом:

```
SCRIPT  
IF( NOT JS1.B2 ) THEN // Если кнопка 2 нажата то  
CMS.A1 = JS1.A1 + A1; // Прибавить смещение X к X джойстика  
CMS.A2 = JS1.A2 + A2; // Прибавить смещение Y к Y джойстика  
ENDIF  
SEQUENCE
```



```

WAIT( JS1.B2 ); // Ждать нажатия Кнопки 2
A1 = JS1.A1 - 128; // Расчитать смещение по X и сохранить в A1
A2 = JS1.A2 - 128; // Расчитать смещение по X и сохранить в A2
ENDSEQUENCE
ENDSCRIPT

```

Должны быть сделаны несколько назначений в ГИП. Во-первых, X и Y оси «CombatStick» должны быть назначены на «None», чтобы не мешали. Во-вторых, CMS.A1 и CMS.A2 значения должны быть назначены X и Y осям на Устройство Диспетчера 1, чтобы оттриммированные значения были доступны Windows.

Пример # 4 – Использование предложения % DEFINE

Этот пример иллюстрирует некоторые возможности, которые можно получить с помощью директивы DEFINE%. Он обеспечивает по существу ту же функцию, что и для Автотриммирования выше, но довольно интенсивно используя директиву %DEFINE.

Сценарий

Скрипт на самом деле создает тот же самый эффект, что и в примере с Автоматическим Триммированием. Основное отличие заключается в применении директивы %DEFINE в верхней части сценария, и затем том, как делаются ссылки по имени. Выглядит это примерно так:

```

// Присвоение имён переменным для получения смещения триммеров
//
%DEFINE xTrimValue A1
%DEFINE yTrimValue A2

// Имена реальных осей джойстика
//
%DEFINE xStickValue JS1.A1
%DEFINE yStickValue JS1.A2

// Понятные Windows имена для осей ГИП
//
%DEFINE xCmsValue CMS.A1
%DEFINE yCmsValue CMS.A2

// Имя константы для значения центрального положения
//
%DEFINE centerStickValue 128

// Имена для некоторых битовых функций.
//
%DEFINE trimButtonPressed JS1.B2
%DEFINE trimButtonReleased NOT JS1.B2

```

```

// Определения произведены. Теперь скрипт.
//
SCRIPT

// Это триммирует значения джойстика и назначают их осям
// Устройства Диспетчера в то время когда кнопка отпущена.
// Это ситуация для нормального режима.
//
IF( trimButtonReleased ) THEN
xCmsValue = xStickValue + xTrimValue
yCmsValue = yStickValue + yTrimValue
ENDIF

// Здесь ожидается нажатие кнопки, затем записывается как
// далеко от центра находятся значения осей X и Y.
// Эти значения используются для триммерной коррекции
// которая будет произведена, когда будет отпущена кнопка.
//
SEQUENCE
WAIT( trimButtonPressed );
xTrimValue = xStickValue - stickCenterValue;
yTrimValue = yStickValue - stickCenterValue;
ENDSEQUENCE

ENDSCRIPT

```

В приведенном выше примере, пожалуй, явный перебор, зато он демонстрирует многие вещи, которые можно сделать. При использовании директивы `DEFINE%`, имейте в виду, что идея такова, чтобы сделать яснее то, что функции и переменные делают на самом деле. Это вполне может “заходить слишком далеко”, и доводить до того, что сценарию будет трудно следовать. Например, вы можете легко определить что-то вроде этого:

```

%DEFINE setXTrim A1 = JS1.A1 - 128
%DEFINE setYTrim A2 = JS1.A2 - 128
SEQUENCE
WAIT( trimButtonPressed ); // Ожидание нажатия
setXTrim; // Set the X trim value
setYTrim; // Set the Y trim value
ENDSEQUENCE

```

Скрипт, который бы получился в точности тот же, но то, как программа устанавливает значения триммера по X и Y получается скрыто и довольно трудно для понимания.

Пример # 5 – Ввод нескольких наборов символов

Бывают моменты, когда желательно, иметь возможность передать несколько команд нажатием одной кнопки. При каждом нажатии отправляя следующую строку в серии. Это не может быть сделано в ГИП, но это, безусловно, возможно при помощи CMS. Подход, в этом примере заключается в том, чтобы просто использовать несколько кнопок Устройства CMS и запрограммировать каждую на разную функцию с помощью графического интерфейса. Сценарий будет контролировать цикличность функций при нажатиях соответствующих кнопок.

Сценарий

Есть несколько способов сделать это. Самый простой, вероятно, использовать последовательность. Мы снова предположим, что у нас один «CombatStick» в карте - JS1 и мы хотим использовать Кнопку 2, как управляющую циклом. Также мы предполагаем, что существует четыре события, и мы хотим вызывать их кнопками с CMS.B1 по CMS.B4. Это будет выглядеть примерно так:

```
SCRIPT
  SEQUENCE
    WAIT(JS1.B2); // Ожидание первого нажатия
    CMS.B1 = TRUE; // Нажатие кнопки1 CMS
    WAIT(JS1.B2); // Ожидание следующего нажатия
    CMS.B1 = FALSE; // Отжатие Кнопки1 CMS
    CMS.B2 = TRUE; // Нажатие кнопки2 CMS
    WAIT(JS1.B2); // Ожидание следующего нажатия
    CMS.B2 = FALSE; // Отжатие Кнопки2 CMS
    CMS.B3 = TRUE; // Нажатие кнопки3 CMS
    WAIT(JS1.B2); // Ожидание следующего нажатия
    CMS.B3 = FALSE; // Отжатие Кнопки3 CMS
    CMS.B4 = TRUE; // Нажатие кнопки4 CMS
    WAIT(JS1.B2); // Ожидание следующего нажатия
    CMS.B4 = FALSE; // Отжатие Кнопки4 CMS
  ENDSEQUENCE
ENDSCRIPT
```

Другой метод может использовать функцию SELECT, чтобы выполнить тоже самое. SELECT будет находиться под контролем одной из внутренних аналоговых переменных, значение которой будет контролироваться JS1.B2. Сначала мы ждем нажатия кнопки, а затем устанавливаем новое значение. Это обрабатывается посредством предложения SELECT, в каждой ситуации включая один из выводов CMS.Bx и запуская функции. Это может выглядеть следующим образом:

SCRIPT

```
SEQUENCE
WAIT( JS1.B2 ); // Ждём нажатия Кнопки2
A1 = A1 + 1;     // Увеличиваем счётчик
IF( [ A1 > 3 ] ) THEN // Если A1 больше трёх, то
A1 = 0;         // Сброс на ноль
ENDIF
```

ENDSEQUENCE

SELECT(A1, POSITION) OF

CASE 0:

CMS.B1 = TRUE; //Задаём бит для первого действия

EXITCASE:

CMS.B1 = FALSE; //Сбрасываем бит для первого действия

BREAK;

CASE 1:

CMS.B2 = TRUE; //Задаём бит для второго действия

EXITCASE:

CMS.B2 = FALSE; //Сбрасываем бит для второго действия

BREAK;

CASE 2:

CMS.B3 = TRUE; //Задаём бит для третьего действия

EXITCASE:

CMS.B3 = FALSE; //Сбрасываем бит для третьего действия

BREAK;

CASE 3:

CMS.B4 = TRUE; //Задаём бит для четвёртого действия

EXITCASE:

CMS.B4 = FALSE; //Сбрасываем бит для четвёртого действия

BREAK;

ENDSELECT

ENDSCRIPT

В ГИП мы должны сделать необходимые для Программного режима назначения «кнопкам» с CMS.B1 до CMS.B4, для определения каждой из четырех функций, которые мы хотим активировать.

Пример # 7 - Функция Аналогового Триммирования

Функции аналогового триммирования руля высоты может быть полезной вещью, так как она обеспечивает более эффективное управление нежели обычные клавиши «Trim +» и «Trim-». Некоторые авиасимуляторы предоставляют такую функцию, которая может быть просто назначена на ось и использоваться непосредственно. Если этого нет, то CMS может помочь в создании такой функции. Опять же, просто арифметика. Нам необходимо взять неиспользуемую ось и использовать её значение как смещение от значения основного управления для оси которая нам нужна для триммирования.

Сценарий

В этом примере мы предполагаем, что у нас есть FighterStick как первое устройство (JS1) и ProThrottle на второй вкладке устройств (JS2). Так как мы используем ProThrottle для управления непосредственно тягой, мы можем использовать ось колеса управления тягой (A3) на FighterStick как нашу ось для триммирования. Мы хотим, чтобы (A2) триммировала руль высоты. Мы также ограничим ход триммера до 10% от полного хода. И мы будем использовать CMS.A1 как значение, обеспечивающее окончательную установку руля высоты.

Опять же, арифметика. Нам нужно, смещать значения триммера, значит, есть положительный и отрицательный диапазон, и 0 в центральном положении. Так как это реальная ось (JS1.A3) мы просто вычитаем 128 из её текущего значения, чтобы получить значение от -128 до +127. Затем делим на 10, чтобы получить 10% хода триммера, и добавляем это значение для JS1.A2. Наконец, мы проверяем, не вышел ли результат из диапазона и задаём ему ограничение, если вышел. Сценарий будет выглядеть примерно так:

```
SCRIPT
CMS.A1 = JS1.A2 + (( JS1.A3 - 128) / 10); // Вычисляем
// значение триммирования
IF( [ CMS.A1 > 255 ] ) THEN // Если результат слишком высок,
то
CMS.A1 = 255; // Задаём максимально возможный предел
ENDIF
IF( [ CMS.A1 < 0 ] ) THEN // Если результат слишком низок, то
CMS.A1 = 0; // Задаём минимально возможный предел
ENDIF
ENDSCRIPT
```

В ГИП, нам нужно будет назначить Ось 1 Устройства CMS на управление осью Y Устройства 1 Диспетчера. Ось Y FighterStick должна быть задана как “None”, чтобы отключить её.

Пример № 8 - Переключение между Трекболом и Миниджойстиком

Если вы используете трекбол в вашей карте, вы вероятно большую часть времени используете его вместо мыши. Хотя, иногда, бывает желательно, перейти на миниджойстик на ProThrottle или другие оси джойстика для управления мышью. Это невозможно средствами одного лишь ГИП, так как вы не можете задать два назначения DX устройству мыши. Однако, это может быть сделано с использованием сценария CMS. Этот пример показывает один из возможных методов. Он также демонстрирует, как использовать метки XRELATIVE, YRELATIVE, и ZRELATIVE.

Сценарий

Для этого примера, мы будем считать, что у нас есть FighterStick как Джойстик1, ProThrottle как Джойстик2, и TrackballPro как Джойстик3. Нам нужен способ определить, откуда принимать данные, из Трекбола или от миниджойстика. Мы можем сделать это, просто проверив, смещен ли миниджойстик относительно центра. Если это так, мы будем считать, что следует принять данные Миниджойстика для перемещения мыши. Если Миниджойстик по центру, мы просто непосредственно используем данные TrackballPro. Таким образом, переключатель должен быть более или менее невидимым. Нам также необходимо одновременно с этим переключать метки ?RELATIVE. Мы возьмём две более или менее независимые оси, чтобы оси реагировали независимо друг от друга. Сценарий может выглядеть следующим образом:

```
SCRIPT
// Задаём ?RELATIVE метку если оси миниджоя отцентрованы
// Задаём трекболу отключиться при смещении мниджоя
// относительно центра.
//
XRELATIVE = (JS2.A1 > 120) AND (JS2.A1 < 136));
YRELATIVE = (JS2.B1 > 120) AND (JS2.B1 < 136));
// Если X в центре, используем Трекбол.
// Иначе, миниджой.
//
IF( XRELATIVE ) THEN
CMS.A1 = JS3.A3;
ELSE
CMS.A1 = JS2.A1;
ENDIF
// Теперь тоже самое для Y.
//
IF( YRELATIVE ) THEN
CMS.A2 = JS3.A4;
ELSE
CMS.A2 = JS2.A2;
ENDIF
ENDSCRIPT
```

В ГИП, программируем CMS.A1 и CMS.A2 управлять осями X и Y мыши соответственно, и всё.

Пример # 9 - Переключение между Устройствами

Бывают случаи, когда желательно использовать несколько устройств и иметь возможность выбирать, какой из них будет активным. Такая ситуация может возникнуть, например, если мы хотим создать двойное управление в кабине, для схем: инструктор/студент или KBC/Второй пилот. Мы должны каким-то образом установить, какое устройство будет активным. Это легко сделать с помощью CMS.

Сценарий

В этом сценарии мы предполагаем, что у нас есть два Штурвала Yoke LE и два набора педалей ProPedals. В карте, они будут настроены, как: Штурвал Командира, на первой вкладке (JS1), педали Командира на второй вкладке (JS2), Штурвал второго пилота, на третьей вкладке (JS3), педали второго пилота, на четвёртой вкладке (JS4). Мы также предположим, что кнопка 1 на штурвале KBC, будет использоваться для переключения от одного Устройства к другому.

Сценарий довольно прост. Мы зададим условия так, чтобы внутренняя переменная B1 находилась под контролем того устройства которое в эксплуатации, затем мы создадим короткую последовательность для переключения B1 каждый раз, когда JS1.B1 будет нажата. Тогда, используя блок IF/THEN/ELSE на основании состояния B1, мы назначим то или иное устройство на оси CMS. Мы будем использовать только оси X, Y, ось тяги, и руля направления в данный момент, чтобы просто продемонстрировать метод. На практике, вероятно, нужно будет сделать аналогичные вещи с кнопками и тормозами. Кроме того, поскольку B1 изначально в значении FALSE, на месте KBC, B1 будет индексировать FALSE, а на месте второго пилота наоборот. Сценарий будет выглядеть следующим образом:

SCRIPT

SEQUENCE

```
WAIT( JS1.B1 ); // Ждём нажатия кнопки1 на штурвале Командира
B1 = NOT B1;    // Инвертируем статус переменной B1
ENDSEQUENCE
```

```
IF( B1 ) THEN // Если b1 TRUE, то применяем второму пилоту
CMS.A1 = JS3.A1; // Копируем ось X второго
CMS.A2 = JS3.A2; // Y ось,
CMS.A3 = JS3.A3; // Тягу,
CMS.A4 = JS4.A3; // и ось Руля Направления осям CMS.
ELSE // Иначе, применяем к KBC
CMS.A1 = JS1.A1; // Копируем ось X KBC,
CMS.A2 = JS1.A2; // Ось Y,
CMS.A3 = JS1.A3; // Тягу,
CMS.A4 = JS2.A3; // и ось Руля направления осям CMS.
ENDIF
```

ENDSCRIPT

В ГИП, назначить управление Устройства CMS на Устройство1 Диспетчера. CMS Ось 1 будет как ось X, CMS Ось 2 - ось Y, CMS ось 3 - Z-Axis, и CMS ось 4 - R. А также необходимо очистить назначение осей X, Y, Z, R и осей штурвала и педалей, чтобы предотвратить конфликты.

Пример # 10 - Изменение Чувствительности Управления в Полёте

В некоторых случаях может понадобиться, изменить чувствительность устройства во время использования в игре или симуляторе. Например, в боевом авиасимуляторе, вам нужно относительно нечувствительное управление в маршрутном полете, а затем переключиться на более чувствительное, для воздушного боя для более оперативного реагирования. Этот пример показывает, как этого можно добиться с помощью функции "SCALE".

Сценарий

Сценарий относительно прост. Нам нужен "переключатель", который указывает, какой из двух наборов параметров чувствительности активен. Эти данные затем можно использовать в блоке IF/THEN/ELSE для активации нужной управляемости джойстика. В этом примере, мы сделаем это посредством последовательности. Этот способ наиболее эффективен, так как последняя функция SCALE остается в силе до тех пор, пока не начнёт выполняться следующая функция SCALE. Использование Последовательности предопределяет, что код SCALE начнёт выполняться только при нажатии кнопки.

Для сценария, мы предположим, что у нас есть FighterStick на первой вкладке, как JS1. Мы собираемся использовать кнопку 4, как кнопку смены чувствительности управляемости. Если наш переключатель имеет значение TRUE, мы выбираем параметры меньшей чувствительности, если переключатель имеет значение FALSE, мы выбираем более чувствительные настройки. Отметим, что до первого нажатия кнопки, будут действовать все параметры, которые были установлены в ГИП, и что первое нажатие будет применять менее чувствительные параметры, как только B1 иницирует значение FALSE. Сценарий будет выглядеть так:

```
SCRIPT
SEQUENCE
WAIT( JS1.B4 ); // Ждём кнопку 4
B1 = NOT B1; // Инвертируем переключатель
IF( B1 ) THEN // Если переключатель TRUE то настраиваем так
SCALE( JS1.A1, 50, 5, GAIN3 ); // Sensitivity=50%, Deadzone=5%
SCALE( JS1.A2, 50, 5, GAIN3 ); // and Gain Curve=3
ELSE // Иначе настраиваем так
SCALE( JS1.A1, 100, 5, GAIN6 ); //Sensitivity=100%,Deadzone=5%
SCALE( JS1.A2, 100, 5, GAIN6 ); // and Gain Curve=6
ENDIF
ENDSEQUENCE
ENDSCRIPT
```


В ГИП, вообще ничего не надо делать. Функция SCALE воздействует на обычные оси X и Y устройства JS1, так что мы просто активируем реальное управление и нет необходимости в назначениях в CMS.

Пример # 11 - Создание повторяющихся последовательностей

Иногда вам может понадобиться создать повторяющуюся последовательность событий на определённый период времени. Примером может служить отстрел ловушек: ложных тепловых целей (ЛТЦ) и дипольных отражателей (ДО) в симуляторе реактивного истребителя. Предложение SEQUENCE, является очевидным решением в этом случае и оно очень похоже на Пример # 5, с тем лишь отличием, что кнопки будут использоваться больше для того, чтобы активировать функции, в то время, как инструкции DELAY будут использоваться для движения по последовательности.

Сценарий

Сценарий следующий. Мы будем считать, что у нас есть FighterStick и мы хотим использовать кнопку 2 для управления последовательностью. Кроме того, нам нужны две команды, одна для диполей и одна для тепловых ловушек. Мы хотим отстреливать их поочерёдно, пока нажата кнопка 2, с 5-секундными интервалами между ними. Так как мы хотим, чтобы действие продолжалось до тех пор, пока нажата кнопка 2, мы будем использовать инструкцию WHILE (), для управления. Мы также используем CMS.B1 для выброса ДО и CMS.B2 для ЛТЦ. Сценарий будет выглядеть следующим образом:

В ГИП, мы должны будем сделать назначения для кнопок 1 и 2, Устройства CMS, для ДО и ЛТЦ. Они должны быть запрограммированы с ведущим значением NULL, чтобы не давать им вводить повторяющиеся символы все 5 секунд каждый раз.

Пример # 12 - Использование блока SELECT

Это не пример для практического использования, он больше для демонстрации некоторых дополнительных функций, которые могут быть реализованы с использованием функции SELECT. Эта функция может быть использована для ряда довольно сложных функций, выходящих за пределы простого воздействия на текущее состояние оси. Этот пример иллюстрирует некоторые из таких возможностей.

Одной из наиболее интересных является возможность вводить или управлять одной группой операций при перемещении аналоговой оси в одну сторону, другой группой, при движении в противоположном направлении. Хотя это можно сделать программированием вида «вверх/вниз» в ГИП и вводя отдельные символы, управление более сложными функциями может быть куда сложнее.

Ключом к решению такой задачи станет отслеживание текущего положения с помощью аналоговой переменной. Затем, когда мы вставим новую “ситуацию” (CASE), мы можем проверить эту переменную, чтобы узнать какой была последняя ситуация и, таким образом, узнать увеличивается или уменьшается значение оси. Основная структура должна выглядеть следующим образом:

```

SCRIPT
    SELECT( JS1.A3, RANGE ) OF
        CASE 0:
            A2 = 0;    // Задаём Состояние 0, что бы было откуда
                       // двигаться к Case 64
            BREAK;

        CASE 64:
            IF( [ A2 EQ 0 ] ) THEN
                // Мы пришли от Case 0
            ELSE
                // Мы должны двигаться от Case 128
            ENDIF
            A2 = 1;    // Задаём Состояние 1
            BREAK;

        CASE 128:
            IF( [ A2 EQ 1 ] ) THEN
                // Мы пришли от Case 64
            ELSE
                // Мы должны двигаться от Case 192
            ENDIF
            A2 = 2;    // Задаём Состояние 2
            BREAK;

        CASE 192:
            A2 = 3;    // Задаём Состояние 3, чтобы было откуда
                       // двигаться послеCase128
            BREAK;
    ENDSELECT
ENDSCRIPT

```

Каждая ситуация выше, проверяет аналоговую переменную A2 и определяет, откуда она пришла. После чего регистрирует идентификатор собственного состояния в аналоговую переменную A2, так, чтобы следующее состояние могло знать, откуда пришло это.

Другой возможностью является то, что мы можем управлять некоторыми группами последовательностей на основе блока SELECT. Мы должны управлять ими извне с помощью внутренних маркеров, которые мы устанавливали в “CASE” части каждой ситуации, а затем очищать их в “EXITCASE” части.

Здесь кроется потенциальная проблема, дело в том, что мы, вероятно, не хотим, чтобы одна ситуация начиналась, пока последовательность инициированная предыдущей ситуацией, не была завершена. Это тоже не так уж трудно гарантировать. Для этого

определим управляющий маркер, там где блок SELECT действительно заметит изменение значения. Каждой последовательности можно задать этот маркер, чтобы заблокировать любые дальнейшие изменения значений в блоке SELECT, до соответствующего завершения последовательности. Сценарий для реализации этой логики может выглядеть примерно так:

```
SCRIPT
// Блокируем логику. Если B1 - FALSE, иначе копируем JS1.A3
// в значение SELECT.Если B1-TRUE, то пропускаем это, значение
// SELECT не изменится, и блок SELECT останется в том
// состоянии, что и в данный момент.
//
IF( NOT B1 ) THEN // Если B1-FALSE, то копируем
A1 = JS1.A3; // реальное значение в наше значение SELECT.
ENDIF;

SELECT( A1, RANGE ) OF
CASE 0:
B1 = TRUE; // Блокируем изменение значений в блоке SELECT
B2 = TRUE; // B2 будет управлять первой последовательностью
EXITCASE:
B2 = FALSE; // Очищаем маркер первой последовательности
BREAK;

CASE 64:
B1 = TRUE; // Блокируем изменение значений в блоке SELECT
B3 = TRUE; // B3 будет управлять второй последовательностью
EXITCASE:
B3 = FALSE; // Очищаем маркер второй последовательности
BREAK;

CASE 128:
B1 = TRUE; // Блокируем изменение значений в блоке SELECT
B4 = TRUE; // B4 будет управлять третьей последовательностью
EXITCASE:
B4 = FALSE; // Очищаем маркер третьей последовательности
BREAK;

CASE 192:
B1 = TRUE; // Блокируем изменение значений в блоке SELECT
B5 = TRUE; // B5 будет управлять четвёртой последовательностью
EXITCASE:
B5 = FALSE; // Очищаем маркер четвёртой последовательности
BREAK;

ENDSELECT
```

```
// Теперь мы можем создать 4 последовательности. Каждая из
// которых по завершению будет каждый раз очищать маркер B1,
// и блок SELECT сможет обновлять значение SELECT.
// Эти последовательности просто ждут соответствующий бит.
// Затем они выполняют задержку DELAY и отменяют блокировку
// обмена данными блока SELECT.
// Реально не делая ничего практического.
```

SEQUENCE

```
WAIT( B2 ); // Ждём маркер последовательности
DELAY( 10 ); // Здесь происходит какая-нибудь полезная функция
B1 = FALSE; // Отключаем блокировку выполнения SELECT
ENDSEQUENCE
```

SEQUENCE

```
WAIT( B3 ); // Ждём маркер последовательности
DELAY( 10 ); // Здесь происходит какая-нибудь полезная функция
B1 = FALSE; // Отключаем блокировку выполнения SELECT
ENDSEQUENCE
```

SEQUENCE

```
WAIT( B4 ); // Ждём маркер последовательности
DELAY( 10 ); // Здесь происходит какая-нибудь полезная функция
B1 = FALSE; // Отключаем блокировку выполнения SELECT
ENDSEQUENCE
```

SEQUENCE

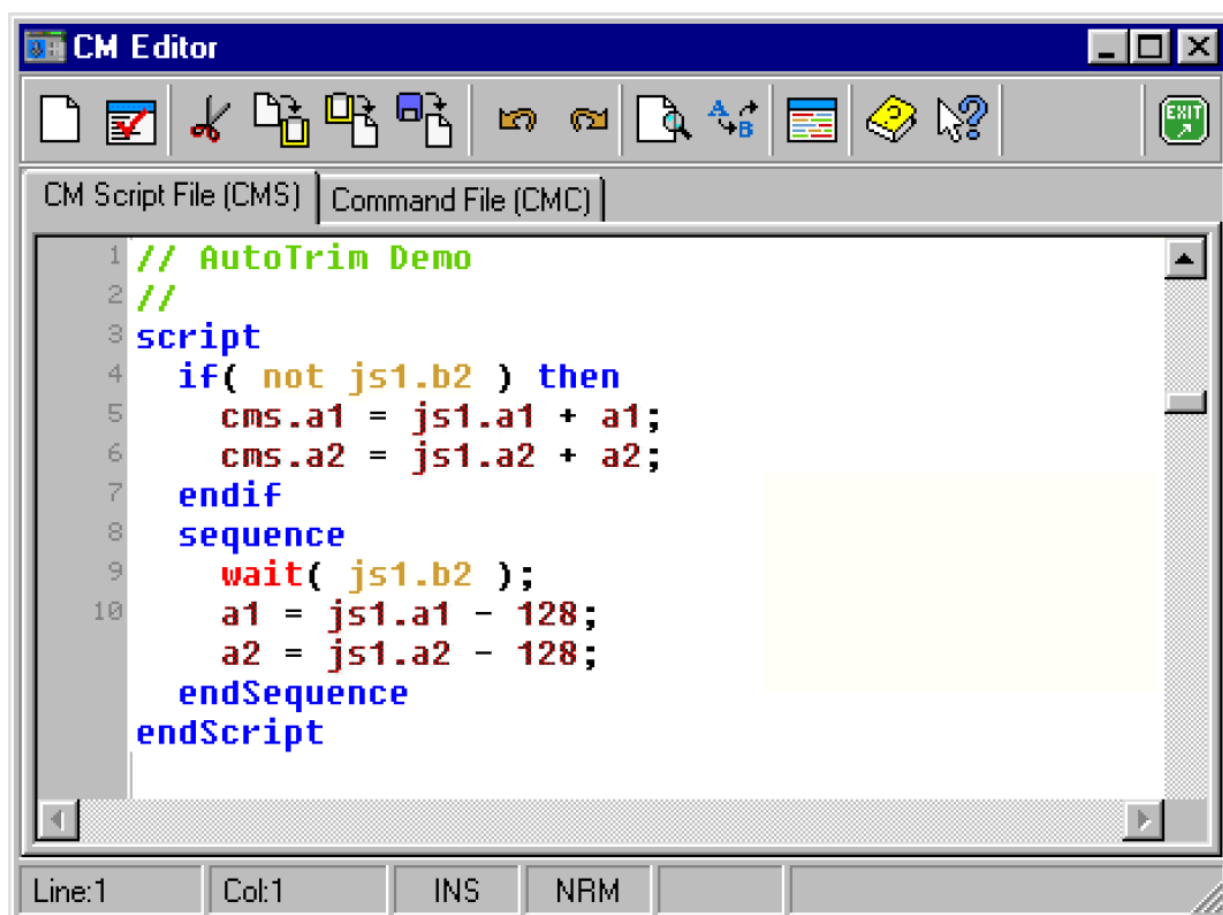
```
WAIT( B5 ); // Ждём маркер последовательности
DELAY( 10 ); // Здесь происходит какая-нибудь полезная функция
B1 = FALSE; // Отключаем блокировку выполнения SELECT
ENDSEQUENCE
```

ENDSCRIPT

Существует еще одна интересная возможность с использованием вышеприведённой логики. Вместо определения ввода в блоке SELECT на основе одной из реальных осей, мы могли бы просто определить ее на основе значения, которое было установлено в каждой из Последовательностей. В этом случае упорядочение различных состояний может полностью зависеть от, например, того каким является значение CURRENTMODE, в результате чего SELECT для перехода из одного состояния в любое из трех других будет зависеть от текущего Режима Карты. В действительности, любой набор условий может быть использован для определения того, каким может быть следующее состояние. Все, что вам нужно сделать, это создать правильное значение управления для предложения SELECT.

Редактор Диспетчера

Редактор используется для редактирования как сценариев (CMS) так и командных файлов (CMC). Редактор Диспетчера доступен посредством кнопки на панели инструментов ГИП. В работе, окно Редактора выглядит примерно так:



Панель Инструментов Редактора

В верхней части окна Редактора расположена панель инструментов, которая содержит кнопки, управляющие большей частью основных операций Редактора. Для более полного изучения этих кнопок и их функций, см. раздел «панель инструментов Редактора» ниже в этом руководстве.

Закладки Файлов

Тут же ниже, вы увидите две вкладки: «CM Script File (CMS)» (Файл Сценария CMS) и «Command File (CMC)» (Командный Файл CMC). Каждый конкретный сценарий может иметь любое сочетание этих вкладок. Если карта включает в себя файл CMS скрипта, то будет доступна вкладка CM Script File (CMS). Если карта ссылается на командный файл,

то появится вкладка Command File (CMC). Если Карта не содержит ничего, кроме самого редактора, то вкладок не будет вовсе.

Экран Редактора

Поле ниже вкладок, предназначено для ввода текста для CMS или CMC файлов. Принцип работы редактора для обоих типов файлов примерно одинаков, однако существуют некоторые различия. О них будет рассказано чуть позже, в разделах посвященных Редактированию CMS файлов и файлов CMC. В левой части окна редактора находится «Поле». По сути оно просто выводит номер строки для каждой линии. Вы можете отключить это поле с помощью кнопки «Параметры» описанной ниже.

Строка состояния

В самом низу экрана Редактора в строке состояния, есть пять показателей, которые индицируют текущее состояние. Эти показатели будут появляться, исчезать или изменяться в зависимости от состояния Редактора и различных других условий. Этими показателями являются:

LIN (Строка) и COL (Колонка)

Разделы Строка (LIN) и Колонка (COL), указывают текущие строку и колонку, в которой находится курсор. **Текст далее в процессе перевода.**