THRUSTMASTER

HOTAS COUGAR

СПРАВОЧНОЕ РУКОВОДСТВО

Приветствуем Вас!

Прежде всего, примите наши ПОЗДРАВЛЕНИЯ и БЛАГОДАРНОСТЬ за приобретение Thrustmaster HOTAS Cougar! Этот мощный и точный манипулятор, который Вы сейчас держите в руках – плод двух лет напряженной исследовательской и инженерной работы, главной целью которой было создание совершенного игрового контроллера для симуляционного жанра, способного удовлетворить самые притязательные ожидания любителей компьютерных игр. И вот он – гордый наследник серии FLCS и F-22 PRO! А ведь сделать так, чтобы этот наследник стал достойной сменой предыдущих легендарных моделей оказалось более чем трудной задачей – и даже это сказано недостаточно сильно.

В самом начале этого безумного проекта, перед компанией Thrustmaster встал выбор: просто усовершенствовать существующие модели, сохранив и расширив их возможности, или начать все с чистого листа и создать действительно новый продукт. Компания выбрала второй путь – самый тернистый. Почему? Да просто потому что каждый раз, когда выходил новый HOTAS от Thrustmaster, это всегда означало поднятие планки качества и функциональности на новый уровень по сравнению с существующими в тот момент времени стандартами. Вот и в случае с HOTAS Cougar – настало время изменить представление о том, каким должен быть джойстик высшего класса.

В результате бессонных ночей и жарких дискуссий появился этот в высшей степени реалистичный, мощный и гибкий манипулятор, с непревзойденным списком возможностей, включая сменные рукоятки, продвинутое (и в то же время подкупающе простое) программирование, безотказную систему plug'n'play, и все это упаковано в солидный корпус, скрывающий множество металлических деталей и узлов. Гордый владелец HOTAS Cougar может не сомневаться: это – на долгие годы!

Кугуар – это вещь для настоящих ценителей симуляционного жанра – поверьте, это отнюдь не дешевая легковесная поделка, оптимизированная исключительно под кошелек. Это – серьезная реплика органов управления истребителя F-16, созданная специально для тех, кто не приемлет компромиссов. Это действительно элитарный летный манипулятор класса high-end, который заставит и врагов, и конкурентов пожалеть, что они вообще поднялись в воздух, а не выбрали в свое время спокойную стезю наземного техника!

Конечно, этот зверь никогда не увидел бы свет без той огромной поддержки и ободрения со стороны сообщества – людей, с которыми мы встречались на выставках, переписывались по электронной почте, общались на форумах. Спасибо вам всем за то, что поддержали нас! И, конечно, нельзя не упомянуть тех, кто заслуживает нашей особой благодарности: прежде всего, команда из Thrustmaster/Guillemot, которая работала над этим

проектом; бета-тестеры, которые вылавливали баги и с риском для собственной жизни испытывали это чрезвычайно опасное оборудование… Спасибо также всем нашим друзьям и близким, которые нас терпели все время, пока мы работали над этим проектом!

Но – хватит эмоций, вернемся с небес на землю.

Если у вас есть опыт общения с предыдущими моделями самолетных манипуляторов Thrustmaster, то вы, будучи, безусловно, поражены новым HOTAS Cougar, почувствуете себя, тем не менее, как дома. Действительно, кое-какие черты очень похожи, но пусть это не сбивает вас с толку – начинка полностью новая. И вы, заядлые вирпилы, лучше чем кто бы то ни был, сможете оценить весь тот труд, который мы вложили в HOTAS Cougar.

В новой версии мы подвергли усовершенствованию все: механику, электронику, ПО. Мы сделали не просто более дружественную версию комплекта семилетней давности (что, в принципе, не является чем-то особенно трудным). Мы стремились дать любому пользователю – как прожженному ветерану, так и новичку – возможность в полной мере использовать всю скрытую мощь этого хищника. Семейство операционных систем Windows позволяет использовать возможности HID-совместимых игровых контроллеров – джойстиков plug'n'play, не требующих для работы какой-либо дополнительной конфигурации. Хотите? Извольте, HOTAS Cougar умеет работать именно так: подключите его к компьютеру, и наслаждайтесь любимыми играми … Очень просто.

Это все хорошо, скажете *вы,* но как насчет продвинутого программирования, ради которого покупают HOTAS Cougar? *Вы* ожидаете от *нас* намного большего, чем просто хороший манипулятор. Так знайте – и это есть! Помимо непревзойденной точности, HOTAS Cougar предлагает все программные возможности своего легендарного предшественника F-22… и намного больше, о чем вы и узнаете из этого Справочного руководства. Программные возможности, которые мы разберем далее, столь обширны и продвинуты, что они позволят вам оптимизировать все любимые игры, и даже изменить и дополнить возможности симулятора. Поэтому лучший путь узнать обо всех возможностях (и при этом уберечь себя от мигрени и ночных кошмаров в виде душащих вас бесконечных цепочек программного кода) – просто прочитать внимательно этот монументальный труд, а затем спокойно подумать: "Ну и что я со *всем этим* буду делать?"

Этот документ скорее заслуживает названия "Справочное руководство", в отличие от привычных "мануалов". Любой, кто видел или пользовался руководством пользователя F-16 FLCS, TQS и F-22 PRO, заметят и оценят отличия данного руководства. Здесь изложено буквально все, что нужно для настройки манипуляторов, использования Панели управления HOTAS Cougar и освоения программирования – от основ до самых продвинутых возможностей,

выделяющих Cougar среди остальных программируемых манипуляторов. Помимо данного Руководства, в помощь вам - подробные справочные файлы, мастера, учебники и прочие полезные и понятные утилиты, которые мы включили в пакет ПО. В общем, на наш взгляд, это самая полная документация, которая когда-либо прилагалась к игровому манипулятору.

Пусть у вас не будет никаких иллюзий по поводу HOTAS Cougar и этого Справочного руководства – вы приобрели серьезный, высококлассный манипулятор. Поэтому не исключено, что какие-то разделы Руководства вам придется перечитать не раз и не два. Но одна из причин, почему это Руководство столь объемно и фундаментально – отзывы пользователей предыдущих моделей манипуляторов TM, которые считали, что руководство было слишком кратким, и, как следствие, освоение джойстика – слишком трудным. Наше руководство, надеемся, будет читаться легко, так как вся информация подается в простой и понятной форме. Оно одинаково дружественно с теми, кому нужна только базовая информация, и внимательно к тем, кто хочет знать все до мельчайших деталей. Что бы вы не хотели получить от этого руководства – вы это получите. Итак, если вы впервые пользуетесь программируемыми манипуляторами Thrustmaster, мы настоятельно советуем вам самым внимательным образом прочитать первую часть Руководства, иначе вы не узнаете, как выжать все из HOTAS Cougar. Это очень гибкое программируемое устройство, которое позволяет вам достичь желаемых результатов самыми разными путями, но помните, что все так или иначе сводится к программированию, а в этой сфере чем вы логичнее и последовательнее, тем лучше.

Итак, без лишних проволочек – принимайте управление! Читайте, изучайте, научитесь выжимать максимум из вашего уникального манипулятора, и покажите всем супостатам, что такое HOTAS Cougar!

Испанская версия данного руководства доступна здесь:
http://www.escuadron111.com

Французская версия данного руководства доступна здесь:
http://www.checksix-fr.com/

Голландская версия данного руководства доступна здесь:
http://thrustmaster.vanree.net/

Немецкая версия данного руководства доступна здесь:
http://www.thrustmaster-x-files.de/

Русская версия данного руководства доступна здесь:
http://www.hotas.ru

ВЫРАЖЕНИЕ ПРИЗНАТЕЛЬНОСТИ

Мы выражаем нашу глубокую благодарность ниже перечисленным лицам и сайтам за их помощь и поддержку, оказанные в рамках этого проекта – спасибо вам всем! ☺

Бета-тестирование

Olivier "Red Dog" Beaumont
Robin "Emacs" Breyl
Jan-Albert "Anvil" van Ree
James "Nutty" Hallows

Компании и эскадрильи

Mark “Frugal” Bush & frugalsworld.com
Wingmen-alliance.com
Escuadron 111.com
Microsimulateur
SimHQ.com
Sim-arena.com
Aimsworth Coproration
Combatsim.com
Ubi Soft
Checksix-fr.com
Dogfighter.com
Desktopsims.com
Gamekult.com
Fast jet Flight Simulation (a.k.a. HAM technologies)

Особая благодарность

Len "Viking1" Hjalmarson
Matt Wagner
James R Campisi
Flavien "Vox" Duhamel
François Pimenta
David "Micro" Vely
Philippe "Twech" Dezeure
Philippe "Jag" Dubois
Lew/+Silat
Rob Coppock
Laurent Espinasse
Fernando Oscar Garcia Minguillán
Guillaume "Ghostrider" Houdayer
Олег Мэддокс
Jim Staud
Jean-Dominique "Bing" Belin
Emmanuel "Judy" Durant
Thomas "Doloop" Coulomb
Denis "Dugin" Blary
David "Zip" Pierron
Ulf Muckel
Hal Bryman
Jose "Oso" Benito
Станислав "huMMer" Вартанян

THRUSTMASTER

HOTAS COUGAR

ПАНЕЛЬ УПРАВЛЕНИЯ
РУКОВОДСТВО

ВВЕДЕНИЕ **1**

ПРОФИЛИ **9**

ПРОФИЛЬ ПО УМОЛЧАНИЮ **9**
СОХРАНЕНИЕ **10**
ЗАГРУЗКА **10**
УДАЛЕНИЕ **10**

РЕЖИМЫ РАБОТЫ ДЖОЙСТИКА **11**

НАСТРОЙКИ ОСЕЙ **11**

ПАРАМЕТРЫ ОСЕЙ **12**

КНОПКИ ПАРАМЕТРОВ ОСЕЙ **12**
Кнопка "Установить" 12
Кнопка "Вызвать" 12
ЗАКЛАДКА "НАСТРОЙКА ОСЕЙ" **13**
Изменение мэппинга осей 13
Инверсия направления оси 16
Блокировка оси 16
Изменение состояния осей в Windows 17
ОСИ В ФИЗИЧЕСКОЙ КОНФИГУРАЦИИ **18**
ОСИ, ВИДИМЫЕ ДЛЯ WINDOWS **19**
ЗАКЛАДКА КОНФИГУРАЦИИ ОСЕЙ **19**
Мертвые зоны 20
Центр калибровки 22
Триммер оси 22
Форма кривой 24
База кривой 25
ЗАКЛАДКА “НАЧАЛЬНАЯ ЗАГРУЗКА И КАЛИБРОВКА” **26**
Параметры начальной загрузки 26
Калибровка 27
Ручная калибровка 29

ДЕЙСТВИЯ И ДРУГИЕ ОПЦИИ **30**

ПЕРЕЗАГРУЗКА **30**
ЭМУЛЯЦИЯ ОСЕЙ И КНОПОК **30**
ЗАГРУЗИТЬ В УСТРОЙСТВО **30**
ОПРОС УСТРОЙСТВА **32**
СВЕРНУТЬ В ПАНЕЛЬ ЗАДАЧ **32**

Введение

Джойстик HOTAS Cougar практически не требует никаких драйверов или иных программ, работающих в фоновом режиме, для назначения различных настроек осей или эмуляции команд. По этой причине важно, чтобы джойстик сам знал обо всех изменениях, касающихся его осей. Внешние программы, такие, как утилита центровки джойстиков CTFJ Боба Черча (Bob Church), и даже утилита калибровки Windows, использоваться не должны, так как в противном случае многие продвинутые настройки осей могут выдавать неожиданные результаты.

ВНИМАНИЕ: НЕ ИСПОЛЬЗУЙТЕ ВСТРОЕННУЮ УТИЛИТУ КАЛИБРОВКИ WINDOWS ДЛЯ КАЛИБРОВКИ HOTAS COUGAR. ВМЕСТО ЭТОГО НЕОБХОДИМО ИСПОЛЬЗОВАТЬ ПАНЕЛЬ УПРАВЛЕНИЯ HOTAS COUGAR ДЛЯ РУЧНОЙ КАЛИБРОВКИ КОМПЛЕКТА.

Панель управления HOTAS Cougar предназначена для регулировки различных параметров выходных сигналов HOTAS Cougar, от мэппинга (назначения) осей до выбора режимов работы джойстика (режим эмуляции, режим Windows, и т. д.) Данное руководство содержит подробные поэтапные разъяснения пользования утилитой, а также описание работы джойстика с учетом вносимых пользователем изменений.

Для отображения текущих настроек джойстика, запустите Калибратор при подключенном устройстве; если все поля мэппинга пусты, джойстик не содержит никакой информации о калибровке.

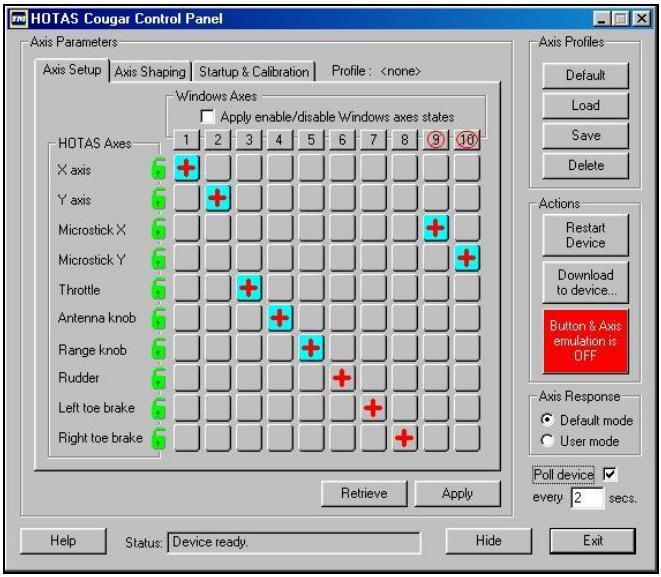


Рис. 1: Окно панели управления HOTAS Cougar в режиме по умолчанию

Профили

Профили используются для быстрой загрузки ранее созданных файлов конфигурации в Калибратор и в джойстик. В разделе "Профили" находятся четыре кнопки: “Default” (По умолчанию), “Load” (Загрузить), “Save” (Сохранить) и “Delete” (удалить). Объяснения этих кнопок приводятся ниже.

Профиль по умолчанию

Нажатие на кнопку "Default" выводит параметры, используемые джойстиком по умолчанию в режиме Windows. Это означает следующее: мэппинг всех осей в соответствии со стандартами DirectX, прямое направление всех осей, значения верхних мертвых зон (ВМЗ) и нижних мертвых зон (НМЗ) - 1536 (2.34%), значение околонулевой мертвой зоны (ОМЗ) - 2048 (3.13%), линейная характеристика кривой отклика (экспонента = 0), база кривой отклика – в центре, и функция "Задействовать видимые оси" - выключена. Если ваш джойстик подключен, и информация на экране лишена всякого смысла, лучше всего загрузить профиль по умолчанию и уже на его основе вносить необходимые изменения.

Сохранение

По нажатии на кнопку "Save" текущая конфигурация, настроенная в разделе "Параметры осей", может быть сохранена в любом месте на диске в виде файла с расширением .f2k. Если была осуществлена калибровка, ее данные также сохраняются в этом файле. Сохранение профилей позволит заготовить файлы конфигурации с различными настройками осей для разных игр, с их последующей загрузкой.

Загрузка

Нажатием на эту кнопку можно загрузить сохраненный профиль в окно программы. Однако, это не означает загрузки данного профиля в джойстик, для этого необходимо нажать кнопку "Set", описанную ниже в данном руководстве.

Удаление

С помощью этой кнопки можно удалить профиль.

Режимы работы джойстика

Калибратор позволяет выбрать режим работы джойстика в части настроек осей, параметров калибровки и способа эмуляции. Выбор режима работы не требует нажатия на кнопку "Set", так как все изменения автоматически выполняются джойстиком.

Настройки осей

В режиме по умолчанию джойстик использует стандартные параметры мэппинга и конфигурации осей. Эти данные всегда загружаются в джойстик и используются при его подключении, кроме пронумерованных кнопок осей, если последний загруженный файл содержал команду включения видимого режима (более подробное описание приведено в разделе "Изменение осей, видимых для Windows). Если же выбран пользовательский режим настроек осей, джойстик будет использовать следующие параметры, заданные пользователем:

- Мэппинг осей
- Инверсия
- Блокировка
- Данные о кривой отклика
- База кривой отклика
- Данные о мертвых зонах
- Установки триммера

Дополнительную информацию о вариантах калибровки см. в разделе “Калибровка”. Дополнительную информацию о режимах эмуляции см. в разделе “Эмуляция осей и кнопок”.

Параметры осей

Калибратор имеет две закладки: Мэппинг осей (Axis Mapping) и Настройка отклика (Axis Shaping). Внесение изменений в любой из этих закладок не отражается сразу на работе джойстика – для того, чтобы устройство начало использовать обновленные параметры, все изменения должны быть загружены в контроллер, и должен быть изменен режим вывода. Для большей наглядности откройте Калибратор и вызовите профиль по умолчанию.

Кнопки параметров осей

Раздел "Параметры осей" (помимо двух вышеописанных закладок) содержит три кнопки: "Сброс" (Reset), "Вызвать" (Retrieve) и "Установить" (Set). Их использование объясняется ниже.

Кнопка "Установить"

Кнопка "Установить" (Set) предназначена для загрузки параметров калибровки в джойстик. До того, как это будет проделано, джойстик не знает ни о каких изменениях, сделанных пользователем в закладках параметров осей, и нажатие на эту кнопку автоматически загружает текущие значения параметров в контроллер. Для того, чтобы джойстик использовал загруженные параметры, необходимо в разделе "Настройки осей" режим "Пользовательские настройки". Этот режим также будет выбран автоматически при нажатии на кнопку "Установить".

Кнопка "Вызвать"

Кнопка "Вызвать" (Retrieve) используется для загрузки параметров из джойстика и их отображения в разделе "Пользовательские настройки осей". Если какие-то изменения вносятся в режиме эмуляции, их также можно отобразить нажатием на эту кнопку. Калибратор автоматически осуществляет эту операцию при загрузке программы, для отображения текущей конфигурации джойстика, если джойстик не подключен – выдается сообщение об ошибке, и загружается профиль по умолчанию. Нажатие кнопки "Вызвать" показывает также текущий режим работы джойстика.

Закладка "Настройка осей"

Закладка "Настройка осей" (Axis Setup) открывает доступ к ряду функций, назначение которых более подробно описано ниже.

- Изменение мэппинга осей
- Инвертирование осей
- Блокировка параметров осей
- Изменение осей, видимых для Windows

Изменение мэппинга осей

Если вы хотите, чтобы определенная физическая ось управляла бы другой осью в определенной конфигурации, необходимо изменить мэппинг (назначение) этой оси. При загрузке по умолчанию таблица мэппинга выглядит следующим образом:

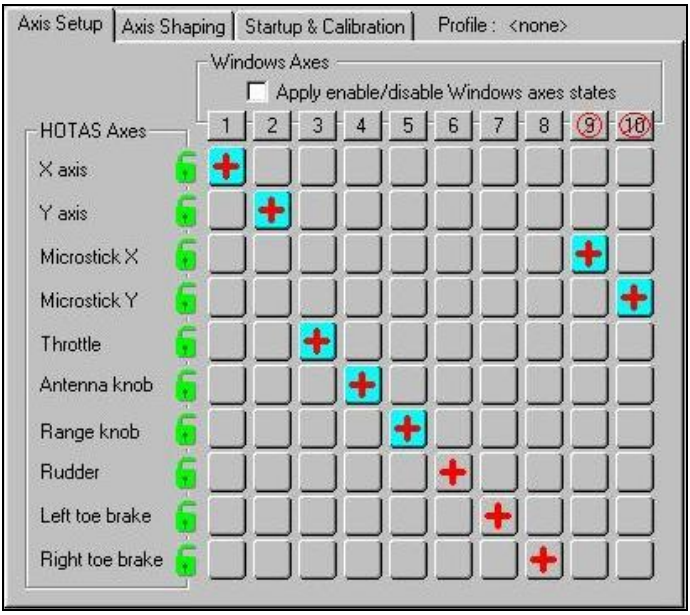


Рис. 2: Матрица мэппинга осей по умолчанию

Матрица из кнопок в центре закладки отображает текущее назначение осей; в данном случае видно, что ось X назначена для управления осью 1 DirectX, ось Y – второй, РУД – третьей, и т. д. Если нужно, чтобы вторая ось (ось Y для DirectX) управлялась сектором газа, достаточно просто кликнуть на вторую ячейку в ряду Throttle, и матрица будет выглядеть следующим образом:

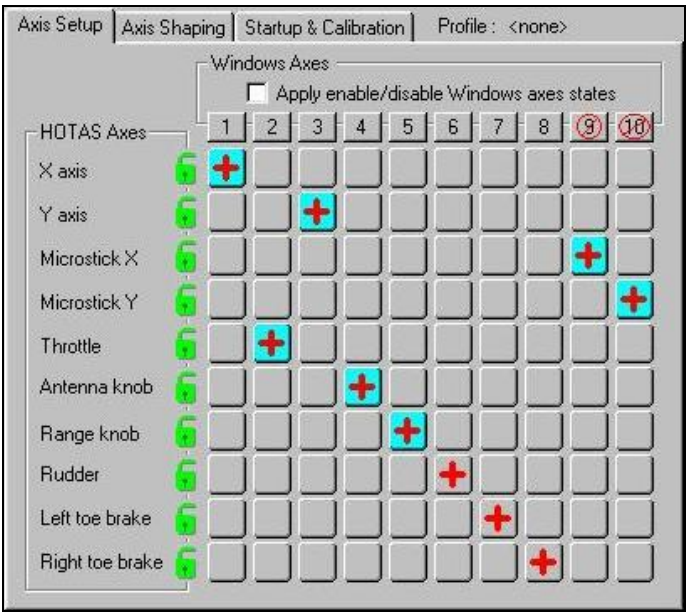


Рис. 2: Матрица мэппинга осей: сектор газа назначен на ось Y

Как видно на рисунке, сектор газа был переназначен на 2 ось, а ось Y джойстика заняла место сектора газа на 5 оси. Таким образом можно поменять местами любые оси, и хотя Windows будет видеть их в измененном порядке, все настройки осей внутри самого джойстика – инвертирование, кривая отклика, мертвая зона и эмуляция - привязаны к физическим осям. Иными словами, если сектор газа (физический) имеет чувствительную кривую отклика, он будет управлять осью Y так же, как управлял бы осью Throttle. Заметьте также, что фон отмеченных ячеек в 9 и 10 колонках – серый, тогда как в остальных – голубой. Это связано с тем, что DirectX может распознать максимум 8 осей, следовательно, при таком мэппинге Windows просто не распознает положение осей X и Y микроджойстика на секторе газа. Если, к примеру, нужно, чтобы микроджойстик контролировал основные оси X и Y, то окно мэппинга выглядело бы следующим образом:

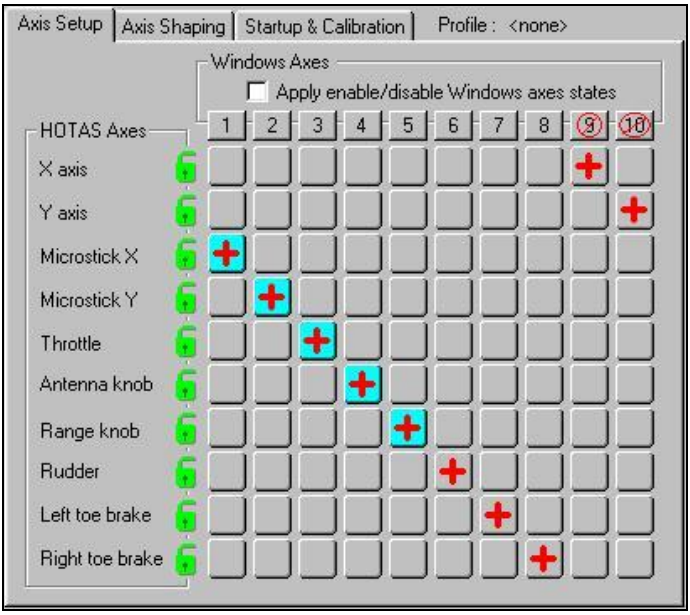


Рис. 3: Матрица мэппинга осей: микроджойстик назначен на оси X и Y

Инверсия направления оси

Для инвертирования оси кликните на нужную ячейку со значком "+" (или "-") – и направление оси изменится. Если изменить направление оси Y, матрица мэппинга осей приобретет следующий вид:

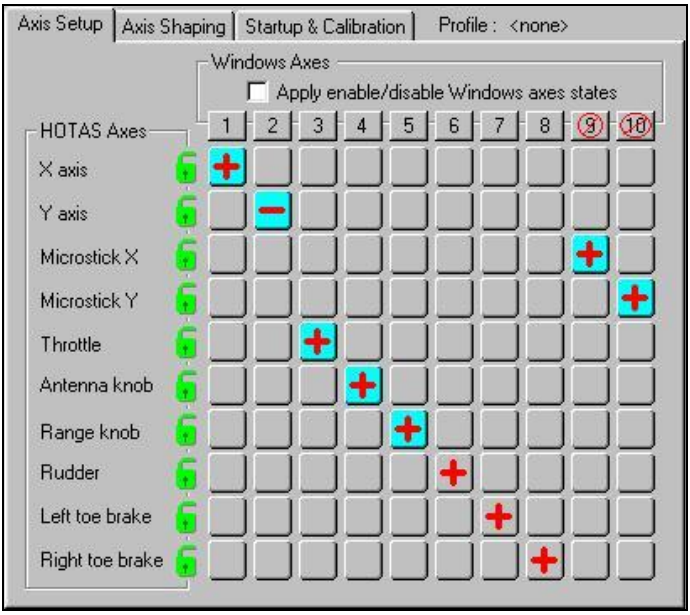


Рис. 4: Матрица мэппинга осей: ось Y инвертирована

Блокировка оси

Справа от названия каждой из осей находится иконка в виде замка. Если замок "открыт", ось не заблокирована и будет работать нормальным образом. Если замок закрыт и красного цвета, ось "заблокирована", и ее назначение не может быть изменено. Для блокировки или разблокировки кликните на иконку.

Изменение состояния осей в Windows

Это название, возможно, не совсем точно отражает всю полезность этой функции, но если не вдаваться в технические подробности, это оптимальное описание. Вверху каждой колонки расположена кнопка с номером. По умолчанию, на кнопках 9 и 10 присутствует перечеркнутый красный кружок – см. рисунок внизу:

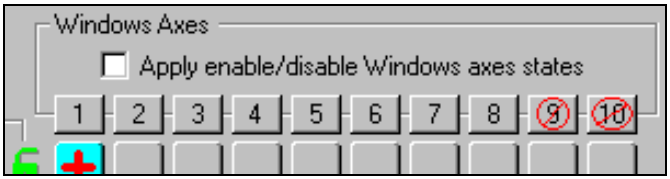


Рис. 5: Пронумерованные кнопки осей в матрице мэппинга

Каждая из этих кнопок представляет собой определенную ось DirectX:

Номер оси	Название DirectX
1	X Axis
2	Y Axis
3	Z Axis
4	Rotational X Axis
5	Rotational Y Axis
6	Rotational Z Axis
7	Slider0
8	Slider1
9	<недоступно>
10	<недоступно>

Таблица 1 Соответствие номеров осей и названий DirectX

Помимо того, что невозможно задействовать оси 9 и 10, невозможно также отключить оси 1 и 2, так как драйвер джойстика подразумевает, что устройство имеет по меньшей мере две оси – X и Y. Если кликнуть на любую кнопку с 3 по 8, можно отключить данную ось (рядом с цифрой появится буква "d"). Когда джойстик подключен, названия осей, а также количество осей, "видимых" для Windows, определяются двумя факторами:

1. Физическая конфигурация манипуляторов
2. Состояние "пронумерованных" кнопок

Более подробные объяснения приводятся на следующей странице.

Оси в физической конфигурации

Физическая конфигурация джойстика определяется тем, какие именно компоненты комплекта подключены к компьютеру в момент, когда Windows обнаруживает джойстик. Всего может быть 6 физических конфигураций HOTAS.

1. Только джойстик
2. Джойстик и сектор газа
3. Джойстик и обычные педали с одной осью
4. Джойстик и новые педали с тремя осями
5. Джойстик, сектор газа и обычные педали с одной осью
6. Джойстик, сектор газа и новые педали с тремя осями

Каждая из этих конфигураций дает различную комбинацию осей в Windows. Ниже приводится таблица для 6 конфигураций и комбинациями осей для каждой из них.

		Названия осей							
		X	Y	Z	R_x	SL_0	R_z	SL_1	R_y
Конфигурация	1	●	●						
	2	●	●	●	●	●			
	3	●	●				●		
	4	●	●				●	●	●
	5	●	●	●	●	●	●		
	6	●	●	●	●	●	●	●	●

Оси, видимые для Windows

Используя пронумерованные кнопки пользователь может изменить список осей, распознаваемых Windows. На следующей странице приводится объяснение, как использовать этот режим.

1. Выберите оси, которые вы хотите сделать видимыми для Windows, переключая соответствующие пронумерованные кнопки до получения требуемой конфигурации.
2. Убедитесь, что флажок поставлен в поле "Включить режим осей, видимых для Windows"
3. Загрузите файл в джойстик, используя кнопку "Установить"
4. Перегрузите джойстик, либо физически переподключив его, либо нажав на кнопку “Reset” утилиты Calibrator.
5. Теперь Windows "видит" джойстик именно в такой конфигурации, которую задал пользователь.

Смысл этой процедуры в том, что, к примеру, даже если у вас нет педалей, или новых педалей с педальными тормозами, можно заставить DirectX “поверить”, что такое устройство присутствует, и использовать любые из свободных физических осей для управления. Важно отметить, что джойстик будет продолжать работать в режиме "Оси, видимые для Windows”, если последний загруженный файл имеет флажок в поле "Включить режим осей, видимых для Windows".

Закладка конфигурации осей

Закладка конфигурации осей позволяет регулировать более продвинутые параметры калибровки джойстика, если пользователь желает настроить каждую ось в соответствии со своими предпочтениями. Выпадающий список осей “Axis” вверху закладки позволяет выбрать ось из десяти возможных, для изменения ее свойств. После регулировки параметров соответствующие изменения отображаются на графике справа от списка параметров. Отображение происходит либо после выбора следующего параметра, либо по нажатию кнопки "Refresh" (обновить). Список регулируемых параметров:

- Верхняя мертвая зона (ВМЗ)
- Нижняя мертвая зона (НМЗ)
- Околонулевая (центральная) мертвая зона (ОМЗ)
- Центровка
- Триммер оси
- Установки кривой (множитель и база)

Мертвые зоны

Меняя эти параметры, вы можете регулировать неактивные области каждой из осей. Внесенные изменения отображаются на графике в виде областей красного цвета. Максимальное значение каждой МЗ равно 65536, хотя, конечно, выбор значения близкого к такому приведет к практически "мертвому" джойстику. К примеру, значение МЗ, используемое джойстиком по умолчанию, равно 1536, что составляет порядка 1% на каждую из МЗ, или около 3% общего хода оси. Ниже приводится внешний вид данной закладки:

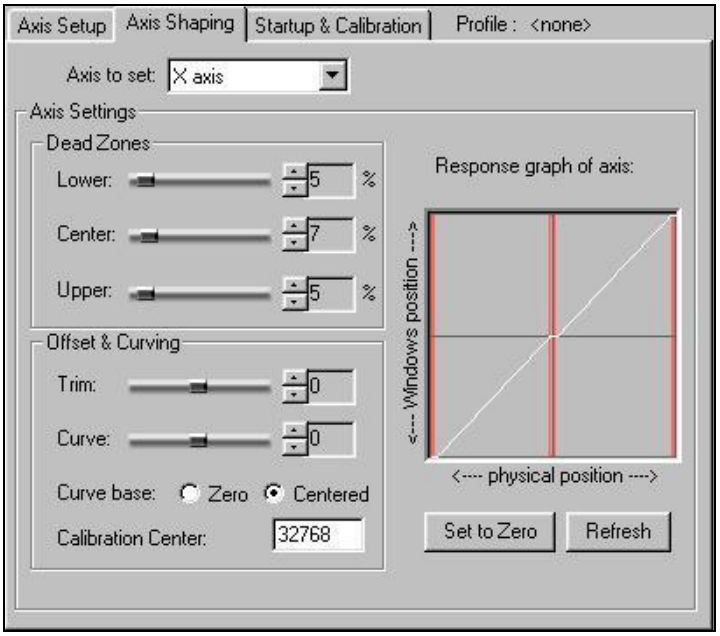


Рис. 6: Закладка конфигурации осей: параметры по умолчанию

На рисунке вверху видны узкие красные области с обоих краев и в центре. Кривая на графике показывает соотношение входного сигнала от осей джойстика и значения, передаваемого в Windows. Для справки, на горизонтальной оси отображаются значения, получаемые джойстиком от его осей, при этом минимум на данном графике находится слева. На вертикальной оси отображаются значения, передаваемые в Windows, минимум на данном графике - снизу. Исходя из этого видно, что в мертвых зонах кривая отклика становится горизонтальной линией, что означает, что физическое перемещение осей в этих областях не будет сопровождаться изменением значений, передаваемых джойстиком в Windows. На

следующем рисунке показано, что происходит, если увеличить значение ОМЗ:

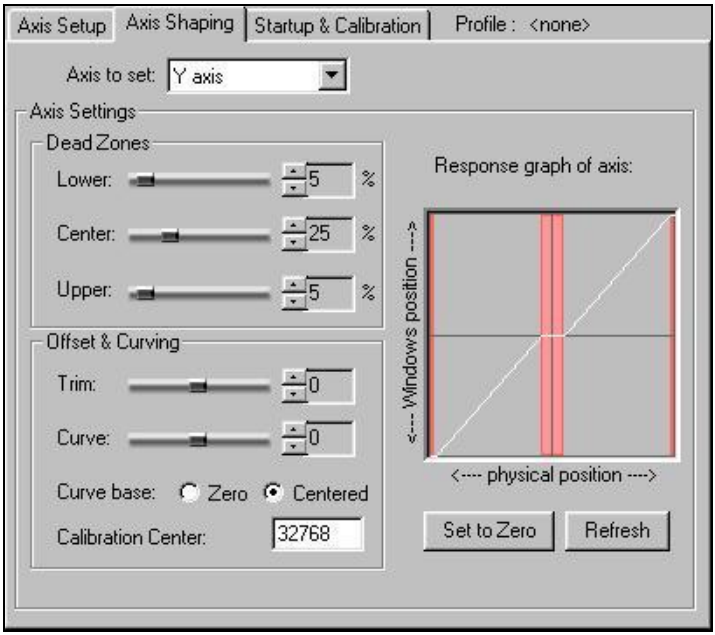


Рис. 7: Закладка конфигурации осей: ОМЗ увеличена

На рис. 8 хорошо видно, что увеличение значения ОМЗ с 7% до 25% выразилось в пропорциональном увеличении выделенной области в центре горизонтальной оси. Горизонтальная часть кривой в центре показывает область, в которой данная ось будет неактивна. Точно также можно настраивать верхнюю и нижнюю мертвые зоны.

Центр калибровки

Эта величина означает положение, которое джойстик будет считать центральным для данной оси. Если указанная величина, к примеру, меньше текущего физического центрального положения джойстика, то в отпущенном состоянии данная ось будет выдавать увеличенное значение. Такого же эффекта можно добиться, используя функцию триммера, чтобы осуществить сдвиг значений, выдаваемых осью джойстика. Основное преимущество триммера по сравнению с регулировкой центрального положения в том, что триммер можно использовать в режиме эмуляции.

Триммер оси

Триммер позволяет задать сдвиг физических значений оси на определенную величину. К примеру, если ось джойстика физически находится в положении 30%, а триммер установлен на –20%, итоговое значение, видимое компьютером, равно 10%. Максимальное значение триммера - +/- 50%; что означает, что при помощи триммера можно сдвинуть значение оси от центра на полный ход в любую из сторон, что проиллюстрировано ниже.

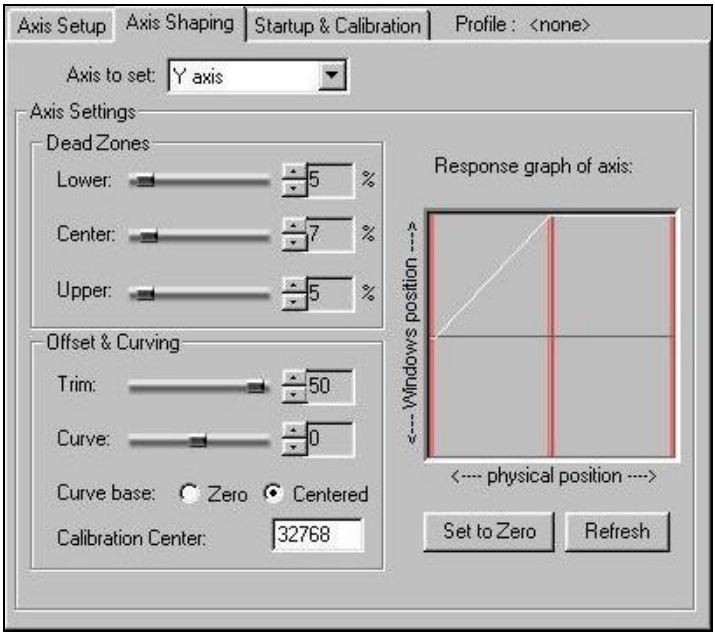


Рис. 8: Триммер установлен на максимум

В данном примере триммер применен к оси Y, что дало следующий результат: в центральном физическом положении оси (центр по горизонтали), выходное значение, получаемое Windows, равно максимальному значению оси Y. Перемещение оси в сторону увеличения не приведет ни к каким изменениям; перемещение в обратном направлении воспринимается как уменьшение значения от максимального до центрального.

Если задать минимальное значение триммера, график будет выглядеть следующим образом:

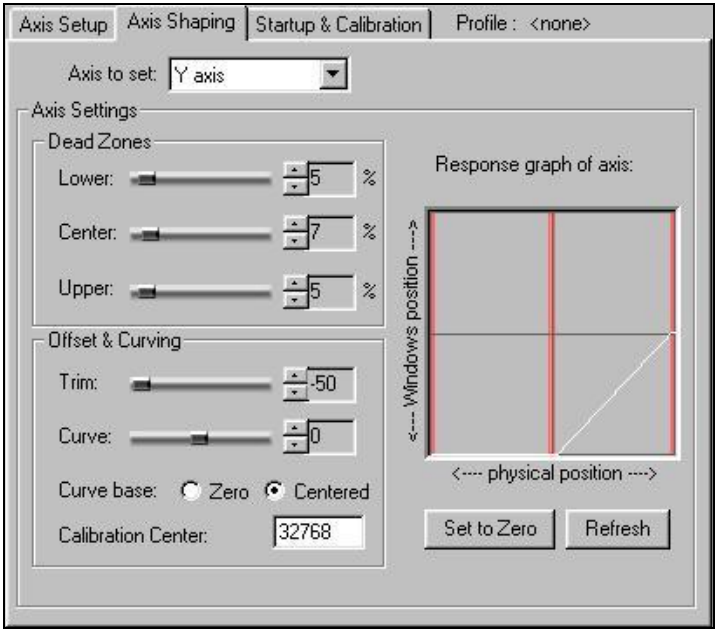


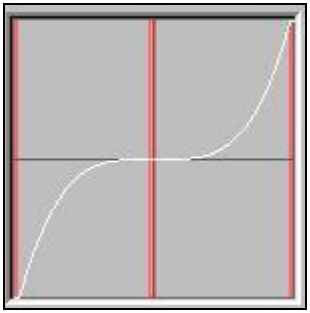
Рис. 9: Триммер установлен на минимум

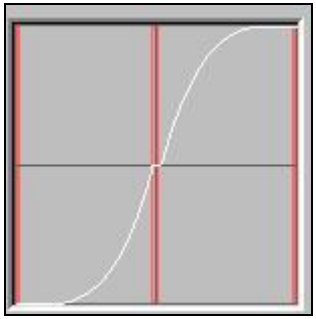
В данном случае ось Y будет в центральном физическом положении выдавать минимальное значение, а перемещение до конца в сторону увеличения даст центральное значение оси.

Форма кривой

Данный параметр может быть изменен для придания отклику оси более экспоненциального характера, вместо линейного. Данный параметр может иметь значение от -32 до 32; однако использование столь высоких значений на нескольких осях приведет к существенному замедлению работы джойстика. Вы заметите, что значения формы кривой больше 20 и меньше –20 практически бесполезны.

Ниже будут рассмотрены кривые с базой в центре, и проиллюстрирована разница между положительным и отрицательным значениями параметра. Для примера взяты две кривых с параметрами 10 и -10.

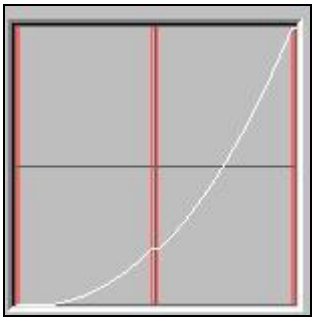


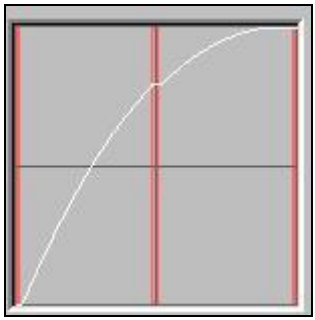


На левом графике изображена кривая оси с экспонентой 10. Как видно, в центре изменения очень несущественные – почти идентичные увеличению ОМЗ. Затем интенсивность кривизны быстро возрастает, пока ветвь оси не становится практически вертикальной. Это означает, что джойстик будет очень чувствительным в крайних областях хода оси. Правый график показывает кривую оси с экспонентой –10, что дает фактически обратный эффект: ось очень чувствительна в центре, тогда как в крайних областях чувствительность сильно уменьшается – почти как при увеличении ВМЗ и НМЗ.

База кривой

Смысл поля "База кривой" (Base of Curve) в следующем: если для осей джойстика и микроджойстика кривые должны меняться относительно центра, то для таких осей, как сектор газа и педальные тормоза более логичным будет экспонирование кривой от нулевого значения. Ниже приведены графики кривых с экспонентами 5 и –5, но с базой в точке ноль.





На левом графике изображена кривая оси с экспонентой 5 – заметьте, насколько ее форма похожа на правую верхнюю ветвь кривой с экспонентой 10 и базой в центре. Чувствительность оси будет очень низкой в нижней части хода, как будто задано большое значение НМЗ. По мере перемещения чувствительность возрастает, достигая пика в верхней части хода оси. Справа представлена кривая с экспонентой –5 и базой в нулевой точке. Опять же, ее форма очень похожа на верхнюю правую часть кривой с базой в центре и экспонентой -10. Чувствительность оси очень высока в начале хода и падает по мере перемещения вверх. Верхняя часть кривой выглядит так, словно задано большое значение ВМЗ.

Закладка "Начальная загрузка и калибровка"

Закладка "Начальная загрузка и калибровка" содержит информацию о том, как ведет себя джойстик при загрузке компьютера, а также о параметрах калибровки. Закладка делится на две области: верхняя часть – параметры начальной загрузки, нижняя – параметры калибровки.

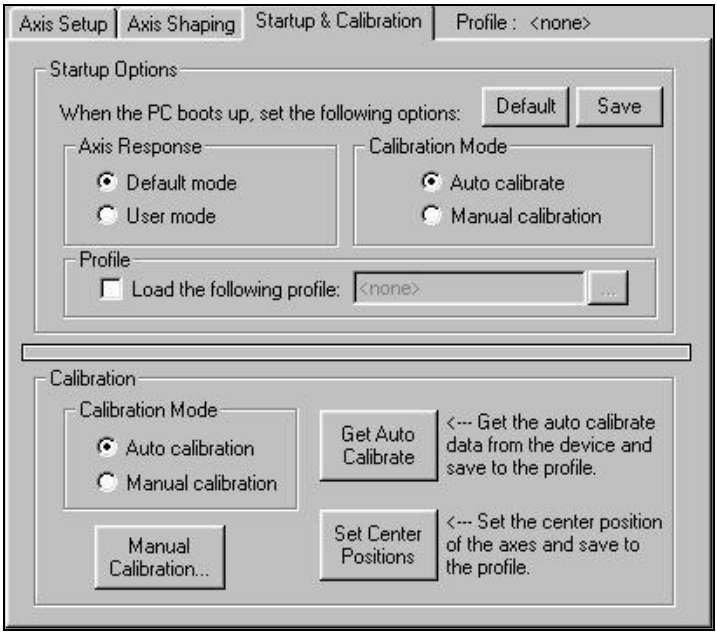


Рис. 11: Закладка "Начальная загрузка и калибровка"

Параметры начальной загрузки

Существуют три основных параметра, которые можно настраивать для автоматической конфигурации джойстика при запуске компьютера: отклик осей (Axis response), режим калибровки (Calibration mode) и загрузка указанного профиля. По умолчанию, джойстик загружается с откликом осей по умолчанию, автоматическим режимом калибровки и последним загруженным в контроллер профилем. Для изменения установок по умолчанию выберете желаемые опции и нажмите кнопку "Save". Для выбора профиля пометьте поле "Load the following profile", затем нажмите '…', или введите имя профиля непосредственно в текстовом поле. Помните,

что профиль должен быть в поддиректории профилей директории HOTAS. Для сброса параметров начальной загрузки, нажмите кнопку “Default”.

Калибровка

В разделе “Калибровка” (Calibration) доступны опции по установке режима калибровки, калибровке манипулятора, вызову данных автокалибровки и заданию центрального положения осей.

Выбор режимов калибровки “Автоматический” (Auto) или "Ручной" (Manual) автоматически переводит джойстик в выбранный режим. Помните, что не требуется нажимать кнопку "Применить" (Apply) – выбранный режим сразу же применяется.

В режиме ручной калибровки будут использоваться данные, загруженные в джойстик после осуществления процедуры ручной калибровки. Эти данные загружаются автоматически вместе с текущими параметрами осей, для использования после осуществления любой процедуры калибровки.

Режим автоматической калибровки означает, что джойстик автоматически настраивает максимальные значения осей, когда пользователь перемещает их в крайние положения. Режим ручной калибровки означает, что джойстик будет использовать калибровочные значения, полученные в ходе выполнения процедуры ручной калибровки, которая будет описана ниже. Помните, что джойстик не может переключиться в режим ручной калибровки, если до этого не была выполнена процедура ручной калибровки. Если джойстик был перезагружен (либо путем физического отключения, либо кнопкой "Перезагрузить" (Restart), и находится в режиме автоматической калибровки, рекомендуется переместить все оси HOTAS Cougar (РУС, РУД, кольцо масштаба, кольцо антенны, микроджойстик и т. д.) в крайние положения, удерживая их а течение примерно трех секунд. Это позволит системе автоматической калибровки считать данные об осях и откалибровать манипулятор.

Если нажать кнопку "Извлечь данные автокалибровки" перед переключением режима калибровки из автоматического в ручной, данные автокалибровки будут скопированы в текущий профиль. Теперь вы можете применить и сохранить данный профиль, и джойстик будет использовать эти данные, никогда не подстраивая значения осей.

Кнопка "Задать центральные положения" (Set Center Positions) используется для задания определенного положения каждой из осей в качестве ее центра. Если вы хотите указать для осей джойстика и кольца масштаба центральные положения, отличные от значений, полученных в результате автокалибровки, нажмите эту кнопку и выберите нужные оси, как показано ниже.

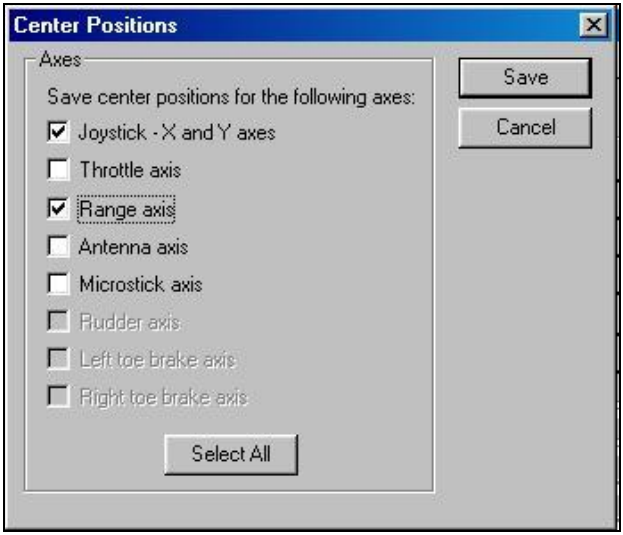


Рис 12: Сохранение центрального положения для осей джойстика и кольца масштаба

После нажатия кнопки "Сохранить" (Save) вам будет предложено поставить указанные оси в центральное положение и нажать "OK". Теперь выбранным осям будут назначены новые центральные положения.

Ручная калибровка

Нажатие на кнопку "Ручная калибровка" (Manual Calibration) вызывает окно процедуры ручной калибровки, как показано ниже:

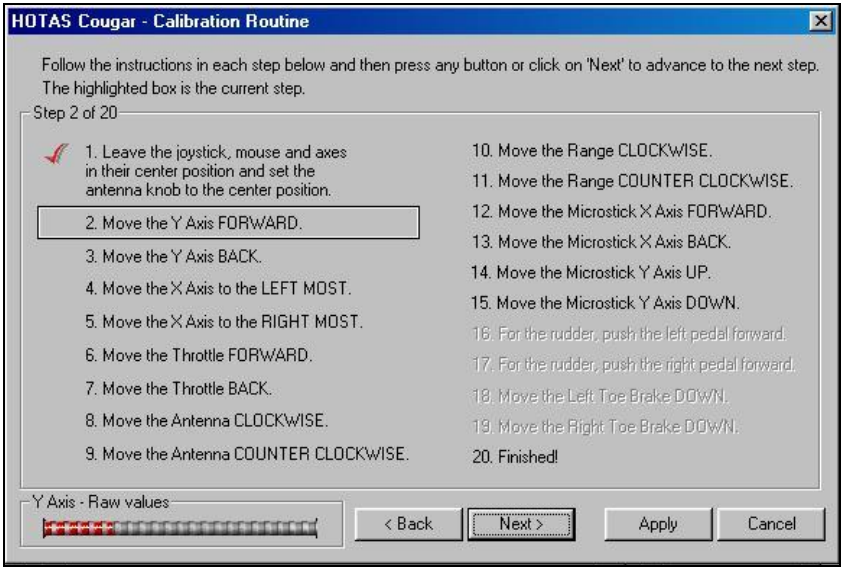


Рис 13: Окно процедуры ручной калибровки

Следуйте инструкциям на экране, последовательно проходя все этапы и нажимая "Далее" (Next), когда та или иная ось достигает положения, которое требуется сохранить на данном этапе. Сохраняется положение оси на момент нажатия кнопки Next. Когда вы завершите процедуру калибровки, программа автоматически загрузит калибровочные данные в джойстик, вместе с текущими данными раздела "Параметры осей". Эти калибровочные данные сохраняются, и могут быть сохранены в виде профиля для дальнейшего использования.

Заметьте, что оси, не подключенные физически, не активны. Процедура калибровка автоматически пропустит отсутствующие оси. Индикатор уровня в нижнем левом углу окна показывает "абсолютное" значение положения осей. Вполне возможно, что даже в крайних положениях органов управления минимальное и максимальное значения по индикатору достигнуты не будут. Индикатор служит лишь для наглядного отображения перемещения оси.

Действия и другие опции

В разделе "Действия" (Actions) доступно три кнопки: "Перезагрузка" (Restart Device), "Эмуляция кнопок и осей" (Button & Axis emulation) и "Загрузить в устройство" (Download to device). Там же доступны маркер автоматического опроса состояния джойстика и кнопка "Спрятать" (Hide).

Перезагрузка

Кнопка "Перезагрузка" предназначена для ручного "отключения" джойстика, что соответствует физическому отключению устройства от USB-порта с последующим подключением. Эта функция может оказаться полезной, если вы хотите изменить состояние осей в Windows. В этом случае вам необходимо отключить и вновь подключить джойстик (см. раздел "Изменение состояния осей в Windows), чему соответствует нажатие кнопки "Restart Device". Помимо этого, при загрузке система автоматической калибровки определяет центральное положение соответствующих осей (X, Y, Rudder и оси микроджойстика); чтобы установить центральное положение этих осей вручную, нажмите "Restart Device", удерживая эти оси в нужном положении.

Эмуляция осей и кнопок

Если эмуляция осей и кнопок включена (фон кнопки зеленого цвета), джойстик будет использовать последний загруженный конфигурационный файл. Конфигурационный файл - это файл, содержащий информацию об эмуляции клавиатуры и мыши, а также различные команды изменения параметров осей. Более подробную информацию см. далее в Справочном руководстве HOTAS Cougar.

В случае, если эмуляция осей и кнопок выключена (фон кнопки красного цвета), джойстик будет работать в обычном режиме, и его кнопки будут восприниматься как стандартные кнопки DirectX в Windows.

Загрузить в устройство

Кнопка "Загрузить в устройство" (Download to device) запускает загрузчик файлов HOTAS Cougar Loader. Загрузчик используется для загрузки конфигурационных файлов (файлы с расширением .TMJ). Более подробную информацию о структуре конфигурационных файлов см. далее в Справочном руководстве “HOTAS Cougar Owner Reference Book”. На следующей странице приводится изображение окна утилиты загрузки.

Главное окно утилиты загрузки содержит 2 кнопки и одно отмечаемое поле. Кнопка "Проверить" (Check) проверяет файлы на наличие ошибок. При этом

файл не загружается в контроллер. Кнопка "Загрузить" (Download) проверяет конфигурационный файл на наличие ошибок и загружает его, если ошибок не обнаружено. Информация в поле "Свойства загруженного файла" показывает, какой файл был последним загружен в джойстик, и дату последней загрузки.

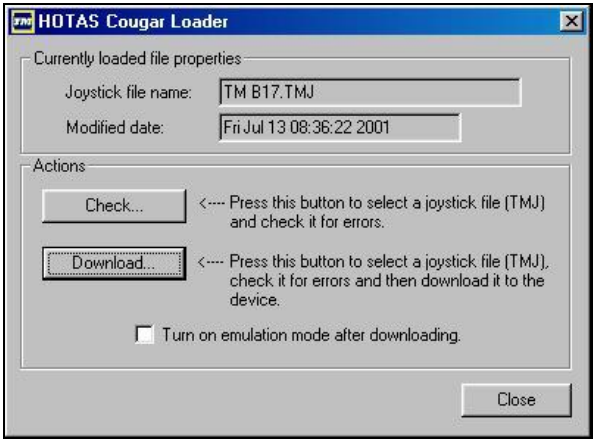


Рис. 14: Загрузчик HOTAS Cougar

Маркер "Включить режим эмуляции после загрузки" позволяет автоматически включить режим эмуляции после загрузки конфигурационного файла в джойстик. Режим эмуляции может также быть включен нажатием кнопки "Эмуляция кнопок и осей" в Панели Управления HOTAS Cougar (см. предыдущий раздел Руководства. При проверке или загрузке файла выводится следующее окно:

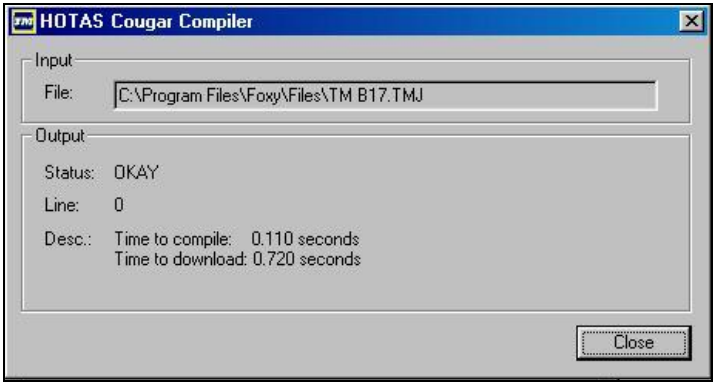


Рис. 15: Утилита "Компилятор HOTAS Cougar", проверка файла на наличие ошибок

Опрос устройства

Маркер опроса устройства используется для включения функции опроса панелью управления HOTAS Cougar текущего состояния устройства. При опросе определяется, подключен ли джойстик, и в каком он режиме. Частота опроса устанавливается в специальном поле.

Свернуть в панель задач

Панель управления HOTAS Cougar может быть свернута в панель задач. Нажмите кнопку "Свернуть" (Hide). Учтите, что данная функция доступна лишь в том случае, если включен опрос устройства. Для того, чтобы восстановить свернутое в панель задач окно, кликните на иконке левой кнопкой мыши и выберите из меню опцию "Открыть панель управления" (Open HOTAS Cougar Control Panel). Помимо этого, в меню доступны также выход из панели управления и запуск загрузчика файлов. Можно переключать режимы эмуляции правым щелчком мыши по иконке в панели задач. Значения различных цветовых сочетаний иконки приводятся в таблице ниже:

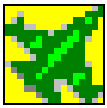
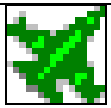
Иконка	Цветовое сочетание	Описание
	*Зеленый самолет/ желтый фон*	Режим эмуляции **ВКЛ**, Отклик осей - в **Пользовательском** режиме
	*Красный самолет/ желтый фон*	Режим эмуляции **ВЫКЛ**, Отклик осей - в **Пользовательском** режиме
	*Зеленый самолет/ серый фон*	Режим эмуляции **ВКЛ**, Отклик осей - в режиме **Windows**
	*Красный самолет/ серый фон*	Режим эмуляции **ВЫКЛ**, Отклик осей - в режиме **Windows**

Таблица 2: Описание иконки панели задач

THRUSTMASTER

HOTAS COUGAR

РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ И СПРАВОЧНАЯ ДОКУМЕНТАЦИЯ

Содержание

1. ЦЕННОЕ ПРИОБРЕТЕНИЕ 39

1.1 ВВЕДЕНИЕ 39
1.2 УСТАНОВКА КОМПЛЕКТА 39
1.3 ЗНАКОМСТВО С ДАННЫМ СПРАВОЧНЫМ РУКОВОДСТВОМ 40

2. БАЗОВЫЕ ПОНЯТИЯ 42

2.1 ОСНОВЫ ПРОГРАММИРОВАНИЯ THRUSTMASTER 42
2.1.1 Введение 42
2.1.2 Концепция HOTAS 42
2.1.3 Как достигается концепция HOTAS в современных летных симуляторах и иных играх? 44
2.1.4 Файл настроек - основы программирования 44
2.1.5 Макрокоманды и файл макрокоманд - основы программирования 45
2.1.6 Каким образом файл настроек определяет, в каком макро-файле содержатся используемые макрокоманды? 46
2.1.7 Подведение итогов описанного выше 47
2.1.8 Загрузка файла настроек в контроллер 48
2.1.9 Структура файлов настроек и файлов макрокоманд 48

3. ВЫРАЖЕНИЯ ДЛЯ КНОПОК И МАКРОСЫ 50

3.1 ВЫРАЖЕНИЯ ДЛЯ КНОПОК И ОБОЗНАЧЕНИЕ КЛАВИШ TM 50
3.2 ОБОЗНАЧЕНИЕ КЛАВИШ THRUSTMASTER 53
3.3 МАКРОСЫ И ПРАВИЛА ИХ СОСТАВЛЕНИЯ 54
3.4 МОДИФИКАТОРЫ ВЫРАЖЕНИЙ 57
3.5 КЛЮЧИ 58
3.5.1 Увеличение количества программируемых функций: 59
3.5.1.1 /U, /M, /D - Верхний, средний, нижний регистры 59
3.5.1.2 /I, /O - Нажато, отжато 60
3.5.2 Разделение макросов, запрограммированных на кнопку: 62
3.5.2.1 /T - Ключ переключения 62
3.5.2.2 Сброс цепочки переключений 64
3.5.2.3 Инверсия порядка переключений 65
3.5.2.4 /P, /R - Нажатие - отпускание 66
3.5.3 Повторяющиеся и неповторяющиеся символы: 68
3.5.3.1 Неповторяющиеся символы 68
3.5.3.2 /A - Автоповтор 68
3.5.3.3 /H - Удержание 68
3.5.4 Правила и иерархия ключей 70
3.5.4.1 Правила использования ключей 70
3.5.4.2 Иерархия ключей 70
3.6 КОМАНДЫ ЗАДЕРЖКИ И ПОВТОРА 71
3.6.1 Выражения с командой DLY( ) 72
3.6.2 Выражения с командой RPT( ) 73

3.7 ГРУППИРОВАНИЕ СИМВОЛОВ – ИСПОЛЬЗОВАНИЕ СКОБОК 74
3.7.1 ( ) Круглые скобки 75
3.7.2 { } Фигурные скобки 75
3.7.3 < > Угловые скобки 77
3.8 НАЗНАЧЕНИЕ КНОПОК DIRECTX (DIRECTINPUT) И РАБОТА С НИМИ 79
3.8.1 Выражение USE ALL_DIRECTX_BUTTONS 80
3.9 ИСПОЛЬЗОВАНИЕ КОДОВ KD, KU И СКАН-КОДОВ USB 83
3.9.1 KD, KU 83
3.9.2 Программирование USB 84
4. ПРОГРАММИРОВАНИЕ ХЭТОВ 85
4.1 ПРОГРАММИРОВАНИЕ ХЭТОВ ДЖОЙСТИКА 85
4.1.1 Программируемые положения хэтов 85
4.1.2 4-позиционные и 8-позиционные хэты: USE HatID FORCED_CORNERS 86
4.1.3 Управление мышью с помощью хэта. 87
4.1.4 Назначение хэта переключателем видов (POV) 88
4.1.5 Использование хэта для эмуляции стрелок клавиатуры 89
4.1.6 Использование хэта для эмуляции клавиш цифровой клавиатуры 90
4.1.7 Каким образом компилятор преобразует выражения USE HatID AS 91
5. КОНФИГУРАЦИОННЫЕ ВЫРАЖЕНИЯ 94
5.1 ВВЕДЕНИЕ 94
5.2 MDEF - ФАЙЛ МАКРОКОМАНД 95
5.3 RATE 96
5.4 S3_LOCK И S3_UNLOCK 97
5.5 НАЗНАЧЕНИЕ ДРУГОЙ КНОПКИ ДЛЯ ВЫПОЛНЕНИЯ ВЫРАЖЕНИЙ С КЛЮЧАМИ /I, /O КОМАНДОЙ SHIFTBTN 98
5.6 USE HAT SENSITIVITY - ЧУВСТВИТЕЛЬНОСТЬ ПРОМЕЖУТОЧНЫХ ПОЛОЖЕНИЙ ХЭТА 98
5.7 USE T1 SENSITIVITY 99
5.8 USE FOXY GRAPHIC И README 100
5.9 NULLCHR - ПУСТОЙ СИМВОЛ ^ 101
5.10 KEYBOARD (AZERTY, QWERTY) 103
5.11 ИСПОЛЬЗОВАНИЕ ПРОФИЛЕЙ ПАНЕЛИ УПРАВЛЕНИЯ HOTAS COUGAR - USE PROFILE 103
5.11.1 Еще несколько слов о профилях 104
5.12 КОНФИГУРАЦИОННЫЕ ВЫРАЖЕНИЯ, ОПИСАННЫЕ В ДРУГИХ РАЗДЕЛАХ РУКОВОДСТВА 106
6. ПРОГРАММИРОВАНИЕ ОСЕЙ 107
6.1 ОСНОВНЫЕ ПРИНЦИПЫ 107
6.1.1 Понятия аналогового и цифрового режимов 107
6.1.2 Оси Кугуара 108

6.2 ВЫРАЖЕНИЯ ЦИФРОВЫХ РЕЖИМОВ 109
6.2.1 Тип 1: повторяющиеся символы 110
6.2.1.1 Модификатор - FORCE_MACROS 111
6.2.1.2 Важные замечания по использованию FORCE_MACROS 113
6.2.2 Тип 2: произвольная последовательность символов, фиксированные зоны 116
6.2.2.1 Модификатор FORCE_MACROS 117
6.2.3 Тип 3: генерирование удерживаемых клавиш 118
6.2.4 Тип 4: импульсное генерирование символов 119
6.2.5 Тип 5: произвольная последовательность символов, изменяемые зоны 120
6.2.5.1 Модификатор FORCE_MACROS 121
6.2.6 Тип 6: повторяющиеся символы, изменяемые зоны 121
6.2.6.1 Модификатор FORCE_MACROS 122
6.2.7 Направление осей: аналоговые значения и цифровые выражения 123
6.2.7.1 Аналоговые значения осей 123
6.2.7.2 Цифровые выражения 1 типа 124
6.2.7.3 Цифровые выражения 2 типа 124
6.2.7.4 Цифровые выражения 3 типа 125
6.2.7.5 Цифровые выражения 4 типа 126
6.2.7.6 Цифровые выражения 5 типа 126
6.2.7.7 Цифровые выражения 6 типа 127
6.3 КРИВЫЕ ОТКЛИКА (CURVE) 128
6.4 ТРИММЕР ОСЕЙ (TRIM) 131
6.5 ОТКЛЮЧЕНИЕ ОСЕЙ 135
6.5.1 Отключение и подключение осей в игре при помощи команд LOCK, UNLOCK 136
6.6 МЭППИНГ ОСЕЙ (SWAP) 139
6.7 ИНВЕРСИЯ НАПРАВЛЕНИЯ ОСИ (REVERSE, FORWARD) 140
6.8 КОНФИГУРАЦИОННОЕ ВЫРАЖЕНИЕ USE AXES_CONFIG 141
7. ПРОГРАММИРОВАНИЕ МЫШИ 143
7.1 ПОНЯТИЯ "УСТРОЙСТВО УПРАВЛЕНИЯ МЫШЬЮ" И "МИКРОДЖОЙСТИК 143
7.2 USE MTYPE – САМЫЙ ПРОСТОЙ СПОСОБ ЗАПРОГРАММИРОВАТЬ МИКРОДЖОЙСТИК НА УПРАВЛЕНИЕ МЫШЬЮ 144
7.3 ВЫРАЖЕНИЕ USE MICROSTICK AS MOUSE 146
7.3.1 Назначение других осей в качестве осей мыши 150
7.4 СОЗДАНИЕ ПОЛЬЗОВАТЕЛЬСКИХ ВЫРАЖЕНИЙ УПРАВЛЕНИЯ МЫШЬЮ ДЛЯ МИКРОДЖОЙСТИКА 152
7.5 ВЫРАЖЕНИЕ USE ZERO_MOUSE 158
7.6 ПРОГРАММИРОВАНИЕ КНОПОК МЫШИ 158
7.7 ОТКЛЮЧЕНИЕ НАЗНАЧЕНИЯ МИКРОДЖОЙСТИКА ДЛЯ УПРАВЛЕНИЯ МЫШЬЮ ПО УМОЛЧАНИЮ 159
7.8 ПРОДВИНУТЫЕ ВЫРАЖЕНИЯ ПЕРЕМЕЩЕНИЯ КУРСОРА 160
7.8.1 Определение экранного разрешения 160
7.8.2 Перемещение в заданную точку экрана 161

7.8.3 Перемещение курсора относительно текущего положения 162
7.8.4 Перемещение по кругу/по многоугольнику 164

8. ЛОГИЧЕСКОЕ ПРОГРАММИРОВАНИЕ 168

8.1 ОСНОВЫ 168
8.1.1 Понятие "флага" 168
8.2 ОПРЕДЕЛЕНИЕ ЛОГИЧЕСКОГО ФЛАГА И ВЫРАЖЕНИЯ С НИМ 169
8.3 ЛОГИЧЕСКИЕ КОМПАРАТОРЫ 171
8.4 ЛОГИЧЕСКОЕ ПЕРЕКЛЮЧЕНИЕ 172
8.5 ИСПОЛЬЗОВАНИЕ ФУНКЦИЙ ЛОГИЧЕСКОЙ ЗАДЕРЖКИ (DELAY) И ЛОГИЧЕСКИХ ИМПУЛЬСОВ (PULSE) 173
8.5.1 Задержка (Delay) 173
8.5.2 Импульс (Pulse) 174
8.5 ПРИМЕРЫ ЛОГИЧЕСКОГО ПРОГРАММИРОВАНИЯ 175
8.5.1 Включение и выключение работы цифровых выражений 4 типа 176
8.5.2 Постепенное триммирование 176

9. УСТРАНЕНИЕ НЕПОЛАДОК 177

9.1 СБРОС КОНТРОЛЛЕРА 177
9.1.1 В игре: EMPTY_BUFFERS и STICK_OFF 177
9.1.2 В среде Windows 178

10. ПРИЛОЖЕНИЯ 180

ПРИЛОЖЕНИЕ 1. СВОДНАЯ ТАБЛИЦА КОМАНД И ВЫРАЖЕНИЙ 180
Выражения для кнопок и модификаторы 180
Ключи 181
Конфигурационные выражения 182
Программирование осей 183
Продвинутые выражения управления мышью 184
Логические выражения 184
Аппаратные команды 184
ПРИЛОЖЕНИЕ 2. СИНТАКСИС КЛАВИШ THRUSTMASTER 185
ПРИЛОЖЕНИЕ 3. СКАН-КОДЫ USB 185
ПРИЛОЖЕНИЕ 4. ОТЛИЧИЯ МЕЖДУ ЯЗЫКОМ ФАЙЛОВ НАСТРОЕЙ ПРЕЖНИХ МАНИПУЛЯТОРОВ TM И ФАЙЛОВ HOTAS COUGAR 190
1. Изменения в обозначении клавиш 190
2. Изменения ключей 190
3. Команды, которые более не поддерживаются 191
4. Имена и расширения файлов 191
5. Действия по умолчанию 191
6. Цифровой и аналоговый режимы работы осей 192
7. Цифровые выражения 1 типа 192
8. Отсутствие РУД 192
9. Символы, запрещенные в именах макрокоманд 193
10. RPT 193

11. Символ комментариев // 193

1. Ценное приобретение

1.1 Введение

Мы рады представить следующее поколение игровых манипуляторов высшего класса от Thrustmaster - HOTAS Cougar. В коробке вы найдете: уникальный, мощный манипулятор, компакт-диск с массой всего полезного, этот *монументальный* документ и бывшее до сегодняшнего дня типичным *краткое* руководство по установке.

На прилагаемом компакт-диске вы найдете все необходимое ПО для работы HOTAS Cougar: Панель управления HOTAS Cougar (подробно описанная в предыдущем разделе), а также пакет приложений Foxy, включая главную утилиту программирования - Foxy HOTAS Cougar Edition и все ее компоненты (такие, как Composer и Korgy), а также вспомогательные утилиты (Foxy GUI, Запуск Приложений и т. д.).

Утилиты Foxy и Foxy GUI не описаны в данном Руководстве. Все необходимые пояснения касательно их использования вы найдете в соответствующих электронных справочных файлах.

Foxy - ключевая утилита, обеспечивающая доступ к полному спектру функциональных возможностей комплекта через работу с файлами программных настроек и макро-файлами с непревзойденной простотой и гибкостью.

Утилита Foxy GUI позволит несколькими щелчками мыши назначить клавиатурные комбинации любой сложности многочисленным кнопкам джойстика без особых познаний в области программного кода Thrustmaster.

Засим, поставьте вашу любимую музыку, усядьтесь поудобней, налейте чего-нибудь ооооооооочень долгоиграющего и приготовьтесь погрузиться в пучину этого Руководства на долгие дни (и ночи)!

1.2 Установка комплекта

Смотрите Краткое руководство по установке в комплекте документации к HOTAS Cougar.

1.3 Знакомство с данным Справочным руководством

Очевидно, что столь продвинутый манипулятор, как HOTAS Cougar, требует наличия обширной справочной документации и вспомогательных компонентов. Итак, после установки Кугуара откройте Панель Управления Cougar, а также "Игровые устройства" панели управления Windows и убедитесь, что устройство опознано и функционирует. Если вы также установили пакет Foxy, **обязательно** запустите Панель Управления Cougar **до** запуска Foxy - это важно!

Мы начнем с самых азов и постепенно будем углубляться в дебри. Итак, как можно использовать HOTAS Cougar:

Уровень 1: Базовое использование

Вам наверняка будет приятно узнать, что в принципе вам больше не нужно почти ничего ни делать, ни читать, чтобы с успехом использовать Кугуара. Для начала переместите все оси комплекта в их крайние положения, удерживая их там на несколько секунд, чтобы система автокалибровки корректно считала все значения. В принципе, теперь можно закрывать все приложения HOTAS Cougar, запускать игру и наслаждаться! Если игра позволяет напрямую назначать команды кнопкам, осям и хэтам, вы можете это сделать в меню настроек игры. Игра может сама определить джойстик и РУД, и автоматически назначить наиболее часто используемые функции управления полетом, иногда охватывая даже дополнительные оси на РУД - Кольцо диапазона и Кольцо антенны. Вот, собственно, и все. Можно взлетать. Кугуар при этом работает в т. н. режиме Windows, или режиме DirectX - то есть все кнопки и хэты не программируются вручную пользователем, а им присваиваются определенные функции средствами игр.

Уровень 2: Программирование HOTAS Cougar с использованием готовых файлов настроек.

На следующем этапе вы, возможно, захотите настроить Cougar при помощи готовых файлов для различных игр, которые поставляются в комплекте с HOTAS. Комплект поставки включает в себя файлы настроек для порядка 30 наиболее популярных игр, и дают пользователю существенно больше возможностей, нежели назначение команд средствами игр. Эти файлы также находятся на прилагаемом компакт-диске, и могут быть загружены в джойстик при помощи Панели Управления HOTAS Cougar. Но есть еще более простой способ, вернее, даже два. Использовать Foxy или Foxy GUI.

Foxy: Войдите в меню "Избранное" (Favourites) Редактора, и кликните название игры, под которую вы хотите настроить комплект. При этом будут

открыты два файла: слева - основной файл, файл настроек, и справа - файл макрокоманд, или макро-файл. На данном этапе вам с ними ничего делать не надо. Идите в меню "Загрузка" (Download), и выберите пункт "Загрузить" (Download). При этом джойстик будет запрограммирован в соответствии с двумя открытыми файлами. Готово! Теперь Кугуар настроен под выбранную игру. При этом можете также кликнуть на цветные кнопки "Графическая раскладка" (Graphical Layout) или "Просмотреть ReadMe-файл (View ReadMe) на панели запуска приложений (вторая сверху), чтобы либо посмотреть в наглядном виде раскладку команд, либо прочитать комментарии создателя файла.

FoxyGUI: Все еще проще. Просто следуйте инструкциям на экране FoxyGUI. Как и в случае с Foxy, вы выбираете игру, нажимаете кнопку "Загрузить" (Download), закрываете FoxyGUI и наслаждаетесь полетом. Опять же, вы можете просмотреть графическую раскладку или прочитать файл ReadMe, нажав соответствующие кнопки.

Уровень 3: Изучение программирования Cougar.

Для вас подготовлена масса справочной документации, которая поможет вам научиться программировать HOTAS Cougar. Просто выберите наиболее удобный для вас путь:

1. Следующий раздел руководства *(2.1 - Основы программирования Thrustmaster),* а также справочные файлы Foxy помогут вам с легкостью ознакомиться с основами программирования HOTAS Cougar.

2. В меню Мастеров (Wizards) Foxy посмотрите Мастер макрокоманд (Macro wizard) и Мастер настроек (Joystick wizard). Многие пользователи освоили программирование с помощью одних лишь этих Мастеров.

3. В том же меню доступны обучающие файлы (Tutorial), которые откроют специальные файлы настроек в Foxy, объясняющие основы и более продвинутые приемы программирования. Вы можете загружать эти файлы, изменять их и наблюдать последствия вносимых изменений - очень удобный способ.

4. FoxyGUI является простым инструментом программирования Cougar, попутно объясняя, как бы все ваши действия выглядели в Foxy. Возможно, в какой-то момент вы захотите перейти к Foxy, так как это более мощный и быстрый редактор для тех, кто постиг азы.

5. В любой момент вы можете нажать F1 в Foxy для вызова справки, или выделить команду в файле настроек и нажать F1 для получения контекстной справки. Справочный файл огромен, и напичкан полезной информацией, охватывающей все подробности, освещенные в данном Руководстве.

6. Освойте утилиты Composer и Korgy из меню "Вставка" (Insert) - и вам вряд ли придется вообще заглядывать в Руководство!

Очень важная вещь: программировать Cougar очень несложно, нужно просто уделить немного внимания справочным материалам, и вы будете вознаграждены! А когда вы освоите в полной мере программирование на основе текстовых файлов, вы никогда больше не вернетесь к графическим интерфейсам, типичным для более простых утилит программирования манипуляторов.

2. БАЗОВЫЕ ПОНЯТИЯ

2.1 Основы программирования Thrustmaster

2.1.1 Введение

Когда речь заходит о программировании, манипуляторы Thrustmaster всегда устанавливали планку, относительно которой измеряли все остальные джойстики. К сожалению, за ними также закрепилась репутация устройств, весьма сложных для программирования, отчасти из-за того, что поставляемое в комплекте ПО было написано под DOS, отчасти из-за того, что люди просто не были готовы посвятить время тому, чтобы с ними разобраться. Так вот, будьте уверены, программирование этих манипуляторов куда проще, чем освоение современного авиасимулятора.

Мы начнем с самых азов, исходя из предпосылки, что вы ранее никогда не имели опыта общения с манипуляторами Thrustmaster и их программирования. Наверное, не очень хорошо, что мы используем слово "программирование" - понятие, как правило ассоциируемое с разработкой ПО и сложными языками программирования. На самом деле, если говорить упрощенно, речь идет просто о создании файлов, которые присваивают клавиатурные команды различным кнопкам на РУС и РУД.

2.1.2 Концепция HOTAS

В настоящее время в любом летном симуляторе можно управлять самим самолетом, вооружением и авионикой исключительно с клавиатуры. Однако можно значительно повысить реализм, если использовать джойстик, а также сектор газа и педали - собирательно, органы управления. Однако по-прежнему возникает необходимость использовать клавиатуру. Этого также можно избежать, разместив на РУС и РУД дополнительные кнопки и хэты, использование которых позволяло бы имитировать нажатие требуемой

клавиши на клавиатуре. Таким образом, не нужно снимать руки с органов управления и можно полностью сосредоточиться на управлении самолетом, ведении огня и т. д. Это и есть концепция HOTAS, торговая марка Thrustmaster (от англ. **H**ands **O**n **T**hrottle **A**nd **S**tick - "руки на РУД и РУС").

2.1.3 Как достигается концепция HOTAS в современных летных симуляторах и иных играх?

Игровые манипуляторы программируются для эмуляции клавиатурных команд, соответствующих командам в игре. Для этого используется два файла: **файл настроек**, в котором определяется, какая клавиша на клавиатуре назначена той или иной кнопке манипулятора, и **файл макрокоманд**, в котором содержится описание тех действий, которые выполняют кнопки и хэты джойстика в конкретной игре.

2.1.4 Файл настроек - основы программирования

Как уже было сказано, файл настроек используется для назначения клавиатурных команд различным кнопкам и хэтам игровых манипуляторов.

Все кнопки и хэты на РУС и РУД имеют свое собственное название. Запоминать длинный список нет необходимости, так как модуль Composer утилиты Foxy отображает их в удобном графическом виде.

Приведем пример: запрограммируем кнопку S2 джойстика для управления автопилотом. На рис. вверху видно, какая именно кнопка является S2 - кнопка в верхнем левом углу головки РУД. Предположим, в игре для включения - выключения автопилота используется клавиша "a".

Что нужно сделать, чтобы джойстик генерировал букву "a" при нажатии кнопки S2? В файле настроек, который представляет из себя обычный текстовый файл, кнопки обозначаются командой "BTN". В данном случае речь идет о кнопке S2. Нужно, чтобы BTN S2 генерировала клавишу "a". В файле конфигурации необходимо добавить следующую строку:

BTN S2 a

Еще один пример: запрограммируем один из хэтов в верхнем положении для генерации клавиши F1. Все хэты также считаются кнопками, поэтому, как и в предыдущем примере, используется команда BTN. Верхнее положение хэта 1 обозначается как H1U, а все выражение выглядит следующим образом:

BTN H1U F1

Это выражение и является основным элементом программирования Thrustmaster.

В современных авиасимуляторах количество команд может доходить до сотни, и запомнить их все чрезвычайно сложно. Можно добавить в файл настроек описание команд, закомментировав его командой REM, например:

```
BTN S2 a  REM Автопилот вкл/выкл
BTN H1U F1    REM Камера обзора вперед
```

Однако, есть и более удобный способ - можно использовать макрокоманды и файл макрокоманд. Прежде чем перейти к ним, напомним что команда REM может быть добавлена в любое место файла настроек, и все, что находится в строке после REM, игнорируется контроллером.

Теперь познакомимся с файлом макрокоманд.

2.1.5 Макрокоманды и файл макрокоманд - основы программирования

Прежде чем перейти к файлу макрокоманд, дадим определение макрокоманды. В рассмотренных выше примерах мы создали файл настроек, состоящий из двух строк:

```
BTN S2 a  REM Автопилот вкл/выкл
BTN H1U F1    REM Камера обзора вперед
```

Макрокоманда - это слово, которое вводится пользователем для более простого запоминания действия, выполняемого той или иной клавишей или комбинацией клавиш в игре. Например:

```
Autopilot = a
Forward_view = F1
```

Соответственно, теперь можно изменить выражения из файла настроек следующим образом:

```
BTN S2 Autopilot
BTN H1U Forward_view
```

Преимущества данного способа могут быть, на первый взгляд, неочевидны. Однако когда число строк в файле настроек переваливает за 100, использование макрокоманд значительно облегчает просмотр и редактирование таких файлов.

Макрокоманды записываются в специальный файл - файл макрокоманд, или макро-файл. Таким образом, файл макрокоманд содержит описание всех необходимых клавиатурных команд в игре, а файл настроек

присваивает эти команды определенным кнопкам джойстика и РУД. Поэтому, как правило, для каждой игры создается пара файлов - файл настроек и файл макрокоманд. И в окне Редактора утилиты Foxy оба эти файла отображаются одновременно, в соседних колонках. Файлы настроек имеют расширение **.tmj** (**T**hrust**M**aster **J**oystick File), а файлы макрокоманд - **.tmm** (**T**hrust**M**aster **M**acro File).

2.1.6 Каким образом файл настроек определяет, в каком макро-файле содержатся используемые макрокоманды?

Для указания файла макрокоманд в текст файла настроек необходимо добавить строку, содержащую команду USE MDEF, после которой указать имя нужного файла макрокоманд. К примеру, в файле настроек для игры Falcon 4, который может называться **falcon4.tmj**, используются макрокоманды, описанные в файле макрокоманд **falcon4.tmm**. В этом случае строка будет выглядеть следующим образом:

USE MDEF Falcon 4.tmm

Утилита Foxy читает строку USE MDEF файла настроек при открытии, и находит соответствующий файл макрокоманд.

2.1.7 Подведение итогов описанного выше

Итак, как же составляются файлы настроек и файлы макрокоманд.

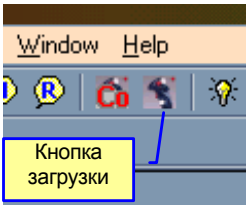
Файл настроек *(Falcon 4.tmj)*	**Файл макрокоманд** *(Falcon 4.tmm)*
REM ---------- REM Falcon 4.tmj REM Файл настроек для Falcon 4 REM REM текст после команды REM не REM исполняется и используется для REM комментариев REM REM ---------- REM Ниже указывается, какой нужно REM использовать файл макрокоманд USE MDEF Falcon 4.tmm REM Программирование кнопок BTN S2 Autopilot BTN H1U Forward_view	REM ---------- REM Falcon 4.tmm REM Файл макрокоманд для Falcon 4 REM REM ---------- REM Описание макрокоманд помогает REM запомнить, что именно та или REM иная клавиша делает в игре. REM Ниже идут описания макрокоманд Autopilot = a Forward_view = F1

Пример условный. Команды в Falcon 4 отличаются.

Итоги вышеизложенного:

1.) Файл макрокоманд содержит макросы, которые просто описывают, что клавиатурные команды делают в игре.
2.) Файл настроек назначает макросы из указанного файла макрокоманд кнопкам и хэтам джойстика и РУД, при помощи команды BTN.
3.) В файле настроек указывается нужный файл макрокоманд, при помощи команды USE MDEF.
4.) Текст после команды REM не исполняется, и используется для написания комментариев.
5.) Файл настроек и файл макрокоманд - обычные текстовые файлы с расширениями **.tmj** и **.tmm** соответственно, и должны находится в одной директории на жестком диске. По умолчанию, устанавливается директория /files/ утилиты Foxy.

2.1.8 Загрузка файла настроек в контроллер



Самый простой способ - нажать на кнопку "Download" (Загрузить) в панели управления Foxy, или "F12" на клавиатуре. Через непродолжительное время файл будет загружен. После этого можно начинать играть, и кнопки и хэты джойстика и РУД будут генерировать клавиатурные команды в соответствии с загруженным файлом настроек. При загрузке файла происходит его обработка одной из утилит HOTAS Cougar - Компилятором. При компиляции текстовые файлы настроек и макрокоманд переводятся в код, понимаемый контроллером. В случае отсутствия ошибок в файлах скомпилированный код загружается в контроллер. После успешного завершения выводится сообщение о подтверждении загрузки, и ваши игровые манипуляторы начинают работать в новой заданной конфигурации.

2.1.9 Структура файлов настроек и файлов макрокоманд

В завершение вводной части приведем общие правила структурирования файлов настроек и файлов макрокоманд, которые проиллюстрированы ниже приведенным примером. На данном этапе, возможно, назначение некоторых строк, команд и символов будет непонятным. Однако задача состоит в другом - дать общее представление о том, какие данные заносятся в файл настроек и файл макрокоманд, и какова их структура. Файл настроек также содержит раздел для конфигурационных команд, назначение которых будет рассмотрено далее.

Разделы	Файл настроек *(Falcon 4.tmj)*	Файл макрокоманд *(Falcon 4.tmm)*
Заголовок Необязателен, но полезен.	Rem ---------- Rem Falcon 4.tmj Rem Rem Файл настроек Falcon 4 Rem Rem Изменен: 1 января 01 Rem Rem ----------	Rem ---------- Rem Falcon 4.tmm Rem Rem Файл макрокоманд Falcon 4 Rem Rem Изменен: 1 января 01 Rem Rem ----------
Конфигурационные команды (***только*** *в файле настроек)*	Rem Rem Конфигурационные команды Rem USE MDEF Falcon 4 USE RATE 60 USE TG1 AS DX1 USE S2 AS DX2	Rem Rem Конфигурационные команды Rem не используются Rem Rem Здесь определяются макрокоманды Rem
Команды Файл настроек Назначения кнопок, Команды осей, Логическое программирование. Файл макрокоманд Определения макросов	Rem ---------- Rem Назначения кнопок Rem ---------- BTN H1U View_up BTN H1D View_Down BTN H1L View_Left BTN H1R View_Right BTN S1 Cycle_MSL_hardpt BTN S2 Pickle_weapon BTN S3 /U Cycle_RDRsubmode /M Ground_Map_FOV /D Cycle_RDRsubmode BTN S4 /T Padlock_view /T 2-D_cockpit Rem ---------- Rem РУД Rem ---------- BTN T2 /T Virtual_Cockpit /T 2-D_cockpit BTN T3 Look_Closer BTN T4 Padlock_Next BTN T5 Padlock_Prev BTN T6 Uncage ANT 1 16 F7 F5 F6 RNG 1 14 SHF F4 SHF F3	Rem ---------- Rem Камера обзора Rem ---------- View_up = KP8 View_Down = KP2 View_Left = KP4 View_Right = KP6 Rem ---------- Rem Вооружение Rem ---------- Cycle_MSL_hardpt = SHF / Pickle_weapon = SPC Rem ---------- Rem Прочее Rem ---------- Cycle_RDRsubmode = F8 Ground_Map_FOV = F9 Padlock_view = 4 2-D_cockpit = 2 Virtual_Cockpit = 3 Look_Closer = l Padlock_Next = KP+ Padlock_Prev = KP- Uncage = u

На этом заканчивается введение в основы программирования. Далее будут более подробно рассмотрены команды файла настроек. Первой будет рассмотрена ранее описанная команда BTN, а также названия всех кнопок и хэтов. Помимо этого, будут более подробно рассмотрены файлы макрокоманд.

3. Выражения для кнопок и макросы

3.1 Выражения для кнопок и обозначение клавиш ТМ

Комплект HOTAS Cougar состоит из нескольких осей, а также ряда кнопок, хэтов, курка и т. д. Все, что не является осью, может быть запрограммировано при помощи команды BTN в соответствии со следующим синтаксисом:

Синтаксис
BTN Button_name ПоследовательностьКлавиш и(или) макрос(ов)

где:

Название_кнопки определяет, какая из кнопок программируется:

На РУД имеются:

10 кнопок: с T1 по T10

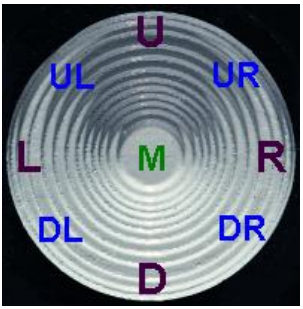
На РУС имеются:

4 хэта: с H1 по H4
4 переключателя: с S1 по S4
Курок на 2 положения: TG1, TG2

Примеры:

```
BTN T3 y                Rem "Вас понял"
BTN S2 Eject            Rem "Катапультирование"
BTN T4 Chaff Flare      Rem Постановка помех
BTN S4 h e l l o        Rem Обратите внимание на пробелы - это не
                        макрос
```

У каждого хэта есть 9 программируемых положений:



Как правило, программируются 4 основных положения. Для Хэта 1:

```
BTN H1U Look_up         Rem Камера вверх
BTN H1R Look_right      Rem Камера вправо
BTN H1D Look_down       Rem Камера вниз
BTN H1L Look_left       Rem Камера влево
```

Однако, промежуточные положения также программируемые:

напр.: `BTN H1UL View_UL Rem Камера влево-вверх`

как и среднее положение:

```
BTN H1M View_forward    Rem Камера вперед
```

Примечания

1. Команда BTN на обязательно должна быть в самом начале строки, но в каждой строке может быть только одна команда BTN. Нельзя указывать название одной и той же кнопки более одного раза в одном и том же файле. То есть:

```
BTN S2 a b c
BTN S3 d e f
```

эти строки правильные, однако если дальше в файле встречается:

```
BTN S3 g h I
```

компилятор выдаст сообщение об ошибке, информируя о наличии повторяющихся кнопок.

2. Между командой BTN и Названием_кнопки должен быть один пробел, то есть:

```
BTNS2
```

содержит ошибку.

3. После BTN Название_кнопки также должен быть один пробел, то есть:

```
BTN S2a b c
```

содержит ошибку.

4. При использовании команды BTN символ или макрос генерируются только один раз, независимо от того, удерживается кнопка или нет. Для многократного генерирования необходимо использовать модификаторы /A (автоповтор) или /H (удержание). Это отличается от прежнего командного синтаксиса TM. *Модификаторы будут рассмотрены в Руководстве далее*.

Дополнительные примечания

Среднее положение хэта заслуживает отдельного разговора. Среднее положение любого хэта может быть запрограммировано, при помощи буквы "M", добавленной к названию хэта. Обратите внимание, что если в командных строках положений хэта вы используете модификаторы /P, /R (более подробно о модификаторах см. ниже), символы по модификатору /R будут генерироваться одновременно с символами команды "M". Таким образом:

```
BTN H1U     /P 1
            /R 2
BTN H1M a
```

при нажатии хэта 1 вверх и последующем отпускании будут сгенерированы следующие символы:

“1” и затем “a” и “2” одновременно. Если вы хотите, чтобы команда H1M исполнялась после того, как будет исполнена команда H1U /R, можно добавить задержку (подробнее см. ниже) в строку H1M:

```
BTN H1M DLY(60) a
```

3.2 Обозначение клавиш Thrustmaster

Напомним синтаксис выражения для кнопки:

Синтаксис
BTN Название_Кнопки Последовательность символов и(или) макросов

Рассмотрим последнюю часть - последовательность символов.

При назначении клавиш клавиатуры через макросы или через файл настроек используется собственная система Thrustmaster для обозначения каждой клавиши. Например, в игре результаты нажатия клавиши "5" на верхней линейке и на цифровой клавиатуре могут быть совершенно разными. В данном примере эти разные клавиши будут обозначаться как "5" и "KP5". Ниже приводится система Thrustmaster для обозначения клавиш на клавиатуре:

ESC	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	
`	1	2	3	4	5	6	7	8	9	0	-	=	BSP
TAB	q	w	e	r	t	y	u	i	o	p	[	]	\
CAPS	a	s	d	f	g	h	j	k	l	;	’	ENT	
LSHF	z	x	c	v	b	n	m	,	.	/	RSHF		
LCTL	LALT	SPC								RALT	RCTL		

PRNTSCRN	SCRLCK	BRK

INS	HOME	PGUP
DEL	END	PGDN

	UARROW	
LARROW	DARROW	RARROW

NUML	KP/	KP*	KP-
KP7	KP8	KP9	KP+
KP4	KP5	KP6	
KP1	KP2	KP3	KPENT
KP0		KP.	

Получить правильные обозначения клавиш можно, воспользовавшись модулем Korgy, виртуальной клавиатурой утилиты Foxy. Данная информация также может быть отображена через редактор Foxy, выбрав пункт "Обозначения клавиш" ("Keyboard Syntax") в пункте меню Help.

Примечания

1. Модифицированные клавиши (*с использованием Shift, ALT или CTRL*) обозначаются как SHF a, ALT b, CTL c. Это не то же самое, что LSHF a, LALT b LCTL c. К примеру, LSHF a соответствует следующим действиям: нажатие левой клавиши Shift, отпускание, и последующее нажатие и отпускание клавиши "a". По умолчанию для модификации клавиш компилятор использует **левые** клавиши Shift, ALT и CTRL.

2. Некоторые символы зарезервированы: ( ) { } < > ; и должны программироваться с помощью SHF:

(= SHF 9
) = SHF 0
{ = SHF [
} = SHF]
< = SHF ,
> = SHF .

3. Рекомендуется всегда использовать комбинацию SHF + буква, а не заглавную букву:

BTN S1 SHF p
BTN S1 P

Оба примера правильные, и выдадут символ "P", но рекомендуется использовать первый вариант.

4. Обозначения клавиш и раскладка клавиатуры соответствуют клавиатуре US Keyboard. Иногда может возникнуть ситуация, когда необходимо воспроизвести символ, существующий лишь в иных раскладках. В этом случае можно напрямую использовать коды USB. Подробнее это объясняется ниже в данном руководстве, хотя, вероятнее всего, использоваться это будет крайне редко.

5. Обозначения клавиш отличаются от предшествующей системы для F22. примеру, из обозначения ряда клавиш был исключен суффикс AUX.

3.3 Макросы и правила их составления

Во Введении были обозначены понятия макрокоманд и файла макрокоманд. Были приведены два примера:

Autopilot = a
Forward_view = F1

Теперь, зная систему обозначения клавиш, вы можете создавать файлы макрокоманд для игры. Программа Foxy содержит ряд полезных утилит, которые помогают создавать файл макрокоманд с правильным синтаксисом и с соблюдением всех правил. Вот эти утилиты: The Macro Wizard, Speedy и Korgy. *См. документацию к программе Foxy для дополнительной информации по этим утилитам.* Прежде чем перейти к правилам составления макросов, позвольте дать вам один совет. Иногда бывает очень утомительно сидеть с картой раскладки клавиатуры вашей игры, переписывая команды в файл. Если в Help-файле к игре есть список клавиатурных команд, попробуйте скопировать их и вставить в файл макрокоманд, а затем отформатируйте текст в соответствии с правилами составления макрокоманд. И еще один момент: очень часто при загрузке файлов настроек компилятор выдает сообщение об ошибке, которая порой неочевидна. Как показывает практика, в 90% случаев ошибка кроется в синтаксисе макрокоманд, поэтому внимательно проверьте файл макрокоманд на предмет правильности синтаксиса и его соответствия правилам.

Итак, правила составления макрокоманд

1. Имена макросов **не могут** содержать пробелов. Используйте нижнюю черту "_" или дефис "-". Таким образом:

```
My macro is = b
```

неверно, тогда как:

```
My_macro_is = b
My-macro-is = b
```

верно.

2. В строке макрокоманды обязательно должны быть пробелы **перед** и **после** знака "=", следующего после имени макроса. Таким образом, обе макрокоманды:

```
Autopilot=  a
Autopilot =a
```

неверны.

3. Следующие символы **нельзя** использовать в именах макрокоманд:

/ = < > { } ( ) ^ , *пробел*

4. Старайтесь избегать написания имен макрокоманд заглавными буквами, напр. RADAR_RANGE_INCREASE. Хотя это допустимо, файл будет намного

легче читаемым, если вы будете использовать строчные буквы. рекомендуется использовать верхний регистр для написания команд TM (напр. MDEF), или сокращений (напр. HUD).

5. В именах макросов не учитывается регистр. Так:

MyMacro = a
mymacro = a

это одно и то же.

6) В названиях макрокоманд нельзя использовать зарезервированные обозначения TM. К ним относятся обозначения клавиш TM, а также другие обозначения, используемые в программных выражениях TM. То есть если в файле настроек имеется:

BTN S2 HOME

и в макро-файле:

HOME = k

компилятор выдаст сообщение об ошибке, так как слово "HOME" является обозначением TM для клавиши HOME.

Дополнительные примечания

Нижеизложенная информация предназначена для действительно продвинутых пользователей.

Макросы могут быть вложены один в другой, то есть вы можете использовать макрокоманду для вызова другой макрокоманды:

Macro_1 = a b c
Macro_2 = Macro_1 d e f

Допускается до 20 вложенных макросов в одной макрокоманде, т. е.:

Macro-1 = a Macro-2
Macro-2 = b Macro-3
Macro-3 = c Macro-4
... и т. д. ...
Macro-20 = d

Но не более 20. Если задать следующее...

```
Macro-1 = a Macro-1
```

…компилятор выдаст сообщение об ошибке: "Слишком длинная цепочка макрокоманд; две макрокоманды могут вызывать сами себя."

3.4 Модификаторы выражений

Выше были рассмотрены простые выражения, такие как:

```
BTN T4 a
```

В результате исполнения этой строки при нажатии и отпускании кнопки T4 на РУД генерируется один символ "a". Однако бывают ситуации, когда нужно не только сгенерировать одиночный символ. На клавиатуре можно нажать и удерживать клавишу для воспроизведения цепочки одинаковых символов с определенным интервалом, можно удерживать несколько клавиш одновременно, и т. д. **Основным показателем успеха программируемого манипулятора является его способность эмулировать то, что может быть получено с использованием клавиатуры**.

Именно для этого служат модификаторы, которые позволяют выйти за рамки воспроизведения одиночных символов.

Модификаторы являются командами, меняющими поведение символов, запрограммированных на определенную кнопку. Они делятся на 5 типов:

1. Ключи /U, /M, /D, /I, /O, /P, /R, /T, /A, /H

```
BTN S4 /H b  Rem Колесные тормоза
BTN T3 /A c f  Rem ДО/ЛТЦ
```

2. Команды задержки (Delay) и повтора (Repeat) DLY( ), RPT( )

```
BTN T6 1 DLY(60) 1 DLY (60) 2 Rem Запрос вектора отхода
BTN S2 RPT(6) c  Rem 6 ДО
```

3. Группировка символов с использованием скобок ( ), { }, < >

```
BTN T2 (a b c)
BTN T3 {a b c}
BTN T4 /P <a b c>
       /R d
```

4. Назначение и работа с кнопками DirectX (Direct Input) DX

```
USE TG1 AS DX1
BTN H2U DX1
USE ALL_DIRECTX_BUTTONS
```

5. Использование скан-кодов KeyDown, KeyUp и USB KD( ), KU( ), USB( )

```
BTN H4U KD(a) DLY(60) KU(a)
BTN H4D /P USB (D51) /R USB (U51) Rem 'Стрелка вниз'
```

3.5 Ключи

Всего существует 10 ключей, которые можно разделить на 3 категории, по их действию:

Увеличение количества программируемых функций:

/U, /M, /D используя переключатель на РУД (T7, T8)
/I, /O используя кнопку S3 на РУС

Разделение макросов на одной кнопке:

/T Переключение
/P, /R Нажать, Отпустить

Повторяющиеся и не повторяющиеся символы:

/A Автоповтор
/H Удержание

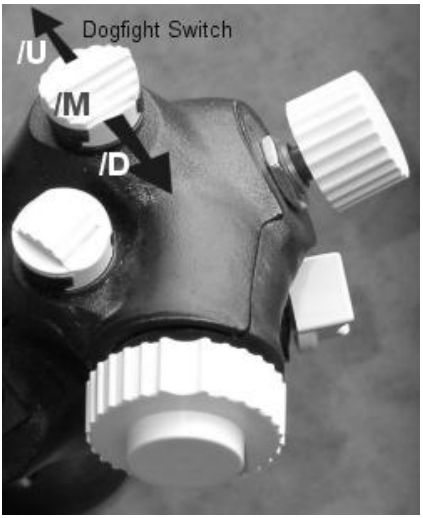
Иерархия и правила использования ключей:

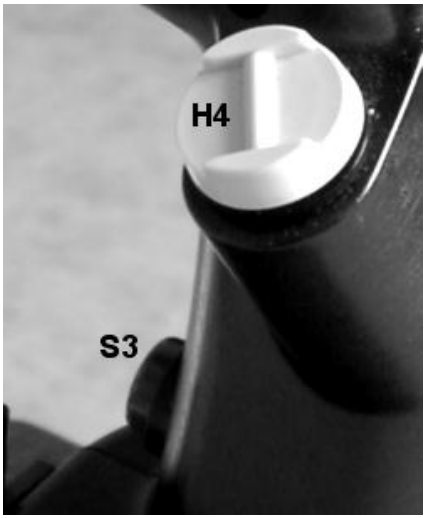
Пользователь должен четко понимать, в каких ситуациях возможно использование ключей, и в каком порядке.

3.5.1 Увеличение количества программируемых функций:

Переключатель "Ближний бой" T7-T8 на РУД имеет три положения: верхнее - (/U), среднее (/M) и нижнее (/D). Кнопка S3 может быть нажата (/I) или отжата (/O), *см. рисунки внизу*. Эти модификаторы могут сочетаться друг с другом для увеличения количества программируемых функций кнопки, хэта или оси максимум в шесть раз.

Ключи
/U, /M, /D используя переключатель "Ближний бой" T7, T8 на РУД
/I, /O используя кнопку S3 на РУС





3.5.1.1 /U, /M, /D - Верхний, средний, нижний регистры

Вы можете увеличить количество программируемых функций кнопки, хэта или цифровой оси, используя положения переключателя "Ближний бой" на РУД, при помощи ключей /U, /M и /D. Например:

BTN H4U /U a
/M b
/D c

Когда хэт 4 нажат вверх, будет сгенерировано следующее:

- символ "a", если переключатель в верхнем (/U) положении
- символ "b", если переключатель в среднем (/M) положении
- символ "c", если переключатель в нижнем (/D) положении

Ключи /U, /M, /D **должны** находиться на разных строках. Если они будут на одной строке, компилятор выдаст сообщение об ошибке. Эти ключи также должны быть расположены только в таком порядке. Таким образом:

```
BTN H4U /D a
        /M b
        /U c
```

приведет к ошибке компиляции.

3.5.1.2 /I, /O - Нажато, отжато

Вы можете увеличить количество программируемых функций кнопки, хэта или цифровой оси, используя кнопку S3, при помощи ключей /I и /O. Например:

```
BTN H4D /I 1
        /O 2
```

Когда хэт 4 нажат вниз, будет сгенерировано следующее:

- символ "1", если кнопка S3 нажата (/I)
- символ "2", если кнопка S3 отжата (/O)

Комбинирование ключей /U, /M, /D с ключами /I, /O в командных строках

Данные ключи могут использоваться в сочетании. Например:

```
BTN H4R /U /I Engage_my_target
           /O Break_right
        /M /I Camera_right
           /O Next_waypoint
        /D /I Engine_right
           /O View_right
```

Теперь нажатие хэта 4 вправо может иметь 6 различных функций, в зависимости от положения переключателя "Ближний бой" (/U, /M, /D) и состояния кнопки S3 (/I, /O).

Примечания

1. Если используются ключи /U, /M, /D, они всегда предшествуют ключам /I, /O.

2. Нельзя использовать ключи /U, /M, /D для программирования рабочих положений переключателя "Ближний бой" (кнопки T7 и T8), так как, очевидно, они используются для определения этих ключей.

3. Ключи /I, /O **должны** быть на разных строках. Это отличается от прежнего синтаксиса TM HOTAS.

4. Команды с ключом /I всегда должны **предшествовать** командам с ключом /O. То есть:

```
BTN H4D /I 1
        /O 2
```

является правильным, тогда как:

```
BTN H4D /O 2
        /I 1

BTN H4D /I 1 /O 2
```

обе эти строчки приведут к ошибке компиляции. Это отличается от прежнего синтаксиса TM HOTAS.

5. Если вы используете ключи /I, /O с кнопкой S3, все команды регистра /O будут игнорироваться. Так же, если вы назначите другую кнопку в качестве модификатора при помощи команды USE *Кнопка* AS SHIFTBTN (*см. ниже*), все команды регистра /O этой кнопки будут игнорироваться. Однако это не приведет к ошибке компиляции.

6. Если файл был создан для джойстика и РУД, но подключен только джойстик, компилятор будет использовать только команды регистра /M для всех кнопок и хэтов, а команды регистров /U и /D будут проигнорированы.

Дополнительные примечания: предотвращение "западания клавиш"

Несколько технических подробностей того, что именно происходит, когда вы меняете состояние кнопки S3 с отжатого на нажатое, и наоборот, если при этом нажата какая-либо кнопка, запрограммированная с ключами /I и /O. Итак, если меняется положение переключателя "Ближний бой", или кнопки S3, контроллер выполняет следующие действия:

1. *Cougar определяет изменение состояния*

2. *Cougar проверяет, какие кнопки нажаты*
3. *Cougar проверяет нажатые кнопки на предмет наличия отличного от текущего действия, запрограммированного для нового состояния ключей.*
4. *Если да, контроллер посылает на выход команду "Отпустить" для предыдущего состояния. В этот момент не исполняются никакие команды нажатия клавиш, однако все иные команды (управление мышью, команды DirectX, команды осей) исполняются.*
5. *Cougar переходит в новое состояние.*

3.5.2 Разделение макросов, запрограммированных на кнопку:

Ключи

/T Переключение между различными макросами на кнопке
/P, /R Действия при нажатии и отпускании кнопки

3.5.2.1 /T - Ключ переключения

Для того, чтобы понять действие ключа /T, посмотрите пример внизу:

```
BTN S2 /T a
       /T b
       /T c
```

Предположим, кнопка S2 нажата 3 раза. При первом нажатии выдается символ "a". При следующем нажатии - символ, "b". И, наконец, при третьем нажатии - символ "c". Если нажать на кнопку еще раз, вновь будет выдан символ "a", и цикл повторится - кнопкой S2 вы будете переключаться между назначенными на нее символами или макросами.

Всего на одно программируемое положение, включая регистры /U, /M, /D, /I и /O, допускается 16 переключений:

```
BTN S2 /U /I 16 переключений
          /O 16 переключений
       /M /I 16 переключений
          /O 16 переключений
       /D /I 16 переключений
          /O 16 переключений
```

Примечания

1. *Ключ переключения **нельзя** использовать **после** ключей /P или /R. То есть:*

Пример 1. BTN TG1 /T /P a
/R b
/T c

Пример 2. BTN TG1 /P /T a /T b
/R c

Пример 3. BTN TG1 /P a
/R /T b /T c

Пример 1 правильный, а 2 и 3 приведут к ошибке компиляции.

2. *Нельзя использовать ключ /T с кнопкой T1, если в файле настроек используется команда* USE T1_SENSITIVITY *(см. ниже).*

3. *Нельзя использовать ключ /T с хэтом джойстика, если в файле настроек используется команда* USE *Хэт№*_SENSITIVITY (*см. ниже*).

4. *Нельзя использовать ключ /T с цифровыми осями (см. ниже).*

5. *Нельзя использовать ключ /T в логическим программировании* (*см. ниже*).

6. *Наличие только одного ключа /T в строке приведет к ошибке компиляции. То есть:*

BTN T4 /T a

недопустимо, так как должно быть более одного ключа /T в командной строке.

7. *Ключи /T могут располагаться как на одной строке, так и на разных. То есть:*

BTN S2 /T a /T b /T c

допускается.

9. *Возможно использование других ключей совместно с* /T, *что было невозможно при программировании предыдущих манипуляторов ТМ. Например:*

BTN S2 /T a /T /H b

допускается.

10. *Нельзя использовать функцию переключения при программировании центрального положения хэта. Пример ниже приведет к ошибке компиляции:*

```
BTN H1M /T a /T b
```

3.5.2.2 Сброс цепочки переключений

Сброс порядка переключений в начало списка командной строки осуществляется следующим образом:

Синтаксис

```
RESET_TOGGLES
```

Таким образом, если есть выражение:

```
BTN S2 /I RESET_TOGGLES
       /O /T 1 /T 2 /T 3 /T 4 /T 5 /T 6 /T 7 /T 8 /T 9 /T 0
```

и вы нажали кнопку S2 семь раз, сгенерировав символ "7", а теперь желаете быстро вернуться в начало цепочки, достаточно нажать S2, удерживая S3 (что включает исполнение команд с ключом /I).

Однако помните, что в данном примере нажатие на S2 с удержанием S3 не приведет к генерированию никаких символов – только к сбросу в начало цепочки переключений. Для того, чтобы сгенерировать "1", необходимо повторно нажать на S2 при отпущенной S3.

Примечания

1. *Эта команда **должна** находиться сразу после ключей /I или /O, при этом цепочка переключений должна быть прописана после соответствующего второго ключа /I или /O. Таким образом следующее выражение приведет к ошибке компиляции:*

```
BTN S4 /U RESET_TOGGLES
       /M /T 1 /T 2 /T 3 /T 4 /T 5 /T 6 /T 7 /T 8 /T 9 /T 0
       /D Some_macro
```

Однако данная строка верна:

```
BTN S4 /U Some_macro
       /M /I RESET_TOGGLES
```

```
/O /T 1 /T 2 /T 3 /T 4 /T 5 /T 6 /T 7 /T 8 /T 9 /T 0
/D Some_macro
```

2. *В строке, содержащей команду* RESET_TOGGLES, *не может быть больше ничего.*

3.5.2.3 Инверсия порядка переключений

Если необходимо инвертировать порядок переключений, это можно сделать следующим образом:

Синтаксис
REVERSE_TOGGLES

```
BTN S2 /I REVERSE_TOGGLES
/O /T 1 /T 2 /T 3 /T 4 /T 5 /T 6 /T 7 /T 8 /T 9 /T 0
```

В обычном случае нажатие кнопки S3 последовательно переключает символы от 1 до 0. Если нажать S2, удерживая S3, то последующие нажатия S2 будут перебирать символы в обратном порядке с того места, в котором была исполнена команда инверсии.

Примечания

1. *Эта команда* ***должна*** *находиться сразу после ключей* /I *или* /O, *при этом цепочка переключений должна быть прописана после соответствующего второго ключа* /I *или* /O. *Таким образом следующая строка приведет к ошибке компиляции:*

```
BTN S4 /U REVERSE_TOGGLES
/M /T 1 /T 2 /T 3 /T 4 /T 5 /T 6 /T 7 /T 8 /T 9 /T 0
/D Some_macro
```

Однако данная строка верна:

```
BTN S4 /U Some_macro
/M /I REVERSE_TOGGLES
/O /T 1 /T 2 /T 3 /T 4 /T 5 /T 6 /T 7 /T 8 /T 9 /T 0
/D Some_macro
```

2. *В строке с командой* REVERSE_TOGGLES *не может быть больше ничего. То есть если изменить вышеприведенный пример:*

```
BTN S4 /U Some_macro
```

```
/M /I /T 1 /T 2 /T 3 /T 4 /T 5 /T 6 /T 7 /T 8 /T 9 /T 0
   /O REVERSE_TOGGLES Macro_here_generates_error
 /D Some_macro
```

макрокоманда в строке после REVERSE_TOGGLES приведет к ошибке компиляции.

3.5.2.4 /P, /R - Нажатие - отпускание

Ключ /P указывает, что следующая после него макрокоманда выполняется при нажатии кнопки или переключателя. Ключ /R указывает, что следующая после него макрокоманда выполняется при отпускании кнопки или переключателя.

Например:

```
BTN S2 /P Chaff
       /R Flare
```

В данном примере для кнопки BTN S2 используются ключи нажатия и отпускания. При нажатии будет выполнена макрокоманда Chaff (ДО), а затем команда Flare (ЛТЦ) при отпускании.

Примечания

Ключи /R и /P должны всегда использоваться вместе. Если нет командной строки с ключом /P, использование ключа /R приведет к ошибке компиляции, и наоборот.

Дополнительные примечания

1. *Если данные ключи используются для программирования положения хэта, вместе с командой программирования центрального положения, то символы по ключу /R будут генерироваться одновременно с символами по команде BTN HxM. Таким образом команды:*

```
BTN H1U /P 1 /R 2
BTN H1M a
```

*сгенерируют **"1"** при нажатии хэта вверх и **"a 2"** одновременно при отпускании.*

2. *Если эти ключи используются в строке программирования центрального положения хэта, то нажатие хэта в любую сторону приведет к генерированию символов по команде HxM /R одновременно с*

символами, запрограммированными на соответствующее положение хэта. Таким образом команды:

```
BTN H1U 1
BTN H1M /P a /R b
```

*сгенерируют при нажатии хэта 1 вверх **"b 1"** одновременно, и затем "a" при отпускании.*

3. *Есть потенциальная опасность вызвать "западание" (постоянный повтор) клавиши при использовании ключей /P /R. Рассмотрим пример:*

```
BTN S4 /P DLY(2000) KD (p)
       /R KU (p)
```

Нажатие на кнопку S4 сначала вызовет задержку в 2 секунды, а затем эмулирует событие "нажатие клавиши "p"" (Key Down) на клавиатуре, а отпускание кнопки S4 эмулирует событие "отпускание клавиши "p"" (Key Up) на клавиатуре.

Однако что произойдет, если кнопка S4 будет отпущена до истечения времени задержки?

Хотя задержка продолжается, и код нажатия "p" будет послан только после истечения 2 секунд, код отпускания "p" будет сгенерирован и обработан компьютером до кода нажатия.

Таким образом, так как команда "нажать клавишу" была выдана компьютеру после команды "отпустить клавишу", клавиша "p" окажется "запавшей".

Избежать подобной ситуации можно двумя способами. Во-первых, осмысленно программировать и использовать HOTAS. Если вы знаете, что необходимо удерживать кнопку нажатой определенное время – значит, надо ее удерживать. Это позволит всегда избежать подобных проблем. Вторым способом является заключение командной строки /P в угловые скобки < >:

```
BTN S4 /P < DLY(2000) KD (p) >
       /R KU (p)
```

что обеспечивает принудительное исполнение всех команд внутри угловых скобок до того, как может быть исполнена командная строка /R, даже если кнопка будет физически отпущена до этого. Однако необходимо отметить, что все макросы и команды на всех других кнопок, если они уже не исполняются, приостанавливаются до

исполнения команды в угловых скобках. Будьте внимательны при использовании!

3.5.3 Повторяющиеся и неповторяющиеся символы:

Ключи

/A Автоповтор
/H Удержание

3.5.3.1 Неповторяющиеся символы

Работа макросов и одиночных символов в командной строке кнопок изменилось по сравнению с прежним синтаксисом TM. По умолчанию, все символы и макросы на кнопках или цифровых осях не повторяются. Генерируется одно нажатие и отпускание на клавишу или исполнение макрокоманды. Для тех, кто пользовался предыдущими моделями TM HOTAS: как будто перед каждой командой в файле, где нет иных ключей, стоит невидимый ключ /N.

Таким образом, необходимость в ключе /N отпала – и, если даже он присутствует в файле, компилятор его игнорирует. Рекомендуется удалить их из старых файлов.

3.5.3.2 /A - Автоповтор

Использование этого ключа приводит к непрерывному воспроизведению символов на кнопке или цифровой оси вплоть до отпускания кнопки.

BTN S2 /A Fire_Missiles

Нажатие кнопки S2 приведет к многократному повторению макроса Fire_Missiles. Это соответствует быстрому многократному нажатию **и** отпусканию клавиши на клавиатуре. Если в этот момент нажимаются другие кнопки, воспроизводимые ими символы "вливаются" в поток повторов, то есть нажатие другой кнопки не прерывает потока. Надо отметить, что макросы, следующие за ключом /A, могут состоять из нескольких символов, и содержать команды DLY, RPT и т. д. Помимо этого, ключ /A заменяет элемент предыдущего синтаксиса TM – заключение символа в круглые скобки для его автоповтора.

3.5.3.3 /H - Удержание

BTN S4 /H b Rem Колесные тормоза

Действие ключа /H сравнимо с нажатием и удержанием клавиши на клавиатуре (*однако см. Примечания ниже*). К примеру, во многих авиасимуляторах для использования колесных тормозов нужно нажать и удерживать клавишу "b". Как только клавиша отпущена, колесные тормоза выключаются.

Дополнительные примеры:

```
BTN S4 /H b  Rem Колесные тормоза
BTN T6 /H Wheelbrakes
BTN T1 /H ALT F6
```

Ключ /H можно также использовать и в более сложных командных строках. В примере ниже генерируется одиночный символ "c", после чего удерживается клавиша "f":

```
BTN S4 /H c f Rem 1 ДО, много ЛТЦ
```

В то же время:

```
BTN S4 /H {c f} Rem Сброс ДО и ЛТЦ
```

генерирует коды KD(c) KD(f) до того момента, пока кнопка не будет отпущена, после чего генерируются коды KU(c) KU(f). (*см. примечания ниже по командам с кодами KD и KU.*) Обратите также внимание на следующий момент:

```
BTN S4 /H {C f}
```

Теперь вспомним, что "C" на самом деле является "SHF c", следовательно, это эквивалентно удержанию клавиши LSHF, и, очевидно, "f" на самом деле будет "F". И действительно, на клавиатуре тоже невозможно воспроизвести символ "f", если вы удерживаете SHIFT.

Преимущества использования ключа /H с несколькими символами может быть продемонстрировано на следующем примере, где вы можете выбрать вооружение и незамедлительно начать непрерывную стрельбу:

```
BTN S4 /H Select_Rockets DLY(600) Fire Rem Выбрать НУРС и открыть
огонь
```

Примечания

1. *Очень важно понять одно существенное различие между командами /H и /A. Обычно, когда на клавиатуре нажимается и удерживается, скажем, клавиша "a", воспроизводится символ "a", затем идет короткая задержка,*

после чего воспроизводится последовательность символов "a", пока кнопка остается нажатой:

a *задержка* aaaaaaaaaaaaaaaa

Именно это происходит с ключом /H. *То есть если исполнить:*

BTN T3 /H a

произойдет именно это. Если надо убрать задержку между первым и последующими символами, можно использовать ключ /A:

BTN T2 /A b

Результатом будет "bbbbbbbbbbbbbbbbbbb" *без задержки.*

2. *Частота, с которой генерируются символы по ключу* /A, *определяется при помощи конфигурационной команды* USE RATE (время_в_мс). *Если такой команды в файле нет, то компилятор сгенерирует команду* USE RATE (0), *и частота генерации будет зависеть от частоты автоповтора клавиатуры.*

3. *Ключи* /H *или* /A *не могут использоваться при программировании среднего положения хэта, или после ключа* /R.

3.5.4 Правила и иерархия ключей

При использовании ключей (напр. /T, /P, /U), необходимо следовать ряду базовых правил их расположения в командной строке.

3.5.4.1 Правила использования ключей

1. Они должны располагаться после кодов кнопок и переключателей (BTN S2 /H)

2. Они должны располагаться до первого макроса или символа (BTN S2 /H SomeMacro), за исключением /T, который может появляться несколько раз в одной строке
3. До и после ключа должно быть по одному пробелу
4. В случае, если в командной строке используются ключи нескольких категорий, они должны располагаться в особом порядке (*см. ниже Иерархия ключей"*).
5. Ключи /U /M /D должны быть на разных строчках.
6. Ключи /I /O должны быть на разных строчках.

3.5.4.2 Иерархия ключей

При использовании нескольких ключей для программирования одной кнопки очень важно использовать правильную иерархию. Ключи должны появляться в следующем порядке:

1. Если используются ключи /U /M /D, они предшествуют всем остальным ключам.
2. Следующими должны располагаться ключи /I и /O.
3. Ключ /T всегда предшествует ключам /P /R.
4. Ключи /P и /R всегда используются после вышеупомянутых.
5. Ключи /H и /A всегда располагаются последними.

Рассмотрим примеры:

```
BTN S4 /U /I macro6
          /O macro7
       /M /P macro8 /R macro9
       /D /T a
          /T /H b c
          /T /A d DLY(30) e DLY(30) f

BTN S2 /I /T  /P macro1 /R macro2
        /T /P macro3 /R macro4
      /O /H macro5
```

Примечания

Командная строка может вообще не содержать макросов или символов, и нет необходимости использовать "пустой" символ, как это было в прежнем TM HOTAS. То есть данная строка:

```
BTN S1  /I Drop_Stores
        /O
```

не приведет к ошибке компиляции и является допустимой.

3.6 Команды задержки и повтора

Команды
DLY (задержка) RPT (число)

где:

задержка - время в миллисекундах (1 секунда = 1000 миллисекунд). Разрешенный диапазон значений - от 0 до примерно 82 800 000. (*Для справки 82 800 000 миллисекунд равно 23 часам*)

Число - количество повторов, максимальное допустимое значение которого зависит от длинны макрокоманды и варьируется в пределах от 2 до 127.

Рассмотри примеры, иллюстрирующие работу данных команд.

3.6.1 Выражения с командой DLY( )

BTN T6 1 DLY(60) 1 DLY (60) 2 Rem Request Vector For Recovery TAW (Запрос курса отхода)

Эта строка приведет к генерации символов "1 1 2", с задержкой в 60 мс между ними. Это может понадобиться в ряде ситуаций. Так, например, в современных сложных авиасимуляторах очень часто для выполнения какой-либо функции требуется нажатие нескольких клавиш. Обычно в авиасимуляторах речь идет об эфирных переговорах, реализованных через систему вложенных меню. Пример из Falcon 4:

VectorToHomePlate = q DLY(60) q DLY(60) 6 (запрос курса на базу)

Если бы в макрокоманде не было задержек:

VectorToHomePlate = q q 6

существует высокая вероятность, что игра не распознает все три символа, так как контроллер воспроизведет их слишком быстро, в то время как игра также занята собственными расчетами и обсчетом графики. Введение задержки между символами позволяет замедлить их воспроизведение, делая его более похожим на физическое нажатие клавиш на клавиатуре, и игра воспримет все символы. Ситуация несколько усложняется, если в файле используется конфигурационная команда USE RATE (время) (*см. ниже*). Команда USE RATE (время) определяет, с какой частотой генерируются непрерывно воспроизводимые символы. Таким образом:

```
USE RATE (60)
BTN S1 q q 6
```

это в точности то же самое, что:

```
USE RATE (0)
BTN S1 q DLY(60) q DLY(60) 6
```

Помните, что компилятор вставляет по умолчанию USE RATE(0), если такой команды нет в файле.

При использовании команды DLY не нужно удерживать кнопку нажатой в течение всего времени исполнения командной строки. Например:

BTN S2 h DLY (2000) e DLY (2000) l DLY (2000) l DLY (2000) o

В результате выводится по буквам слово "hello", с 2-секундной задержкой между буквами. Для исполнения всей строки не нужно удерживать кнопку джойстика нажатой. Обратите внимание, что если вы дважды нажмете кнопку S2 а течение 2 секунд, результат будет следующим:

"hheellllloo"

Это объясняется особенностью HOTAS Cougar – а именно, способностью осуществлять параллельную обработку команд. Чтобы этого избежать, либо будьте внимательны при использовании, либо заключите команды в угловые скобки < >.

3.6.2 Выражения с командой RPT( )

BTN S2 RPT(6) c Rem Сброс 6 ДО … самое время!

Использование команды RPT позволяет повторить указанное количество раз символы или макросы после команды RPT (n), где n = число повторов. Повторяется только символ или макрос, стоящий **сразу после** команды RPT (т.е. повтор может быть применен к любому одиночному символу или макросу, или группе таковых, заключенной в скобки).

Так, в примере выше нажатие на кнопку S2 генерирует 6 символов "c" с частотой, определенной конфигурационной командой USE RATE(x) в файле.

Дополнительные примеры:

```
BTN S1 RPT(10) a b
```

генерирует 10 символов "a", после которых следует 1 символ "b".

```
BTN S1 RPT(10) (a b)
```

генерирует символы "a b" десять раз.

```
BTN S1 /A RPT(10) (a DLY(60)) DLY(2000)
```

генерирует 10 символов "a" с задержкой в 60 миллисекунд между символами, затем 2-секундная задержка, а затем повторяется вся комбинация.

Команды RPT могут использоваться совместно или быть вложенными друг в друга:

```
BTN S1 RPT(10) a RPT(10) b

BTN S1 RPT(10) (a RPT(10) b)
```

Если определение макрокоманды выглядит так:

```
Macro1 = a b c
```

и командная строка выглядит так:

```
BTN S2 RPT (3) Macro1
```

при нажатии S2 результат будет: a a a b c

Чтобы это исправить, необходимо заключить в скобки либо макрокоманду, либо символы в ее определении:

```
BTN S2 RPT (3) (Macro1)
```

или

```
BTN S2 RPT (3) Macro1
```

где

```
Macro1 = (a b c)
```

3.7 Группирование символов – использование скобок

Модификаторы

```
BTN T2 (a b c)
BTN T3 {a b c}
BTN T4  /P <a b c>
        /R d
```

Прежде, чем перейти к рассмотрению использования скобок для группировки символов, необходимо запомнить одно важное правило. Скобки **( ) { }** и **< > зарезервированы** для программного синтаксиса TM. Таким образом, эти символы не могут быть напрямую назначены на кнопки или включены в макрокоманды. Вместо этого необходимо использовать обозначение верхнего регистра:

(= SHF **9**
) = SHF **0**
{ = SHF **[**
} = SHF **]**
< = SHF **,**
> = SHF **.**

То есть макрокоманда:

Left_ToeBrake = <

приведет к ошибке компиляции, а верным будет:

Left_ToeBrake = SHF.

3.7.1 ( ) Круглые скобки

Круглые скобки используются в целом ряде выражений программного языка TM, например DLY ( ), RPT ( ), USB ( ), KU ( ), командах цифровых режимах осей, и т. д., для группировки элементов. На предыдущей странице был приведен пример их использования в сложной командной строке с командами RPT и DLY:

BTN S1 /A RPT(10) (a DLY(60)) DLY(2000)

Важно отметить, что в синтаксисе прежнего языка TM заключение в круглые скобки символов или макросов вызывало их автоповтор. Это не работает с HOTAS Cougar, так как в новом языке используется ключ /A, и круглые скобки используются для группировки символов, макросов и команд. Следует отметить, однако, что когда круглые скобки используются со значениями параметров, варианты их написания интерпретируются компилятором одинаково: DLY (20), DLY(20), DLY (20) и т. д.

3.7.2 { } Фигурные скобки

Группировка символов с помощью фигурных скобок позволяет генерировать их так, как если бы все соответствующие клавиши были нажаты

одновременно. Группы внутри фигурных скобок интерпретируются при обработке как единое целое.

Пример:

BTN S4 {a b c}

генерируется как нажатие "a", нажатие "b", нажатие "c", отпускание "a", отпускание "b", отпускание "c", как если бы вы нажали все эти клавиши по очереди, и затем по очереди отпустили. (как в ситуации с CTRL ALT DEL ☺)

Группы, заключенные в фигурные скобки, могут использоваться с ключом /H, в этом случае все клавиши в группе будут оставаться в нажатом состоянии, пока не будет отпущена кнопка. Например:

BTN S4 /H {CTL ALT DEL}

вполне допустимо (*и часто необходимо!*)

Дополнительные примечания

1.) С фигурными скобками нельзя использовать команду DLY.

2.) Реализация выражений с { } на USB-устройствах отличается от таковой для gameport. С предыдущими моделями TM HOTAS символы, заключенные в "{ }", воспроизводились в том порядке, в котором они были расположены. Таким образом:

BTN S4 {a b c}

выдавало нажатие "a", нажатие "b", нажатие "c", отпускание "a", отпускание "b", отпускание "c". Однако в случае с USB порядок символов определяется их USB-кодами, и для букв соответствует алфавитному порядку. Поэтому:

BTN S4 {c b a}

все равно нажатие "a", нажатие "b", нажатие "c", отпускание "a", отпускание "b", отпускание "c". Однако, на самом деле все эти символы воспроизводятся **одновременно, в рамках одного цикла,** то есть ситуация скорее соответствует именно одновременному нажатию и отпусканию клавиш “a”, “b” и “c” на клавиатуре. Если необходимо действительно их разделить, чтобы их воспроизведение происходило в нужном порядке, нужно использовать команды KD и KU, например:

BTN S4 KD(c) KD(b) KD(a) KU(c) KU(b) KU(a)

3.7.3 < > Угловые скобки

Угловые скобки являются новым элементом синтаксиса HOTAS Cougar. Все, что заключено в угловые скобки, исполняется контроллером до конца в принудительном порядке, до того, как будут исполнены все другие команды. Например, рассмотрим следующее выражение:

```
BTN H1D q DLY(60) q DLY(60) 6 Rem Запрос курса на базу, Falcon 4
```

Будет весьма неприятно, если в это время будет нажата какая-то другая кнопка, так как данный эфирный запрос в Falcon 4 превратится во что-нибудь совершенно иное. Добавим в выражение угловые скобки < >:

```
BTN H1D <q DLY(60) q DLY(60) 6> Rem Запрос курса на базу
```

Когда хэт 1 нажат вниз, все символы данного выражения генерируются в принудительном порядке до того, как будет обработано любое другое нажатие. Этот прием полезен для предотвращения "залипания" клавиш:

```
BTN T4 /P < DLY(2000) KD (b) >
          /R KU (b)
```

Нажатие кнопки T4 в данном случае гарантирует исполнение всего выражения по ключу /P, даже если кнопка будет отпущена до окончания исполнения. Таким образом, команда "KU (b)" будет выполнена только после команды нажатия "b". В ином случае команда "KU (b)" была бы выполнена незамедлительно при отпускании кнопки T4, возможно, раньше команды "KD (b)", и клавиша "b", получив команду нажатия, так и не получила бы команду отпускания, то есть как бы "залипла" в нажатом положении.

Примечания

1. *Угловые скобки < > нельзя использовать внутри фигурных { }, таким образом:*

```
BTN S2 { < a b > }
```

приведет к ошибке компиляции.

2. *Многократное вложение угловых скобок < > не допускается, таким образом:*

```
BTN T4 < < a b > >
```

приведет к ошибке компиляции.

3. *Угловые скобки < > не обязательно должны заключать все выражение, то есть:*

BTN S1 a b <c d > e f

допускается.

4. *Если имеется выражение:*

BTN S2 /H <a b c>

как только будут сгенерированы символы “a” и “b”, результатом данного выражения станет удержание клавиши "c". Помните, что выражение с ключом /H, *в которое входит несколько символов, в результате приведет к удержанию последнего символа.*

5. *Угловые скобки < > действуют следующим образом: если в этот момент уже выполняется какое-либо выражение, то оно продолжает выполняться. То есть если вы держите нажатой кнопку, запрограммированную с ключом* /H, *выражение будет продолжать исполняться, даже когда вы нажмете кнопку, запрограммированную с < >. Все остальные команды нажатия кнопок сохраняются в буфере, и будут исполнены, как только закончится выполнение команд внутри < >. Соответственно, нужно быть осторожным при использовании < >, и стараться не включать внутрь команды* DLY ( ) *с большим временем задержки, так как существует опасность "подвесить" контроллер. Более подробно об это см. ниже в разделе "Устранение проблем".*

Дополнительные примечания

Один из вышеприведенных примеров:

BTN S4 KD(c) KD(b) KD(a) KU(c) KU(b) KU(a)

В данном выражении каждая команда нажатия (KeyDown) *и отпускания* (KeyUp) *выполняется отдельным циклом, всего 6 циклов. Если в файле присутствует конфигурационная команда* RATE, *пользователь может предпочесть, чтобы все команды* KeyUp *выполнялись одновременно, а не через интервалы, заданные командой* RATE. *Это достигается следующим образом:*

BTN S4 KD(c) KD(b) KD(a) KU({c b a})

где все команды KeyUp *исполняются быстрее, за 1 цикл.*

3.8 Назначение кнопок DirectX (DirectInput) и работа с ними

В случае с простыми джойстиками, имеющими, скажем, только курок, в игре нажатие на этот курок открывает огонь, и не потому, что кто-то его запрограммировал, а потому, что Windows сообщили игре о том, что у джойстика есть курок, и игра самостоятельно назначила ему типичную функцию. Курок был обозначен как кнопка DirectX - то есть кнопка, функция которой назначается самой игрой.

Конфигурационное выражение

USE *идентификатор_кнопки* или *логический_флаг* AS DX*n*

Синтаксис команды

BTN *button_identifier* DX*n*

где:

идентификатор_кнопки - это H1U, T6, S2 и т. д.
логический_флаг - это флаг с X1 по X32 (*описано далее в руководстве*)
n - номер с 1 по 28

HOTAS Cougar (считая новые педали) состоит из 10 аналоговых осей, 28 кнопок и одного переключателя видов (Hat 1). Если Кугуар работает в режиме Windows, всем кнопкам могут быть назначены функции внутри самой игры, благодаря тому, что Windows информирует игру о возможностях подключенного игрового манипулятора - осях, кнопках, POV и т. д. При создании файла настроек можно при помощи конфигурационных выражений указать Windows, что именно говорить игре.

В программируемом режиме, по умолчанию, ни одна кнопка не видится игрой как кнопка DirectX. Если вы хотите назначить функции каким-либо кнопкам прямо из игры, нужно показать их. Чаще всего именно курок назначается как кнопка DirectX:

```
USE TG1 AS DX1
```

Больше ничего делать не надо - курок будет виден в игре, и ему будет назначена функция, как правило, огонь из пушек или запуск ракет. Еще примеры:

```
USE H1U AS DX2
USE X4 AS DX3    Rem использован логический флаг - см. далее
```

```
USE T4 AS DX5
```

Можно также включить команды кнопок DirectX в другие выражения, и делать всякие интересные вещи:

```
BTN S2 /H a DLY(2000) DX2
```

В данном примере при нажатии и удержании кнопки S2 будет сгенерирован символ "a", а затем, через 2 сек. задержки, будет нажата кнопка DX2. В данном случае мы "почти" определяем S2 как DX2, потому что мы не использовали команды USE. "Почти", потому что это - не то же самое, что и:

```
USE S2 AS DX2
```

так как в этом случае сигнал кнопки DX2 был бы сгенерирован сразу при нажатии S2, тогда как в нашем примере DX2 выполняется только после задержки DLY(2000).

3.8.1 Выражение USE ALL_DIRECTX_BUTTONS

Выше описывалось, как назначить отдельные кнопки кнопками DirectX. Теперь ознакомимся с конфигурационным выражением USE ALL_DIRECTX_BUTTONS.

Конфигурационное выражение

```
USE ALL_DIRECTX_BUTTONS
```

Это выражение назначает все кнопки как кнопки DirectX. То есть в файле настроек с таким выражением все незапрограммированные кнопки будут видны, как кнопки DirectX, но при этом по-прежнему доступны все команды изменения кривых отклика, эмуляции мыши и т. д. Помните при этом, что данное выражение отменяет все настройки по умолчанию, заданные в Foxy.

То есть можно составить очень простой файл:

```
Rem Назначить все кнопки как кнопки DirectX
USE ALL_DIRECTX_BUTTONS
Rem Назначить микроджойстик для управления мышью (см. далее в руководстве)
USE MTYPE A3
Rem Назначить Hat1 на джойстике как переключатель видов POV
USE HAT1 AS POV
```

Загрузив его в контроллер, получаем джойстик и сектор газа, с хэтом 1 как POV, всеми кнопками как DirectX, и с мышкой на микроджойстике. Затем

можно добавлять выражения для кнопок, постепенно создавая конфигурационный файл.

Примечания

1. *Эти новые команды и выражения заменяют выражения* PORT Bx IS *для прежних манипуляторов TM HOTAS.*

2. *Таблица назначения номеров DirectX для кнопок HOTAS Cougar по умолчанию в режиме Windows:*

DX	BTN ID	DX	BTN ID	DX	BTN ID	DX	BTN ID
1	TG1	8	H2R	15	H4U	22	T4
2	S2	9	H2D	16	H4R	23	T5
3	S3	10	H2L	17	H4D	24	T6
4	S4	11	H3U	18	H4L	25	T7
5	S1	12	H3R	19	T1	26	T8
6	TG2	13	H3D	20	T3	27	T9
7	H2U	14	H3L	21	T2	28	T10

при этом Hat1 *по умолчанию назначается как POV (переключатель видов).*

3. *Если в файле настроек использовано конфигурационное выражение* USE ALL_DIRECTX_BUTTONS*, и вы запрограммировали одну из кнопок, эта кнопка не будет видна как кнопка DirectX. Компилятор интерпретирует выражение* USE ALL_DIRECTX_BUTTONS *следующим образом::*

```
BTN TG1 /H DX1
BTN S2 /H DX2
```

и т. д. Если компилятор находит в файле командное выражение для одной из кнопок, он не назначает ее как кнопку DirectX.

4. *Если вы используете выражения* USE ALL_DIRECTX_BUTTONS*, или любое сочетание выражений* USE MTYPE *или* USE HATn AS POV (*см. далее), эти выражения* ***должны*** *идти* ***перед*** *выражениями для кнопок, иначе компилятор выдаст ошибку. Помните, что в структуре файла конфигурационные выражения всегда должны идти перед выражениями кнопок или осей.*

5. *Выражение* USE MTYPE *(подробнее о нем - далее в руководстве) позволяет назначать кнопки T1 и T6 как правую и левую кнопки мыши, в*

зависимости от типа выражения MTYPE. *Таким образом, если выражение* USE MTYPE *используется вместе с* USE ALL_DIRECTX_BUTTONS, *и если в выражении* MTYPE *какие-то кнопки назначаются как кнопки мыши, то эти кнопки не назначаются, как кнопки DirectX.*

В зависимости от типа выражения MTYPE (A1 - A5), *кнопки мыши назначаются следующим образом:*

A1: *T1 = Левая кнопка, T6 = Правая кнопка*
A2: *T1 = Правая кнопка, T6 = Левая кнопка*
A3: *T1 = Левая кнопка*
A4: *T6 = Левая кнопка*
A5: *Кнопки мыши не назначаются.*

6. *Если вы назначаете другой хэт как переключатель видов (см. далее в руководстве), например:*

USE HAT3 AS POV

компилятор не назначает положения этого хэта как кнопки DirectX.

Дополнительные примечания

1. *Рассмотрим один из вышеприведенных примеров:*

```
USE TG1 AS DX1
```

компилятор транслирует это выражение в следующее:

```
BTN TG1   /P KD (DX1)
          /R KU (DX1)
```

Мы знаем, что можно использовать команды KD *и* KU *для разделения действий по нажатию и отпусканию кнопки DirectX. Рассмотрим забавный пример:*

```
BTN TG1   /I   /A KD (DX1) DLY(50) KU (DX1) DLY (200)
          /O /H DX1
```

Теперь, если в игре огонь из пушек назначен на кнопку DirectX 1, то при использовании такого выражения при отпущенной кнопке S3 пушки будут просто стрелять, а при нажатой - стрелять короткими очередями.

3.9 Использование кодов KD, KU и скан-кодов USB

Иногда может возникнуть потребность более полно контролировать процесс нажатия и отпускания клавиш, или послать прямой скан-код для эмуляции специальных символов, доступных только в неамериканских раскладках клавиатуры. Этого можно достичь следующим образом:

Синтаксис

```
KD(символ клавиатуры/кнопки DX/кнопки мыши)
KU(символ клавиатуры/кнопки DX/кнопки мыши)

USB(действие_с_клавишей_код_HID)
```

3.9.1 KD, KU

Данные выражения используются для программного управления действиями, которые обычно совершаются при нажатии клавиши на клавиатуре - то есть, очевидно, сначала клавиша нажимается (**KD**), а затем она отпускается (**KU**). Иногда может потребоваться выполнить определенные действия между этими двумя событиями, например:

```
BTN H1U KD(UARROW) DLY(20) KU(UARROW)
```

В этом примере нажатие хэта 1 вверх приводит к нажатию стрелки вверх на 20 миллисекунд, после чего клавиша отпускается. Команды KD и KU могут использоваться с любыми клавишами при использовании правильного обозначения клавиш TM.

В выражениях с KD, KU можно также комбинировать символы.

```
BTN T4 KD(a b c) DLY(20) KU(a b c)
```

KD и KU могут также использоваться с кнопками DirectX, кнопками мыши и логическими флагами (*логические флаги описаны далее в руководстве*), например:

```
BTN T6 KD(MOUSE_LB) DLY(2000) KU (MOUSE_LB)
```

Нажатие на кнопку T6 эмулирует нажатие левой кнопки мыши на 2 секунды, и ее отпускание.

3.9.2 Программирование USB

Существует возможность посылать прямые USB-коды для эмуляции любых символов, которые не поддерживаются в синтаксисе TM. Это может оказаться особенно полезным на неамериканских раскладках клавиатуры. USB-коды приведены в таблице в Приложении 3 к данному Руководству, и для каждого кода символа используется либо “D” для обозначения нажатия, либо “U” для отпускания. Пример:

```
BTN T3 /P USB (D51) /R USB (U51) Rem 'Стрелка вниз'
BTN T4 USB (DE1 D04 UE1 U04) Rem ‘Shift a’
```

В данном примере все USB-коды в разных циклах. Если вы хотите, чтобы все действия с клавишами выполнялись одновременно, используйте фигурные скобки. Пример:

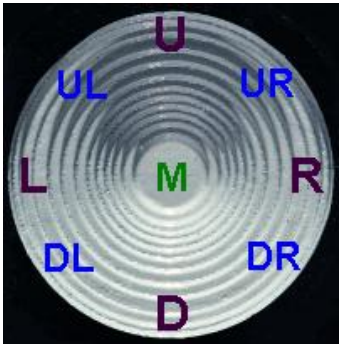
```
BTN T4 USB (DE1 D04) DLY (2000) USB ({UE1 U04}) Rem ‘Shift a’
```

В данном случае отпускание клавиш Left Shift и ‘a’ произойдет одновременно. Необходимо отметить при этом, что задержка между циклами очень мала - около 30 миллисекунд, так что вряд ли вы сможете заметить разницу.

4. Программирование хэтов

4.1 Программирование хэтов джойстика

4.1.1 Программируемые положения хэтов



Хэты имеют 9 программируемых положений, хотя чаще всего вы будете программировать 4 основных направления. Пример для HAT 1:

BTN H1U Look_up
BTN H1R Look_right
BTN H1D Look_down
BTN H1L Look_left

Однако, промежуточные положения также программируемы:

BTN H1UL View_UL
BTN H1UR View_UR
BTN H1DL View_DL
BTN H1DR View_DR

как и центральное положение:

BTN H1M View_forward

Важно отметить, что все эти программируемые положения являются отдельными, и промежуточные положения по умолчанию не являются комбинацией того, что было запрограммировано на соседние основные положения. То есть если вы запрограммировали на основные положения клавиши 8, 2, 4 и 6 дополнительной клавиатуры, то перемещение хэта вправо-вверх не приведет к одновременному нажатию 8 и 6, или к нажатию 9. Оно вообще ни к чему не приведет. Естественно, все зависит от того, насколько точно вы будете отклонять хэт в промежуточные положения.

4.1.2 4-позиционные и 8-позиционные хэты: USE HatID FORCED_CORNERS

Если нужно заставить хэт генерировать в промежуточных положениях сочетание команд соседних основных положений, для этого используется следующее конфигурационное выражение:

Конфигурационное выражение
USE *HatID* FORCED_CORNERS

где: *HatID* - один из хэтов: HAT1, HAT2, HAT3, HAT4
напр. USE HAT1 FORCED_CORNERS

(*Для того, чтобы обеспечить более уверенное отслеживание промежуточных положений, можно использовать выражение USE HatID_SENSITIVITY(nnnn), описанное далее в руководстве*)

Любой хэт может быть сконфигурирован как обычные программируемые кнопки, как мышь, как переключатель видов, как стрелки курсора или как цифры на дополнительной цифровой клавиатуре. Для этого используются 4 специальных конфигурационных выражения:

Конфигурационные выражения
USE *HatID* AS MOUSE (rate) [- *дополнительные модификаторы*]
USE *HatID* AS POV [-*дополнительные модификаторы*]
USE *HatID* AS ARROWKEYS [-*дополнительные модификаторы*]
USE *HatID* AS KEYPAD [-*дополнительные модификаторы*]

где:

HatID - любой из хэтов: HAT1, HAT2, HAT3, HAT4, RADIOSWITCH

(*переключатель RADIOSWITCH не очень похож на хэт, но это именно он и есть. Он состоит из 4 кнопок: T2 (вверх), T3 (вниз), T5 (влево), T4 (вправо). Однако есть одно небольшое отличие от стандартных хэтов: RADIOSWITCH работает так, как будто по умолчанию к нему всегда применено выражение FORCED_CORNERS.*)

Rate (скорость) - число от 1 до 127, используется только в выражении USE *HatID* AS MOUSE (rate).

[-*дополнительные модификаторы*] Используются с конфигурационными выражениями для изменения поведения хэта:

REVERSE_UD, REVERSE_LR, FORCED_CORNERS, NOHOLD, KP5.

Помните, что не все дополнительные модификаторы могут использоваться с любыми конфигурационными выражениями - подробнее см. соответствующие разделы руководства, или просто используйте утилиту Foxy Composer.

4.1.3 Управление мышью с помощью хэта.

Конфигурационное выражение

USE *HatID* AS MOUSE (rate) [- *дополнительные модификаторы*]

Rate: - скорость мыши, от 1 до 127
Разрешенные дополнительные модификаторы: REVERSE_UD, REVERSE_LR

напр.: USE HAT1 AS MOUSE (2)

Проще говоря, хат 1 используется для управления мышью. Значение в скобках (в данном примере 2) определяет скорость перемещения курсора мыши по экрану – низкое значение означает медленное перемещение, высокое - быстрое. Если хэт перемещается в промежуточное положение, курсор движется по диагонали.

Если необходимо инвертировать вертикальное перемещение мыши, используется следующее выражение:

USE HAT1 AS MOUSE (2) - REVERSE_UD

аналогично, для инверсии горизонтального перемещения, используйте:

USE HAT1 AS MOUSE (2) - REVERSE_LR

Допускается и одновременная инверсия обоих осей:

USE HAT1 AS MOUSE (2) - REVERSE_UD, REVERSE_LR

Примечания

Можно управлять мышью одновременно с помощью одного из хэтов и микроджойстика, или двух и более хэтов. Возможно также назначить управление мышью одним из хэтов при определенном положении переключателя "дальний-ближний бой" (/U, /M, /D) *и (или) кнопки S3* (/I, /O).

Подробнее об этом см. в разделе "Устройство управления мышью, микроджойстик".

4.1.4 Назначение хэта переключателем видов (POV)

Конфигурационное выражение
USE *HatID* AS POV [- *дополнительные модификаторы*]

Разрешенные дополнительные модификаторы: REVERSE_UD, REVERSE_LR

напр.: USE HAT3 AS POV

При использовании манипулятора в режиме Windows, или при отсутствии выражений для HAT 1 (BTN H1x) в файле настроек, HAT 1 по умолчанию назначается как стандартный переключатель видов (POV HAT). Стандартный переключатель видов - это специальный 8-позиционный хэт, функции которого назначаются автоматически во многих авиасимуляторах. К примеру, в игре Falcon 4, если HAT 1 никак не запрограммирован, он будет работать переключателем камер обзора (видов). Как и в предыдущих выражениях, можно использовать модификаторы инверсии направлений:

USE HAT4 AS POV - REVERSE_UD
USE HAT1 AS POV - REVERSE_LR
USE HAT3 AS POV - REVERSE_UD, REVERSE_LR

Можно также напрямую присваивать кнопкам положения переключателя видов, используя POVU, POVD и т. д., даже если ни один из хэтов не был назначен как POV. Можно выборочно запрограммировать нужные положения, и не использовать ненужные. Положения POV работают точно также, как кнопки DirectX - они есть, но чтобы их использовать, их надо присвоить органам управления Cougar. Если аппаратная часть определяет, что на используемой вами рукоятке присутствует POV, его положения можно присваивать. См. Примечания внизу.

Примечания

Мы знаем, что любой хэт легко назначить как переключатель видов (POV). Но также важно упомянуть, что положения POV могут быть назначены в файле настроек. Синтаксис схож с обозначениями положений хэтов джойстика:

POVU, POVD, POVL, POVR, POVUL, POVDL, POVUR, POVDR

Выражение выглядит следующим образом:

BTN T5 POVL
BTN T4 POVR

4.1.5 Использование хэта для эмуляции стрелок клавиатуры

Конфигурационное выражение

USE *HatID* AS ARROWKEYS [- *дополнительные модификаторы*]

Разрешенные дополнительные модификаторы: REVERSE_UD, REVERSE_LR, NOHOLD

Напр.: USE HAT2 AS ARROWKEYS

Очень часто в авиасимуляторе возникает необходимость запрограммировать стрелки на хэт. Это достигается использованием данного выражения. При этом клавиши стрелок удерживаются, пока хэт удерживается в соответствующем положении. Как и в предыдущих примерах, можно использовать инверсию:

USE HAT3 AS ARROWKEYS - REVERSE_UD
USE HAT4 AS ARROWKEYS - REVERSE_LR
USE HAT1 AS ARROWKEYS - REVERSE_UD, REVERSE_LR

Помните, что в случае, если вы переместите хэт в промежуточное положение, то будут сгенерированы одновременно две стрелки соседних основных положений - то есть в данном выражении как бы по умолчанию встроен модификатор FORCED_CORNERS.
Если вы не хотите, чтобы клавиши стрелок удерживались, используйте модификатор NOHOLD:

USE HAT3 AS ARROWKEYS - NOHOLD

В этом случае при перемещении хэта будет сгенерирован только один символ. Все перечисленные модификаторы могут использоваться совместно:

USE HAT1 AS ARROWKEYS - REVERSE_UD, REVERSE_LR, NOHOLD

4.1.6 Использование хэта для эмуляции клавиш цифровой клавиатуры

Конфигурационное выражение

USE *HatID* AS KEYPAD [- *дополнительные модификаторы*]

Разрешенные дополнительные модификаторы:
REVERSE_UD, REVERSE_LR, FORCED_CORNERS, NOHOLD, KP5

напр.: USE HAT4 AS KEYPAD

Очень часто необходимо при помощи хэта воспроизводить клавиши цифровой клавиатуры. В приведенном примере хэт 4 эмулирует клавиши цифровой клавиатуры, при этом в промежуточных положениях (UL UR, DL, DR) генерируются клавиши "7 9 1 3" соответственно.

Проблема с цифровой клавиатурой заключается в том, что в различных авиасимуляторах она работает по-разному. Более того, в некоторых играх еще имеет значение статус кнопки Num LOCK, или игры в принудительном порядке включают или выключают ее. Однако в большинстве случаев цифровая клавиатура работает согласно названию - то есть воспроизводит цифры.

Как и в предыдущих примерах, можно использовать инверсию:

USE HAT1 AS KEYPAD - REVERSE_UD
USE HAT2 AS KEYPAD - REVERSE_LR
USE HAT3 AS KEYPAD - REVERSE_UD, REVERSE_LR

Как было сказано, использование данного выражения позволяет при перемещениях хэта генерировать клавиши цифровой клавиатуры. Но там есть еще одна клавиша - "5". В некоторых симуляторах эта клавиша используется для центровки вида или положения чего-либо, и можно запрограммировать эту клавишу на центральное положение хэта:

USE HAT4 AS KEYPAD - KP5

Можно также использовать принудительное генерирование промежуточных положений:

USE HAT4 AS KEYPAD - FORCED_CORNERS

В данном случае при отклонении хэта 4, скажем, вправо-вверх будет сгенерировано сочетание "KP8" и "KP6" вместо KP9, то есть комбинация символов, назначенных на соседние основные положения.

В некоторых симуляторах это может оказаться полезным.

Опять же, если вы не хотите, чтобы клавиши цифровой клавиатуры удерживались, используйте модификатор NOHOLD:

```
USE HAT3 AS KEYPAD - NOHOLD
```

В этом случае при перемещении хэта в определенное положение будет генерироваться одиночный символ.

Все описанные модификаторы могут использоваться совместно:

USE HAT4 AS KEYPAD - REVERSE_UD, REVERSE_LR, FORCED_CORNERS, NOHOLD, KP5

4.1.7 Каким образом компилятор преобразует выражения USE HatID AS

Данный раздел предназначен для продвинутых пользователей, желающих ознакомиться со всеми подробностями!

Для того, чтобы понять данный раздел, необходимо прочесть другие разделы руководства. Итак, для любителей технических подробностей: как компилятор преобразует выражения USE HATx AS *что-то_полезное*. Начнем с выражения USE HATx AS MOUSE.

USE HAT1 AS MOUSE (2) преобразуется компилятором в следующее:

```
USE HAT1 FORCED_CORNERS
BTN H1U /P MSY(2-) /R MSY(2+)
BTN H1R /P MSX (2+) /R MSX (2-)
BTN H1D /P MSY (2+) /R MSY (2-)
BTN H1L /P MSX (2-) /R MSX (2+)
```

Точно также:

```
USE HAT1 AS MOUSE (2) - REVERSE_UD
```

преобразуется в:

```
USE HAT1 FORCED_CORNERS
BTN H1U /P MSY(2+) /R MSY(2-)
BTN H1R /P MSX (2+) /R MSX (2-)
BTN H1D /P MSY (2-) /R MSY (2+)
BTN H1L /P MSX (2-) /R MSX (2+)
```

и наконец:

```
USE HAT1 AS MOUSE (2) - REVERSE_UD, REVERSE_LR
```

в:

```
USE HAT1 FORCED_CORNERS
BTN H1U /P MSY(2+) /R MSY(2-)
BTN H1R /P MSX (2-) /R MSX (2+)
BTN H1D /P MSY (2-) /R MSY (2+)
BTN H1L /P MSX (2+) /R MSX (2-)
```

Теперь рассмотрим преобразование выражения ARROWKEYS:

```
USE HAT2 AS ARROWKEYS
```

преобразуется в:

```
USE HAT2 FORCED_CORNERS
BTN H2U /H UARROW
BTN H2R /H RARROW
BTN H2D /H DARROW
BTN H2L /H LARROW
```

соответственно:

```
USE HAT2 AS ARROWKEYS - REVERSE_UD, NOHOLD
```

преображается в:

```
USE HAT2 FORCED_CORNERS
BTN H2U  DARROW
BTN H2R RARROW
BTN H2D UARROW
BTN H2L LARROW
```

Заметьте, что верхнее и нижнее положения поменялись местами, а добавление модификатора NOHOLD убирает из выражений ключ /H.

И наконец:

```
USE HAT4 AS KEYPAD
```

преобразуется в:

```
BTN H4U /H KP8
```

```
BTN H4R /H KP6
BTN H4D /H KP2
BTN H4L /H KP4
BTN H4UR /H KP9
BTN H4DR /H KP3
BTN H4DL /H KP1
BTN H4UL /H KP7
```

соответственно:

```
USE HAT4 AS KEYPAD - REVERSE_UD, NOHOLD
```

в:

```
BTN H4U KP2
BTN H4R KP6
BTN H4D KP8
BTN H4L KP4
BTN H4UR KP3
BTN H4DR KP9
BTN H4DL KP7
BTN H4UL KP1
```

Отметьте, что при инверсии верхнего и нижнего положений изменяются автоматически и промежуточные положения.

То же самое справедливо и для REVERSE_LR:

```
USE HAT4 AS KEYPAD - REVERSE_LR
```

преобразуется в:

```
BTN H4U /H KP8
BTN H4R /H KP4
BTN H4D /H KP2
BTN H4L /H KP6
BTN H4UR /H KP7
BTN H4DR /H KP1
BTN H4DL /H KP3
BTN H4UL /H KP9
```

Если используется принудительное генерирование промежуточных положений:

```
USE HAT4 AS KEYPAD - FORCED_CORNERS
```

компилятор выполняет следующие преобразования:

```
USE HAT4 FORCED_CORNERS
BTN H4U /H KP8
BTN H4R /H KP6
BTN H4D /H KP2
BTN H4L /H KP4
```

И, наконец, при:

```
USE HAT4 AS KEYPAD - KP5
```

компилятор генерирует дополнительную строку:

```
BTN H4M KP5
```

Заметьте, что на центральном положении ключ /H не используется - там нужен единичный символ.

5. Конфигурационные выражения

5.1 Введение

В начале данного руководства обсуждалась структура файла настроек. В начале файла, до основных программных выражений, располагаются конфигурационные выражения, такие, как USE MDEF. В данном разделе эти выражения будут рассмотрены подробно.

Конфигурационные выражения применяются ко всему файлу настроек - они не относятся к какому-то конкретному органу управления, хотя некоторые из них могут быть запрограммированы непосредственно через выражения для кнопок. Эти выражения сообщают компилятору общие настройки джойстика для той или иной игры: какой файл макрокоманд использовать, какова должна быть частота генерирования символов, какие оси отключены и т. д.

Многие конфигурационные выражения подробно описаны в других разделах руководства, посвященных тем аспектам работы контроллера, на которые они влияют. В данном разделе основное внимание будет уделено тем выражениям, которые не были детально рассмотрены в других частях Руководства.

Синтаксис конфигурационных выражений для HOTAS Cougar изменился, и ниже приведено Золотое Правило:

Все конфигурационные выражения начинаются с USE или DISABLE

Все конфигурационные выражения логического программирования начинаются с DEF

Это отличается от прежнего программного синтаксиса TM. Что касается логического программирования, данный аспект предназначен скорее для самых продвинутых любителей программирования, и будет описан подробно в конце Руководства.

5.2 MDEF - Файл макрокоманд

Конфигурационное выражение

USE MDEF *имя_макрофайла*

Данное выражение применяется только в том случае, если файл настроек использует макрокоманды (*что всячески приветствуется*). *Имя_макрофайла* - это имя файла макрокоманд с или без расширения (.tmm).

Например, если есть файл настроек: Janes WW2 Fighters.tmj, которому соответствует файл макрокоманд Janes WW2 Fighters.tmm, выражение для MDEF будет выглядеть следующим образом:

USE MDEF Janes WW2 fighters

Оба этих файла должны обязательно находится в одной директории, по умолчанию это папка Files утилиты Foxy. В принципе, возможно написать программное обеспечение так, чтобы это ограничение было снято, однако на наш взгляд, это весьма практичное правило, унаследованное от предыдущих манипуляторов TM HOTAS.

Примечания

1. *Для макрофайлов поддерживаются длинные имена с пробелами.*

2. *Наличие или отсутствие расширения макрофайла в конфигурационном выражении значения не имеет:*

USE MDEF Janes WW2 fighters
USE MDEF Janes WW2 fighters.tmm

одинаково верны.

3. *В именах файлов настроек, файлов макрокоманд и в именах самих макрокоманд регистр не учитывается:*

Mig Alley.tmm

это то же самое, что и:

mig alley.tmm

4. *В файлах настроек предыдущих манипуляторов TM HOTAS можно было использовать несколько выражений* MDEF*, таким образом обращаясь к макрокомандам в разных макрофайлах. В случае с HOTAS Cougar это невозможно.*

5.3 RATE

За исключением данного и нескольких нижеизложенных конфигурационных выражений, большинство из них относится к программированию осей, и будет рассмотрено в соответствующем разделе Руководства.

Конфигурационное выражение

USE RATE (*nnnn*)

Синтаксис команды

RATE (*nnnn*)

где *nnnn* - время в миллисекундах (1 с = 1000 мс). Данный параметр определяет частоту повтора символов. При отсутствии данного выражения в файле компилятор автоматически генерирует выражение USE RATE (0), при этом символы будут генерироваться с частотой, равной системной частоте автоповтора клавиатуры. Чем **больше** значение параметра, тем **медленнее** скорость автоповтора. Допустимые значения параметра - от 0 до 655350 (*чуть более 10 минут!)*.

Значение параметра RATE можно также менять в реальном времени через программное выражение для кнопки:

```
BTN S4 /I RATE (100)
       /O RATE (0)
```

или для оси:

ANT 2 5 RATE (0) RATE (30) RATE (60) RATE (90) RATE (120)

(*о программировании цифровых режимов осей см. далее в Руководстве*)

Дополнительные примечания

HOTAS Cougar воспроизводит символы циклами. Циклы повторяются примерно каждые 30 мс при RATE(0) *(или без выражения RATE). В рамках одного цикла Кугуар может сгенерировать до 16 символов. Если количество символов превышает 16, они буферизируются и переносятся на следующий цикл. Параметр* RATE *определяет задержку между циклами.*

5.4 S3_LOCK и S3_UNLOCK

Конфигурационное выражение

USE S3_LOCK

Синтаксис команды

S3_LOCK
S3_UNLOCK

Обычно, кнопка S3 на джойстике используется для выполнения выражений с ключами /I (кнопка нажата) и /O (кнопка отжата).

Выражение USE S3_LOCK позволяет переключаться между состояниями кнопки S3 повторными нажатиями, то есть если кнопка S3 нажата один раз, выполняются выражения с ключом /I; если кнопка нажата еще раз - выполняются выражения с ключом /O.

Если вы хотите переключаться между двумя этими режимами работы кнопки S3, вы можете использовать выражение типа:

BTN S2 /T S3_LOCK /T S3_UNLOCK

Для того, чтобы использовать прямые команды S3_LOCK, S3_UNLOCK, наличие конфигурационного выражения USE S3_LOCK не требуется. В чем же6 отличия между USE S3_LOCK и S3_LOCK? Выражение USE S3_LOCK применяется ко всему файлу и выполняется сразу после загрузки файла настроек в джойстик, тогда как S3_LOCK, запрограммированное через выражение для кнопки, вступает в действие только при нажатии этой кнопки.

Примечания

1. *Если вы назначили другую кнопку (положение хэта) в качестве S3 (см. следующий раздел), вышеописанные выражения будут применяться к такой кнопке, но синтаксис их не меняется.*

2. *С командой* S3_LOCK *нельзя использовать ключ* /H. *Таким образом:*

```
BTN S1 /H S3_LOCK
BTN S4 /H Some_macro S3_LOCK
```

приведет к ошибке компиляции.

5.5 Назначение другой кнопки для выполнения выражений с ключами /I, /O командой SHIFTBTN

Конфигурационное выражение

```
USE название_кнопки AS SHIFTBTN
```

Синтаксис команды

```
SHIFTBTN (название кнопки)
```

Примеры:

```
USE S4 AS SHIFTBTN
BTN T6 SHIFTBTN (T10)
```

Определяет кнопку, которая будет использоваться вместо S3 для выполнения ключей /I /O. Если данное конфигурационное выражение отсутствует (а чаще всего так оно и будет), ключи /I, /O выполняются по состоянию кнопки S3.

5.6 USE HAT SENSITIVITY - чувствительность промежуточных положений хэта

Иногда бывает достаточно непросто гарантированно переместить хэт в промежуточное положение. Помните, что физически хэты являются 4-позиционными переключателями. Так как контроллер обрабатывает команды очень быстро, нередко перед выполнением команды,

запрограммированной на промежуточное положение, выполняется команда одного из соседних основных. Эту проблему можно решить, как бы уменьшив чувствительность хэта:

Конфигурационное выражение
USE *HatID*_SENSITIVITY (*nnnn*)

где:

HatID - один из хэтов: HAT1, HAT2, HAT3, HAT4
nnnn - величина от 0 (максимальная чувствительность) до 1000 (минимальная чувствительность). Значение *nnnn* на самом деле является временем задержки в миллисекундах. Пример:

USE HAT1_SENSITIVITY (100)

Данное выражение означает, что все выражения, запрограммированные для данного хэта, будут исполняться лишь после того, как хэт будет удержан в нужном положении по меньшей мере 100 мс, что позволяет более уверенно отрабатывать промежуточные положения, нажимая оба соседних микропереключателя.

Примечания

При наличии выражения USE *HatID*_SENSITIVITY (*nnnn*) *для данного хэта нельзя использовать ключ* /T*.*

Таким образом:

```
USE HAT1_SENSITIVITY (60)
BTN H1U /T a /T b /T c
```

приведет к ошибке компиляции.

5.7 USE T1 SENSITIVITY

Если вы обнаружите, что постоянно случайно нажимаете кнопку T1 микроджойстика, вы можете снизить ее чувствительность.

Конфигурационное выражение
USE T1_SENSITIVITY (*nnnn*)

где:

nnnn - значение от 0 (максимальная чувствительность) до 1000 (минимальная чувствительность). *nnnn* на самом деле является временем задержки в мс, которое проходит между тем, как кнопка T1 физически нажата, и тем, когда нажатие регистрируется.

Пример:

USE T1_SENSITIVITY (1000)

В этом случае нажатие на кнопку T1 будет зарегистрировано только по прошествии секунды.

Примечания

При наличии выражения USE T1_SENSITIVITY (*nnnn*) *с кнопкой* T1 *нельзя использовать ключ* /T*.*

5.8 USE FOXY GRAPHIC и README

Конфигурационное выражение

USE FOXY GRAPHIC *файл изображения*
USE FOXY README *текстовый файл*

Эти два конфигурационных выражения игнорируются компилятором и используются исключительно утилитой Foxy для своих целей. Когда Foxy открывает файл настроек, проверяется наличие выражения USE FOXY GRAPHIC *файл изображения*, и при наличии такового в окно просмотра изображений будет загружен соответствующий графический файл. Данная функция весьма полезна при обмене файлами настроек, позволяя создателю включить графическую раскладку символов, запрограммированных на кнопки и хэты комплекта. Пример:

USE FOXY GRAPHIC Total Air War.bmp

Поддерживаемые форматы графических файлов: .bmp, .jpg .gif.

Точно также, выражение USE FOXY README сообщает утилите Foxy о том, какой текстовый файл загрузить в Редактор шаблонов. Данный файл может содержать какую-либо дополнительную информацию: как был создан файл, какие настройки в игре необходимо сделать, и т. д. Пример:

USE FOXY README Total Air War.rtf

Поддерживаемые форматы: .txt и .rtf. Файлы в расширенном текстовом формате (rtf) позволяют использовать различные цветные шрифты, форматирование и т. д., что делает их более приятными и удобными для восприятия.

Примечания

Графический и текстовый файлы должны находится в той же директории, что и файл настроек. По умолчанию, это папка Files утилиты Foxy.

5.9 NULLCHR - пустой символ ^

Конфигурационное выражение

USE NULLCHR *символ*

Пустой символ - это символ в выражении, который не приводит к генерированию чего-либо. По умолчанию пустым символом является галочка ^. Зачем он нужен? Для некоторых выражений требуется фиксированное число параметров для того, чтобы выражение было обработано компилятором. Хорошим примером являются некоторые выражения цифровых режимов осей, которые будут рассмотрены в следующем разделе. Пример:

RDDR 3 L ^ R

Данным выражением педали (руль поворота) программируются таким образом, что при нажатии на левую педаль генерируется символ "L", на правую - символ "R". Когда педали находятся в нейтральном положении, не происходит ничего, поэтому добавлен пустой символ ^. При этом нельзя просто написать:

RDDR 3 L R

так как данное выражение требует наличия трех символов/макросов после команды "RDDR 3", в противном случае компилятор выдаст сообщение об ошибке. Пустой символ - это как бы заглушка для тех случаев, когда в файле настроек требуется наличие символа, но вам не нужно, чтобы что-то генерировалось контроллером.

Возвращаясь к конфигурационному выражению, повторим, что по умолчанию пустым символом является галочка (**^**). Если вы хотите, вы можете использовать любой другой символ:

```
USE NULLCHR TAB
USE NULLCHR z
```

В случае, если галочка используется в игре, вы можете ее программировать через сочетание клавиш SHF 6, т .е. BTN S1 SHF 6

Примечания

1. *С предыдущими контроллерами TM нельзя было оставлять пустые программные строчки, добавляя пустой символ, например:*

```
BTN S2 /U Fire_Missile
       /M ^
       /D  ^
```

С Cougar ситуация изменилась. Теперь вы можете запросто иметь в файле настроек:

```
BTN S2 /U Fire_Missile
       /M
       /D
```

2. *Пустой символ на самом деле генерирует код* USB (00), *по которому ничего не происходит. Если хотите, вы можете создать макрокоманду в макрофайле:*

```
Do_Nothing = USB(00)
```

и затем использовать:

```
RDDR 3 L Do_Nothing R
```

Конечно, намного быстрее и удобнее вставить ^, *собственно поэтому данная команда и была введена.*

3. *С выражением* USE NULLCHR *нельзя использовать сочетания клавиш. Следующие примеры приведут к ошибке компиляции:*

```
USE NULLCHR SHF F1
USE NULLCHR ALT  p
```

5.10 KEYBOARD (AZERTY, QWERTY)

Конфигурационное выражение

USE KEYBOARD *Тип раскладки*

Тип раскладки: AZERTY or QWERTY.

Если вы используете французскую раскладку клавиатуры AZERTY, и в игре у вас возникают проблемы с неправильным выполнением клавиатурных команд, используйте конфигурационное выражение USE KEYBOARD AZERTY для исправления ситуации. Более подробно см. раздел "Тестер символов" (Key Tester).

Примечания

Использовать выражение USE KEYBOARD QWERTY *нет никакой необходимости - именно такую раскладку компилятор использует по умолчанию.*

5.11 Использование профилей Панели управления HOTAS Cougar - USE PROFILE

Конфигурационное выражение

USE PROFILE *Профиль* (Режим калибровки)

где:

Профиль: профиль, созданный в панели управления Cougar и сохраненный с расширением .tmc в директории Profiles панели управления.
Режим калибровки: либо AUTO (автоматический), либо CUSTOM (пользовательский). Используйте этот параметр для того, чтобы выбрать использование данных автокалибровки или ручной калибровки, заданной вами.

Примеры:

USE PROFILE Crimson Skies.tmc (AUTO)
USE PROFILE Mechwarrior 4 (CUSTOM)

Как вы скоро узнаете из раздела, посвященного программированию осей, оси можно менять местами, отключать, задавать разные кривые отклика и т.

д. Это можно сделать и через программные выражения, но это достаточно сложно. Куда проще использовать ранее созданный и сохраненный профиль панели управления. Использование сохраненных профилей имеет и еще одно преимущество: возможность использовать мертвые зоны, чего нельзя сделать программными выражениями. К тому же, опыт показывает, что файлы, в которых используются профили, загружаются быстрее, чем файлы с выражениями DISABLE или USE AXES_CONFIG (*см. далее в Руководстве.*)

5.11.1 Еще несколько слов о профилях.

Немного о профилях и о том, почему так полезно включать их в файлы настроек. Итак, профили создаются в Панели управления HOTAS Cougar (CCP - *Cougar Control Panel*). В них содержится вся информация о настройках, регулируемых через CCP: мэппинг осей, мертвые зоны, кривые отклика и т. д.
Если вы загружаете файл настроек, в котором так или иначе изменяются параметры осей, или файл, в котором предусмотрена возможность, скажем, менять кривые отклика "налету", то когда вы выйдете из игры, вся эта информация останется в памяти манипуляторов. Помните, что HOTAS Cougar работает без драйверов и резидентных программ, и все данные сохраняются непосредственно в ПЗУ джойстика. Если впоследствии вы загрузите новый файл настроек для другой игры, в котором нет никаких элементов, связанных с параметрами осей, то все их настройки для предыдущей игры сохранятся. Если это доставляет проблемы, есть два пути их решения. Во-первых, вы можете использовать выражение USE PROFILE во всех ваших файлах настроек, используя либо специально созданный профиль под конкретную игру, либо профиль DEFAULT.tmc (*профиль, создаваемый по умолчанию Панелью управления при первой загрузке*). Во-вторых, в меню Download утилиты Foxy можно указать компилятору при загрузке файла настроек прежде всего сбрасывать параметры осей, применяя выбранный вами профиль.
И последнее: так как профиль содержит данные калибровки, вы должны указать компилятору, следует ли ему использовать калибровочные параметры из профиля, или использовать данные Автокалибровки.

Примечания

1. *Утилита Foxy и компилятор изначально считают, что все профили находятся в папке Profiles Панели управления. Именно там сохраняются все профили, созданные в Панели управления. По умолчанию, это директория* C:\Program Files\Hotas\Profiles. *При компиляции или загрузке файла, содержащего выражение* USE PROFILE *компилятор выполняет следующие действия:*

(a) Ищет профиль в директории по умолчанию (Profiles).
(b) Если профиль найден, он применяется, и компилятор больше нигде его не ищет.
(c) Если профиль не найден, компилятор ищет его в той же директории, где находится загружаемый файл настроек, то есть в папке Files утилиты Foxy.
(d) Если профиль не найден и там, выдается сообщение об ошибке.

2. *Профили имеют расширение* **tmc**. *Наличие или отсутствие расширения в строке выражения* USE PROFILE *значения не имеет, то есть:*

USE PROFILE Crimson Skies.tmc

равноценно

USE PROFILE Crimson Skies

3. *Если вы используете утилиту Foxy для открытия чужого дистрибутива файла настроек в формате .zip, и в архиве содержится профиль, все файлы извлекаются в папку Files. Затем, на ваше усмотрение, вы либо оставляете профиль там, либо переносите его в директорию по умолчанию (Profiles). Рекомендуется хранить все профили в директории HOTAS Profiles* (*обычно* C:\Program Files\HOTAS\Profiles).

4. *В профиле хранится следующая информация об осях Кугуара: мэппинг, направление осей, центр, данные калибровки, мертвые зоны, кривые отклика, настройки триммера, информация об отключенных осях, а также информация об использовании настроек осей, видимых для Windows.*

5.12 Конфигурационные выражения, описанные в других разделах руководства

Некоторые конфигурационные выражения не попали в данный раздел, так как были подробно описаны в других частях Руководства, вместе с параметрами, которыми они управляют. Ниже приводится список конфигурационных выражений, описанных в других разделах:

Конфигурационное выражение	Раздел
USE *Btn* AS DXn	Раздел 3.8: Назначение кнопок DirectX (Direct Input) и работа с ними
USE ALL_DIRECTX_BUTTONS	Раздел 3.8.1: Назначение всех кнопок как кнопок DirectX
USE HAT AS MOUSE, POV, ARROWKEYS, KEYPAD	Раздел 4.1: Программирование хэтов джойстика
USE CURVE	Раздел 6.3: Кривые отклика (CURVE)
DISABLE AXIS	Раздел 6.5: Отключение осей
USE SWAP	Раздел 6.6: Мэппинг осей (SWAP)
USE REVERSE	Раздел 6.7: Инверсия оси
USE AXES_CONFIG	Раздел 6.8: Выражение USE AXES_CONFIG
USE MTYPE	Раздел 7.2: USE MTYPE - самый простой способ запрограммировать мышь на микроджойстик
USE Axis_Identifier AS Mouse_Axis	Раздел 7.3.1: Назначение других осей осями мыши
USE ZERO_MOUSE	Раздел 7.5: Выражение USE ZERO_MOUSE
DISABLE MOUSE	Раздел 7.7: Отключение назначения микроджойстика для управления мышью
USE SCREEN_RESOLUTION	Раздел 7.8.1: Определение разрешения экрана
DEF Xn	Раздел 8.2: Определение логических флагов и выражения с ними

6. Программирование осей

6.1 Основные принципы

6.1.1 Понятия аналогового и цифрового режимов

В данном разделе вы узнаете, как можно программировать различные оси HOTAS Cougar, как в цифровом виде, так и меняя их аналоговые параметры. Но прежде чем перейти к этой теме, еще раз проясним отличие между аналоговой и цифровой осью.

Большинство джойстиков на рынке имеют практически одинаковую электромеханическую концепцию: в них используется два потенциометра, или резистора.

Эти два резистора используются для считывания положения джойстика в двух плоскостях: движение вправо-влево - ось X, движение вперед-назад - ось Y. Потенциометр - прибор аналоговый, или *не дискретный,* бесступенчатый. Он выдает определенное значение для каждого своего положения. Соответственно, ось, значения которой считываются с аналогового потенциометра в виде не дискретных значений, является *аналоговой,* в отличие от цифровых устройств, которые в принципе сводятся к считыванию двух состояний - 0 и 1 (клавиши клавиатуры, например.

Теперь рассмотрим эту концепцию применительно к HOTAS Cougar. В идеале, потенциометры являются прекрасным устройством для считывания положения оси, выдавая точные и постоянные значения - но только в идеале. В реальности есть такие вещи, как загрязнение и износ, приводящие к неприятным явлениям, таким как дрожание и скачки сигнала. Так как Кугуар является USB-устройством, сигнал с потенциометров не может быть подан напрямую в компьютер: его значения считываются встроенным процессором, обрабатываются, отфильтровываются анормальные значения, и на выходе получается более точный и стабильный цифровой сигнал. То есть хотя потенциометры и являются аналоговым устройством, их сигнал переводится в цифровой вид. Однако для целей настоящего Руководства ось, которая выдает оцифрованный сигнал с потенциометра, который считывается DirectInput как значение оси DirectX, будет называться *аналоговой*.

Однако HOTAS Cougar позволяет также программировать оси еще в одном режиме, который мы назовем *цифровым*. Что это означает? Допустим, ось РУД (сектор газа) выдает аналоговые значения в диапазоне от 0 до 100

(условно). Данный диапазон можно разделить, к примеру, на 5 полос: полосе 1 будут соответствовать значения от 0 до 19, полосе 2 - от 20 до 39. и т. д. Кугуар позволяет при помощи специальных выражений запрограммировать ось таким образом, что, скажем, когда ее значение попадает в полосу 1, генерируется символ 'a', во второй полосе - символ 'b' и т. д. Именно такой способ работы оси и будет называться в данном руководстве "цифровым режимом" - то есть когда перемещение оси приводит к генерированию некоторой последовательности клавиатурных символов.

6.1.2 Оси Кугуара

Полный комплект HOTAS Cougar (РУС, РУД и педали с педальными тормозами) имеет 10 физических аналоговых осей. Эти оси могут работать следующим образом:

- как чисто аналоговые (не дискретные) устройства *(при условии что они поддерживаются DirectX и игрой*), которым присваиваются органы управления в игре;
- как чисто цифровые устройства, которые генерируют исключительно клавиатурные символы
- сочетание двух вышеописанных режимов.

Далее, с аналоговыми осями можно совершать следующие действия:

- полностью отключать их
- применять к ним различные кривые отклика
- применять триммирование
- инвертировать их направление
- изменение мэппинга осей, как для всего файла настроек, так и в зависимости от положения переключателя "дальний-ближний бой" или кнопки S3.

Одной из сильных сторон Cougar является разнообразие настроек осей. Однако, эта тема может показаться на первый взгляд очень сложной. Во избежание трудностей в понимании прежде всего запомните **6 типов выражений цифрового режима**. Эти выражения используются для программирования осей на генерирование клавиатурных символов. Затем будут рассмотрены настройки для *аналогового* режима работы осей.

Ниже приводятся обозначения всех 10 физических осей комплекта:

Обозначение TM	Ось
JOYX	Ось X джойстика
JOYY	Ось Y джойстика
THR	РУД
RNG	Кольцо масштаба

ANT	Кольцо антенны
MIX	Ось X микроджойстика*
MIY	Ось Y микроджойстика
LBRK	Левый педальный тормоз
RBRK	Правый педальный тормоз
RDDR	Руль поворота

* Оси X, Y микроджойстика не эквивалентны осям X, Y мыши.

Примечания

1. *Оси предыдущих комплектов TM HOTAS могли работать либо в цифровом, либо в аналоговом режиме, то есть либо ось была видна операционной системе, и ее функция определялась настройками игры, либо она программировалась на генерирование клавиатурных символов. С HOTAS Cougar ситуация иная. По умолчанию, все оси видны как аналоговые, и если вы просто их программируете на воспроизведение клавиатурных символов, они* ***одновременно*** *будут работать в аналоговом* ***и*** *цифровом режимах. Если требуется перевести ось исключительно в цифровой режим, то ее необходимо отключить как аналоговую ось.*

2. *В отличие от предыдущих манипуляторов HOTAS, вам не нужно ничего менять в Control Panel > Gaming Options для того, чтобы использовать ось исключительно в цифровом виде.*

3. *Применительно ко всем выражениям цифровых режимов осей:*

- *Можно использовать ключи* /U, /M, /D, /I, /O
- *Любые настройки аналогового режима (кривая отклика, мэппинг, инверсия и т. д.) не влияют на работу оси в цифровом режиме. Они остаются привязаны к физической оси комплекта.*

6.2 Выражения цифровых режимов

В данном разделе рассматриваются способы программирования осей для эмуляции клавиатурных символов. Такое программирование осуществляется с использованием одного из 6 типов выражений цифровых режимов. Ниже рассмотрены примеры всех 6 типов, и проанализированы генерируемые клавиатурные символы.

Примечание: *Для упрощения понимания все приведенные примеры касаются эмуляции клавиатурных команд. Однако это не единственное, что можно запрограммировать в цифровом*

режиме. Это могут быть логические флаги, команды управления мышью, изменения кривой отклика и т. д.

6.2.1 Тип 1: повторяющиеся символы

Синтаксис:

Обозн. оси	Тип цифрового выражения	количество символов или макросов (макс. 50)	символ (макрос) перем. вверх	символ (макрос) перем. вниз	Символ (макрос) центра *(дополн.)*	FORCE_ MACROS *(дополн.)*
напр. ANT	1	10	u	d	c	- FORCE_MACROS

в файле настроек это примет такой вид:

ANT 1 10 u d c - FORCE_MACROS

Вращение кольца антенны из крайнего положения по и против часовой стрелки приведет к генерированию следующих символов:

u u u u u c u u u u u d d d d d c d d d d d

Использоваться могут не только единичные клавиатурные символы, можно использовать сложные макрокоманды. Например, в макрофайле определены следующие макрокоманды:

```
Chaff_Flare =  c DLY(30) f          rem ДО, ЛТЦ
Getting_desperate = RPT(20) (c f)   rem Дело пахнет керосином!
```

А затем кольцо масштаба запрограммировано через выражение цифрового режима 1 типа:

RNG 1 5 Chaff_Flare Getting_desperate

Отметьте, что количество символов определяет суммарное число символов (за вычетом символа центрального положения), генерируемых за **полный** ход перемещения оси. Это отличается от прежнего выражения цифрового режима 1 типа, где данный параметр определял число символов, генерируемых на участке от крайнего положения оси до ее центра (и, соответственно, от центра до другого крайнего положения). Причина изменения заключается в том, что теперь определение символа центрального положения не является обязательным. Например:

RNG 1 6 u d

При вращении кольца масштаба от упора до упора и обратно будут сгенерированы следующие символы:

u u u u u u d d d d d d

Помните, что отсутствие символа центрального положения **отличается** от использования пустого символа (*по умолчанию* ^) в центре. Так, при выражении:

RNG 1 6 u d ^

будут сгенерированы символы:

u u u *мертвая зона* u u u d d d *мертвая зона* d d d

Пустой символ "^" не приводит к генерированию чего-либо, создавая своего рода мертвую зону. Значение параметра Количество символов **должно быть четным числом** *в случае, если* определен символ центрального положения. Если таковой отсутствует, это значение может быть любым.

6.2.1.1 Модификатор - FORCE_MACROS

Использование этого модификатора опционально, и возможно только в выражениях 1, 2, 5 и 6 типа. Приведем в качестве примера выражение 1 типа, однако во всех остальных типах действие этого модификатора аналогично.

Имеем выражение:

RNG 1 50 u d

Вращение кольца масштаба от упора до упора сгенерирует 50 символов "u" (или 50 символов "d", в зависимости от направления). Если вращать кольцо быстро, и проверить результат в Блокноте или Тестере символов утилиты Foxy, вы не увидите 50 символов - 10, может быть 20. В чем же дело? Почему есть пропуски, ведь Кугуар обрабатывает команды очень быстро? Да, это действительно так, и в этом-то и проблема. Если вращать кольцо слишком быстро, команды нажатия и отпускания клавиш будут обрабатываться параллельно, и в случае, если за цикл (около 30 мс) выполняется 16 символов, компьютер, возможно, воспримет их как одно нажатие клавиши. Ситуацию можно изменить, заставив компьютер воспринять все символы:

RNG 1 50 (< u >) (< d >)

собственно, именно это и делает модификатор - FORCE_MACROS. Он как бы заключает каждый символ (макрос) выражения в угловые скобки. Однако стоит быть осторожным с использованием этого модификатора - если вы программируете принудительное исполнение команд этого выражения, то выполнение остальных команд не начнется, пока не будут выполнены символы (макросы) в угловых скобках. Убедитесь также, что макрокоманды,

используемые в выражении с модификатором - FORCE_MACROS уже не имеют внутри себя угловых скобок < >. Угловые скобки не допускают многократных вложений, а именно это и получится, если есть макрокоманды:

```
Macro_1 = < a b c>
Macro_2 = d
```

и:

```
RNG 1 50 Macro_1  Macro_2 - FORCE_MACROS
```

так как компилятор преобразует все это в:

```
RNG 1 50 ( < < a b c> > ) ( < d > )
```

Примечания

1. *Недопустимо:* RNG 1 3 a b c d e f

однако возможно: RNG 1 3 (a b c) (d e f)
или RNG 1 3 ABC_macro DEF_macro

и в макрофайле:
```
ABC_macro = a b c
DEF_macro = d e f
```

2. *Со всеми типами выражений цифрового режима можно использовать ключи* /U, /M, /D, /I, /O, *например:*

```
RNG  /U 1 3 F1 F2
     /M 1 5 (SHF UARROW) (SHF DARROW)
     /D  /I 1 6 e t F5
         /O 1 4 [ ] KP5
```

3. *В выражениях 1, 2, 5 и 6 типов можно также использовать ключи* /P, /R *и* /H *как внутри макрокоманд, так и непосредственно в строке выражения. Пример:*

```
RNG 1 3 Macro_1 Macro_2
```

и в макрофайле:

```
Macro_1 = /P a /R b
Macro_2 = /H d   Rem Непонятно правда, зачем это может понадобиться…
```

4. *В выражениях 1 типа (как и в* ***любых других*** *выражениях цифрового режима) нельзя использовать ключ* /T.

5. *Выражение 6 типа является частным случаем выражения 1 типа, поэтому для кольца антенны их действие будет одинаковым* (u u c u u d d c d d):

ANT 1 4 u d c

ANT 6 5 (0 20 40 60 80 100) u d c

6.2.1.2 Важные замечания по использованию FORCE_MACROS

Дополнительные примечания

На первый взгляд может показаться, что стоит использовать модификатор FORCE_MACROS *во всех случаях, но это не совсем так. Первое, на что стоит обратить внимание - какое влияние оказывает использование* FORCE_MACROS *не только на выполнение команд остальных кнопок и хэтов, но и на работу самой запрограммированной оси. Рассмотрим пример, на котором также будут объяснены ряд других моментов.*

Возьмем выражение цифрового режима 2 типа (*этот тип еще не был рассмотрен, ему будет посвящен следующий раздел, однако он достаточно простой*):

ANT 2 26 a b c d e f g h i j k l m n o p q r s t u v w x y z

Не надо быть Эйнштейном, чтобы понять, что в данном выражении ось кольца антенны запрограммирована на воспроизведение английского алфавита.

Если добавить это выражение в файл настроек и загрузить в джойстик, то при вращении кольца антенны будут воспроизводиться буквы алфавита, что можно наблюдать в окне "Тестер символов" утилиты Foxy. Вы наверняка заметите, что символы генерируются очень быстро. Если затем модифицировать это выражение, добавив FORCE_MACROS*:*

ANT 2 26 a b c d e f g h i j k l m n o p q r s t u v w x y z - FORCE_MACROS

то вы увидите, что вывод символов оси кольца антенны очень сильно замедлился (примерно в 4 раза). Это происходит из-за того, что компилятор конвертирует каждый символ (например, "a") в 4 команды: < KD(a) KU(a) *>. Каждая такая четверка команд занимает отдельный цикл.*

Здесь необходимо напомнить понятие "цикла". HOTAS Cougar выдает запрограммированные команды каждые 30 мс. Если хотите, такова тактовая частота Кугуара. Каждый цикл может содержать несколько команд - помните, что Кугуар поддерживает параллельную обработку, до 32 макрокоманд одновременно. Поэтому в приведенном примере столь высокая скорость генерирования алфавита объясняется тем, что контроллер выдает несколько символов одновременно (в зависимости от того, насколько быстро вы вращаете кольцо антенны). Это легко увидеть в Тестере символов Foxy. Несколько команд Key Down *посылаются одновременно в рамках одного цикла, за которым следует второй цикл, содержащий соответствующие команды* Key Up. *К этой теме мы еще вернемся.*

Вернемся к модификатору FORCE_MACROS. *Теперь каждый символ (к примеру, "a") конвертируется в* < KD(a) KU(a) >, *то есть на каждый символ приходится не одно, а 4 действия, которые, к тому же, принудительно исполняются до конца в рамках одного цикла, прежде чем могут быть исполнены другие команды. Именно по этой причине генерирование символов происходит* ***намного*** *медленнее.*

На том же самом примере цифрового режима оси ANT *проиллюстрируем способность Cougar генерировать несколько символов за цикл и некоторые последствия этой способности в части порядка символов. Вернемся к примеру:*

ANT 2 26 a b c d e f g h i j k l m n o p q r s t u v w x y z

Кажется очевидным, что с данным выражением вращение кольца антенны по часовой стрелке генерирует буквы в алфавитном порядке, а против часовой стрелки - в обратном порядке. Однако если вращать кольцо достаточно быстро, то в Тестере символов утилиты Foxy вы увидите, что символы идут не по порядку. Это особенно хорошо заметно при вращении кольца против часовой стрелки. Результат может быть таким:

"x y z u v w p q r s t m n o j k l f g h i a b c d e"

Также в окне Up/Down/Keycode видно, что символы генерируются группами кодов нажатия, за которыми следуют группы кодов отпускания, то есть Down (z y x) Up (z y x), *а не* Down (z) Up (z) Down (y) Up (y) *и т. д. Почему это происходит при вращении кольца в обратную сторону? Является ли это багом? Нет, не является. Это связано с тем, как USB-устройства посылают клавиатурные коды." Предыдущие котроллеры TM посылали клавиатурные символы, использу я стандарт PS2, который работает через индивидуальные коды нажатия и отпускания. В случае с USB-устройствами это происходит следующим образом: символы посылаются*

как таковые, затем операционная система считывает буфер клавиатуры и определяет, какие клавиши нажаты, а какие нет. Когда Cougar изменяет содержимое буфера, чтобы указать, что клавиши z, x и y нажаты, операционная система определяет это при считывании и воспроизводит все нажатые в данный момент символы в строго определенном порядке - для букв это ***алфавитный*** *порядок. Именно поэтому, несмотря на то, что вы запрограммировали кольцо антенны выдавать символы z, x и y именно в таком порядке, при быстром вращении кольца Кугуар помещает все три символа в один цикл, в итоге компьютер видит три нажатых клавиши и выдает их в* ***алфавитном*** *порядке внутри цикла, то есть* x, y *и* z.

Конечно, если добавить модификатор FORCE_MACROS*, алфавит будет воспроизводиться строго по порядку, так как все символы будут посылаться в рамках отдельного цикла, но намного медленнее.*

Надеемся, вы поняли все вышесказанное. Подумайте хорошенько, когда нужно, а когда не нужно использовать FORCE_MACROS*. Помните основное правило: самое главное - как в итоге ваши выражения воспринимаются операционной системой. Именно реальное генерирование символов в игре и есть окончательный этап тестирования. Пробуйте различные способы, загружайте файлы и смотрите, что получится. Последнее замечание: иногда при тестировании приведенного в качестве примера выражения* (*без использования* FORCE_MACROS) *при слишком быстром вращении кольца антенны некоторые символы могут выпадать, а вместо них могут появляться коды отпускания клавиш, типа* SHF *или* CTL*. Нам известно об этом явлении, и это не похоже на баг Кугуара. Скорее всего, это связано с драйвером клавиатуры Microsoft. Похоже, при переполнении буфера драйвер иногда повторно инициализируется, в результате чего на выходе появляются неправильные коды отпускания. Кугуар просто не может их выдавать.*

6.2.2 Тип 2: произвольная последовательность символов, фиксированные зоны

Синтаксис выражения 2 типа:

Обозн. оси	Тип цифрового выражения	Количество символов или макросов (макс. 50)	Последовательность символов, макросов, логических флагов	FORCE_ MACROS *(дополнительно)*
напр. ANT	2	5	a b c d e	- FORCE_MACROS

а так выглядит командная строка в файле настроек:

ANT 2 5 a b c d e - FORCE_MACROS

Вращение кольца масштаба от упора до упора по, а затем против часовой стрелки, выдает следующие символы:

b c d e d c b a

Как и в случае с выражением 1 типа, генерируются одиночные символы (*но логические флаги остаются включенными – см. далее в руководстве*). То же самое справедливо и для макрокоманд. Отметьте также, что если начинать вращать кольцо антенны из своего крайнего левого положения по часовой стрелке, то первый символ, который будет сгенерирован - "b", а не "a", как вы могли подумать. Если вы повернете кольцо масштаба от упора до упора несколько раз, символы будут генерироваться следующим образом:

b c d e d c b a b c d e d c b a b *и т. д.*

а не:

a b c d **e e** d c b **a a** b c d **e e** d c b a *etc. etc.*

Отметьте, что в отличие от синтаксиса предыдущих контроллеров TM, параметр Количество символов или макросов может быть как четным, так и нечетным. Естественно, если вы программируете, к примеру, кольцо антенны, наверное, удобнее иметь нечетное количество символов в последовательности, чтобы средний символ как раз приходился на центральное положение кольца.

Точно также в выражении 2 типа могут использоваться макрокоманды:

RNG 2 5 Emcon-1 Emcon-2 Emcon-3 Emcon-4 Emcon-5

и в макрофайле:

```
Emcon-1 = e DLY(40) 1
Emcon-2 = e DLY(40) 2
Emcon-3 = e DLY(40) 3
Emcon-4 = e DLY(40) 4
Emcon-5 = e DLY(40) 5
```

Мы еще не обсуждали логические флаги, но если вы действительно доберетесь до этой стороны программирования HOTAS, то просто запомните, что в выражении 2 типа можно использовать логические флаги, в том числе функции их переключения:

RNG 2 4 X1 X2* X3* X4

Это будет описано более подробно в разделе логического программирования. В выражениях 2 типа допускается одновременное использование одиночных символов, макрокоманд и логических флагов:

ANT 2 5 a Emcon-2 c ^ X1

6.2.2.1 Модификатор FORCE_MACROS

Подробное объяснение использования этого модификатора см. в раздел 6.2.1.1 посвященном выражениям 1 типа.

Примечания

1. *Со всеми выражениями цифрового типа можно использовать ключи* /U, /M, /D, /I, /O. *Однако, будьте внимательны при комбинировании ключей цифрового выражения с ключами в выражениях логического программирования (см. далее в Руководстве), так как результат может быть неожиданным. Пример (для самых продвинутых пользователей):*

```
ANT /U 1 6 a b c
    /M 1 6 d e f
    /D 2 3 (DLY(5000) X1) X2 X3
BTN X1 /U a
       /M b
       /D c
```

При вращении кольца антенны с переключателем в положении /D, логический флаг будет генерировать "a", "b" или "c" в зависимости от того, изменилось ли положение переключателя во время 5-секундной задержки.

2. *Однако нельзя использовать ключи /U, /M, /D, /I, /O* ***внутри*** *самого выражения (это относится к выражениям цифровых режимов всех типов). Следующее выражение приведет к ошибке компиляции:*

RNG 2 3 a b macro_1

если macro_1 = /I KP1 /O KP2

3. *Допустим, у вас есть:*

ANT /I 2 3 SM1 SM2 SM3
/O 2 3 SM4 SM5 SM6

где: SM1 = a
SM2 = /H b
SM3 = c
SM4 = d
SM5 = /H e
SM6 = f

Допустим, кнопка S3 отжата, а кольцо антенны стоит в центральном положении (генерируя удерживаемый символ "e"), и в этот момент вы нажимаете S3. В этом случае "e" будет прервано, и автоматически будет удерживаться "b". Когда кнопка S3 будет отпущена, возобновится удерживаемое "e".

4. *Ключи /P, /R, /H можно использовать в выражениях 1, 2, 5 и 6 типов.*

6.2.3 Тип 3: генерирование удерживаемых клавиш

Синтаксис выражения 3 типа:

Назв. оси	Тип цифрового выражения	Символ левого положения	Символ центр. положения	Символ правого положения
напр. RDDR	3	l	c	r

командная строка в файле настроек:

RDDR 3 l c r

Нажатие левой педали приведет к генерированию удерживаемого "l", точно так же, как при использовании ключа /H.

Примечания

1. *Ось разбивается не на равные зоны, а скорее как на схеме внизу, иначе центральное положение будет иметь слишком большой ход:*

Левая зона	Центр. зона	Правая зона
/H l	c	/H r

2. *В выражениях 3 типа можно использовать логические флаги.*

3. *Если вы не хотите генерировать ничего в центральном положении, используйте пустой символ* (^):

RDDR 3 l ^ r

6.2.4 Тип 4: импульсное генерирование символов

Синтаксис выражения 4 типа:

Назв. оси	Тип цифр. выр-я	Длит. импульса (мс)	Символ левого положения	Символ центр. положения	Символ правого положения
напр. RNG	4	1000	l	c	r

командная строка в файле настроек:

RNG 4 1000 l c r

Импульсным генерированием мы называем воспроизведение символа каждые x миллисекунд. Выражения 4 типа являются нововведением в программировании TM. В приведенном примере при вращении кольца масштаба против часовой стрелки символ "l" воспроизводится каждые 1000 миллисекунд (или раз в секунду). Соответственно, при вращении кольца обратно в центральном положении генерируется одиночный символ "c", а затем раз в секунду генерируется "r".

Примечания

1. *Допускается также использование макросов и логических флагов.*

2. *Вместо символа центрального положения можно использовать пустой символ:* ^

RNG 4 60 l ^ r

3. *Частота импульса - это время в миллисекундах, и допустимые значения - то 0 до 82800000 (23 часа!)*

6.2.5 Тип 5: произвольная последовательность символов, изменяемые зоны

Синтаксис выражения 5 типа:

Назв. оси	Тип цифрового выражения	Кол-во зон (макс. 50)	Ширина зон (в процентах)	Последоват-ть символов/ макросов/логич. флагов	FORCE_ MACROS *(дополн.)*
напр. THR	5	4	(0 20 45 70 100)	a b c d	- FORCE_MACROS

командная строка в файле настроек:

THR 5 4 (0 20 45 70 100) a b c d - FORCE_MACROS

Выражения 5 типа могут показаться сложнее других, но на самом деле это просто частный случай выражения 2 типа. Напомним, что в выражениях 2 типа последовательность символов равномерно распределялась на весь ход оси. В выражении 5 типа ось разбивается на зоны, или полосы, определяемые пользователем, по которым и распределяются символы.

В приведенном примере задано 4 зоны:

- От 0 до 20% хода оси: символ "a"
- 21 - 45%: символ "b"
- 46% - 70%: символ "c"
- 71% - 100%: символ "d"

Во всех остальных аспектах выражения 5 типа абсолютно идентичны выражениям 2 типа. Цифровые режимы работы оси могут запросто сосуществовать ее аналоговым функционированием. Например, можно использовать стандартный аналоговый РУД, но использовать выражение 5 типа для генерирования символов в определенной зоне ее рабочего хода. Таким способом можно легко запрограммировать реверс, или включение колесных тормозов при посадке на минимальной тяге, используя, к примеру, ключи положения переключателя ближний-дальний бой:

```
THR /U
    /M
    /D 5 1 (0 5) Wheelbrakes
```

и в макрофайле:

Wheelbrakes = /P b /R b

6.2.5.1 Модификатор FORCE_MACROS

Подробное объяснение использования этого модификатора см. в раздел 6.2.1.1 посвященном выражениям 1 типа.

Примечания

1. *С предыдущими контроллерами TM HOTAS можно было запрограммировать положение минимальной тяги с помощью выражения* BTN MT*, но только в случае, если РУД не использовалась как аналоговая ось. Выражение* BTN MT *более не поддерживается, так как его с успехом можно заменить, используя обширные возможности по комбинированию аналогового и цифровых режимов работы осей. Так,* BTN MT *можно заменить на выражение из примера:* THR 5 1 (0 5) макро_команда

2. *Регулировки кривых отклика, заданные для аналогового режима осей, никак не влияют на цифровые режимы всех типов. В цифровом режиме отклик оси является линейным.*

3. *Как уже было сказано, в выражениях 1, 2, 5 и 6 типов можно использовать ключи* /P, /R, /H*, однако они должны входить в макрокоманду, или быть заключены в круглые скобки. То есть:*

THR 5 1 (0 5) Wheelbrakes

где в макрофайле:

Wheelbrakes = /P b /R b

верно. Можно также использовать:

THR 5 1 (0 5) (/P b /R b)

однако:

THR 5 1 (0 5) /P b /R b *приведет к ошибке компиляции.*

6.2.6 Тип 6: повторяющиеся символы, изменяемые зоны

Синтаксис выражения 6 типа:

Назв. оси	Тип цифр. выр-я	Кол-во зон (макс. 50)	Ширина зон (%)	Увелич.	Уменьш.	Центр *(допол.)*	FORCE_ MACROS *(дополн.)*
напр. ANT	6	5	(8 20 40 45 70 80)	u	d	c	- FORCE_MACROS

командная строка в файле настроек:

ANT 6 5 (8 20 40 45 70 80) u d c - FORCE_MACROS

Выражение 6 типа - это, по сути, выражение 1 типа, но ход оси не делится автоматически на равные зоны, а зоны эти задаются пользователем.

Однако, в отличие от выражения 1 типа, **если вы используете символ центрального положения, количество зон должно быть нечетным.**

В приведенном примере ось разбита на 5 зон:

- от 8 до 20% хода
- 21 - 40%
- 41 - 45%
- 46 - 70%
- 71 - 80%

И точно также, как и с выражением 1 типа, при вращении кольца антенны получаем:

u u c u u d d c d d

Если бы было выражение:

ANT 6 5 (8 20 40 45 70 80) u d

тогда воспроизводилось бы:

u u u u u d d d d d

6.2.6.1 Модификатор FORCE_MACROS

Подробное объяснение использования этого модификатора см. в раздел 6.2.1.1 посвященном выражениям 1 типа.

6.2.7 Направление осей: аналоговые значения и цифровые выражения

В данном разделе показаны направления всех осей и соответствие аналоговых значений, а также примеры выражений цифрового режима всех типов.

6.2.7.1 Аналоговые значения осей

Положение оси	Аналоговое значение
JOYX - влево	0
JOYX - вправо	100%
JOYY - на себя	100%
JOYY - от себя	0
THR - вперед	100%
THR - назад	0
RNG - против ч. с. [прим. 1]	100%
RNG - по ч. с.	0
ANT - против ч. с. [прим. 1]	100%
ANT - по ч. с.	0
MIX - влево [прим. 2]	-
MIX - вправо	-
MIY- вниз	-
MIY - вверх	-
RDDR - лев. педаль вперед	0
RDDR - прав. педаль вперед	100%
LBRK, RBRK - отпущено	100%
LBRK, RBRK - нажато	0

Примечания

1. *С кольцом масштаба происходит постоянная путаница направлений. Запомните, что для определения направления вращения по или против ч. с. на кольца надо смотреть* ***сверху****.*

2. *Микроджойстик* (MIX, MIY) *по умолчанию не виден в Windows как аналоговое устройство. Однако это не означает, что он не может работать в аналоговом режиме, так как на него можно вешать другие оси.*

6.2.7.2 Цифровые выражения 1 типа

JOYX 1 6 r l — При перемещении оси X джойстика слева направо генерируются символы "r", справа налево - "l".

JOYY 1 6 f b — При перемещении оси X джойстика вперед генерируются символы "f", назад - "b".

THR 1 6 f b — При перемещении РУД вперед генерируются символы "f", назад - "b".

RNG 1 6 r l — При вращении кольца масштаба по часовой стрелке генерируются символы "r", против часовой стрелки - "l".

ANT 1 6 r l — При вращении кольца антенны по часовой стрелке генерируются символы "r", против часовой стрелки - "l".

MIX 1 6 r l — При перемещении оси X микроджойстика слева направо генерируются символы "r", справа налево - "l".

MIY 1 6 u d — При перемещении оси Y микроджойстика вверх генерируются символы "u", вниз - "d".

RDDR 1 6 l r — Нажатие на левую педаль вперед выдает символы "l", на правую - "r".

LBRK 1 6 d u — Нажатие на левый педальный тормоз выдает символы "d", отпускание - "u". (То же самое справедливо для RBRK).

6.2.7.3 Цифровые выражения 2 типа

JOYX 2 5 a b c d e — При перемещении оси X джойстика слева направо генерируются символы "a b c d e", справа налево - "e d c b a".

JOYY 2 5 a b c d e — При перемещении оси X джойстика вперед генерируются символы "a b c d e", назад - "e d c b a".

THR 2 5 a b c d e — При перемещении оси РУД вперед генерируются символы "a b c d e", назад - "e d c b a".

RNG 2 5 a b c d e — При вращении кольца масштаба по часовой стрелке генерируются символы "a b c d e" , против часовой стрелки - "e d c b a".

ANT 2 5 a b c d e — При вращении кольца антенны по часовой стрелке генерируются символы "a b c d e" , против часовой стрелки - "e d c b a".

MIX 2 5 a b c d e — При перемещении оси X микроджойстика слева направо генерируются символы "a b c d e" , справа налево - "e d c b a".

MIY 2 5 1 2 3 4 5 — При перемещении оси Y микроджойстика вверх генерируются символы "1 2 3 4 5" , вниз - "5 4 3 2 1".

RDDR 2 5 a b c d e — Из исходного положения левой педали до конца вперед, перемещение правой педали вперед до конца выдает "a b c d e", затем перемещение левой педали обратно до конца вперед выдает "e d c b a".

LBRK 2 5 a b c d e — При нажатии на педальный тормоз выдается "a b c d e", при отпускании - "e d c b a". (То же для RBRK).

6.2.7.4 Цифровые выражения 3 типа

JOYX 3 l ^ r — При отклонении оси X джойстика влево удерживается клавиша "l", вправо - "r".

JOYY 3 b ^ f — При отклонении оси Y джойстика назад удерживается клавиша "b", вперед - "f".

THR 3 b ^ f — При положении оси РУД менее середины хода удерживается клавиша "b", более середины хода - "f".

RNG 3 l ^ r — При положении кольца масштаба по часовой стрелке от центра удерживается клавиша "r", против часовой стрелки от центра - "l".

ANT 3 l ^ r — При положении кольца антенны по часовой стрелке от центра удерживается клавиша "r", против часовой стрелки от центра - "l".

MIX 3 l ^ r — При отклонении оси X микроджойстика влево удерживается клавиша "l", вправо - "r".

MIY 3 d ^ u — При отклонении оси Y микроджойстика вниз удерживается клавиша "d", вверх - "u".

RDDR 3 l ^ r — При нажатой вперед левой педали удерживается клавиша "l", при правой - "r".

LBRK 3 u ^ d — При положении педали тормоза от полностью нажатого до середины хода удерживается клавиша "d", от середины хода до полностью отпущенного положения - "u". (То же для RBRK).

6.2.7.5 Цифровые выражения 4 типа

JOYX 4 300 l ^ r — При отклонении оси X джойстика влево "пульсирует" клавиша "l", вправо - "r".

JOYY 4 300 b ^ f — При отклонении оси Y джойстика назад "пульсирует" клавиша "b", вперед - "f".

THR 4 300 b ^ f — При положении оси РУД менее середины хода "пульсирует" клавиша "b", более середины хода - "f".

RNG 4 300 l ^ r — При положении кольца масштаба по часовой стрелке от центра "пульсирует" клавиша "r", против часовой стрелки от центра - "l".

ANT 4 300 l ^ r — При положении кольца антенны по часовой стрелке от центра "пульсирует" клавиша "r", против часовой стрелки от центра - "l".

MIX 4 300 l ^ r — При отклонении оси X микроджойстика влево "пульсирует" клавиша "l", вправо - "r".

MIY 4 300 d ^ u — При отклонении оси Y микроджойстика вниз "пульсирует" клавиша "d", вверх - "u".

RDDR 4 300 l ^ r — При нажатой вперед левой педали "пульсирует" клавиша "l", при правой - "r".

LBRK 4 300 u ^ d — При положении педали тормоза от полностью нажатого до середины хода "пульсирует" клавиша "d", от середины хода до полностью отпущенного положения - "u". (То же для RBRK).

6.2.7.6 Цифровые выражения 5 типа

JOYX 5 5 (0 20 40 60 80 100) a b c d e — При перемещении оси X джойстика слева направо генерируются символы "a b c d e", справа налево - "e d c b a".

JOYY 5 5 (0 20 40 60 80 100) a b c d e — При перемещении оси X джойстика вперед генерируются символы "a b c d e", назад - "e d c b a".

THR 5 5 (0 20 40 60 80 100) a b c d e	При перемещении оси РУД вперед генерируются символы "a b c d e", назад - "e d c b a".
RNG 5 5 (0 20 40 60 80 100) a b c d e	При вращении кольца масштаба по часовой стрелке генерируются символы "a b c d e" , против часовой стрелки - "e d c b a".
ANT 5 5 (0 20 40 60 80 100) a b c d e	При вращении кольца антенны по часовой стрелке генерируются символы "a b c d e" , против часовой стрелки - "e d c b a".
MIX 5 5 (0 20 40 60 80 100) a b c d e	При перемещении оси X микроджойстика слева направо генерируются символы "a b c d e" , справа налево - "e d c b a".
MIY 5 5 (0 20 40 60 80 100) 1 2 3 4 5	При перемещении оси Y микроджойстика вверх генерируются символы "1 2 3 4 5" , вниз - "5 4 3 2 1".
RDDR 5 5 (0 20 40 60 80 100) a b c d e	Из исходного положения левой педали до конца вперед, перемещение правой педали вперед до конца выдает "a b c d e", затем перемещение левой педали обратно до конца вперед выдает "e d c b a".
LBRK 5 5 (0 20 40 60 80 100) a b c d e	При нажатии на педальный тормоз выдается "a b c d e", при отпускании - "e d c b a". (То же для RBRK).

6.2.7.7 Цифровые выражения 6 типа

JOYX 6 5 (0 20 40 60 80 100) r l	При перемещении оси X джойстика слева направо генерируются символы "r", справа налево - "l".
JOYY 6 5 (0 20 40 60 80 100) f b	При перемещении оси X джойстика вперед генерируются символы "f", назад - "b".
THR 6 5 (0 20 40 60 80 100) f b	При перемещении РУД вперед генерируются символы "f", назад - "b".

RNG 6 5 (0 20 40 60 80 100) r l	При вращении кольца масштаба по часовой стрелке генерируются символы "r", против часовой стрелки - "l".
ANT 6 5 (0 20 40 60 80 100) r l	При вращении кольца антенны по часовой стрелке генерируются символы "r", против часовой стрелки - "l".
MIX 6 5 (0 20 40 60 80 100) r l	При перемещении оси X микроджойстика слева направо генерируются символы "r", справа налево - "l".
MIY 6 5 (0 20 40 60 80 100) u d	При перемещении оси Y микроджойстика вверх генерируются символы "u", вниз - "d".
RDDR 6 5 (0 20 40 60 80 100) l r	Нажатие на левую педаль вперед выдает символы "l", на правую - "r".
LBRK 6 5 (0 20 40 60 80 100) d u	Нажатие на левый педальный тормоз выдает символы "d", отпускание - "u". (То же для RBRK).

На этом мы закончим тему программирования осей в цифровом режиме. Теперь перейдем к аналоговым аспектам работы оси и способам их изменения через файлы настроек.

6.3 Кривые отклика (CURVE)

Все 10 осей по умолчанию имеют линейную характеристику отклика. Иными словами, физическое перемещение оси на определенную величину приводит к изменению сигнала, воспринимаемого игрой, на величину, прямо пропорциональную исходной. При перемещении оси на 10% хода игра видит изменение сигнала на 10% от полного диапазона. Эту характеристику можно изменить для каждой из 10 осей, изменяя их кривые отклика, или, иными словами, их чувствительность.

Существуют два выражения для определения и изменения кривых отклика:

Конфигурационное выражение

USE CURVE (Название_оси, Чувствительность)

Синтаксис команды

CURVE [Ключи] (Название_оси, Чувствительность)

где:

Название_оси - одно из ниже перечисленного:

JOYX, JOYY	(совместно именуемые **JOYSTICK**)
THR	
RNG, ANT	(совместно именуемые **ROTARIES**)
MIX, MIY	(совместно именуемые **MICROSTICK**)
LBRK, RBRK	(совместно именуемые **TOEBRAKES**)
RDDR	

Чувствительность - значение от –32 до 32 (*хотя значения более 20 приводят к такому результату, который вы вряд ли захотите использовать!*)

Отрицательные числа означают сниженную чувствительность – удобно при посадке, дозаправке в воздухе, полетах в формациях. Значение 0 приводит отклик оси к линейной характеристике (и имеет приоритет над выражениям USE CURVE), а положительные числа означают увеличение чувствительности (удобно в ближнем воздушном бою, или исполнении резких маневров).

Ключи *(дополнительно)*

Разрешено использование /U, /M, /D и /I, /O

На примерах ниже мы объясним различия между действием выражений USE CURVE и CURVE.
USE CURVE является **конфигурационным выражением** – (как и **все** выражения, начинающиеся с USE). Это выражение находится в начале файла настроек, и не может быть запрограммировано на определенную кнопку. Это выражение определяет **кривые отклика осей по умолчанию для всего файла**. Изначально, как уже говорилось, отклик всех осей имеет линейную характеристику, то есть выражение для оси X джойстика:

```
USE CURVE (JOYX, 0)
```

бесполезно, так как именно такой кривая отклика была бы и в отсутствие конфигурационного выражения. Если же в файле настроек присутствует выражение:

```
USE CURVE (JOYSTICK, 2)
```

в этом случае отклик обеих осей (X и Y) джойстика будет более чувствительным. Отметьте, что для обозначения обеих осей было использовано групповое название "JOYSTICK", которое компилятор автоматически превратит в:

```
USE CURVE (JOYX, 2)
USE CURVE (JOYY, 2)
```

Таким образом, выражение USE CURVE используется для изменения отклика осей по умолчанию.

Напротив, выражение CURVE, использующее такой же синтаксис, может быть запрограммировано на кнопку или положение хэта, включено в строку цифрового режима оси или может использоваться обособлено (с ключами). Это не конфигурационное выражение. Предположим, мы хотим менять отклик оси Y джойстика в зависимости от положения переключателя "дальний - ближний бой":

```
CURVE  /U (JOYY, 2)   Rem Более чувствительная ось для ближнего
                      воздушного боя
       /M (JOYY, 0)   Rem  Нормальный отклик
       /D (JOYY, -2)  Rem  Меньшая чувствительность для посадки
```

Или для микроджойстика с кнопкой S3:

```
CURVE  /I (MICROSTICK, 2)   Rem Менее чувствительный отклик
       /O (MICROSTICK, 0) Rem  Нормальный отклик
```

Можно также составлять комбинации:

```
CURVE  /U /I (MICROSTICK, 2) (THR, 2)
          /O (MICROSTICK, 0)
       /M /I (RDDR, -2)
          /O (RDDR, 0) (TOEBRAKES, 2)
       /D (JOYY, -2)
```

Выражение CURVE может программироваться непосредственно на орган управления:

```
BTN T7 /P CURVE (JOYX, 3)  Rem Чувствительно
       /R CURVE (JOYX, 0)  Rem  Нормально
```

Можно также включать выражения CURVE в выражения цифрового режима осей. Допустим, мы хотим изменять чувствительность оси X джойстика в зависимости от положения РУД:

```
THR  2  5  CURVE(JOYX,  -3)  CURVE  (JOYX,  -1)  CURVE  (JOYX,  0)
CURVE (JOYX, 2) CURVE (JOYX, 5)
```

На малых значениях тяги джойстик будет менее чувствительным, затем его чувствительность будет повышаться по мере увеличения тяги. Кстати, это

хороший пример того, как РУД может работать как аналоговая ось и одновременно изменять определенные параметры в цифровом режиме.

Примечания

1. *Если одна ось переносится на другую, то установки кривой отклика также переносятся.*

2. *С помощью выражения* CURVE *нельзя задавать мертвые зоны - их необходимо задавать в Панели управления, сохранять профиль и задействовать его с помощью конфигурационного выражения USE PROFILE (см. Раздел 5).*

3. *В файле настроек нельзя использовать более одного обособленного выражения* CURVE. *В примере ниже второе выражение приведет к ошибке компиляции:*

```
CURVE  /I (MICROSTICK, 2)   Rem More responsive
       /O (MICROSTICK, 0) Rem  Normal

CURVE  /I (ROTARIES, 2)   Rem More responsive
       /O (ROTARIES, 0) Rem  Normal
```

Не путать с выражением CURVE, *запрограммированным на кнопку или иной орган управления, которое можно использовать неоднократно.*

4. *Если вы используете выражение* CURVE *отдельно, а не как функцию какого-либо органа управления, то вы должны обязательно использовать ключи* /U, /M, /D *или* /I, /O. *Или используйте конфигурационное выражение. То есть:*

```
CURVE (JOYSTICK, 10)          приведет к ошибке, а:

USE  CURVE (JOYSTICK, 10)     нормально.
```

5. *Вместо программирования кривых отклика в файле настроек можно использовать сохраненный профиль Панели управления (см. конфигурационное выражение* USE PROFILE *выше). Преимущество такого способа в том, что можно задавать мертвые зоны.*

6.4 Триммер осей (TRIM)

Триммирование оси позволяет эмулировать определенное отклонение осей при отпущенных органах управления. К примеру, вы выполняете горизонтальный полет, но ваш самолет стремится к набору. Вам приходится

постоянно компенсировать это легким отклонением РУС от себя. Вместо этого вы можете использовать триммер, так что при отпущенной ручке управления игра будет считать, что она немного отклонена вперед. Функция триммирования может быть использована на любой из 10 аналоговых осей Кугуара.

Синтаксис команды

TRIM (Название_оси, Величина_триммирования)

и:

HOLDTRIM (Название_оси)

где:

Название_оси - одно из ниже перечисленного:

JOYX, JOYY	(совместно именуемые **JOYSTICK**)
THR	
RNG, ANT	(совместно именуемые **ROTARIES**)
MIX, MIY	(совместно именуемые **MICROSTICK**)
LBRK, RBRK	(совместно именуемые **TOEBRAKES**)
RDDR	

Величина_триммирования - число от -128 до 127, или параметр TO_CURRENT.

Значение, равное 0 означает отсутствие триммирования.
Положительное число означает триммирование в сторону увеличения, отрицательное - в сторону уменьшения.
Параметр TO_CURRENT означает, что контроллер считывает текущее положение оси и устанавливает триммер на это положение.

Теперь рассмотрим пример, как при помощи программирования колец антенны и масштаба регулировать триммер осей X и Y джойстика при помощи цифровых выражений 1 типа.

```
RNG 1 12 TRIM (JOYX, 20+) TRIM (JOYX, 20-)
ANT 1 12 TRIM (JOYY, 20-) TRIM (JOYY, 20+)
```

Вращение кольца антенны по часовой стрелке будет постепенно уменьшать значение оси Y с шагом 20, что имитирует отклонение ручки управления от себя. Таким образом вы можете компенсировать стремление самолета к набору высоты при отпущенной ручке управления.

Можно запрограммировать кнопку S2 на джойстике для отключения триммера:

```
BTN S2 TRIM (JOYX, 0) TRIM (JOYY, 0)  Rem снятие триммера с обеих осей
```

или:

```
BTN S2 TRIM (JOYSTICK, 0)
```

Можно также одной кнопкой применить триммирование определенной оси на определенную величину:

```
BTN S4 TRIM (JOYX, 5) TRIM (JOYY, -10)
```

И наконец, можно отклонить ручку управления в нужное положение, и установить триммера на это положение, с тем, чтобы при отпускании ручки значения осей остались теми же:

```
BTN S2 /I TRIM (JOYSTICK, TO_CURRENT) Rem Триммировать на текущее положение
        /O TRIM (JOYSTICK, 0) Rem Снять триммирование
```

Однако, здесь существует одна тонкость. Обратите внимание, что было сказано: "при отпускании ручки". Попробуем прояснить ситуацию. Предположим, вы выполняете горизонтальный полет, и постоянно даете ручку немного от себя для компенсации стремления самолета к набору высоты. Затем вы вышеописанным способом триммируете оси на текущее положение. Теперь, как вы думаете, вы можете вернуть ручку в центральное положение, и самолет продолжит горизонтальный полет. Правильно, но! Только в том случае, если вы умудритесь в рамках одного и того же 30-миллисекундного цикла поставить триммер и вернуть ручку в центральное положение. Скорее всего, это вам не удастся. Что же произойдет? На какую-то секунду самолет резко клюнет носом вниз. Почему? Дело в том, что вы установили триммер на величину, равную отклонению ручки. Теперь, при ручке в центре игра будет получать это значение. Но ручку вы еще не успели отпустить (наверняка) - поэтому в этот момент игра получит значение оси, равное величине триммера ПЛЮС отклонение ручки, или (грубо) в два раза больше! Вряд ли это вам очень понравится на бреющем полете :).

Как обойти это явление?
Есть два способа: простой и сложный. Начнем с простого:

```
BTN S2 HOLDTRIM (JOYSTICK)
```

Эта функция используется так: вы отклоняете ручку управления в положение, при котором самолет выдерживает горизонтальный полет; нажимаете (и не отпускаете) кнопку S2; возвращаете ручку в нейтральное положение, и отпускаете кнопку S2. Пока кнопка S2 нажата, игра все время получает одно и то же значение осей - то, которое было на момент нажатия. Теперь можете класть ноги на приборную панель и пить кофе :).

Теперь сложный способ, который на самом деле не так уж и сложен. Он состоит в использовании выражений TRIM TO_CURRENT в сочетании с LOCK и UNLOCK (*см. далее*):

```
BTN S2 /P LOCK (JOYSTICK, LASTVALUE) TRIM(JOYSTICK, TO_CURRENT)
       /R UNLOCK (JOYSTICK)
```

С точки зрения ваших действий происходит все то же самое. А теперь как это работает внутри: Первое, что происходит - оси джойстика как бы "блокируются" в текущем положении. Затем происходит триммирование осей на эти значения. Затем, когда ручка возвращается в центральное положение, блокировка снимается. Именно так компилятор конвертирует выражение BTN S2 HOLDTRIM (JOYSTICK).

Примечания

1. *Изменение величины триммера TRIM приводит к прибавлению или вычитанию целочисленного значения из значения оси, и никак не влияет на кривую отклика - просто происходит сдвиг всего графика кривой на заданную величину.*

2. *При триммировании оси с линейным графиком отклика ее рабочий диапазон "обрезается" с одной или другой стороны.*

3. *Инверсия оси не влияет на направление триммирования. Цифровые режимы работы осей не инвертируются при инверсии аналоговой оси.*

4. *Будьте внимательны, с какой стороны от числа вы ставите знак "+" или "-" - эффект будет различный (это касается не только триммирования). Знак слева от числа означает абсолютную величину триммирования (и ее знак), знак справа от числа означает прибавление или вычитание указанной величины от текущего значения триммера, то есть означает относительное изменение. См. раздел "Управление мышью и микроджойстик" для более подробного объяснения различий действия знаков в зависимости от их положения справа или слева от числа. Модуль Composer утилиты Foxy поможет вам избежать ошибок.*

5. *Все ниже приведенные примеры использования выражения HOLDTRIM допустимы:*

```
BTN T6 a b HOLDTRIM (RNG) c d
BTN S4 /P a HOLDTRIM (RNG)
       /R b
BTN S1 a { HOLDTRIM (JOYY) b HOLDTRIM (ANT) }
```

Отметьте, что при использовании нескольких выражений HOLDTRIM *их необходимо заключать в фигурные скобки.*

6. *Нельзя называть макрокоманду* TRIM*, (заглавными буквами), но можно, например,* Trim *или* Trim_Hold*.*

7. *Можно использовать ключ автоповтора* (/A) *в сочетании с командой* TRIM *для плавного управления триммерами при помощи кнопок или хэта, например:*

```
BTN H1U /A TRIM (JOYY, 5-) DLY(120)
BTN H1D /A TRIM (JOYY, 5+) DLY(120)
BTN H1L /A TRIM (JOYX, 5-) DLY(120)
BTN H1R /A TRIM (JOYX, 5+) DLY(120)
```

Помимо этого, триммерами можно управлять при помощи цифровых режимов осей, которых не существует физически (например, педалей или педальных тормозов. (*См. раздел руководства* "Оси, видимые в Windows" *для более подробной информации о том, как заставить Windows поверить в присутствие осей, которых физически нет*).

6.5 Отключение осей

По умолчанию операционная система (и игра) видит все аналоговые оси, **кроме** осей микроджойстика. Однако в ряде случаев может понадобиться отключить некоторые оси, к примеру, если вы хотите, чтобы они работали только в цифровом режиме. Оси можно отключать при помощи конфигурационных выражений в файле настроек:

Конфигурационное выражение
DISABLE Название_оси

где:

Название_оси - одно из ниже перечисленного:

THR	
RNG, ANT	(совместно именуемые **ROTARIES**)
LBRK, RBRK	(совместно именуемые **TOEBRAKES**)
RDDR	

Как вы видите, некоторые оси объединены в группы с одним названием, для упрощения их обозначения:

DISABLE ROTARIES	конвертируется компилятором в	DISABLE RNG DISABLE ANT
DISABLE TOEBRAKES		DISABLE LBRK DISABLE RBRK

Все конфигурационные выражения для каждой оси должны располагаться на отдельной строке, в которой допускается наличие только комментариев REM.

То есть выражения:

```
DISABLE THR Rem Отключить РУД
DISABLE ANT Rem Отключить кольцо антенны на РУД
```

верны, тогда как:

```
DISABLE (THR, ANT, RNG)  неверно.
```

Новая команда DISABLE пришла на смену команде USE NO из синтаксиса предыдущих программируемых манипуляторов TM (USE NOMOUSE, USE NOTHR). И еще одно: теперь по умолчанию все оси представлены, то есть необходимость в выражениях типа USE RCS, USE TQS и т. д. отпала, и они более не поддерживаются.

6.5.1 Отключение и подключение осей в игре при помощи команд LOCK, UNLOCK

Итак, мы узнали, как отключить оси с помощью конфигурационного выражения. Когда ось отключена таким образом, игра ее не увидит в аналоговом виде вообще, и ось либо совсем не работает, либо может быть запрограммирована в цифровых режимах.

Однако возможна ситуация, когда вы захотите использовать видимую в игре аналоговую ось только в цифровом режиме, так, чтобы ее аналоговое значение не изменялось. К сожалению, временно удалить ось из числа видимых аналоговых осей во время игры нельзя, так как это может привести к зависанию системы, вылету игры и вообще всяким подобным неприятным последствиям.

Если нельзя удалить, а затем добавить ось, единственным способом остается на время заблокировать аналоговое значение этой оси, чтобы при ее перемещении выполнялись только команды цифрового режима. Аналоговое значение оси может быть заблокировано с помощью команды LOCK и вновь разблокировано с помощью команды UNLOCK:

Синтаксис команды

LOCK (Название_оси, Блокируемое_значение%)

UNLOCK (Название_оси)

где:

Название_оси - одно из ниже перечисленного:

JOYX, JOYY	(совместно именуемые **JOYSTICK**)
THR	
RNG, ANT	(совместно именуемые **ROTARIES**)
MIX, MIY	(совместно именуемые **MICROSTICK**)
LBRK, RBRK	(совместно именуемые **TOEBRAKES**)
RDDR	

Блокируемое_значение:

либо величина от 0 до 100%, или LASTVALUE

Пример:

```
BTN S2 /I LOCK (THR, 100%)
       /O UNLOCK (THR)
```

Таким образом, команда LOCK используется для того, чтобы система видела некое постоянное аналоговое значение оси независимо от ее физического перемещения, либо в диапазоне от 0 до 100%, либо последнее значение на момент применения команды (с использованием параметра LASTVALUE). Восстановить аналоговую работу оси можно командой UNLOCK.

Зачем это может понадобиться? Поясним на примерах:

1. Допустим, нам нужно, чтобы кольцо масштаба виделось исключительно как аналоговая ось, за исключением того, когда кнопка S3 нажата, и в этом случае нам нужно, чтобы кольцо работало в цифровом режиме, а система видела бы его последнее аналоговое значение.

```
RNG /I LOCK (RNG, LASTVALUE) 2 5 a b c d e
    /O UNLOCK (RNG)
```

Теперь, когда S3 отжата, кольцо масштаба видно системе как стандартная аналоговая ось, функция которой назначается средствами игры. При нажатии на кнопку S3 фиксируется последнее аналоговое значение RNG, а при его вращении генерируются символы "a b c d e", запрограммированные

по ключу /I с помощью выражения цифрового режима 2 типа. Теперь посмотрим, насколько мощный этот инструмент:

2.
```
ANT  /U /I LOCK (ANT, LASTVALUE) 2 10 1 2 3 4 5 6 7 8 9 0
     /O UNLOCK (ANT) Rem Ось назначена в игре
     /M UNLOCK (ANT) Rem Ось назначена в игре
     /D LOCK (ANT, 0%) 3 Lower_flaps ^ Raise_Flaps Rem Выпустить
     закрылки - убрать закрылки
```

Когда переключатель "дальний - ближний бой" находится в положении /D, кольцо антенны работает в цифровом режиме 3 типа, управляя закрылками, а игра в этот момент получает аналоговое значение оси, равное нулю. Когда переключатель переводится в положение /M, кольцо работает в аналоговом режиме, в соответствии с назначенной в игре функцией. Если переключатель в положении /U, и кнопка S3 отжата, ось по-прежнему работает в аналоговом режиме. Когда S3 нажата, фиксируется последнее аналоговое значение кольца антенны, а при его вращении, через цифровое выражение 2 типа генерируются символы от 1 до 0.

Использование команд LOCK и UNLOCK в сочетании с цифровыми режимами осей является очень мощным инструментом программирования осей. Однако будьте внимательны - существует опасность натворить бед!

Примечания

1. *Нельзя отключить оси джойстика или микроджойстика при помощи конфигурационного выражения* DISABLE. *Оси X и Y джойстика обязательно должны присутствовать - это требование DirectX, а оси микроджойстика и так отключены как аналоговые оси по умолчанию.*

2. *Когда вы отключаете оси при помощи конфигурационных выражений в файле настроек, манипулятор должен обменяться новыми данными о конфигурации осей с операционной системой. Это занимает определенное время, поэтому такой файл будет загружаться несколько дольше.*

3. *Если команды* LOCK *и* UNLOCK *используются в выражениях цифровых режимов осей, им обязательно должны предшествовать ключи* /U, /M, /D, /I, *или* /O. *Таким образом, следующее выражение приведет к ошибке компиляции.*

```
ANT LOCK (RNG, LASTVALUE)
```

но: `BTN S2 LOCK (RNG, LASTVALUE)` *верно.*

6.6 Мэппинг осей (SWAP)

Мэппинг осей позволяет менять назначение физических осей как до начала игры, так и "налету" с использованием команды SWAP.

Конфигурационное выражение

USE SWAP (Название_оси, Название_оси)

Синтаксис команды

SWAP (Название_оси, Название_оси)

где:

Название_оси - одно из ниже перечисленного:

JOYX, JOYY	(совместно именуемые **JOYSTICK**)
THR	
RNG, ANT	(совместно именуемые **ROTARIES**)
MIX, MIY	(совместно именуемые **MICROSTICK**)
LBRK, RBRK	(совместно именуемые **TOEBRAKES**)
RDDR	

Пример:

```
USE SWAP (ANT, RNG)
```

в этом случае оси колец антенны и масштаба меняются местами. Еще примеры:

```
BTN S1 SWAP(JOYY, THR)          REM меняются местами ось Y и Throttle
BTN S2 SWAP(JOYSTICK, MICROSTICK)   REM меняются местами оси
                                    джойстика и оси
                                    микроджойстика
```

Последний пример преобразуется компилятором в:

```
BTN S2 SWAP(JOYX, MIX) SWAP(JOYY, MIY)
```

таким образом, очень важно при использовании групповых наименований использовать группы, включающие одинаковое количество осей:

```
BTN S2 SWAP(JOYSTICK, MICROSTICK)  нормально, но

BTN S2 SWAP(JOYSTICK, THR)
```

приведет к ошибке компиляции, так как наименование JOYSTICK включает в себя 2 оси (JOYX и JOYY), тогда как THR - одна ось.

Примечания

Перемена местами осей влечет за собой переход кривых отклика, однако выражения цифровых режимов местами не меняются.

6.7 Инверсия направления оси (REVERSE, FORWARD)

Можно изменить направление оси при помощи команды REVERSE:

Конфигурационное выражение

USE REVERSE (Название_оси)

Синтаксис команды

REVERSE (Название_оси)
FORWARD (Название_оси)

где:

Название_оси - одно из ниже перечисленного:

JOYX, JOYY	(совместно именуемые **JOYSTICK**)
THR	
RNG, ANT	(совместно именуемые **ROTARIES**)
MIX, MIY	(совместно именуемые **MICROSTICK**)
LBRK, RBRK	(совместно именуемые **TOEBRAKES**)
RDDR	

Эта функция может оказаться, к примеру, весьма полезна в вертолетном симуляторе, если вы хотите пользоваться РУД в обратном направлении, по сравнению с самолетными симуляторами, или, допустим, вы хотите использовать педали не как в настоящем самолете, а наоборот.

Если нужно поменять направление оси по умолчанию для всего файла настроек, можно воспользоваться конфигурационным выражением USE REVERSE:

USE REVERSE (RDDR)

Напомним, что такое выражение должно находиться в начале конфигурационного файла, на отдельной строке. Другой пример:

```
BTN H1U   /I REVERSE (JOYY)
          /O FORWARD (JOYY)
```

Нажатие хэта 1 вверх при нажатой кнопке S3 приведет к инверсии оси Y джойстика. Нажатие этого же хэта вверх при отжатой кнопке S3 приведет ось Y в нормальное состояние. Помните, что при перемене осей местами функция инверсии также переходит, однако цифровые режимы работы осей не зависят от инверсии.

6.8 Конфигурационное выражение USE AXES_CONFIG

Мы обсудили в деталях, как отключать, менять местами и инвертировать отдельные оси. Существует возможность объединить все это в конфигурационное выражение USE AXES_CONFIG, которое будет конвертировано компилятором в ряд соответствующих конфигурационных выражений, рассмотренных выше. Это сложное выражение, которое, скорее всего, не будет часто использоваться, однако может оказаться полезным в некоторых случаях.

Рассмотрим синтаксис

Конфигурационное выражение:

USE AXES_CONFIG **(***DX-axis1*, HOTASaxis1), (*DX-axis2*, HOTASaxis2) и т.д.

напр. ось DX-axis1 назначена на ось HOTASaxis1

Итак, что такое оси DX и оси HOTAS? Начнем с последнего: оси HOTAS - это любые из 10 аналоговых осей Кугуара (то есть JOYX, JOYY, THR, RDDR, ANT, RNG, MIX, MIY, LBRK, RBRK). Оси DX - это стандартные аналоговые оси, воспринимаемые операционной системой и видимые в игре.

Кугуар показывает наличие оси DirectX, которой потом игра присваивает функцию. Кугуар имеет 10 осей, однако DirectX 8 поддерживает только 8 осей для USB-устройств (для справки: DirectX 7 -только 6). Для упрощения мы говорим Windows, “*Использовать ось X джойстика как DX-axis1, ось Y джойстика как DX-axis 2*” и т. д. При этом, собственно, не важно, что стоит на вашем столе - джойстик с РУД, штурвал, или даже автомобильный руль: DX работает с видимыми осями.

Ниже в таблице приведены названия осей DX, соответствующие им оси Кугуара и их программные обозначения:

Оси DirectX	Соотв. оси HOTAS	Синтаксис
X	Ось X джойстика	JOYX
Y	Ось Y джойстика	JOYY
Z	РУД	THR
Rotation X	Кольцо антенны	ANT
Rotation Y	Левый педальный тормоз	LBRK
Rotation Z	Педали	RDDR
Slider 0	Кольцо масштаба	RNG
Slider 1	Правый педальный тормоз	RBRK

Примечание: Возможно, с выпуском педалей Cougar назначения педальных тормозов будут поменяны местами

Возвращаясь к использованию данного выражения, предлагаем вам запомнить четыре основных правила:

1. Любая ось (ось DirectX или HOTAS Cougar), которая не будет видна операционной системе, будет отключена в своем аналоговом качестве.
2. Оси, заключенные в круглые скобки, меняются друг с другом.
3. Любая ось HOTAS Cougar со знаком '-' перед ее названием будет инвертирована.
4. Оси 1 и 2 обязательно должны присутствовать - таково требование Windows.

Пример:

```
USE AXES_CONFIG (1, RNG), (2, ANT), (3, -THR)
```

В данном примере:

1. Ось кольца масштаба назначена как ось DirectX 1, и будет видна в системе как ось X джойстика.
2. Ось кольца антенны назначена как ось DirectX 2, , и будет видна в системе как ось Y джойстика.
3. Ось РУД назначена как ось DirectX 3, что привычно, однако знак "-" перед ее названием говорит о ее инверсии - полезно для вертолетных симуляторов.
4. Все остальные оси не заявлены, и поэтому не видны как аналоговые оси в системе и в игре. Естественно, их можно по-прежнему программировать в цифровых режимах средствами Cougar.

Таким образом, с помощью одного конфигурационного выражения можно менять местами, отключать и инвертировать любые оси.

Примечания

1. *Если вы используете конфигурационное выражение* USE AXES_CONFIG*, то вы не можете одновременно с ним использовать конфигурационные выражения* DISABLE, USE REVERSE *или* USE SWAP*. Это приведет к ошибке компиляции. Однако в выражениях для кнопок вы по-прежнему можете использовать команды* SWAP, REVERSE*.*

2. *Намного проще использовать* USE PROFILE *и профиль, созданный в Панели управления (см. ранее в Руководстве)*.

3. *DirectX делает свой мэппинг осей, которые он видит, поэтому если вы используете профиль, созданный в Панели управления, результат может отличаться от того, что вы ожидали. Поэкспериментируйте, чтобы получить желаемый результат.*

7. Программирование мыши

7.1 Понятия "Устройство управления мышью" и "Микроджойстик

В последней части данной главы мы рассмотрим принципы работы устройства управления мышью и покажем некоторые интересные приемы. Однако прежде чем перейти непосредственно к синтаксису выражений, необходимо понять, как осуществляется управление мышью.

Если вы уже познакомились с HOTAS Cougar, наверняка вы использовали микро для управления мышью, и считали его контроллером мыши. Однако, важно понять один момент:

Микроджойстик – это НЕ мышь.

Микроджойстик – это, в точном соответствии с названием, маленький джойстик. Он, *как правило,* управляет мышью, потому что так по умолчанию задает компилятор, который по умолчанию назначает управление мышью на оси микроджойстика (MIX и MIY), о чем вы можете и не подозревать.

Следующий важным момент, который необходимо понять: мышь, как устройство, **не имеет** осей как таковых. Что это значит? Рассмотрим джойстик: когда вы его отклоняете, в каждый момент времени он посылает определенные значения осей X и Y. И если бы джойстик мог управлять курсором мыши так же, как он управляет перекрестием в окне анализатора

джойстика утилиты Foxy, **перемещение** джойстика точно соответствовало **перемещению** мыши. Соответственно, если бы движение джойстика остановилось (в любом положении), то прекратилось бы и движение курсора мыши. Но это происходит не так. Микроджойстик, запрограммированный на управление мыши, работает по-другому. Если вы его отклоните от центрального положения и **остановите,** курсор мыши будет продолжать двигаться. Говоря по-простому, микроджойстик не говорит системе: "*Переместить курсор мыши в точку с координатами X, Y,*". Вместо этого он говорит: "*Перемещать курсор мыши со скоростью "a" по оси X, и со скоростью "в" по оси Y, до получения иных инструкций.*" Затем, если вы отпустите микроджойстик, и он вернется в центральное положение, курсор мыши не возвратится в центр экрана – он замрет на том месте, где он был, потому что микроджойстик посылает команду: "Скорость перемещения по оси X = 0, по оси Y = 0", а не "Вернуть курсор мыши в центр"

Таким образом, курсор мыши движется в соответствии со значениями (отличными от нуля) параметров MouseX и (или) MouseY (MSX и MSY), которые определяют скорость перемещения курсора по соответствующим осям координат. Микроджойстик просто изменяет эти значения, таким образом управляя мышью.

Наверное, вы уже начинаете догадываться, зачем это было сделано именно таким образом. А дело в том, что теперь мышь может управляться чем угодно, что будет изменять значения параметров MSX и MSY. Это могут быть микроджойстик, РУС, хэт, кнопка, логические флаги. И что самое важное, они могут это делать *одновременно*. То есть вы можете, к примеру, использовать микроджойстик для быстрого перемещения мыши, а хэт для точного.

7.2 USE MTYPE – самый простой способ запрограммировать микроджойстик на управление мышью

Далее в данной главе вы узнаете в самых мельчайших подробностях, как повесить мышь на микроджойстик. Однако это потребует глубокого понимания цифровых режимов работы осей и использования параметров MSX и MSY – занятие не для слабонервных :). К счастью, есть пара выражений, которые позволяют назначить микроджойстик для управления мышью легко и непринужденно: USE MTYPE и USE MICROSTICK AS MOUSE. Начнем с первого.

Конфигурационное выражение:

USE MTYPE Тип - REVERSE_*тип*

где:

Тип: от A1 до A5, описывает, какие кнопки на РУД будут назначены правой и левой кнопками мыши:

Тип	Левая кнопка	Правая кнопка
A1	T1	T6
A2	T6	T1
A3	T1	нет
A4	T6	нет
A5	нет	нет

T1 – это кнопка, приводимая в действие нажатием на головку микроджойстика, T6 – нажатие на кольцо масштаба.

REVERSE_*тип* это REVERSE_UD и (или) REVERSE_LR

Команда REVERSE_UD инвертирует направление вертикальной оси мыши (Y), а команда REVERSE_LR – горизонтальной (X).

Пример:

USE MTYPE A3

Управление мышью назначено на микроджойстик, кнопка T1 работает как левая кнопка мыши.

USE MTYPE A5 - REVERSE_UD

Управление мышью назначено на микроджойстик, ось Y инвертирована, кнопки мыши не назначены.

Примечания

1. *Чувствительность (скорость) мыши устанавливается компилятором на определенное значение, и должно быть приемлемым для экранных разрешений до 1024x768. При использовании выражения* USE MTYPE *менять чувствительность мыши нельзя, в этом случае необходимо пользоваться выражением* USE MICROSTICK AS MOUSE*, которое будет рассмотрено в следующем разделе.*

2. *Как уже говорилось, микроджойстик является аналоговым устройством с 4 осями, а не системой из 4 кнопок, как это было в предыдущей модели HOTAS. Поэтому кнопки T11 - T14 более не*

существуют. Однако если необходимо – их можно эмулировать при помощи выражений Type *– см. Справку Foxy, раздел "Конвертирование командных строк TQS T11 - T14 для использования с микроджойстиком HOTAS Cougar".*

3. *Если вы настраивали кривые отклика осей микроджойстика, то они никак не влияют на управление мышью, так как это – цифровой режим работы оси, а кривые отклика в цифровых режимах не учитываются.*

4. *Если вы используете в файле настроек выражение* USE MTYPE*, которое назначает кнопки мыши на T1 или T6, то эти кнопки РУДа становятся недоступны для программирования. То есть если есть выражение:*

USE MTYPE A3

которое компилятор автоматически превращает, в том числе, и в выражение:

BTN T1 /H MOUSE_LB

то если где-то ниже в файле будет:

BTN T1 *макрокоманда* или BTN T1 /H MOUSE_RB

произойдет ошибка компиляции. Если вы хотите программировать эти позиции, то используйте соответствующий тип выражения USE MTYPE*, оставляющий их свободными.*

7.3 Выражение USE MICROSTICK AS MOUSE

Вторым способом назначить микроджойстик для управления мышью является использование выражения USE MICROSTICK AS MOUSE. Преимущество по сравнению с USE MTYPE заключается в возможности вносить изменения в чувствительность мыши, иными словами, вы можете менять скорость перемещения курсора мыши по экрану. Это выражение по умолчанию назначает левую кнопку мыши на T1, но эту функцию можно отключить, добавив модификатор - NO_BUTTON. Для большинства авиасимуляторов, однако, настройка кнопки T1 как левой кнопки мыши удобна. Выражение USE MICROSTICK AS MOUSE говорит компилятору настроить управление мышью на оси микроджойстика в форме выражения цифрового режима 6 типа, или, если указано начальное значение, в виде выражения 5 типа.

Рассмотрим синтаксис:

Конфигурационное выражение:

Для выражений 6 типа: *(без начального значения)*

USE MICROSTICK AS MOUSE (значение шкалы, значение шага) - *Модификатор*

Для выражений 5 типа: *(с указанием начального значения)*
USE MICROSTICK AS MOUSE (значение шкалы, значение шага, начальное значение) - *Модификатор*

*(**Примечание**: выражение USE MICROSTICK AS MOUSE ( ) также автоматически назначает левую кнопку мыши на T1, если не используется модификатор - NO_BUTTON)*

где:

Значение шкалы: Число от 2 до 12, описывает, на сколько зон делятся оси микроджойстика.

Значение шага: Значение шага для каждой зоны, число от 1 до 63. *(Внимание! Произведение значения шкалы и значения шага должно быть меньше 128. Даже не спрашивайте, почему!)*

Начальное значение: Начальное значение для выражений 5 типа, от которого начинается прибавление значения шага. Это первоначальная скорость перемещения курсора мыши при отклонении микроджойстика от центрального положения.

Модификатор:

REVERSE_UD: Инверсия оси Y.
REVERSE_LR: Инверсия оси X.
NO_BUTTON: Указывает компилятору не программировать T1 как левую кнопку мыши, позволяя программироваться ее через выражение BTN T1.

Несколько примеров.

USE MICROSTICK AS MOUSE (12, 2)

Данное выражение программирует микроджойстик для управления мышью, назначая левую кнопку мыши на T1. Еще два примера:

USE MICROSTICK AS MOUSE (6, 4)
USE MICROSTICK AS MOUSE (7, 3, 2)

И в этих двух случаях микроджойстик программируется на управление мышью, а кнопка T1 является левой кнопкой мыши. Эти три разных выражения делают одно и то же – программируют микроджойстик на управление мышью, вешая левую кнопку на T1, однако при этом будет отличаться отклик мыши. Ниже приводится объяснение чисел в скобках.

Детальное описание всех этих параметров будет дано в следующем разделе, и оно довольно сложное для восприятия. Для начала – более простое объяснение.

Рассмотрим отдельно ось X микроджойстика. Выражение USE MICROSTICK AS MOUSE разбивает ось на несколько зон, как показано на диаграмме внизу. (*Для удобства объяснения эти зоны в приведенном примере одинакового размера, однако на самом деле это не совсем так в реальности (это - для самых въедливых:))*

Итак, зона “C” представляет из себя центральное положение микроджойстика (он отпущен). Здесь, естественно, нам нужно чтобы мышь не двигалась. В обе стороны от центра – две зоны с номером 1. Когда микроджойстик перемещается в эти зоны, мышь должна начать двигаться. И есть еще по две зоны (2 и 3), в которые микроджойстик попадает по мере его отклонения от центра.

Вернемся к синтаксису:

USE MICROSTICK AS MOUSE (Значение шкалы, Значение шага, *дополнительно* Начальное значение)

Значение шкалы определяет, на сколько зон делится ось микроджойстика. Число, определяющее значение шкалы, на самом деле не равно точному числу зон, на которое будет разбита ось. Оно равно числу зон (КРОМЕ центральной) с одной стороны от центра. То есть, например, если значение шкалы равно 3, то количество зон будет равно 7 (по три с каждой стороны от центра плюс центральная зона).

Значение шага определяет, на какую величину увеличится скорость перемещения курсора мыши при перемещении микроджойстика из зоны в зону. Поясним на примере что происходит при перемещении микроджойстика от центрального положения в крайнее.

USE MICROSTICK AS MOUSE (4, 2)

Зона C: Микроджойстик находится в центральном положении, мышь не двигается (скорость – 0).
Зона 1: Мышь начинается двигаться со скоростью, равной сумме предыдущего значения и значения шага. То есть в нашем случае - со "*скоростью 2*".
Зона 2: Скорость мыши увеличивается еще на одно значение шага, то есть мышь движется со "*скоростью 4*".
Зона 3: Скорость мыши увеличивается еще на одно значение шага, то есть мышь движется со "*скоростью 6*".
Зона 4, 5, 6 и т. д.

По мере перемещения микроджойстика от центра по зонам скорость движения курсора увеличивается. Соответственно, по мере перемещения по зонам к центральному положению скорость движения курсора уменьшается.

Теперь рассмотрим использование Начального значения.

```
USE MICROSTICK AS MOUSE (4, 2, 1)
```

Начальное значение явным образом определяет скорость, с которой курсор мыши движется при отклонении джойстика от центрального положения в зону 1. По мере перемещения дальше от центра выражение работает точно также, как и предыдущее: при переходе в каждую последующую зону скорость перемещения курсора увеличивается на значение шага. То есть:

Зона C: Микроджойстик в центре, курсор неподвижен.
Зона 1: Мышь начинает двигаться со скоростью, определяемой Начальным значением, или со "*Скоростью 1*".
Зона 2: Скорость мыши увеличивается на значение шага, и теперь равна *3 (1 + 2, Начальное значение + Значение шага)*.
Зона 3: Скорость мыши увеличивается на значение шага, и теперь равна *5*.
Зона 4, 5, 6 и т. д.

Запомним основное правило: чем больше Начальное значение и Значение шага, тем быстрее будет двигаться курсор мыши (быстрее, но не дальше).

Также можно инвертировать оси микроджойстика:

```
USE MICROSTICK AS MOUSE (7, 3, 2) - REVERSE_UD
USE MICROSTICK AS MOUSE (7, 3, 2) - REVERSE_LR
```

В конце данного раздела рассмотрим еще один вариант использования этих выражений, а именно: программирование других осей для управления мышью.

7.3.1 Назначение других осей в качестве осей мыши

Выражение USE MICROSTICK AS MOUSE является, на самом деле, особым случаем другого выражения (особенность в том, что левая кнопка мыши по умолчанию назначается на T1):

Конфигурационное выражение:

Выражения 6 типа:

USE *Название_оси* AS *Ось_мыши* (Значение шкалы, Значение шага) - REVERSE_*тип*

Выражения 5 типа:

USE *Название_оси* AS *Ось_мыши* (Значение шкалы, Значение шага, Начальное значение) - REVERSE_*тип*

где:

Название_оси - одно из ниже перечисленного:

JOYX, JOYY	(совместно именуемые **JOYSTICK**)
THR	
RNG, ANT	(совместно именуемые **ROTARIES**)
MIX, MIY	(совместно именуемые **MICROSTICK**)
LBRK, RBRK	(совместно именуемые **TOEBRAKES**)
RDDR	

Ось_мыши – одно из ниже перечисленного:

MOUSE
MOUSEX
MOUSEY
MOUSEZ (колесо прокрутки)

Значение шкалы: Число от 2 до 12, описывает, на сколько зон делятся выбранные оси.

Значение шага: Значение шага для каждой зоны, число от 1 до 63. *(Внимание! Произведение значения шкалы и значения шага должно быть меньше 128. Даже не спрашивайте, почему!)*

Начальное значение: Начальное значение для выражений 5 типа, от которого начинается прибавление значения шага. Это первоначальная скорость перемещения

курсора мыши при отклонении выбранной оси от центрального положения.

REVERSE_*тип*: инверсия осей, одно из ниже перечисленного:

REVERSE_UD: используется, если *Название_оси* представляет собой 2 оси (JOYSTICK, ROTARIES, MICROSTICK, TOEBRAKES).

REVERSE_LR: используется, если *Название_оси* представляет собой 2 оси (JOYSTICK, ROTARIES, MICROSTICK, TOEBRAKES).

REVERSE_DIR: инверсия одной оси. Не может использоваться вместе с REVERSE_UD, or REVERSE_LR.

Таким образом, можно точно также использовать любые другие оси для управления мышью. Несколько примеров:

```
USE ROTARIES AS MOUSE (6, 4)
USE JOYSTICK AS MOUSE (11, 2, 0)
USE ANT AS MOUSEY (5, 2) - REVERSE_DIR
USE JOYY AS MOUSEZ (9, 3)
```

В последнем примере ось Y джойстика управляет колесом прокрутки мыши!

Таким образом, мы рассмотрели конфигурационные выражения, с помощью которых можно назначить микроджойстик или другие оси на управление мышью. Однако, если вы помните, управлять мышью можно и с помощью хэта, например, через выражение:

```
USE HAT1 AS MOUSE (2)
```

На самом деле можно одновременно запрограммировать и микроджойстик, и хэт для управления мышью: с помощью микроджойстика курсор быстро перемещается в нужную область, а хэт используется для точного позиционирования! *[В этом месте идут аплодисменты :)]*

Теперь попробуем объяснить, что же именно происходит, или как именно компилятор интерпретирует такие выражения, формируя выражения цифровых режимов для осей микроджойстика. Тема эта непроста для восприятия, поэтому следующий раздел предназначается исключительно для самых продвинутых пользователей. Но если в ней разобраться, вы сможете создавать пользовательские настройки управления мышью для любых осей, чтобы добиться самых изощренных результатов. Более того, вы сможете комбинировать выражения управления мышью с любыми другими выражениями цифровых режимов на микроджойстике, чтобы,

например, при нажатой S3 микроджойстик управлял курсором целеуказателя, а при отпущенной – просто мышью. Неплохо!

7.4 Создание пользовательских выражений управления мышью для микроджойстика

Мы уже видели, что микроджойстик может быть настроен на управление мышью при помощи выражений USE MTYPE и USE MICROSTICK AS MOUSE. Эти выражения просто программируют оси микроджойстика при помощи выражений цифровых режимов. В данном разделе мы рассмотрим, как именно это происходит и как можно создавать собственные цифровые выражения для микроджойстика или любых других осей.

Начнем с выражений 1 и 2 типов, хотя можно использовать выражения любого из 6 типов. Итак, выражение 1 типа:

```
MIX 1 14 MSX (2+) MSX (2-) MSX (0)
MIY 1 14 MSY (2-) MSY (2+) MSY (0)
```

Очень похоже на обыкновенное выражение 1 типа, с той лишь разницей, что появились знаки "+" и "-". Чтобы понять предназначение этих знаков, рассмотрим, что именно происходит при отклонении микроджойстика. Напомним, что такое выражение 1 типа. Если есть выражение:

```
MIX 1 6 l r c
```

то при перемещении микроджойстика по оси X из крайнего левого положения в крайнее правое, и обратно, будут сгенерированы следующие символы:

```
r r r c r r r l l l c l l l
```

В нашем примере выражение выглядит так:

```
MIX 1 14 MSX (2+) MSX (2-) MSX (0)
```

Теперь мы не генерируем последовательность символов – число в скобках означает, сколько нужно прибавить (или вычесть) к текущему значению буфера мыши, или, иными словами, насколько быстрее (или медленнее) должен двигаться курсор.

Предположим, что курсор мыши неподвижен, а микроджойстик в центральном положении. Центральному положению оси микроджойстика соответствует "символ" центрального положения – в нашем примере, MSX (0). Этот символ означает нулевую скорость, то есть курсор не движется по оси X. При перемещении микроджойстика вправо курсор начинает

двигаться вправо со *"скоростью 2"*, так как к буферу добавляется 2. Перемещаем микроджойстик дальше вправо – мышь движется со скоростью 4…перемещаем до конца вправо – мышь движется со скоростью 14 (7 x 2). Отпускаем микроджойстик – скорость мыши постепенно падает, по мере вычитания значений из буфера, пока не достигнет 0 (а микроджойстик – центрального положения). Курсор остановился.

То же самое происходит и по оси Y.

Если нужно, чтобы микроджойстик управлял мышью так же, как и джойстик (то есть на себя – курсор вверх), выражение должно выглядеть следующим образом:

MIY 1 14 MSY (2+) MSY (2-) MSY (0)

Дело в том, что если значение буфера мыши для оси Y увеличивается, мышь движется вниз, и наоборот. Очень важно запомнить правила синтаксиса: знаки "+" и "-" должны быть *внутри* скобок, и должны стоять **после значения**. Они указывают на то, сколько нужно прибавить (или вычесть) к буферу. Вы поймете это лучше когда мы перейдем к выражениям 2 типа. Надеюсь, вы все еще с нами... пока!

Итак, использование выражений 2 типа для назначения осей микроджойстика для управления мышью:

MIX 2 9 MSX(-8) MSX(-4) MSX(-2) MSX(-1) MSX(0) MSX(1) MSX(2) MSX(4) MSX(8)
MIY 2 9 MSY(8) MSY(4) MSY(2) MSY(1) MSY(0) MSY(-1) MSY(-2) MSY(-4) MSY(-8)

Это обычное выражение 2 типа. Оси делятся на 9 равных зон, и для каждой зоны указывается **абсолютное значение** (а не *относительное изменение*) буфера, или скорости курсора. При перемещении микроджойстика по оси X курсор сначала движется со скоростью 1, затем 2, 4 и, наконец, 8. Обратите внимание на положение знака "-". Теперь он стоит **перед значением**, что означает, что данная величина не отнимается от значения буфера, а становится его абсолютным значением.

С точки зрения синтаксиса отметим, что следующие выражения абсолютно одинаковы:

MIX 2 7 MSX(-4) MSX(-2) MSX(-1) MSX(0) MSX(1) MSX(2) MSX(4)
MIX 2 7 MSX(-4) MSX(-2) MSX(-1) MSX(0) MSX(+1) MSX(+2) MSX(+4)

то есть при отсутствии знака "+" **перед** значением он подставляется автоматически.

А вот выражения:

MIX 2 7 MSX(-4) MSX(-2) MSX(-1) MSX(0) MSX(1) MSX(2) MSX(4)
MIX 2 7 MSX(4-) MSX(2-) MSX(1-) MSX(0) MSX(1+) MSX(2+) MSX(4+)

будут означать *очень* разное поведение мыши.

Теперь, усвоив это, рассмотрим, что делает компилятор с выражением USE MICROSTICK AS MOUSE. Компилятор преобразует его в цифровые выражения 5 и 6 типов, которые, как мы помним (правда ведь?...), являются ни чем иным, как выражениями 2 и 1 типов, только с настраиваемыми зонами. Освежим в памяти синтаксис этих выражений:

USE MICROSTICK AS MOUSE (Значение шкалы, Значение шага, *дополнительно* Начальное значение)

Как уже было сказано, Значение шкалы определяет, на сколько зон делится ось. Количество зон определяется по следующей формуле:

Количество зон = (Значение шкалы x 2) - 1

Затем компилятор создает эти зоны разного размера, используя очень сложную формулу, приводить которую нет смысла, и формирует цифровое выражение 6 типа, если нет Начального значения, и выражение 5 типа, если Начальное значение указано.

Выражения 6 типа

USE MICROSTICK AS MOUSE (2, 2)

раскладывается на:

MIX 6 3 (2 24 75 98) MSX(2+) MSX(2-) ^
MIY 6 3 (2 24 75 98) MSY(2-) MSY(2+) ^
BTN T1 /H MOUSE_LB Rem Удерживать нажатой левую кнопку мыши, пока нажата T1

Обратите внимание на отличия: теперь выражения MSX(0) и MSY(0) не используются в качестве символа центрального положения, как здесь:

MIX 6 3 (2 24 75 98) MSX(2+) MSX(2-) MSX (0)
MIY 6 3 (2 24 75 98) MSY(2-) MSY(2+) MSY(0)

Использование "пустого символа" (^) вместо MSX (0), MSY (0) имеет свои преимущества и недостатки. Выражения управления мышью 6 типа отличаются от выражений 5 типа тем, что они уменьшают или увеличивают на определенную величину значение буфера по осям X и Y, тогда как выражения 5 типа устанавливают абсолютные значения. При

использовании MSX(0) и MSY(0) в качестве символов центрального положения буфер мыши обнуляется, и курсор гарантированно останавливается, когда соответствующая ось попадает в центральное положение. В общем и целом это хорошо. Однако прелесть использования "пустых символов" для центрального положения заключается в том, что в этом случае можно назначить еще и, к примеру, хэт для более тонкого управления мышью. Например:

```
USE MICROSTICK AS MOUSE (2, 2)
BTN H1L MSX(1-)
BTN H1R MSX(1+)
```

В этом случае микроджойстик используется для управления мышью вообще, а отклонение хэта 1 влево - вправо для более точного позиционирования курсора по горизонтали (оси X). Важно помнить, что выбор того или иного метода зависит от того, что именно вам нужно получить. К сожалению, нельзя заставить компилятор автоматически использовать MSX(0), MSY(0) в качестве команд центрального положения при преобразовании выражений USE MICROSTICK AS MOUSE. Для этого придется самостоятельно задавать выражения для MIX и MIY, или задавать Начальное значение, при наличии которого компилятор создает выражения 5 типа, в которых как раз используются MSX(0), MSY(0) для центрального положения. Рассмотрим еще несколько примеров: как компилятор раскладывает выражения USE MICROSTICK AS MOUSE на выражения 6 типа.

USE MICROSTICK AS MOUSE (3, 4)

преобразуется в:

```
MIX 6 5 (2 16 32 68 84 98) MSX(4+) MSX(4-) ^
MIY 6 5 (2 16 32 68 84 98) MSY(4-) MSY(4+) ^
BTN T1 /H MOUSE_LB
```

USE MICROSTICK AS MOUSE (4, 2)

преобразуется в:

```
MIX 6 7 (2 12 23 36 65 78 89 98) MSX(2+) MSX(2-) ^
MIY 6 7 (2 12 23 36 65 78 89 98) MSY(2-) MSY(2+) ^
BTN T1 /H MOUSE_LB
```

USE MICROSTICK AS MOUSE (12, 3)

преобразуется в:

```
MIX 6 23 (2 4 6 8 11 14 17 21 25 30 36 43 58 65 71 76 80 84 87 90 93 95 97 98) MSX(3+) MSX(3-) ^
MIY 6 23 (2 4 6 8 11 14 17 21 25 30 36 43 58 65 71 76 80 84 87 90 93 95 97 98) MSY(3-) MSY(3+) ^
BTN T1 /H MOUSE_LB
```

USE MICROSTICK AS MOUSE (4, 2) - REVERSE_UD

преобразуется в:

```
MIX 6 7 (2 12 23 36 65 78 89 98) MSX(2+) MSX(2-) ^
MIY 6 7 (2 12 23 36 65 78 89 98) MSY(2+) MSY(2-) ^
BTN T1 /H MOUSE_LB
```

Теперь рассмотрим, что происходит с выражениями USE MICROSTICK AS MOUSE, если указывается Начальное значение. Компилятор преобразует их в цифровые выражения 5 типа.

Цифровые выражения 5 типа

USE MICROSTICK AS MOUSE (2, 2, 3)

преобразуется в:

```
MIX 5 3 (0 24 75 100) MSX(-3) MSX(0) MSX(3)
MIY 5 3 (0 24 75 100) MSY(3) MSY(0) MSY(-3)
BTN T1 /H MOUSE_LB
```

Отметьте, что количество создаваемых зон равно 3. Центральная зона обеспечивает неподвижность курсора, а две зоны по обеим сторонам от нее задают скорость перемещения курсора, равную Начальному значению. Таким образом, в данном конкретном примере Значение шага не играет никакой роли – просто нет зон, в которых бы оно начало работать. Теперь сравните это выражение с примером ниже:

USE MICROSTICK AS MOUSE (3, 2, 3)

преобразуется в:

```
MIX 5 5 (0 14 31 69 86 100) MSX(-5) MSX(-3) MSX(0) MSX(3) MSX(5)
MIY 5 5 (0 14 31 69 86 100) MSY(5) MSY(3) MSY(0) MSY(-3) MSY(-5)
BTN T1 /H MOUSE_LB
```

Теперь, как вы видите, значение шага учитывается.

USE MICROSTICK AS MOUSE (3, 8, 1)

преобразуется в:

MIX 5 5 (0 14 31 69 86 100) MSX(-9) MSX(-1) MSX(0) MSX(1) MSX(9)
MIY 5 5 (0 14 31 69 86 100) MSY(9) MSY(1) MSY(0) MSY(-1) MSY(-9)
BTN T1 /H MOUSE_LB

USE MICROSTICK AS MOUSE (12, 2, 3)

преобразуется в:

MIX 5 23 (0 2 4 6 9 12 16 20 25 30 36 43 59 66 72 77 82 86 90 93 96 98 99 100)
MSX(-23) MSX(-21) … MSX(-3) MSX(0) MSX(3) … MSX(21) MSX(23)
MIY 5 23 (0 2 4 6 9 12 16 20 25 30 36 43 59 66 72 77 82 86 90 93 96 98 99 100)
MSY(23) MSY(21) … MSY(3) MSY(0) MSY(-3) … MSY(-21) MSY(-23)
BTN T1 /H MOUSE_LB

USE MICROSTICK AS MOUSE (3, 8, 1) - REVERSE_UD, REVERSE_LR

преобразуется в:

MIX 5 5 (0 14 31 69 86 100) MSX(9) MSX(1) MSX(0) MSX(-1) MSX(-9)
MIY 5 5 (0 14 31 69 86 100) MSY(-9) MSY(-1) MSY(0) MSY(1) MSY(9)
BTN T1 /H MOUSE_LB

И для полноты примеров – назначение других осей для управления осями мыши:

USE JOYSTICK AS MOUSE (3, 2, 3)

преобразуется в:

JOYX 5 5 (0 14 31 69 86 100) MSX(-5) MSX(-3) MSX(0) MSX(3) MSX(5)
JOYY 5 5 (0 14 31 69 86 100) MSY(5) MSY(3) MSY(0) MSY(-3) MSY(-5)

USE JOYY AS MOUSEZ (4,2)

преобразуется в:

JOYY 6 7 (2 12 23 36 65 78 89 98) MSY(2-) MSY(2+) ^

7.5 Выражение USE ZERO_MOUSE

Это выражение очень полезно, если вы вручную задаете настройки управления мышью с помощью микроджойстика, и при этом привязываете их использование к положению кнопки S3 через ключи /I и /O. С помощью этого выражения можно предотвратить зависание мыши.

В предыдущем разделе мы рассмотрели использование цифровых выражений для создания пользовательских настроек управления мышью при помощи микроджойстика. Выражение USE ZERO_MOUSE используется для предотвращения "зависания" мыши при использовании выражений управления мышью в сочетании с ключами /I и /O. Рассмотрим пример:

```
MIX    /I 1 6 MSX(2+) MSX(2-)
       /O 1 6 RARROW LARROW
MIY    /I 1 6 MSY(2-) MSY(2+)
       /O 1 6 UARROW DARROW
```

В данном примере, если кнопка S3 нажата, микроджойстик перемещает курсор мыши. Если вы отпустите кнопку S3 в то время, как курсор движется, то курсор будет продолжать двигаться, пока не достигнет края экрана, где и "застрянет". Это явление можно предотвратить, использовав выражение USE ZERO_MOUSE, которое заставит курсор остановиться, как только будет кнопка S3 сменит состояние.

Стоит отметить, что подобное "застревание" мыши не является багом. Курсор ведет себя именно так, как был запрограммирован микроджойстик. "Зависания" всегда можно избежать, если возвратить микроджойстик в центральное положение ДО того, как отпустить кнопку S3. Вообще, золотое правило, которого надо придерживаться во избежание всевозможных "запавших клавиш" и "застрявших" мышей, звучит так: **всегда используйте кнопки, хэты и оси именно так, как это было задумано и реализовано в программировании.** Ну, а выражение USE ZERO_MOUSE – для тех из нас, кто никогда этому правилу не следует :)

7.6 Программирование кнопок мыши

Нажатие кнопок мыши может быть запрограммировано на кнопки, оси и в прочих выражениях.

Синтаксис для кнопок мыши:

MOUSE_LB
MOUSE_RB
MOUSE_MB

(MB = *средняя кнопка для трехкнопочных мышей*)

Пример:

```
BTN T1 /I /H MOUSE_RB
       /O /H MOUSE_LB
```

В данном выражении кнопка T1 (головка микроджойстика) управляет левой кнопкой мыши при отпущенной S3, и правой – при нажатой.

С кнопками мыши можно также использовать команды KD и KU (нажатие и отпускание):

```
BTN S1 KD(MOUSE_LB) DLY(2000) KU (MOUSE_LB)
```

При нажатии кнопки S1 левая кнопка мыши нажимается, удерживается в течение 2 секунд и затем отпускается.

7.7 Отключение назначения микроджойстика для управления мышью по умолчанию

В окне Preferences (настройки) утилиты Foxy есть закладка "Defaults" (параметры по умолчанию). Эти параметры определяют, что именно автоматически делает компилятор в случае, если в файле настроек нет противоречащих выражений.

Один из таких параметров - "Назначить микроджойстик для управления мышью". Если эта опция отмечена, то при загрузке файла настроек в контроллер компилятор автоматически назначает микроджойстик для управления мышью, а кнопку T1 программирует на нажатие левой кнопки мыши. Это сделано для удобства, так как часто пользователи предпочитают, чтобы микроджойстик управлял мышью, но не хотят разбираться досконально с соответствующими выражениями.

Если вы хотите иметь возможность программировать оси микроджойстика иначе, необязательно отключать эту опцию в настройках Foxy. Можно использовать конфигурационное выражение в файле настроек:

Конфигурационное выражение:

```
DISABLE MOUSE
```

В таком случае, *даже если эта опция включена в настройках Foxy,* компилятор для данного файла все равно не будет автоматически назначать микроджойстик для управления мышью.

При этом мышь можно назначить на хэт, или другие оси. Помните, что интерфейс мыши работает все время – нужно просто назначить что-нибудь для управления.

7.8 Продвинутые выражения перемещения курсора

Мы уже видели, как при помощи соответствующих выражений осуществлять перемещением курсора. В последнем разделе данной главы мы изучим, как запрограммировать более сложные движения.

7.8.1 Определение экранного разрешения

Для всех выражений, которые будут рассмотрены ниже, **необходимо** определить экранное разрешение через конфигурационное выражение:

Конфигурационное выражение:

```
USE SCREEN_RESOLUTION (X, Y)
```

Пример:

```
USE SCREEN_RESOLUTION (800,600)
```

Минимальное значение для X и Y – 640 и 480 соответственно.

Теперь можно использовать следующие выражения программирования мыши:

Синтаксис команды:

Переместить курсор в заданную точку экрана
MOUSEXY (*Исх*, X, Y)

Переместить курсор относительно текущего положения
MOUSEMOVE (X, Y)

Перемещение по кругу/по многоугольнику

MOUSEROTATE (*Исх*, Центр, Радиус, Начальный угол, [Макрос 1], Направление, Конечный угол, кол-во вершин, [Макрос 2])

7.8.2 Перемещение в заданную точку экрана

Синтаксис команды:

MOUSEXY (*Исх*, X, Y)
MOUSEXY (*Исх*, X%, Y%)

где:

- *Исх* – исходное положение, один из углов экрана: UL, DL, UR, DR.
- X,Y - координаты X и Y точки на экране, куда нужно переместить курсор.
- X%, Y% - координаты, выраженные в процентах от ширины и (или) высоты экрана, 0 до 100% (с точностью до 3 знака после запятой). Использование процентного выражения позволяет менять разрешение экрана, не меняя параметров выражения.

Во время игры невозможно отследить в любой момент времени где находится курсор мыши, поэтому для того, чтобы переместить его в заданную точку на экране необходимо сначала в один из углов экрана, так как координаты этой точки будут нам всегда известны исходя из параметров выражения SCREEN_RESOLUTION. После этого компилятор сможет обсчитать относительное перемещение курсора из этой точки. Это происходит очень быстро, поэтому никаких проблем быть не должно, однако надо убедиться, что на этапе позиционирования никакая из кнопок мыши не будет нажата.

Итак, есть выражение:

BTN H3D MOUSEXY (UL, 400, 300)

Сначала курсор очень быстро переместится в левый верхний угол экрана, а затем в точку с координатами 400, 300, то есть в центр экрана при разрешении 800 на 600. Теперь предположим, что в игре нужно нажать F2 для вида кокпита, затем переместить курсор мыши в определенную точку (на кнопку в кокпите), нажать левую кнопку мыши, и вернуться к обзору вперед клавишей F1. Все это можно сделать в рамках одного выражения:

BTN H3D F2 MOUSEXY (UL, 400, 300) MOUSE_LB F1

Если изменить это выражение следующим образом:

BTN H3D F2 MOUSEXY (UL, 50%, 50%) MOUSE_LB F1

то даже при смене экранного разрешения на 1600x1200, и внесении соответствующих изменений в выражение USE SCREEN_RESOLUTION, все сработает так же, как и при разрешении 800x600.

Примечания

1. *Для того, чтобы использовать такие выражения, в файле обязательно должно присутствовать конфигурационное выражение* USE SCREEN_RESOLUTION ( ).

2. *На все выражения программирования мыши влияет параметр* RATE. *Если курсор перемещается слишком медленно, уменьшите этот параметр (чтобы увеличить скорость, так как RATE - на самом деле время задержки). Помните, что если в файле нет конфигурационного выражения* USE RATE (*nnnn*), *то его значение автоматически устанавливается равным нулю - что означает самый быстрый отклик.*

7.8.3 Перемещение курсора относительно текущего положения

Синтаксис команды:

```
MOUSEMOVE (X,Y)
MOUSEMOVE (X%,Y%)
```

где:

- X,Y - количество пикселей, на которое вы хотите сместить курсор относительно текущего положения.
- X%, Y% - выражение этой же величины в процентах от высоты (ширины) экрана в диапазоне от 0 до 100% (с точностью до 3 знака после запятой. Использование процентного выражения позволяет менять разрешение экрана, не меняя параметров выражения.

Как правило, для того, чтобы знать, где именно находится курсор мыши, сначала используется выражение MOUSEXY для его позиционирования.

Рассмотрим примеры:

```
BTN TG1 MOUSEMOVE (20, 50)
```

При нажатии на курок курсор мыши переместится на 20 пикселей вправо по горизонтали и на 50 пикселей вниз по вертикали.

BTN S1 MOUSEMOVE (20%, 50%)

При нажатии на кнопку S1 курсор мыши переместится на 20% ширины экрана вправо и на 50% высоты экрана вверх. При разрешении 800x600 курсор переместится на 160 пикселей вправо (20% от 800) и на 300 пикселей вниз (50% от 600).

BTN H2R MOUSEMOVE (30, 0)

При нажатии Хэта 2 вправо курсор сместится только по горизонтали, на 30 пикселей вправо. Если необходимо переместить курсор влево, перед значением перемещения необходимо поставить знак "-".

BTN H2U MOUSEMOVE (0%, -50%)

При нажатии на Хэт 2 вверх курсор переместится вертикально вверх на 50% высоты экрана - или на 300 пикселей при разрешении 800x600.

Примечания

1. *Для того, чтобы использовать такие выражения, в файле обязательно должно присутствовать конфигурационное выражение* USE SCREEN_RESOLUTION ( ).

2. *На все выражения программирования мыши влияет параметр* RATE. *Если курсор перемещается слишком медленно, уменьшите этот параметр (чтобы увеличить скорость, так как RATE - на самом деле время задержки). Помните, что если в файле нет конфигурационного выражения* USE RATE (*nnnn*)*, то его значение автоматически устанавливается равным нулю - что означает самый быстрый отклик.*

3. *Нельзя одновременно использовать абсолютные и процентные значения. То есть:*

BTN H2U MOUSEMOVE (0%, -50%) *верно, тогда как:*

BTN H2U MOUSEMOVE (0, -50%)

приведет к ошибке компиляции.

4. *При использовании выражения* MOUSEMOVE *курсор одновременно перемещается по обеим осям. То есть с выражением:*

BTN S1 MOUSEMOVE (100, 100)

курсор мыши переместится по диагонали вправо-вниз, а не на 100 пикселей вправо и затем на 100 пикселей вниз.

7.8.4 Перемещение по кругу/по многоугольнику

Синтаксис команд:

MOUSEROTATE (*Исх*, Центр, Радиус, Начальный угол, [Макрос 1], Направление, Конечный угол, Кол-во вершин, [Макрос 2])

Где:

- *Исх* - один из четырех углов экрана: UL, DL, UR, DR.
- *Центр* - точка центра вращения, с координатами X,Y или в %.
- *Радиус* - радиус вращения, выражается в пикселях или в процентах (*см. примечания*)
- *Начальный угол* - точка на дуге, в которой начинается вращение. Считается от 12 часов. Выражается в градусах, от 0 до 360°
- *Макрос 1, 2* - просто макрос, как правило, это нажатие на кнопку мыши (*однако см. примечания по поводу существующих ограничений*.) Используйте пустой символ, если никакого макроса не нужно. Макрос должен быть заключен в квадратные скобки [ ].
- *Направление* - направление вращения, CW или CCW (по или против часовой стрелки)
- *Конечный угол* - угловые координаты точки на дуге, в которой происходит остановка вращения, выражается в градусах, от 0 до 1800° (*пять полных оборотов*)
- *Количество вершин* определяет, насколько плавная описываемая траектория. Значения: 1 или 2. Им соответствует следующие траектории:

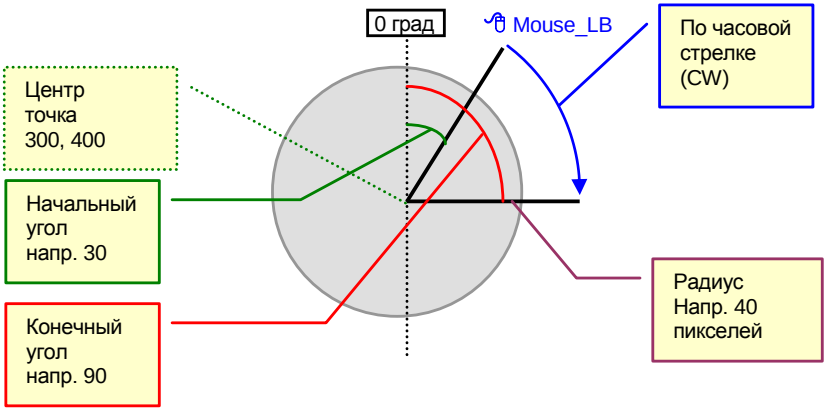
  Number of steps = 1: по квадрату (*4 вершины*),
  Number of steps = 2: по восьмиугольнику (*8 вершин*)

При увеличении количества вершин несколько снижается скорость движения курсора, однако несущественно.

Все числовые значения могут указываться с точностью до 1 знака после запятой (напр. 244,3)

Эта функция была добавлена для того, чтобы пользователь мог в игре одним нажатием кнопки повернуть на нужный угол различные вращающиеся регуляторы интерактивного кокпита в игре.

Рассмотрим пример, в котором необходимо, нажав левую кнопку мыши, повернуть ручку настройки частоты эфира на интерактивной приборной панели. Предположим, центр этой ручки находится в точке с координатами 300, 400 при разрешении экрана 1024x768.



Для того, чтобы достичь необходимого результата, нужны следующие выражения:

```
USE SCREEN_RESOLUTION (1024,768 )
BTN S2 MOUSEROTATE (DL, 300, 400, 40, 30, [MOUSE_LB], CW, 90, 2)
```

Рассмотрим пошагово, что происходит. Итак, нажата кнопка S2:

1. DL - курсор очень быстро перемещается в нижний левый угол экрана, для его дальнейшего позиционирования.
2. 300,400 - курсор перемещается в центр вращения ручки настройки частоты эфира, чтобы оттуда рассчитать остальные параметры вращения.
3. 40 - определяется радиус дуги.
4. 30 - начальный угол по отношению к 12 часам (0 град.). В сочетании с ранее заданными значениями определяется фактическая точка, в которой будет нажата левая кнопка мыши.
5. [MOUSE_LB] - левая кнопка мыши нажимается и удерживается.
6. CW - определяется направление вращения по часовой стрелке
7. 90 - вращение происходит, пока курсор не достигнет точки на дуге с угловыми координатами 90 градусов от вертикали (0 град.)

8. 2 - траектория перемещения - восьмиугольник.
9. Выражение выполнено до конца - макрокоманда прекращается, левая кнопка мыши отпущена.

Сложно? Может быть. Но программное описание вращения мыши - тоже непростое дело :-)

Примечания по макрокомандам

1. *Клавиши и кнопки в макрокоманде всегда удерживаются нажатыми - как если бы присутствовал ключ* /H. *Команда отпускания клавиш и кнопок макрокоманды будет послана автоматически как только движение курсора завершится.*

2. *Список разрешенных действий в макрокоманде:*

- *Простые клавиши* (a, b, 1, 2, `, *и т. д.*)
- *Сочетания клавиш с SHF, ALT, CTL (A, ALT b, и т. д.)*
- *Кнопки DirectX и положения POV* (DX1, POVL, *и т. д.*)
- *Кнопки мыши:* MOUSE_LB / RB / MB
- *Логические флаги* (X1, X2, *и т. д.*)
- *Скан-коды* USB

3. *В таких макрокомандах запрещено использование ключей, а также команд* DLY( ) *или* RPT( ).

Примечания

1. *Выражения* MOUSEROTATE *не могут быть сгруппированы с другими клавишами. То есть:*

BTN S2 a b DLY(30) MOUSEROTATE (*и т. д. и т. п.)* PRNTSCRN

нормально, тогда как:

BTN S2 a b DLY(30) **{**MOUSEROTATE (*и т. д. и т. п.*) PRNTSCRN**}**

приведет к ошибке компиляции.

2. *Для того, чтобы использовать такие выражения, в файле обязательно должно присутствовать конфигурационное выражение* USE SCREEN_RESOLUTION ( ).

3. *На все выражения программирования мыши влияет параметр* RATE. *Если курсор перемещается слишком медленно, уменьшите этот*

параметр (чтобы увеличить скорость, так как RATE - на самом деле время задержки). Помните, что если в файле нет конфигурационного выражения USE RATE (*nnnn*), *то его значение автоматически устанавливается равным нулю - что означает самый быстрый отклик.*

4. *В данных выражениях можно использовать не только собственно команды клавиш и кнопок, но и названия макрокоманд, определенных в макрофайле. Хотя наиболее типичным примером, конечно, будут* MOUSE_LB *или* MOUSE_RB (*левая или правая кнопки мыши*).

5. *Комбинации клавиш как таковые не работают в выражениях* MOUSEROTATE. *Так как все равно все клавиши макрокоманды в выражениях вращения курсора удерживаются одновременно, необходимо использовать* LSHF *вместо* SHF.

6. *Радиус также может быть выражен в % от разрешения экрана по горизонтали. Это может показаться странным, но попробуем пояснить. Радиус - некая фиксированная величина, как правило, выраженная в пикселях. Предположим, разрешение экрана равно 800x600, а радиус вращения равен 600 пикселям, с точкой центра вращения, лежащей на нижней границе экрана. Ситуация маловероятная, но теоретически возможная. Если перевести радиус в проценты, получим: (600/800*100) = 75%. Теперь если разрешение изменится на 1024x768, радиус будет составлять уже 75% от 1024, или 768. Так как теоретически возможно иметь радиус вращения больше высоты экрана, целесообразно выражать его в процентах именно от ширины экрана. И еще один важный момент: большое значение имеет соотношение разрешения по ширине (X) и высоте (Y) экрана. Для разрешений 800x600, 1024x768, 1152x864, 1600x1200 и т. д. это соотношение одинаково и равно 1.333. Однако если вы используете разрешение, к примеру, 1280x1024 (соотношение = 1.25), то из-за другого соотношения радиус, выраженный в %, не сможет нормально масштабироваться.*

7. *Это достаточно сложное выражение, поэтому шанс допустить ошибку весьма велик. Пожалуйста, используйте Мастер Продвинутого Программирования Мыши (Advanced Mouse Programming Wizard) утилиты Foxy. Это была чертовски сложная часть для программирования, и будет обидно, если наши силы были потрачены впустую :)*

8. Логическое программирование

8.1 Основы

8.1.1 Понятие "флага"

Логическое программирование построено вокруг концепции **флагов**. Итак, что же такое флаг? Начнем с некоторых его базовых характеристик, которые должны вас окончательно запутать, а мы затем все проясним при помощи всего лишь одного наглядного примера. Сначала - факты:

- Флаг может быть включен или выключен.
- Всего существует 32 флага, которые называются X1 - X32.
- Флаг сам по себе ничего не делает. Он просто включен или выключен.

Запутались? Тогда в качестве объяснения концепции флага используем клавишу Caps Lock на клавиатуре.

Скажем, что клавиша Caps Lock - это логический флаг, который называется X1. Когда вы нажимаете Caps Lock, на клавиатуре загорается соответствующий светодиод, то есть мы считаем, что флаг X1 включен. Клавиатура, зная о том, что X1 включен, посылает в компьютер, например, символ "W" вместо "w", если нажать на клавишу w. Если же выключить Caps Lock, нажав на него повторно, светодиод погаснет, то есть флаг X1 выключен, и в компьютер будет послан символ "w". Иным словами, сама по себе клавиша Caps Lock ничего не делает - она просто включает или выключает логический флаг X1, а вот уже от его состояния зависит, какой именно символ посылается в компьютер - заглавная или строчная "w".

Эту концепцию очень важно понять. Флаг сам по себе ничего не делает. Он просто включается или выключается. А вы уже используете его включенное или выключенное состояние для изменения того, что вы получаете на выходе из джойстика. То есть логическое программирование - это определение событий, которые происходят в зависимости от того, включен или выключен логический флаг.

8.2 Определение логического флага и выражения с ним

Начнем с того, как включить флаг:

Конфигурационное выражение :

DEF X1 S2

Выражение логического программирования:

BTN X1 /H Fire_rockets

В приведенном примере мы определили логический флаг X1 через выражение DEF, в котором мы указали, что его состояние зависит от состояния кнопки S2. Иными словами, когда кнопка S2 нажата, флаг X1 включен, а когда она отпущена, флаг X1 выключен.

После того, как вы определили логический флаг, его можно запрограммировать при помощи команды кнопки (BTN). В нашем примере когда кнопка S2 нажата, будет осуществляться многократный пуск НУРСов. Это происходит потому, что в файле настроек определено, что именно происходит, если флаг X1 включен, а он остается включенным, пока нажата кнопка S2. Важно отметить, что выражения для логических флагов подчиняются тем же правилам, что и выражения для кнопок, с той разницей, что в них нельзя использовать ключ цепочки переключений /T. Это означает, что указанная макрокоманда по умолчанию будет выполнена однократно, как и любая другая команда в выражении для кнопки, даже несмотря на то, что флаг остается включенным. Однако вы можете использовать ключ /H, чтобы удерживать нажатой эту, скажем так, логическую кнопку.

Вы, наверное сейчас чешете затылок и задаетесь вопросом: "*А почему бы просто не написать:*

BTN S2 /H Fire_rockets

в файле настроек? Зачем нужны какие-то логические флаги?" Понятное дело, что в данном примере не нужны совершенно. Но далее в руководстве будут приведены примеры того, что никак не может быть реализовано без использования логического программирования - пока просто объясняется синтаксис.

Логические флаги можно также определять непосредственно в выражении для кнопок и в цифровых выражениях 1 типа:

```
RNG 2 5 X1 X2 X3 X4 X5
BTN H1L X8
```

Однако есть некоторая разница между определением флага через конфигурационное выражение и определением его непосредственно вы выражении для кнопки. Возьмем пример:

```
DEF X20 S1
```

Флаг X20 включается и остается включенным все время, пока нажата кнопка S1. Возьмем другой пример:

```
BTN S1 X20
```

(*без выражения DEF*) В этом случае флаг X20 будет вести себя точно так же, как любая другая команда, то есть он включится и сразу выключится, даже если кнопка S1 останется нажата. А вот если составить такое выражение:

```
BTN S1 /H X20
```

то результат будет полностью идентичен DEF X20 S1. Помните, что нельзя в файле настроек определить флаг через конфигурационное выражение, и далее определить его же непосредственно в выражении для кнопки или оси.

Как мы сказали выше, флаг, определенный непосредственно через выражение для кнопки, ведет себя точно так же, как и любая другая команда. Таким образом, возможно определять флаги и так:

```
BTN S1 KD(X8) DLY(2000) KD(X6) KU(X6 X8)
```

Это может показаться не особо ценной возможностью, однако допустим вам необходимо выполнить цепочку действий, состоящую из 2 этапов: однократного установочного этапа и автоповтора. Это можно сделать через следующее выражение:

```
BTN X1 /A Fire_Main_Guns Rem Огонь из пушек
BTN S1 /H Switch_to_Main_Guns X1  Rem Переключиться на пушки и - огонь!
```

Ключ /H, как известно, действует на последнюю команду в выражении- то есть к флагу X1. В итоге макрокоманда Switch_to_Main_Guns (переключиться на пушки) выполняется однократно, а макрокоманда Fire_Main_Guns (огонь из пушек) выполняется в режиме автоповтора через удержание логического флага. Круто, да?

Примечания

Будьте осторожны при определении одного и того же логического флага как через DEF*, так и непосредственно в выражении для кнопки. Пример:*

```
DEF X1 S4
BTN S2 X1
```

на самом деле то же самое, что и:

```
DEF X1 S4 OR S2
```

то есть флаг X1 включится, если будет нажата либо S4, либо S2.

8.3 Логические компараторы

Существует три логических компаратора: AND, NOT и OR (И, НЕ и ИЛИ), с которыми можно также использовать круглые скобки.

Конфигурационные выражения:

```
DEF X1 S2 OR T6
DEF X2 S4 AND S3 AND H1U
DEF X3 S1 AND NOT X1
```

Выражения логического программирования:

```
BTN X1 Fire_missile
BTN X2 Engines_off Gather_belongings Eject Rem Двигатель_выкл, Пакуем_вещички Катапультирование
BTN X3 AutoPilot
```

Рассмотрим первую пару:

```
DEF X1 S2 OR T6
BTN X1 Fire_missile
```

Флаг X1 может быть включен нажатием либо на кнопку S2, либо на кнопку T6. Таким образом, нажатие на S2 или T6 осуществляет однократный запуск ракеты. То же самое могло бы быть достигнуто так:

```
BTN S2 Fire_missile
BTN T6 Fire_missile
```

Теперь рассмотрим то, что не может быть реализовано без логического программирования:

```
DEF X2 S4 AND S3 AND H1U
BTN X2 Engines_off Gather_belongings Eject Rem Двигатель_выкл, Пакуем_вещички Катапультирование
```

В данном примере флаг X2 включается только тогда, когда одновременно нажаты кнопки S4, S3 и Хэт 1 вверх. Вряд ли такое может произойти случайно. Но если уж вы умудритесь - приземляться будете не на три точки, а на одну, но пятую :) И наконец:

```
DEF X1 S2 OR T6
DEF X3 S1 AND NOT X1
BTN X3 AutoPilot
```

Мы определили, что флаг X3 включается, если одновременно выполняются два условия: кнопка S1 нажата и кнопки S2 или T6 НЕ нажаты. Таким образом, нажатие на S1 включит автопилот, но только в том случае, если не нажаты кнопки S2 или T6.

Еще раз напомним, что логические флаги, запрограммированные непосредственно через выражения для кнопок, воспроизводятся однократно, как и все другие команды.

И в заключение добавим, что логические компараторы можно группировать в круглые скобки. Выражения DEF могут состоять из любой комбинации компараторов AND, OR, и NOT, скобок, названий кнопок и флагов - главное, чтобы в итоге получалось корректное логическое уравнение, например:

```
DEF X1 (S1 AND NOT S2) OR (X5 AND (H1U OR H2U))
```

8.4 Логическое переключение

В приведенных выше примерах логические флаги были определены таким образом, что они включались, когда кнопка (или комбинация кнопок) была нажата, и выключались, когда кнопки отпускались. Можно также сделать так, чтобы флаги "залипали" - то есть оставались включенными, даже если кнопка отпущена, и выключались при повторном нажатии. Иными словами, происходит переключение между состояниями флага: включен-выключен. Для этого после названия кнопки или флага необходимо добавить звездочку "*".

Конфигурационное выражение:

```
DEF X1 S4*
```

Выражения логического программирования:

BTN X1 /A Chaff DLY(30) Flare DLY(30)

Когда кнопка S4 нажата, флаг X1 включается и остается включенным, даже если S4 отпущена. Повторное нажатие кнопки S4 выключает флаг. В приведенном примере при нажатии и отпускании кнопки S4 вы начинаете сбрасывать ДО и ЛТЦ, пытаясь отвертеться сразу от самонаводящейся ракеты и от Сайдуиндера (*неудачный денек, надо сказать!*). Повторное нажатие на S4 остановит сброс ДО и ЛТЦ (если, конечно, к этому моменту они сами не кончились).

Примечания

1. *Нельзя программировать логическое переключение непосредственно в выражении для кнопки. Например:*

BTN T6 X1*

приведет к ошибке компиляции (в отличие от логического программирования предшественника - TM F-22 PRO). Логическое переключение можно задавать только в конфигурационных выражениях DEF*. Например, вот так:*

DEF X1 T6*

2. *Помните, что, в отличие от обычных кнопок, ключ цепочки переключений* /T ***не может*** *использоваться в выражениях программирования логических флагов, то есть:*

BTN X7 /T a /T b

приведет к ошибке компиляции. Однако следующее возможно:

BTN S4 /T X1 /T X2 /T X3

8.5 Использование функций логической задержки (DELAY) и логических импульсов (PULSE)

8.5.1 Задержка (Delay)

Функция задержки (DELAY) добавляет определенное время между тем, когда логическое уравнение становится верным, и моментом, когда включается логический флаг. Синтаксис:

Конфигурационное выражение:
DEF Xflag DELAY(Время_задержки) *Логическое уравнение*

Рассмотрим пример:

```
DEF X1 DELAY(1000) S1 AND S4
BTN X1 Eject
```

В данном примере флаг X1 включается через одну секунду (1000 миллисекунд) **после того,** как кнопки S1 и S4 будут нажаты и удержаны одновременно. Если одна из кнопок будет отпущена до истечения 1 секунды, отсчет задержки обнуляется. Если флаг X1 еще не включился, он и не включится. Если он уже включился, он сразу же выключится. Если кнопки S1 и S4 вновь нажаты одновременно, отсчет 1 секунды начнется заново.

Примечания

1. *Логическая задержка* (DELAY) *не имеет ничего общего с обычной задержкой* DLY( ). *Допустимые значения логической задержки - от 0 до 327670 (примерно 5,46 минут - даже не думайте спрашивать почему!*).

2. *Команда* DELAY *должна стоять первой в выражении логического программирования. То есть:*

```
DEF X1  DELAY(1000) S1 AND S4      верно, но:

DEF X1 X2 AND DELAY(1000) S1 AND S4      приведет к ошибке компиляции.
```

8.5.2 Импульс (Pulse)

Функция логического импульса (PULSE) посылает последовательность изменений состояния флага (включен-выключен), пока логическое уравнение верно. Два числовых параметра определяют, соответственно, период включенного состояния и период выключенного состояния. Синтаксис:

Конфигурационное выражение:

DEF Xflag PULSE(Время_вкл Время_выкл) *Логическое уравнение*

Например:

```
DEF X1  PULSE(100 1000) H1U
BTN X1 Trim_up_increase Rem Триммер вверх
```

Нажатие на Хэт 1 вверх немедленно включит флаг X1, который будет оставаться включенным в течение 1/10 секунды, после чего выключится на 1 секунду, затем вновь включится на 1/10 секунды, и т. д. Это будет происходить все время, пока Хэт 1 нажат вверх. В данном примере триммер руля высоты отклоняется на один шаг вверх каждые 1,1 секунды. заметьте, именно 1,1 секунды. Если нам надо, чтобы весь цикл происходил за 1 секунду ровно, выражение надо изменить:

```
DEF X1  PULSE(100 900) H1U
```

Отметьте также, что два значения должны быть разделены пробелом. Использование запятой или любого другого разделителя приведет к ошибке компиляции.

Примечания

1. *Команда* PULSE *должна стоять первой в выражении логического программирования. То есть:*

```
DEF X1  PULSE(100 1000) S1 AND TG1      верно, тогда как
DEF X1 X2 AND PULSE(100 1000) S1        приведет к ошибке.
```

2. *Фактическое разрешение команд* DELAY, RATE, PULSE - *около 30 миллисекунд. Минимальное ощутимое значение также составляет 30 миллисекунд. Таком образом, любое значение от 1 до 30 дает задержку примерно в 30 мс, от 31 до 60 - 60 мс, от 61 до 90 - 90 мс, и так далее.*

3. *Допустимые значения логического импульса* (PULSE) - от *0 до 82800000 (то есть 23 часа).*

8.5 Примеры логического программирования

8.5.1 Включение и выключение работы цифровых выражений 4 типа

Напомним, что цифровое выражение 4 типа позволяет запрограммировать на ось (например, кольцо масштаба) генерирование многократно повторяющихся символов (импульсное генерирование):

```
RNG 4 1000 a ^ b
```

Единственным способом остановить импульсное генерирование символов в данном случае будет установить кольцо масштаба в центральное положение, которому соответствует "пустой символ" (^). Но допустим, мы хотим иметь возможность остановить импульсное генерирование, нажав на кнопку T6 (головка кольца масштаба). Это легко реализовать с помощью логического программирования:

```
DEF X3 T6*
DEF X4 X1 AND NOT X3
DEF X5 X2 AND NOT X3
RNG 4 1000 X1 ^ X2
BTN X4 a
BTN X5 b
```

Как это работает? Теперь кольцо антенны вместо того, чтобы генерировать повторяющиеся символы, включает и выключает флаги X1 и X2. Логические кнопки X4 и X5 однократно генерируют символы a и b, но только если флаг X3 выключен. Так как кнопка T6 переключает состояния флага X3, теперь нажатием этой кнопки можно останавливать и вновь включать импульсное генерирование символов кольцом масштаба.

8.5.2 Постепенное триммирование

Спасибо Марку за это элегантное решение!

Предположим, что в авиасимуляторе мы используем клавиши KP7 и KP1 для триммирования руля высоты самолета. Мы запрограммируем один из хэтов на функцию постепенного триммирования, то есть если мы нажмем Хэт 1 вверх и отпустим, то каждые 5 секунд генерируется клавиша KP7, триммируя руль высоты вверх, а повторное нажатие останавливает триммирование. То же самое происходит с нажатием Хэта 1 вниз и клавишей KP1. Выражения:

```
DEF X1 H1U* AND (NOT X2)
DEF X2 H1D* AND (NOT X1)
BTN X1 /A KP7 DLY(5000)
BTN X2 /A KP1 DLY(5000)
```

Теперь объясним, что именно происходит. Когда Хэт 1 нажимается вверх, включается флаг X1 и остается включенным, так как используется логическое переключение (*). Когда флаг X1 включен, выполняется выражение логической кнопки X1, и клавиша KP7 генерируется каждые 5 секунд. Когда Хэт 1 повторно нажат вверх, флаг X1 выключается, и генерирование KP7 прекращается. То же самое для X2. Включение в уравнения компараторов (NOT X1) и (NOT X2) сделано для того, чтобы избежать одновременного генерирования KP1 и KP7 если Хэт 1 нажат, скажем, вниз в то время, как флаг X1 включен.

В директории Files находится еще несколько файлов, составленных Марком (Mark Mooney), в которых дается еще ряд примеров логического программирования.

9. Устранение неполадок

9.1 Сброс контроллера

Есть несколько способов борьбы с возможными проблемами, возникающими в работе манипуляторов.

9.1.1 В игре: EMPTY_BUFFERS и STICK_OFF

Теоретически возможно (*хотя и очень маловероятно!*), что HOTAS Cougar выдаст одновременно слишком много символов - либо вы слишком быстро нажмете слишком много кнопок (*что вряд ли*), либо вы неграмотно составите файл настроек (*вполне возможно*), и джойстик перестанет нормально работать. Это может проявиться следующим образом - джойстик как будто зависнет: аналоговые оси будут работать, но кнопки не отзываются. В такой ситуации можно очистить буфер команд, не удаляя файл настроек из памяти, и продолжить игру.

Синтаксис команды:

EMPTY_BUFFERS

Пример:

BTN S2 EMPTY_BUFFERS

Конечно, такая команда будет использоваться крайне редко, и имеет смысл задать ее через логическое программирование, чтобы для ее выполнения потребовалось нажать какую-нибудь совершенно неудобоваримую комбинацию кнопок, которую уж точно нельзя нажать случайно.

```
DEF X1  DELAY(2000) S1 AND S4
BTN X1 EMPTY_BUFFERS
```

В приведенном примере нужно нажать одновременно и удерживать кнопки S1 и S4 в течение 2 секунд, прежде чем сработает флаг X1, который, в свою очередь, пошлет команду EMPTY_BUFFERS.

Таким же способом можно просто отключить контроллер прямо из игры, принудительно переводя кнопки в режим Windows, при помощи команды STICK_OFF.

Синтаксис команды:

```
STICK_OFF
```

Пример:

```
BTN S2 STICK_OFF
```

Конечно, такая команда будет использоваться крайне редко, и имеет смысл задать ее через логическое программирование, чтобы для ее выполнения потребовалось нажать какую-нибудь совершенно неудобоваримую комбинацию кнопок, которую уж точно нельзя нажать случайно.

```
DEF X1  DELAY(2000) S1 AND S4
BTN X1 STICK_OFF
```

Если вы прибегли к этой команде, то включить обратно джойстик из игры уже нельзя, поэтому это - на крайний случай, когда вдруг джойстик не переставая выдает клавиатурные команды, и вы не можете остановить их. После этого надо выйти из игры и устранить неполадки уже в Windows.

9.1.2 В среде Windows

В меню утилиты Foxy вы найдете пункты, позволяющие осуществлять сброс различных аспектов работы джойстика. Приводим их в порядке возрастания "аварийности".

1. Empty buffers (очистка буфера)

То же самое - можно очистить буфер, оставив настройки в памяти и не меняя режима работы джойстика.

2. Disabling programmed mode (отключение программируемого режима)

Из окна Foxy очень просто переключить джойстик в режим Windows, если вдруг он начал неконтролируемо выдавать клавиатурные команды. Если вам особенно не повезло, и "залипшей" клавишей оказалась "F1", вам предоставляется уникальная возможность любоваться калейдоскопом окон справки, пока вы не долезете до USB-кабеля и не выдернете его к… совсем!

3. Clear memory (очистка памяти)

Джойстик переходит в режим Windows. Память очищается от файла настроек.

4. Flash memory (перепрошивка ПЗУ)

В случае, если с джойстиком произошла серьезная неприятность, когда он распознается в Windows, но оси и кнопки не работают, или если вышла новая версия прошивки, воспользуйтесь этим пунктом меню.

5. Звонок в техническую поддержку!

Если дело совсем плохо - джойстик не распознается, кнопки и оси не работают - то, вероятно, это либо сбой в операционной системе, либо некорректно установлены драйвера, либо серьезная проблема в аппаратной части. Свяжитесь со службой технической поддержки, чтобы выяснить, требуется ли замена джойстика.

10. Приложения

Приложение 1.
Сводная таблица команд и выражений

Выражения для кнопок и модификаторы

Команда	Расшифровка	Описание
BTN	Кнопка	Определяет программируемую кнопку. Сюда входят хэты с 1 по 4, кнопки S1 - S4, T1 - T10, TG1 и 2.
REM	Комментарий	Весь текст, стоящий в строке после REM, игнорируется компилятором. Используется для заголовков, комментариев и т. д.
RESET_TOGGLES	Сброс цепочки переключений	Сбрасывает цепочку переключений на команду первого ключа /T.
REVERSE_TOGGLES	Инверсия цепочки переключений	Воспроизводит цепочку переключений в обратном порядке
DLY	Задержка	Добавляет задержку между символами или макрокомандами.
RPT	Повтор	Многократно повторяет символ или макрокоманду
( )	Заключение в круглые скобки	Объединяет символы или макрокоманды в различных выражениях, включая выражения цифровых режимов
{ }	Заключение в фигурные скобки	Объединяет символы или макрокоманды таким образом, что сначала посылаются все коды нажатия, а затем – все коды отпускания. Соответствует удержанию группы клавиш в нажатом состоянии.
< >	Заключение в угловые скобки	Все команды внутри выполняются до конца, прежде чем могут быть выполнены какие-либо другие команды
DX	Кнопки DirectX	Кнопки DirectX, которые могут программироваться через

		любые выражения для кнопок
KD KU	Нажатие Отпускание	Позволяет полностью контролировать события нажатия и отпускания клавиш.
USB	Скан-коды USB	Используется для прямого программирования кодов нажатия и отпускания клавиш.
REVERSE_UD REVERSE_LR REVERSE_DIR	Инверсия направления	Инвертирует направление осей или хэтов

Команда	Расшифровка	Описание
NOHOLD, KP5	Используется с выражениями USE HAT AS	Команда NOHOLD запрещает удержание клавиш при исполнении выражений для хэтов. Команда KP5 добавляет центральное положение KP5 в выражении USE HAT AS KEYPAD.
FORCED_CORNERS	Обработка промежуточных положений хэта	Промежуточные положения d в принудительном порядке воспроизводятся как сочетание двух соседних основных положений
S3_LOCK, S3_UNLOCK	Блокировка/разблокировка S3	Используется для перевода кнопки S3 в "залипающий" режим
SHIFTBTN	Назначение кнопки как S3	Назначает другую кнопку для использования в качестве S3
POV*n*, (n = U, D, L,R,UL, DL,UR, DR)	Положения переключателя видов (POV)	Позволяет программировать положения стандартного переключателя видов DirectX
MOUSE_LB, MOUSE_RB, MOUSE_MB	Кнопки мыши	Позволяет программно эмулировать нажатие кнопок мыши

Ключи

Ключ	Расшифровка	Описание
/U, /M, /D	Верхнее, среднее, нижнее положения	Утраивает количество программируемых позиций для каждой из кнопок и хэтов (кроме T7 и T8) в зависимости от положения переключателя "дальний-ближний бой" на РУД.
/I, /O	Нажато, Отжато	Удваивает количество программируемых позиций для каждой из кнопок и хэтов в зависимости от состояния кнопки S3 на РУС (*не может использоваться на самой S3*)
/P, /R	Нажать,	Позволяет раздельно программировать

	отпустить	нажатие и отпускание любой кнопки или хэта.
/T	Цепочка переключений	С каждым последующим нажатием происходит переключение на следующий символ (макрокоманду) в цепочке
/H	Удержание	Генерирует удержание клавиши в то время, как нажата соответствующая кнопка, независимо от того, нажимаются ли в это время другие кнопки. Может использоваться в сочетании с другими ключами.
/A	Автоповтор	Автоматически повторяет все команды в данной строке.

Конфигурационные выражения

Синтаксис	Описание
USE MDEF	Указывает макрофайл, в котором содержатся определения макрокоманд для данного файла настроек.
USE *Btn* AS DXn	Назначает кнопки джойстика как кнопки DirectX. Заменяет прежнюю команду PORTB1 IS
USE ALL_DIRECTX_BUTTONS	Назначает Хэт 1 как стандартный переключатель видов (POV), а все кнопки – как кнопки DirectX
USE HAT*n* FORCED_CORNERS	Превращает 8-позиционный хэт в 4-позиционный, при этом в промежуточных положениях исполняется комбинация команд соседних основных положений
USE HAT*n* AS MOUSE, POV, ARROWKEYS, KEYPAD	Определяет, каким образом использовать хэт вместо обычного программирования его положений
USE RATE	Частота генерирования (повтора) символов
USE S3_LOCK USE S3_UNLOCK	Делает кнопку S3 "залипающей"
USE S3 AS SHIFTBTN	Определяет другую кнопку для использования вместо S3 с ключами /I, /O
USE HATx_SENSITIVITY	Снижает "чувствительность" хэта для более четкой фиксации положений
USE T1_SENSITIVITY	Снижает "чувствительность" кнопки T1
USE FOXY	Используется для внутренних функций утилиты Foxy, напр. USE FOXY GRAPHIC USE FOXY README
USE NULLCHR	Определяет "пустой" символ, по умолчанию - ^
USE KEYBOARD	AZERTY – для французской раскладки клавиатуры
USE PROFILE	Для использования профиля, сохраненного в Панели управления Cougar
USE CURVE	Определяет кривую отклика оси
DISABLE AXIS	Отключает ось
USE SWAP	Меняет оси местами

USE REVERSE	Инвертирует ось
USE AXES_CONFIG	Определяет список используемых осей и их параметры
USE MTYPE	Назначает микроджойстик для управления мышью, назначает кнопки мыши
USE MICROSTICK AS MOUSE – NO_BUTTON	Назначает микроджойстик для управления мышью, кнопка T1 работает как левая кнопка мыши (если не используется - NO_BUTTON)
USE Axis_Identifier AS Mouse_Axis	Назначает любую ось для управления перемещением курсора
USE ZERO_MOUSE	Предотвращает "зависание" мыши при использовании /I, /O
DISABLE MOUSE	Отключает назначение мыши по умолчанию
USE SCREEN_RESOLUTION	Используется для программирования сложных движений курсора

Программирование осей

Выражение	Расшифровка	Описание
JOYX, JOYY	Joystick X, Y	Оси РУС
THR	Throttle	Ось РУД
RDDR	Rudder	Педали (руль поворота)
ANT	Antenna knob	Ось кольца антенны
RNG	Range knob	Ось кольца масштаба
MIX, MIY	Microstick X, Y	Оси микроджойстика
LBRK, RBRK	Left, Right Toe Brakes	Оси педальных тормозов
MSX, MSY, MSZ	Mouse X, Y, Z	Оси мыши. Ось Z – колесико прокрутки
Цифровые выражения 1 – 6 типов		Позволяет программировать оси для генерирования клавиатурных символов. Также используются для программирования мыши, кривых отклика и в логическом программировании.
FORCE_MACROS	Принудительное выполнение макрокоманд	Принудительное выполнение всех макрокоманд в цифровых выражениях 1, 2, 5 и 6 типов
CURVE	Кривые осей	Изменение кривых отклика осей
TRIM TO_CURRENT HOLDTRIM	Триммирование оси	Триммирование оси на заданную величину
LOCK, UNLOCK, LASTVALUE	Блокировка оси	Блокировка аналогового значения оси "на лету" для ее использования в цифровом режиме
SWAP	Переброс осей	Меняет оси местами
REVERSE FORWARD	Направление оси	Инвертирует ось, или возвращает нормальное направление

Продвинутые выражения управления мышью

Выражение	Описание
MOUSEXY	Помещает курсор в заданную точку на экране
MOUSEMOVE	Перемещает курсор относительно текущего положения
MOUSEROTATE	Программирует круговые движения курсора для программного управления вращающимися регуляторами интерактивных кокпитов

Логические выражения

Логический синтаксис	Расшифровка	Описание
DEF	Определить	Определяет логические флаги через логические уравнения
BTN X*n*	Кнопка	Программирует виртуальную логическую кнопку X1 - X32
X1 - X32	Логические флаги	Логические флаги, бывают включены или выключены
AND, OR, NOT	Логические компараторы	Используются при построении логических уравнений
*	Логическое переключение	Переключается между состояниями флага – вкл/выкл
DELAY	Логическая задержка	Вводит задержку, по истечении которой логическое условие считается выполненным
PULSE	Логический импульс	Воспроизводит символы (состояния флага) через каждые *nn* миллисекунд

Аппаратные команды

Синтаксис	Описание
EMPTY_BUFFERS	Очищает буфер контроллера
STICK_OFF	Выключает контроллер во время игры

Приложение 2. Синтаксис клавиш Thrustmaster

ESC	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	
`	1	2	3	4	5	6	7	8	9	0	-	=	BSP
TAB	q	w	e	r	t	y	u	i	o	p	[	]	\
CAPS	a	s	d	f	g	h	j	k	l	;	’	ENT	
LSHF	z	x	c	v	b	n	m	,	.	/	RSHF		
LCTL	LALT	SPC								RALT	RCTL		

PRNTSCRN	SCRLCK	BRK

INS	HOME	PGUP
DEL	END	PGDN

	UARROW	
LARROW	DARROW	RARROW

NUML	KP/	KP*	KP-
KP7	KP8	KP9	KP+
KP4	KP5	KP6	
KP1	KP2	KP3	KPENT
KP0		KP.	

Примечания

1. *Синтаксис для клавиатурных комбинаций (с клавишами Shift, ALT или CTRL):* SHF a, ALT b, CTL c, *а не* LSHF a, LALT b LCTL c

2. *Некоторые символы зарезервированы:* ( ) { } < >, *и должны быть представлены в виде комбинаций с* SHF*:*

(= SHF 9
) = SHF 0
{ = SHF [
} = SHF]
< = SHF ,
> = SHF .

Приложение 3. Скан-коды USB

Пример:

BTN S2 /P USB (D04) Rem скан-код нажатия “a”
/R USB (U04) Rem скан-код отпускания “a”

Клавиша	Синтаксис TM	Скан-код USB
a A	a	04
b B	b	05
c C	c	06
d D	d	07
e E	e	08
f F	f	09
g G	g	0A
h H	h	0B
i I	I	0C
j J	j	0D
k K	k	0E
l L	l	0F
m M	m	10
n N	n	11
o O	o	12
p P	p	13
q Q	q	14
r R	r	15
s S	s	16
t T	t	17
u U	u	18
v V	v	19
w W	w	1A
x X	x	1B
y Y	y	1C
z Z	z	1D
1 !	1	1E
2 @	2	1F
3 #	3	20
4 $	4	21
5 %	5	22
6 ^	6	23
7 &	7	24
8 *	8	25
9 (	9	26
0)	0	27
Return	ENT	28
Escape	ESC	29
Backspace	BSP	2A
Tab	TAB	2B
Space	SPC	2C
- _	-	2D

= +	=	2E
[{	[	2F
] }	]	30
\ |	\	31
Europe 1 (*см. примечания*)		32
; :	;	33
' "	'	34
` ~	`	35
, <	,	36
. >	.	37
/ ?	/	38
Caps Lock	CAPS	39
F1	F1	3A
F2	F2	3B
F3	F3	3C
F4	F4	3D
F5	F5	3E
F6	F6	3F
F7	F7	40
F8	F8	41
F9	F9	42
F10	F10	43
F11	F11	44
F12	F12	45
Print Screen	PRNTSCRN	46
Scroll Lock	SCRLCK	47
Break (Ctrl-Pause)	BRK	48
Pause	BRK	48
Insert	INS	49
Home	HOME	4A
Page Up	PGUP	4B
Delete	DEL	4C
End	END	4D
Page Down	PGDN	4E
Right Arrow	RARROW	4F
Left Arrow	LARROW	50
Down Arrow	DARROW	51
Up Arrow	UARROW	52
Num Lock	NUML	53
Keypad /	KP/	54
Keypad *	KP*	55
Keypad -	KP-	56
Keypad +	KP+	57
Keypad Enter	KPENT	58
Keypad 1 End	KP1	59
Keypad 2 Down	KP2	5A
Keypad 3 PageDn	KP3	5B

Keypad 4 Left	KP4	5C
Keypad 5	KP5	5D
Keypad 6 Right	KP6	5E
Keypad 7 Home	KP7	5F
Keypad 8 Up	KP8	60
Keypad 9 PageUp	KP9	61
Keypad 0 Insert	KP0	62
Keypad . Delete	KP.	63
Europe 2 (*см. примечания*)		64
Keypad =		67
F13		68
F14		69
F15		6A
F16		6B
F17		6C
F18		6D
F19		6E
F20		6F
F21		70
F22		71
F23		72
F24		73
Keyboard Execute		74
Keyboard Help		75
Keyboard Menu		76
Keyboard Select		77
Keyboard Stop		78
Keyboard Again		79
Keyboard Undo		7A
Keyboard Cut		7B
Keyboard Copy		7C
Keyboard Paste		7D
Keyboard Find		7E
Keyboard Mute		7F
Keyboard Volume Up		80
Keyboard Volume Dn		81
Keyboard Locking Caps Lock		82
Keyboard Locking Num Lock		83
Keyboard Locking Scroll Lock		84
Keypad , (Brazilian Keypad .)		85
Keyboard Equal Sign		86
Keyboard Int'l 1 (Ro)		87
Keyboard Intl'2 (Katakana/Hiragana)		88
Keyboard Int'l 2 ¥ (Yen)		89

Keyboard Int'l 4 (Henkan)		8A
Keyboard Int'l 5 (Muhenkan)		8B
Keyboard Int'l 6 (PC9800 Keypad ,)		8C
Keyboard Int'l 7		8D
Keyboard Int'l 8		8E
Keyboard Int'l 9		8F
Keyboard Lang 1 (Hanguel/English)		90
Keyboard Lang 2 (Hanja)		91
Keyboard Lang 3 (Katakana)		92
Keyboard Lang 4 (Hiragana)		93
Keyboard Lang 5 (Zenkaku/Hankaku)		94
Keyboard Lang 6		95
Keyboard Lang 7		96
Keyboard Lang 8		97
Keyboard Lang 9		98
Keyboard Alternate Erase		99
Keyboard SysReq/Attention		9A
Keyboard Cancel		9B
Keyboard Clear		9C
Keyboard Prior		9D
Keyboard Return		9E
Keyboard Separator		9F
Keyboard Out		A0
Keyboard Oper		A1
Keyboard Clear/Again		A2
Keyboard CrSel/Props		A3
Keyboard ExSel		A4
Left Control	LCTL	E0
Left Shift	LSHF	E1
Left Alt	LALT	E2
Left GUI		E3
Right Control	RCTL	E4
Right Shift	RSHF	E5
Right Alt	RALT	E6
Right GUI		E7

Примечания

Эти клавиши различаются в зависимости от типа раскладки и места изготовления клавиатуры. Europe 1, как правило, это 42 клавиша по стандарту AT-101, и находится рядом с клавишей Enter. Europe 2, как

правило, это 45 клавиша по стандарту AT-101, и находится между левой клавишей Shift и Z.

Приложение 4. Отличия между языком файлов настроек прежних манипуляторов TM и файлов HOTAS Cougar

Существует ряд отличий между файлами настроек для SWF Digital chips, F22, FLCS, FCS, TQS, WCS MkII. В данном разделе представлены сводные данные об этих отличиях. Более подробно см. документацию к соответствующим манипуляторам Thrustmaster.

1. Изменения в обозначении клавиш

Старые обозначения	Новые обозначения
LSFT	LSHF
RSFT	RSHF
none	PRNTSCRN
AUXUAROW, UAROW	UARROW
AUXDAROW, DAROW	DARROW
AUXLAROW, LAROW	LARROW
AUXRAROW, RAROW	RARROW
AUXENT	KPENT
AUX/	KP/
AUXINS	INS
AUXHOME	HOME
AUXPGUP	PGUP
AUXPGDN	PGDN
AUXDEL	DEL
AUXEND	END

2. Изменения ключей

Ключи	Комментарии
/U	Как и прежде
/M	
/D	
/I	Как и было, но теперь строки /I всегда должны предшествовать строкам /O, и ключи /I, /O обязательно должны быть на разных строках
/O	
/P	Как и прежде
/R	
/T	Как и прежде

/A	Теперь означает автоповтор
/H	Как и прежде, но символы воспроизводятся многократно, а в сложных выражениях ключ действует на последний символ в строке
/F	Более не поддерживается
/Q	Более не поддерживается
/N	Более не поддерживается – все символы теперь выполняются однократно, если нет ключей /H, /A

3. Команды, которые более не поддерживаются

Команда	Комментарии
RAW ( )	Заменено USB ( ) *(коды HID)*, однако проще использовать команды KD( ) и KU( )
BTN MT	Используйте цифровые выражения 5 типа – у них намного больше возможностей
BTN T11 – T14 (они более не существуют)	Микроджойстик теперь является устройством с 2 аналоговыми осями, а не 4 кнопками, и может программироваться через цифровые выражения 1 - 6 типов
USE RCS	Более не требуется
USE TQS	Более не требуется
USE WCS	Более не требуется
USE RCSPRO	Более не требуется
USE NOMOUSE	Более не требуется
USE NOTHR	Более не требуется
USE MTYPE (B или C)	Типы мыши более не поддерживаются, хотя возможна эмуляция трехкнопочной мыши.

4. Имена и расширения файлов

Файл настроек должен иметь расширение “.tmj”, а файл макрокоманд - “.tmm”. Имена файлов настроек и файлов макрокоманд теперь могут иметь более 8 символов и содержать пробелы. Но как и прежде, файл настроек и файл макрокоманд должны находиться в одной директории. По умолчанию это папка Files утилиты Foxy.

5. Действия по умолчанию

В зависимости от выбранных опций в настройках Foxy компилятор автоматически выполняет ряд действий. Эти действия отменяются либо через настройки Foxy, либо если в файле настроек есть противоречащие тому выражения.

Действия:

- Назначение Хэта 1 (или другого, по выбору) в качестве стандартного переключателя видов (POV)
- Назначение TG1 как DX1, S2 как DX2 – то есть как кнопок DirectX? функции которых назначаются в игре.
- Если нет конфигурационного выражения USE MDEF, но в файле настроек присутствуют макрокоманды, компилятор будет искать файл с таким же именем и расширением “.tmm” в той же директории, где находится файл настроек
- Микроджойстик управляет мышью, кнопка T1 работает как левая кнопка мыши.

Это сделано для упрощения задачи для начинающих. Допустим, если есть файл настроек с одной строкой:

```
BTN S1 Autopilot
```

и файл макрокоманд с одной строкой:

```
Autopilot = a
```

то можно смело загружать файл в джойстик, и курок будет работать, равно как и мышь, переключатель видов и т. д.

Проблемы могут возникнуть лишь при передаче файла кому-то другому, у кого изменены настройки по умолчанию.

6. Цифровой и аналоговый режимы работы осей

В прежних манипуляторах TM ось могла работать либо в аналоговом, либо в цифровом режиме. Оси HOTAS Cougar могут работать одновременно как в аналоговом, так и в цифровом режимах. Для того, чтобы получить "чистую" цифровую ось, ее аналоговую составляющую надо отключить при помощи конфигурационного выражения DISABLE. Помните также, что теперь вам не надо ничего менять в закладке "Gaming Options" панели управления для использования оси в цифровом режиме – то, что нужно было делать со старыми джойстиками TM.

7. Цифровые выражения 1 типа

Их синтаксис изменился – см. соответствующий раздел Руководства.

8. Отсутствие РУД

Если вы составили файл для комбинации РУС и РУД, и отключили РУД, то из всех выражений с ключами /U, /M, /D будут использованы только выражения с ключом /M, а /U, /D будут проигнорированы, равно как и выражения для осей РУД.

9. Символы, запрещенные в именах макрокоманд

В именах макрокоманд **НЕЛЬЗЯ** использовать следующие символы:

= < > { } ( ) ^ , *пробелы*

10. RPT

Если есть макрокоманда: Macro1 = a b c

и выражение: BTN S2 RPT (3) Macro1

то при нажатии S2 получим: a a a b c

Чтобы измежать этого, заключите в круглые скобки либо имя макрокоманды, либо символы в ее определении, то есть:

BTN S2 RPT (3) (Macro1)

или

BTN S2 RPT (3) Macro1

Где: Macro1 = (a b c)

11. Символ комментариев //

При программировании цифровых чипов SWF Digital Chips можно было использовать двойную косую черту // вместо команды REM. Это более не поддерживается.