THRUSTMASTER

HOTAS COUGAR

MANUALE DI RIFERIMENTO

Versione 1.4a (IT)

INTRODUZIONE

Ciao!

Prima di tutto, CONGRATULAZIONI e GRAZIE per aver acquistato l'HOTAS Cougar di Thrustmaster! La potentissima ed incredibilmente precisa periferica che adesso hai tra le mani è il fiero risultato di due anni di studi e ricerche guidate dal profondo desiderio di creare una periferica al vertice della gamma e che soddisfi le aspettative dei più esigenti. Ed eccolo qui – osservate il degno successore della serie FLCS e F-22 PRO! Molto del lavoro fatto per creare un successore degno della precedente serie di periferiche è sembrato divertente, dare vita a questo fantastico prodotto si è dimostrato molto più che semplicemente… impegnativo – e questo è riduttivo!

Quando questo folle progetto iniziò, Thrustmaster si trovò davanti a due scelte: aggiornare i joystick esistenti, mantenendone e migliorandone le capacità, oppure, iniziare da zero e inventare un prodotto completamente nuovo. Chiaramente, scegliendo la seconda via, Thrustmaster ha anche scelto la via più difficile. Per quale ragione? Semplicemente perché se Thrustmaster avesse deciso di rilasciare un nuovo HOTAS, il nuovo joystick sarebbe stato di qualità superiore e avrebbe superato ogni limite che gli standard potessero definire – quindi, con l'HOTAS Cougar, arrivò il momento di rivedere la definizione del joystick hardcore.

Il risultato di infinite riunioni e notti insonni è una periferica estremamente realistica, versatile, flessibile ed enormemente potente - oltre ogni dubbio, con caratteristiche come le barre intercambiabili, la potenza della programmazione (nonostante la semplicità), la semplicità del vero plug'n'play, insieme all'enorme numero (e peso) delle parti metalliche rendono l'HOTAS Cougar il vero punto di riferimento per gli anni a venire.
Questo è anche un successo per l'intera comunità della simulazione di volo – credeteci, questa meraviglia non è un'economica, elegante, leggera contrazione ottimizzata per i coin-op del passato, ma una replica fedele dei controlli dell'F-16, studiata per chi vuole niente più che il meglio – la periferica che farà si che i rivali ed i nemici desiderino aver scelto una carriera da personale di terra.

Naturalmente, quest'opera di ingegno non avrebbe mai visto la luce senza il supporto e l'incoraggiamento che abbiamo ricevuto dalle persone incontrate durante le fiere, che hanno parlato con noi sui forum e hanno partecipato a discussioni tramite e-mail – quindi, a tutti voi, grazie per averci aiutati! Non dimentichiamo le persone che meritano un ringraziamento speciale da parte nostra; primi tra tutti, lo staff di Thrustmaster/Guillemot che ha lavorato sul progetto ed i beta tester, che hanno fatto un ottimo lavoro alla ricerca di tutti quei bug e lavorato con tutto quel materiale pericoloso a rischio della propria vita… Un ringraziamento anche ai nostri amici e alle nostre famiglie, che hanno tenuto duro con noi per tutto il tempo in cui abbiamo lavorato su questo progetto!

Ok, basta con i sentimentalismi, arriviamo al punto!

Se hai esperienza con i precedenti prodotti Thrustmaster per la simulazione di volo, il potente HOTAS Cougar t'impressionerà. Alcune caratteristiche potrebbero sembrare identiche ai predecessori, ma non farti ingannare – è tutto nuovo, e da smaliziato pilota virtuale, tu, meglio di chiunque altro, sarai la persona che trarrà vantaggio da tutto il lavoro che abbiamo fatto per offrirti tutto ciò.

Con questa nuova uscita, abbiamo migliorato e innovato tutti i punti – componenti meccanici, elettronici e software. Nonostante offrire una periferica più usabile che un joystick vecchio di sette anni non sia stato eccessivamente difficile, quello che abbiamo realizzato è studiato per dare ad ogni giocatore, compresi i principianti, la possibilità di arrivare al meglio. Sicuramente, l'uscita di Microsoft Windows ha permesso l'introduzione delle periferiche HID – joystick plug'n'play che non necessitano di particolari configurazioni per funzionare. Bene, questo è esattamente quello che l'HOTAS Cougar ha in serbo per voi – connettilo e gioca ai tuoi giochi preferiti… è semplicissimo.

Bene, queste sono le buone notizie – ma un sacco di voi, che hanno comprato l'HOTAS Cougar per le capacità di programmazione avanzata che possiede, si aspettano molto di più da noi che semplicemente una grande periferica. Per pareggiare l'ineguagliabile precisione che possiede, l'HOTAS Cougar offre le stesse caratteristiche di programmabilità dell'F-22… e molto, molto altro, così come imparerai da questo manuale di riferimento. Le caratteristiche di programmabilità di cui parleremo più avanti sono così potenti e ampie che ti permetteranno di ottimizzare tutte le tue configurazioni per ottenere il meglio di ogni gioco, oppure per correggere gli errori o le funzionalità non previste nel simulatore. La soluzione migliore è quindi, quella di leggersi attentamente ed accuratamente questo manuale.

Come abbiamo detto sopra, questo documento è quello che definiamo come "Manuale di Riferimento" piuttosto che il solito "Manuale Utente". Chiunque abbia usato o visto i manuali degli F-16 FLCS, TQS e F-22 PRO apprezzerà la differenza con questa guida. Essa contiene tutto ciò che avrai bisogno di sapere dalla configurazione della periferica, il pannello di controllo, le basi della programmazione del Cougar, fino alle informazioni dettagliate su tutti gli aspetti che differenziano il Cougar da tutte le altre periferiche programmabili. Per integrare questo manuale esistono file di help dettagliati, wizard, tutorial ed altre utili ed intuitive applicazioni all'interno del software di programmazione. Alla fine, pensiamo di aver fornito la più ampia documentazione che sia mai stata fornita con una periferica.

Senza illusioni sull'HOTAS Cougar e sulla documentazione – questo è lo stato dell'arte delle periferiche. E questa guida ha parti che sicuramente capiterà di rileggere più di una volta. Una delle ragioni per cui questo manuale è così completo è che siamo stati avvisati che con le precedenti periferiche della Thrustmaster, le persone hanno trovato il manuale troppo sbrigativo e quindi troppo difficile familiarizzare con le periferiche. Ad ogni modo, tu troverai che questo manuale è molto facile da consultare e introdurrà i concetti in maniera semplice. Sarà utile a chi vorrà una conoscenza di base sulle funzionalità, così come per chi vorrà approfondire gli argomenti. Qualunque sia il tuo livello, abbiamo pensato a te :) Dunque, se stai usando l'HOTAS Cougar per la prima volta, ti suggeriamo di leggere con calma la prima parte della guida dall'HOTAS Cougar. L'HOTAS è un dispositivo flessibile, ci sono molte possibilità e tool per ottenere lo stesso risultato, ma dovrai programmarlo ,e su questo terreno, più sarai metodico, meglio sarà per te.
Bene – ora tocca a te! Leggi e impara ad ottenere il massimo da questa periferica e dimostra agli avversari cos'è l'HOTAS Cougar!

Per la traduzione italiana di questo manuale, visitate:
http://thrustmaster.it - http://www.amvi.it

Per la traduzione spagnola di questo manuale, visitate:
http://www.escuadron111.com

Per la traduzione francese di questo manuale, visitate:
http://www.checksix-fr.com/

Per la traduzione olandese di questo manuale, visitate:
http://thrustmaster.vanree.net/

Per la traduzione tedesca di questo manuale, visitate:
http://www.thrustmaster-x-files.de/

Per la traduzione russa di questo manuale, visitate:
http://www.hotas.ru

RINGRAZIAMENTI

I nostri più profondi ringraziamenti vanno a tutte le persone e siti internet per il loro aiuto e supporto durante lo sviluppo di questo impegnativo progetto – un saluto a tutti ☺

Beta tester

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

Gruppi e squadriglie

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)

Ringraziamenti speciali

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
Oleg Maddox
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
Stanislav "huMMer" Vartanian

Thrustmaster Italia ringrazia Egon “Rider” Carusi, Maurizio “Jester” Massasso e Francesco “MIX” Missarino e tutti i membri dell’ A.M.V.I. per aver collaborato alla traduzione del manuale.

Bel lavoro ragazzi!!

THRUSTMASTER

HOTAS COUGAR

PANNELLO DI CONTROLLO (CCP)

Introduzione

L'HOTAS Cougar è virtualmente un joystick senza driver, cioè che non richiede alcun programma in background per avere diversi settagli sugli assi o per ottenere emulazioni di alcun tipo. Per questa ragione è importante che il joystick conosca le variazioni sugli assi e, comunque, altre applicazioni come il CTFJ (Centering Tool For Joystick – tool di centratura per joystick – di Bob Church) e la procedura di centratura di windows non vengano usate – viceversa i risultati saranno imprevedibili.

NOTA: NON USARE LA CALIBRAZIONE DI WINDOWS PER CALIBRARE L'HOTAS COUGAR. USARE IL PANNELLO DI CONTROLLO DELL'HOTAS COUGAR (HOTAS COUGAR CONTROL PANEL) PER CALIBRARE MANUALMENTE .

Il pannello di controllo dell'HOTAS Cougar (CCP) è usato per variare diversi parametri dell'HOTAS Cougar, dalla mappatura degli assi alle diverse modalità di output (modo di emulazione, modo Windows…). Ciò che troverete in questo manuale di riferimento sarà una spiegazione dettagliata, pulsante per pulsante dell'applicazione e come i cambiamenti su tali pulsanti si rifletteranno sul funzionamento del joystick.
Per visualizzare le impostazioni che sono attualmente attive sul joystick, aprire il CCP con il joystick connesso.

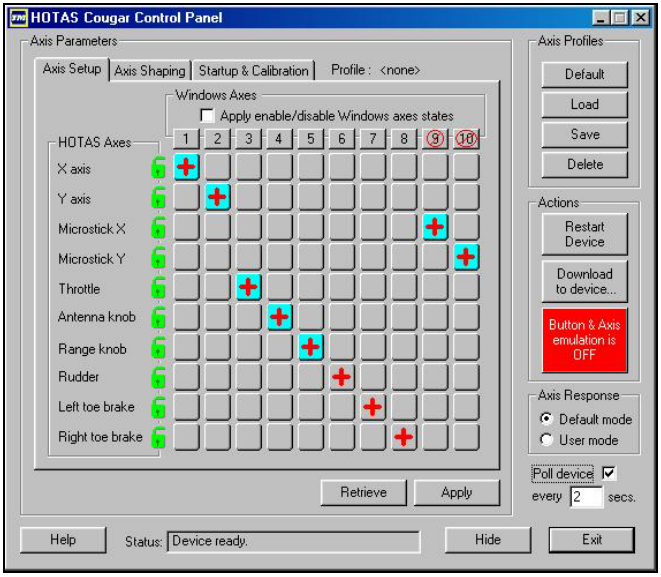


Figura 1: Il pannello di controllo nella configurazione di default

Profili degli Assi

I profili sono usati per caricare velocemente configurazioni predefinite nel CCP e nel joystick. Nella sezione Profiles del CCP, troveremo quattro pulsanti: “Default”, “Load”, “Save” and “Delete”, le cui funzioni saranno spiegate qui sotto.

Default (Predefinito)

Premendo il pulsante Default verranno visualizzati i parametri predefiniti che saranno usati nel modo Windows (Windows mode). Ciò include la mappatura degli assi secondo gli standard Windows, la maggior parte di essi con direzione positiva, dead zone superiore ed inferiore del 5%, dead zone centrale del 7%, curva lineare (0), base della curva centrata e il pulsante “apply enable/disable Windows axes states” su disabled (abilita/disabilita stato Windows degli assi su disabilitato). Se con il joystick connesso capitasse di avere una schermata del CCP senza alcun senso, la soluzione migliore è quella di premere il pulsante Default e modificare la situazione predefinita.

Save (Salva)

Premendo il pulsante Save si salverà la configurazione correntemente definita nella sezione Axes Parameters (Parametri degli Assi) nella cartella HOTAS/Profiles in file con estensione ‘.TMC’ (Thrustmaster Configuration). Anche un’eventuale calibrazione sarà salvata in tale file. Salvando i profili in questo modo significa che diverse impostazioni possono essere salvate per diversi giochi e ricaricate in futuro.

Load (Carica)

Premendo il pulsante Load si caricherà una configurazione in precedenza salvata nell’applicazione grafica (formata dalle varie cartelle del CCP). Ciò non significa caricare il file nel joystick, operazione che sarà possibile premendo il tasto Apply (Applica) trattato più avanti in questo Manuale di riferimento.

Delete (Cancella)

Questo pulsante è usato per cancellare i profili salvati.

Modalità del Joystick

Il CCP permette di configurare i modi di funzionamento del joystick relativi agli Axis Parameters (Parametri degli Assi), Calibration Options (Opzioni di Calibrazione), and Emulation features (Funzioni di Emulazione). Impostare questi modi non richiede la pressione del tasto Apply visto che ogni variazione della modalità di risposta degli assi (Axis response mode) verrà segnalata automaticamente al joystick.

Axis Response (Risposta degli Assi)

Nella modalità predefinita, il joystick userà configurazioni e trasformazioni predefinite per gli assi. Questa modalità è sempre presente nel joystick e viene usata ogni volta che esso viene collegato con l'esclusione con i pulsanti numerati degli assi di cui l'ultimo download dovrebbe aver abilitato la visualizzazione (questo è spiegato meglio nella sezione "Modificare il Windows Axes States"). Quando la risposta degli assi viene impostata su User Defined Mode (Modalità Definita dall'Utente), il joystick userà i seguenti dati:

- Mappatura degli Assi
- Dati di Inversione
- Dati di Blocco
- Informazioni sulla Curva
- Origine della curva
- Informazioni sulla Dead Zone
- Impostazioni del Trim

Per informazioni sulle opzioni di calibrazione, riferirsi alla sezione Calibratura.
Per informazioni sulle modalità di Emulazione, riferirsi alla sezione Emulazione di Pulsanti ed Assi.

Assi e Parametrizzazione

Il CCP contiene tre cartelle: Axis Setup (Configurazione degli assi), Axis Shaping (Trasformazione degli assi) e Startup & Calibration (Inizializzazione e Calibrazione). Cambiando i valori all'interno di queste cartelle, non ci sarà alcuna variazione immediata sul joystick visto che ogni variazione dovrà prima essere applicata al joystick e la Risposta degli assi impostata su User mode. Per poter proseguire con le spiegazioni, aprire il CCP e premere sul pulsante Default.

Pulsanti di Axis Parameter

La sezione Axis Parameters (esclusi le sopraccitate cartelle) contiene due pulsanti: “Retrive” e “Apply”. Il loro uso è spiegato nelle sezioni successive.

Apply (Applica)

Il pulsante Apply permette di scaricare la configurazione descritta in Axis Parameters sul joystick. Finché non sarà eseguita quest’operazione, il joystick non saprà nulla di ciò che verrà modificato nelle cartelle Axis Parameters. Il pulsante Apply scaricherà tutto ciò che è presente nelle cartelle. Affinché il joystick sia in grado di usare questi dati, l’Axis Respinse va impostato su User Mode, operazione che viene eseguita in automatico cliccando il pulsante Apply.

Retrieve (Recupera)

Il pulsante Retrive è usato per recuperare la configurazione che è presente sul joystick. Qualunque variazione nella modalità di emulazione verrà recuperata usando il pulsante Retrive e verificando l’apposita sezione. Il CCP esegue un’operazione di recupero ad ogni avvio per visualizzare la configurazione corrente del joystick; se il joystick non è connesso viene visualizzato un messaggio d’errore e viene visualizzato il profilo predefinito.
Premendo il pulsante Retrieve è possibile ottenere il modo corrente di funzionamento del joystick.

Cartella Axis Setup (Configurazione Assi)

Nella cartella Axis Setup sono presenti molte funzioni; il loro scopo può essere riassunto così:

- Variazione della configurazione degli assi
- Inversione della direzione degli assi
- Blocco dei valori degli assi
- Cambiamento degli assi riconosciuti da Windows

Queste funzioni sono spiegate nelle rispettive sezioni a seguire.

Variazione della configurazione degli assi

Se è necessario che un asse controlli direttamente un altro asse in una configurazione, la strada più semplice è quella di modificare l'Axis Setup.
Se caricato nella sua forma predefinita solo con il TQS (manetta) collegato, l'Axis Setup apparirà così:

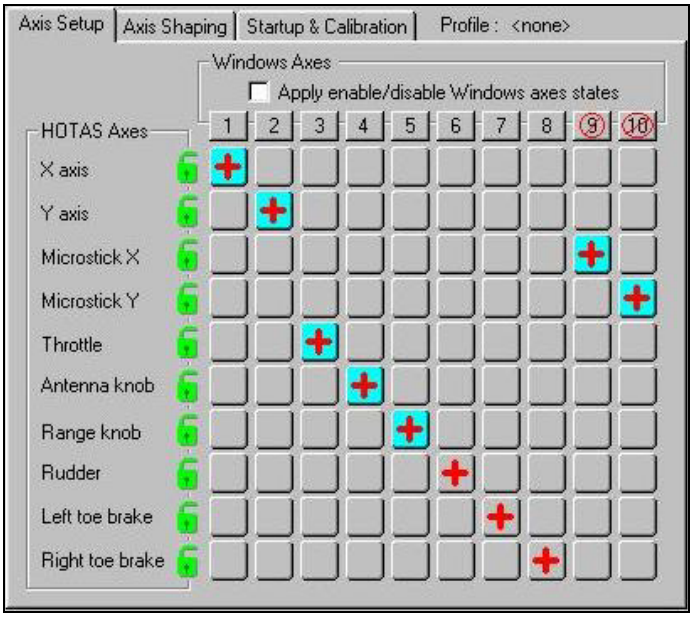


Figura 2: Cartella Axis Setup nella configurazione predefinita

I numerosi pulsanti presenti al centro rappresentano la mappatura corrente; la figura illustra che l'asse X (X axis) è mappato sul primo asse, l'asse Y sul terzo, la manetta sul secondo, ecc… Se volessimo che il secondo asse (l'asse Y visto da DirectX) venga controllato dalla manetta (tipico delle simulazioni automobilistiche), dovremo semplicemente premere il pulsante sulla riga della manetta (Throttle) e la colonna contrassegnata dal numero due e la finestra assumerà questo aspetto:

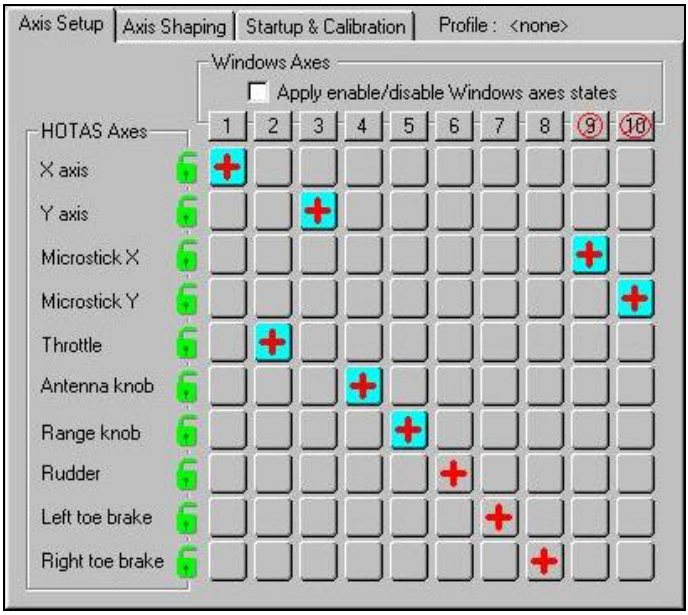


Figura 3: Axis Setup con la manetta mappata sull'asse Y

Come si può notare la manetta è stata assegnata alla posizione numero due e l'asse Y ha preso il posto della manetta sull'asse numero tre. Tutti gli scambi di assi sono eseguiti in questo modo e anche se Windows "vedrà" gli assi rimappati in questo modo, tutta la programmazione degli stessi all'interno del joystick (incluse inversioni, curve, dead zone, emulazioni e programmazioni) rimarrà legata all'asse fisico; in pratica, se sulla manetta viene impostata una curva specifica e poi rimappata sull'asse Y (DirectX), la curva apparterrà sempre alla manetta (fisicamente) mentre Windows ne leggerà i valori come se provenissero dall'asse Y.

Il colore di sfondo delle colonne numero sei, sette ed otto è grigio diversamente dagli altri (che sono in azzurro); ciò è dovuto al fatto che in questa configurazione, il RCS (pedaliera e freni sui pedali) non è connesso. I pulsanti con lo sfondo azzurro indicano che gli assi sono fisicamente connessi. Per far si che gli assi X e Y vengano controllati dal Microstick, l'Axis Setup dovrà essere impostato così:

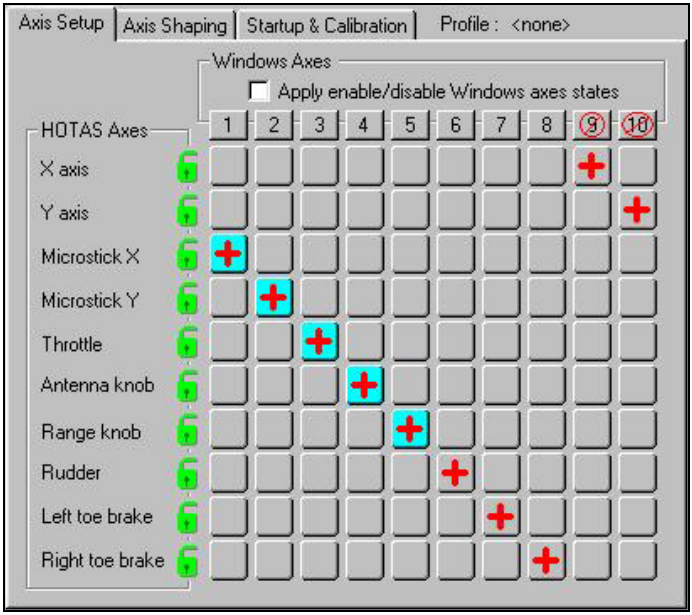


Figura 4: Axis Setup con Microstick associato agli assi X & Y

Invertire la direzione di un asse

Per invertire la direzione di un asse basta premere un pulsante che presenta il segno più (o meno) e la direzione dell'asse verrà invertita. Nel caso in cui si volesse cambiare la direzione dell'asse Y, l'Axis Setup apparirebbe così:

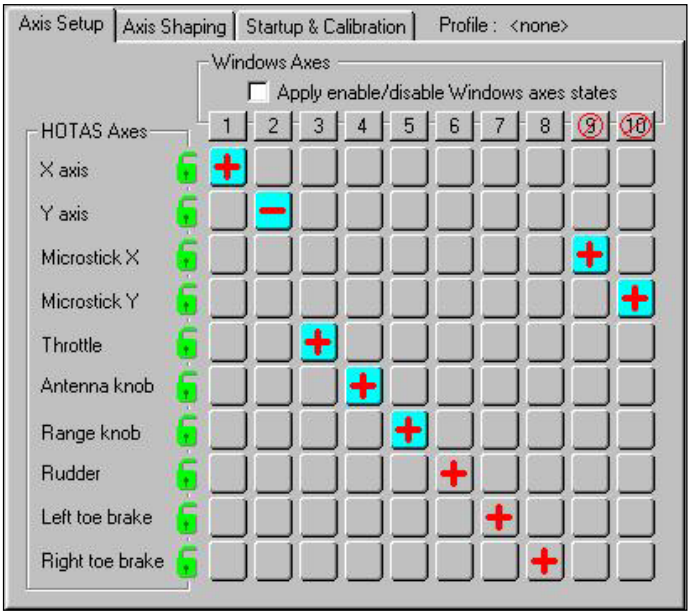


Figura 5: Axis Setup con l'asse Y invertito

Bloccare un Asse

Sulla sinistra di ogni asse si trova un'icona a forma di lucchetto. Se il lucchetto risulta aperto e di colore verde, l'asse è sbloccato e si comporterà normalmente. Se il lucchetto è chiuso e di colore rosso, il relativo asse è bloccato e il suo valore non può esser modificato. Per variare il blocco di un asse è sufficiente premere sull'icona del lucchetto ed esso varierà tra i due stati appena descritti.

Modificare il Windows Axes States

Questo titolo potrebbe non risultare chiarissimo, ma cercheremo di spiegarlo senza andare troppo nel tecnico. Sopra ogni colonna esiste un pulsante. Nel modo di Default, i pulsanti 9 e 10 hanno un cerchio barrato rosso disegnato sopra il numero. Questi pulsanti sono rappresentati qui sotto:

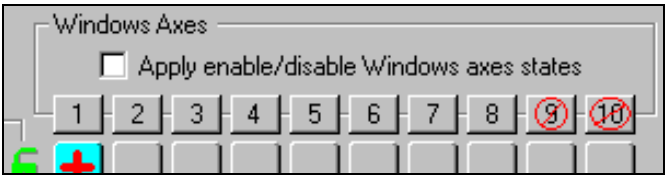


Figura 6: I pulsanti numerati degli assi in Axis Setup

Ognuno di questi pulsanti rappresenta un diverso asse DirectX, questi sono:

Numero dell'asse	Nome DirectX
1	Asse X
2	Asse Y
3	Asse Z
4	Asse Rotativo X
5	Cursore 0
6	Asse Rotativo Z
7	Cursore 1
8	Asse Rotativo Y
9	<non disponibile>
10	<non disponibile>

Tabella 1: Numeri degli assi e nomi DirectX

Oltre a non poter abilitare gli assi 9 e 10, è altresì impossibile disabilitare i primi due in quanto un joystick prevede che almeno questi due esistano. Se si clicca su uno dei pulsanti numerati da 3 ad 8, il cerchio rosso apparirà o scomparirà. Quando il joystick è collegato, i nomi degli assi, così come il numero degli assi "visti" da Windows sono determinati da:

1. L'attuale configurazione fisica del joystick
2. Lo stato di questi pulsanti numerati

La spiegazione di queste due voci si trova nella pagina seguente.

Assi e configurazione fisica

Cosa significa configurazione fisica del joystick? La configurazione fisica è rappresentata da altre periferiche collegate al joystick che in pratica definiscono il numero di assi che il Cougar presenterà a Windows. Le configurazioni fisiche possibili per l'HOTAS sono 6:

1. Il joystick è collegato da solo
2. Il TQS è collegato al joystick
3. Il joystick è collegato ad un RCS (versione da 1 asse)
4. Il joystick è collegato al nuovo RCS (versione da 3 assi)
5. Il joystick è collegato al TQS e al RCS
6. Il joystick è collegato al TQS e al nuovo RCS

Ognuna di queste possibilità verrà rappresentata da una diversa combinazione degli assi in Windows. La tabella qui sotto rappresenta le sei configurazioni elencate e gli assi che saranno disponibili per ognuna.

Configurazione	NOMI DEGLI ASSI							
	X	Y	Z	R_x	SL_0	R_z	SL_1	R_y
1	●	●						
2	●	●	●	●	●			
3	●	●				●		
4	●	●				●	●	●
5	●	●	●	●	●	●		
6	●	●	●	●	●	●	●	●

Assi e "Enable Windows Axis States"

Usando i pulsanti numerati, è possibile cambiare gli assi che saranno riconosciuti da Windows. La procedura per abilitare questi pulsanti è la seguente:

1. Selezionare gli assi che si vorrebbe far riconoscere da Windows usando i pulsanti numerati fino a raggiungere la configurazione desiderata.
2. Selezionare “Apply enable/disable Windows axes states”.
3. Caricare la configurazione nel joystick premendo il pulsante “Apply” e poi “OK” per eseguire un restart del joystick.
4. Il joystick verrà visto da Windows, con le caratteristiche specificate.

Se non si possiede una pedaliera RCS o la nuova RCS (con freni sui pedali), si può far credere alle DirectX che entrambi questi dispositivi siano connessi e controllare i valori d’ingresso con uno degli altri assi disponibili o tramite un file di emulazione.

È importante notare che il joystick caricherà la configurazione definita del modo “Enable Windows Axes States” fintanto che l’ultimo file caricato avrà il parametro “Apply enable/disable Windows axes states” attivato**.**

Axis Shaping (Curva dell’asse)

La cartella Axis Shaping contiene dei controlli per la configurazione avanzata degli assi e delle preferenze dell’utente. La tendina “Axis to set” in testa alla maschera seleziona l’asse desiderato nella lista dei dieci disponibili. Una volta che un parametro viene variato, il cambiamento viene rappresentato dal grafico disposto alla destra della lista dei parametri. La variazione del grafico è scatenata variando i parametri stessi oppure premendo il pulsante Refresh (Aggiorna) posto nell’angolo in basso a destra. I parametri modificabili sono:

- Upper Dead Zone o UDZ (zona morta superiore)
- Lower Dead Zone o LDZ (zona morta inferiore)
- Center Dead Zone o CDZ (zona morta centrale)
- Calibration Center (centro di calibrazione)
- Axis Trim (trim dell’asse)
- Curve Settings (curva) – fattore e base

I parametri qui elencati possono essere abilitati per selezionare diverse modalità di uscita del joystick.

Dati sulle zone morte

Variando i dati sulle zone morte (UDZ è la zona morta superiore, LDZ è la zona morta inferiore, CDZ è la zona morta centrale), è possibile alterare le regioni inattive di ogni asse. L'effetto dei valori impostati è visualizzato sul grafico presente a destra dell'area dove i parametri vengono inseriti; le zone morte vengono visualizzate in rosso. Tutti i valori hanno un massimo del 100%, dove il 100% è pari al 30% del movimento del joystick. Per esempio, il valore predefinito è pari all'1% per ogni zona morta, che si traduce nel 3% del movimento totale dell'asse. Ecco un esempio grafico di questa cartella:

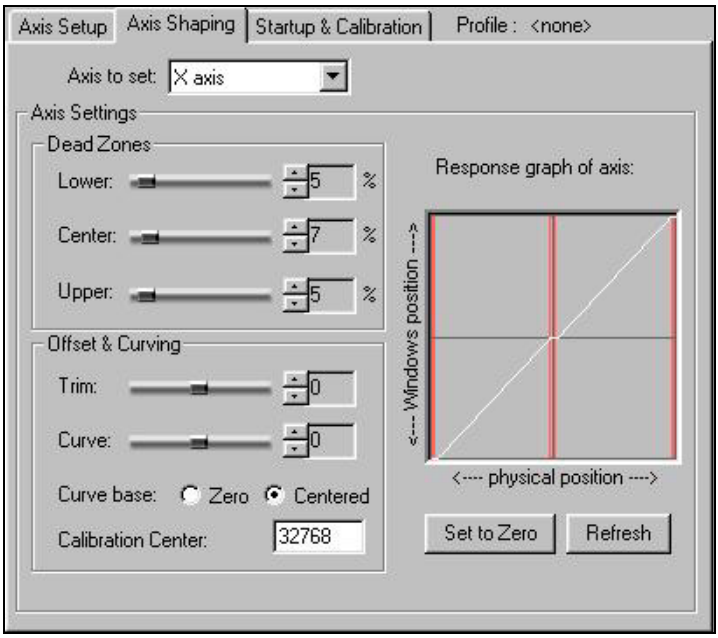


Figura 7: Axis Shaping con valori predefinti

Nella figura qui sopra, le piccole aree rosse appaiono sui bordi e al centro. Il grafico mostra la relazione tra la posizione fisica dell'asse ed i valori forniti a Windows. Per meglio chiarire, l'asse orizzontale del grafico mostra i valori che il joystick raccoglie dal suo asse con il minimo sulla sinistra. L'asse verticale del grafico rappresenta il valore fornito a Windows con il valore minimo in basso. Nelle zone morte il grafico si appiattisce, rappresentando che in quell'area i valori non varieranno al variare della posizione fisica dell'asse. La zona morta inferiore è rappresentata a sinistra (linea orizzontale nell'area rossa), la zona morta superiore si trova nella parte destra. La figura seguente illustra cosa accade variando il valore della CDZ:

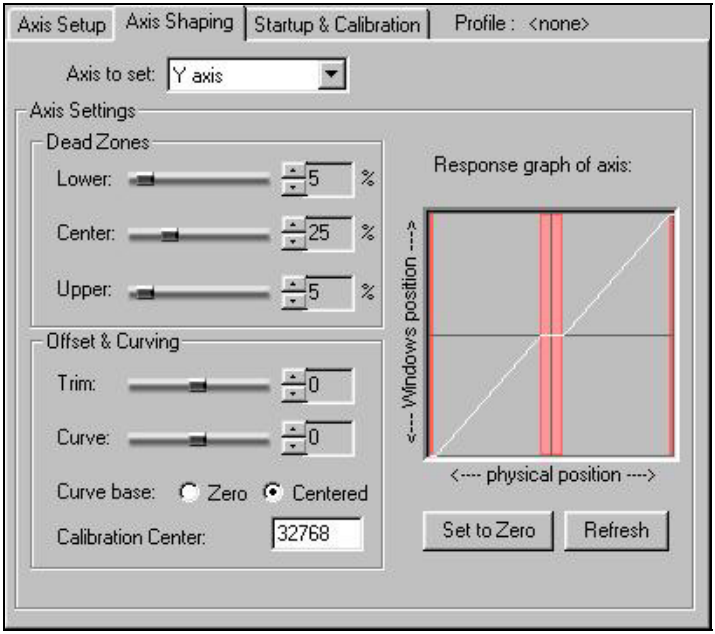


Figura 8: Axis Shaping con CDZ dell'asse Y aumentato

La figura 8 mostra chiaramente che aumentando la CDZ dal 7% al 25% si aumenta proporzionalmente la banda rossa al centro dell'asse orizzontale e la riga bianca orizzontale al suo interno rappresenta la regione dove il joystick sarà inattivo. La UDZ e la LDZ lavorano in maniera simile.

Calibration Center (Centro di calibrazione)

Questo valore è la posizione che il joystick considererà come centro dell'asse. Se la posizione specificata è inferiore a quella fisica del joystick, allora l'asse avrà un valore più elevato del normale quando si troverà a riposo (o al centro). Un risultato simile si può ottenere variando il valore del Trim, che aggiunge un valore costante (positivo o negativo) alla lettura del joystick; il vantaggio principale del Trim rispetto alla posizione centrale è che il valore del Trim può essere alterato durante l'emulazione.

Trim dell'asse

La funzione di Trim è usata per spostare il valore fisico attuale di un asse ad un nuovo valore. Per esempio: se l'asse del joystick è in una posizione fisica del 30% mentre il trim è impostato al -20%, allora la posizione risultante sarà pari a 10%. I valori limite del Trim sono +/- 50%; ciò significa che è possibile raggiungere con la posizione centrale gli estremi del grafico. Questo è dimostrato dalla figura seguente.

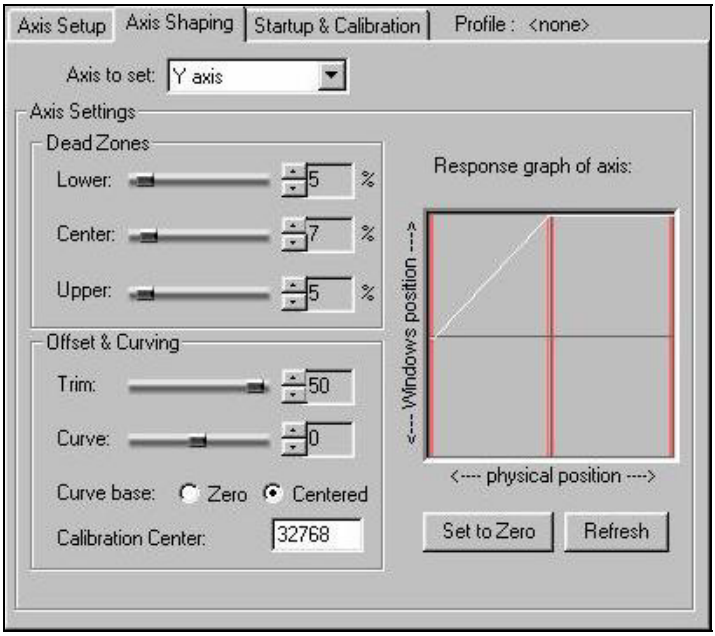


Figura 9: Trim al massimo

Con il Trim applicato all'asse Y, la reazione può essere osservata immediatamente, lasciando l'asse al suo centro fisico (la metà dell'asse orizzontale del grafico), i dati in uscita che Windows riceverà saranno pari al massimo valore per l'asse Y. Se l'asse verrà mosso nella direzione in cui i valori dovrebbero aumentare, i valori d'uscita non subiranno variazioni; se l'asse verrà mosso nella direzione opposta, si potrà notare che i valori diminuiranno, ma raggiungeranno un minimo pari a quello della normale posizione centrale.

Se si riducesse il Trim al suo valore minimo, il grafico sarebbe:

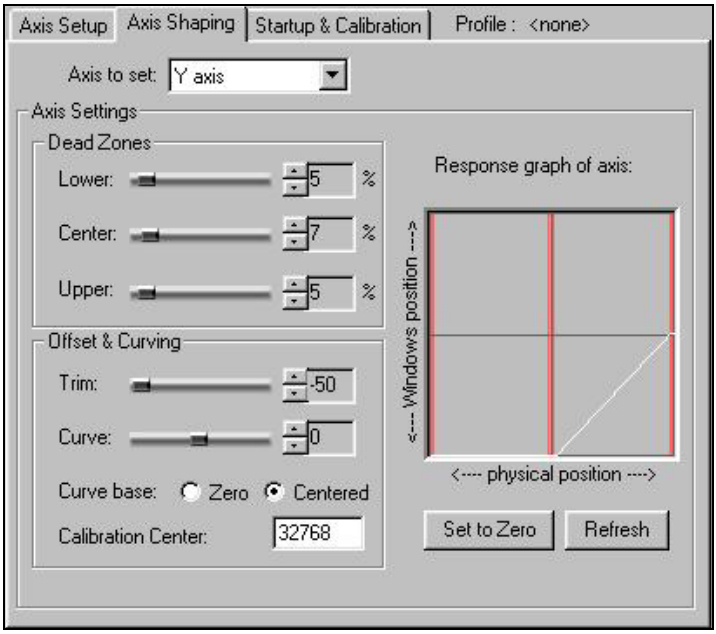


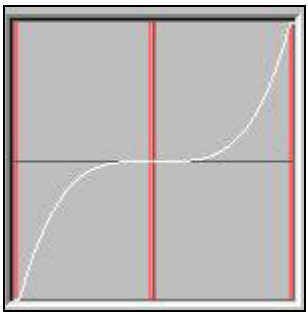
Figura 10: Trim al minimum

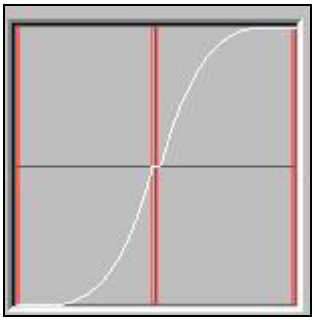
Applicando questa modifica, l'asse Y sarà letto al minimo quando la barra sarà tenuta in posizione centrale sull'asse Y e verrà letto in posizione centrale quando la barra verrà spinto al suo massimo.

Impostazioni di Curve (Curva)

Questo parametro può essere variato per ottenere un asse che risponda in maniera più o meno esponenziale variando la curvatura. Il parametro Curve può assumere valori da –32 a +32, ma impostare diversi assi a valori elevati di curvatura rallenterà notevolmente il joystick. Valori più alti di 20 o minori di –20 sono praticamente inutili e la variazione di curvatura oltre tali limiti non giustifica l'uso di tali valori.

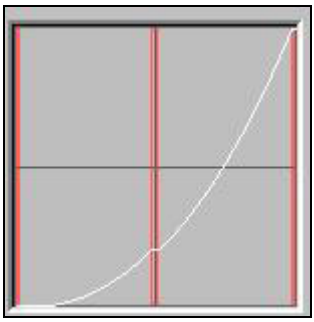
Inizieremo ad osservare i grafici con base centrata, quale può essere la differenza di una curvatura positiva rispetto ad una negativa? Ecco la risposta quando il parametro Curve viene impostato a valore 10 positivo e negativo.

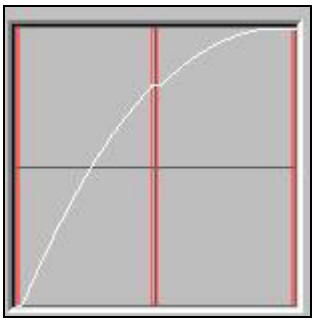




La figura a sinistra rappresenta l'asse con una curvatura di –10; come si può vedere, intorno alla posizione centrale, ci sono cambiamenti minimi nell'uscita, quasi come se la zona morta centrale fosse stata allargata. La pendenza aumenta progressivamente fino ad ottenere un grafico quasi verticale. Ciò significa che l'asse sarà molto sensibile vicino al limite fisico dell'asse. La figura a destra rappresenta il grafico dell'asse con una curvatura di +10. Il risultato è molto simile al precedente, sebbene invertito; l'asse è molto più sensibile vicino alla posizione centrale rispetto agli estremi in cui sembra che le due zone morte (superiore ed inferiore) siano state allargate.

Sebbene la risposta del joystick e del microstick debba essere curva come negli esempi precedenti, l'idea di 'Base of Curve (base delle curva) è che una rampa unica sia più appropriata per assi come i freni e la manetta. Ecco i grafici per una base di curva impostata a Zero (zero) con valori di curvatura 5 positivo e negativo (in questo caso un valore 10 è eccessivo).





La figura a sinistra rappresenta il grafico per un asse con curvatura –5, il risultato è molto simile al quadrante superiore sinistro dell'esempio precedente (curvatura –10, base centrata). Il joystick adesso reagirà molto lentamente al limite inferiore dell'asse e la sensibilità aumenterà fino a raggiungere il picco massimo al limite superiore dell'asse. Sulla destra si trova il grafico del valore +5 e si presenta la stessa similitudine con il caso precedente. La risposta aumenterà molto velocemente vicino al limite inferiore e sarà molto meno ripida avvicinandosi al

limite superiore. Vicino al limite superiore la forma del grafico assomiglia al caso di una zona morta superiore estesa.

Startup & Calibration (Avvio e Calibrazione)

La cartella 'Startup & Calibration' contiene informazioni su come il joystick si comporterà all'avvio del computer e sulla calibrazione degli assi. Questa cartella è divisa in due sezioni, la parte superiore è dedicata ai parametri di avvio, mentre quella inferiore è dedicata alla calibrazione. Entrambe verranno descritte qui sotto.

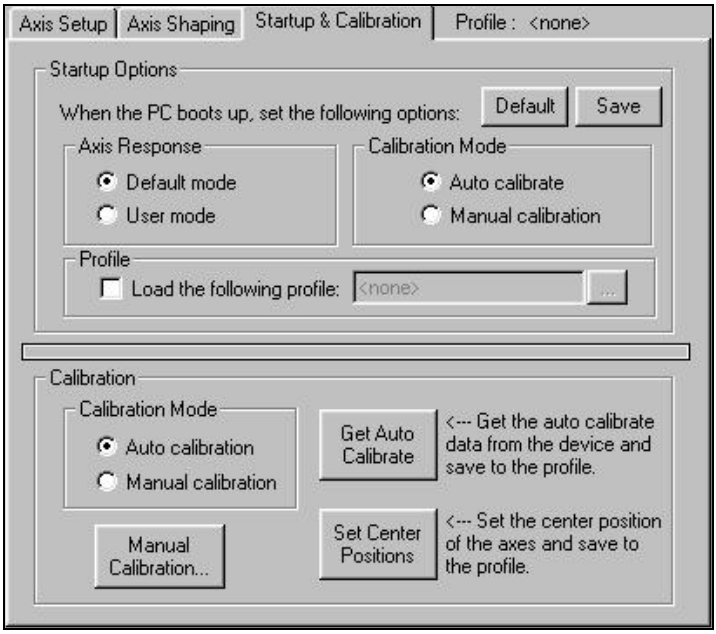


Figura 11: Startup & Calibration

Startup Options (Parametri di avvio)

Ci sono tre parametri che il joystick può configurare autonomamente all'avvio del computer. Questi parametri sono: la risposta degli assi, il modo di calibrazione e il caricamento di un profilo specifico. L'impostazione predefinita è in Default mode (predefinito) per l'Axis Mode, in Auto Calibrate (calibrazione automatica) per il Calibration Mode (modo di calibrazione) e con l'ultimo profilo caricato nel joystick attivo. Per cambiare le impostazioni predefinite, premere sulle opzioni desiderate e poi sul pulsante 'SavÈ. Per selezionare un profilo, premere sul segno di spunta

vicino a ‘Load the following profilÈ (carica il seguente profilo) e poi sul pulsante ‘…’ oppure digitare il nome del profilo nell’apposita casella (il profilo deve essere esistere nella directory Profiles della cartella HOTAS). Per riportare queste opzioni alla situazione predefinita, premere il pulsante ‘Default’.

Calibration (Calibrazione)

La sezione di calibrazione contiene le opzioni relative al modo di calibrazione, alla calibrazione del joystick, recupero dei dati di calibrazione e settaggio delle posizioni centrali degli assi.

Selezionare la modalità Auto (automatica) o Manual (manuale) nel Calibration Mode cambierà automaticamente il joystick nel modo scelto. Non è necessario premere il pulsante ‘Apply’, questo parametro viene inviato automaticamente al joystick.

Nella modalità di calibrazione manuale, i dati usati saranno quelli caricati nel joystick dopo una Manual Calibration (calibrazione manuale). Questi dati vengono inviati al joystick per l’uso, insieme ai parametri degli assi, dopo ogni esecuzione della procedura di calibrazione.

Il modo automatico di calibrazione significa che il joystick configurerà i valori massimi degli assi ogni volta che questi verranno mossi fino agli estremi. La calibrazione manuale significa che il joystick userà i valori massimi forniti dalla procedura di calibrazione manuale descritta più avanti in questa sezione. Il joystick non può essere impostato sul modo manuale della calibrazione se non è mai stata eseguita la procedura di calibrazione manuale. Se il joystick viene resettato (riconnettendolo o premendo il pulsante Restart) e si trova nel modo di calibrazione automatica, è consigliabile muovere gli assi (joystick, manetta, Range, Antenna, Microstick, ecc.) nelle posizioni massime e minime e mantenendo tali posizioni per circa 3 secondi. Questo consente alla calibrazione automatica di raccogliere i dati dagli assi e di calibrarli correttamente.

Premendo su “Get auto Calibration”, ottieni la calibrazione automatica. Prima di impostare il modo di calibrazione da automatico a manuale, i dati della calibrazione automatica vengono copiati nel profilo corrente. Ora è possibile applicare e salvare il profilo in modo che il joystick usi questi dati e non provi a modificarli.

Il pulsante Set Center Positions (imposta le posizioni centrali) serve per salvare una specifica posizione di ogni asse come la posizione centrale di quel dato asse. Se si desidera avere gli assi del joystick e del Range in posizioni centrali diverse da quelle predefinite, allora occorre premere questo pulsante e specificare gli assi desiderati come mostrato nella pagina seguente:

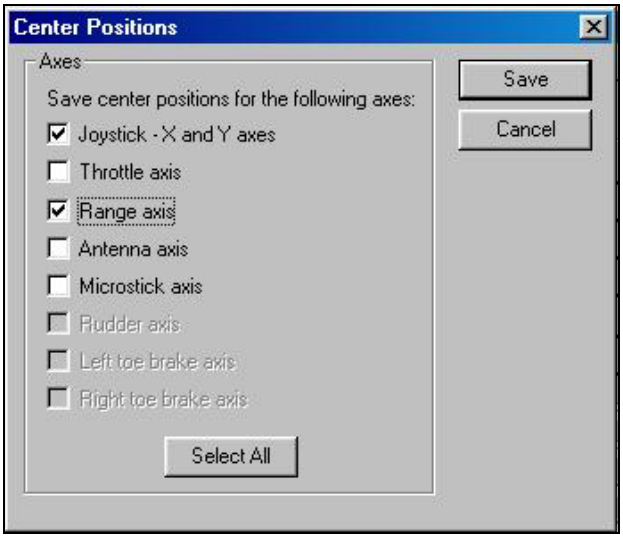


Figura 12: Memorizzazione delle posizioni centrali di joystick e Range

Una volta premuto il pulsante ‘SavÈ, verrà chiesto di muovere gli assi specificati nella nuova posizione centrale e premere ‘OK’. Questo concluderà la procedura di centramento manuale degli assi specificati.

Manual Calibration (Calibrazione Manuale)

Il pulsante Manual Calibration richiamerà la finestra Calibration Routine rappresentata qui sotto:

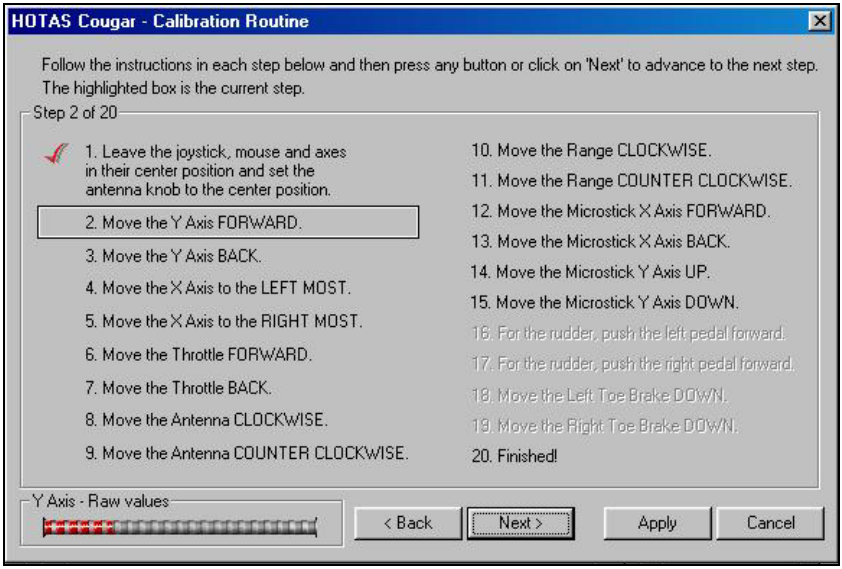


Figura 13: finestra di calibrazione manuale

Seguire le istruzioni in ordine premendo il pulsante Next quando l'asse ha raggiunto la posizione che deve essere memorizzata per quel particolare passo. Ad esempio, se il secondo passo richiede di muovere l'asse Y in avanti (al massimo), il joystick salverà la posizione che l'asse avrà alla pressione del pulsante Next. Quando la calibrazione sarà terminata, l'applicazione invierà automaticamente i dati al joystick, insieme a tutti i parametri impostati per gli assi. I dati di calibrazione vengono salvati nei profili e possono essere riusati successivamente.

I passi relativi ad assi non collegati vengono rappresentati in grigio e la procedura li salterà. La barra nell'angolo inferiore sinistro della finestra indica i valori 'grezzi' (non elaborati) forniti dal joystick. Potrebbe non essere possibile raggiungere gli estremi di tale barra muovendo gli assi, ma il suo scopo è di indicare l'asse su cui si sta eseguendo la calibrazione e la direzione di movimento.

Azioni ed altre opzioni

Nella sezione Actions, ci sono tre pulsanti: Restart Device, Button & Axis emulation e Download to device. Ci sono anche opzioni che riguardano il polling automatico del joystick e un pulsante Hide. Tutte queste azioni e opzioni sono descritte qui sotto.

Restart Device (Riavvia il dispositivo)

Il pulsante Restart Device causerà un riavvio del joystick come se il joystick sia stato scollegato e ricollegato alla porta USB. Questa funzione è utile ogni volta che viene cambiato i Windows Axes States. In questo caso, sarebbe necessario scollegare e ricollegare il joystick (come descritto nella sezione Variazione dello stato degli assi), ma questa operazione è possibile anche premendo il pulsante Restart Device. Altra condizione utile è quella che al riavvio il joystick esegue la funzione di centratura automatica, muovendo gli assi nella posizione desiderata e premendo Restart è possibile variare le posizioni misurate manualmente.

Button & Axis emulation (Emulazione assi e pulsanti)

Se il pulsante Button & Axis emulation è ON (sfondo verde), allora il joystick userà l'ultimo file di emulazione caricato al suo interno. Il file di emulazione è quello che controlla l'emulazione di mouse e tastiera, così come la variazione di diversi parametri degli assi. Per ulteriori informazioni, riferirsi al Manuale Utente.

Se il pulsante Button & Axis emulation è OFF (sfondo rosso), allora il joystick si comporterà come un semplice joystick con i pulsanti che saranno mappati sui pulsanti direct in Windows.

Download to device (Carica sul dispositivo)

Questo pulsante apre l'applicazione HOTAS Cougar Loader. L'applicazione è usata per caricare i file sul joystick (file Thrustmaster Joystick o .TMJ). Per ulteriori informazioni sulla struttura di un file di emulazione, riferirsi al Manuale Utente. Nella pagina successiva si trova un'immagine dell'applicazione HOTAS Cougar Loader.

Nella finestra dell'applicazione si trovano due pulsanti e una checkbox (segno di spunta). Il pulsante Check serve a verificare che il file non contenga errori. Tramite questo pulsante il file non verrà caricato, ma semplicemente verificato. Il pulsante Download controllerà il file e lo invierà al joystick se non verranno rilevati errori.

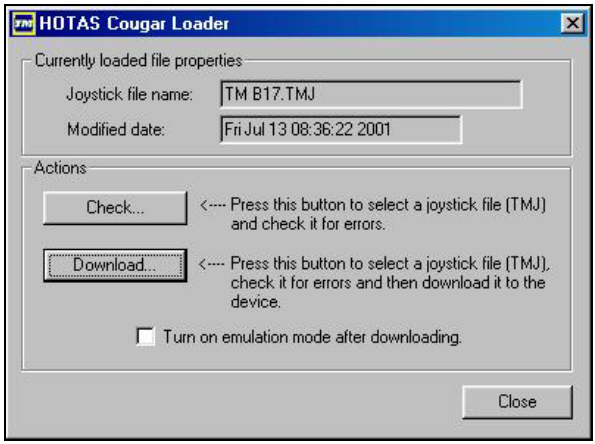


Figura 14: HOTAS Cougar Loader per caricare i file nel joystick

La sezione 'Currently loaded file properties' mostra quale file è presente nel joystick e quando è stato caricato. Il checkbox 'Turn on emulation mode after downloading' serve ad abilitare l'emulazione subito dopo il caricamento del file nel joystick. Dopo aver caricato un file nel joystick è necessario abilitare l'emulazione in modo che il joystick possa emulare la pressione dei tasti e il movimento degli assi che sono programmati nel file. Questa operazione può essere fatta anche usando il pulsante 'Button & Axis emulation' nel CCP.
Controllando o caricando il file, apparirà la seguente finestra.

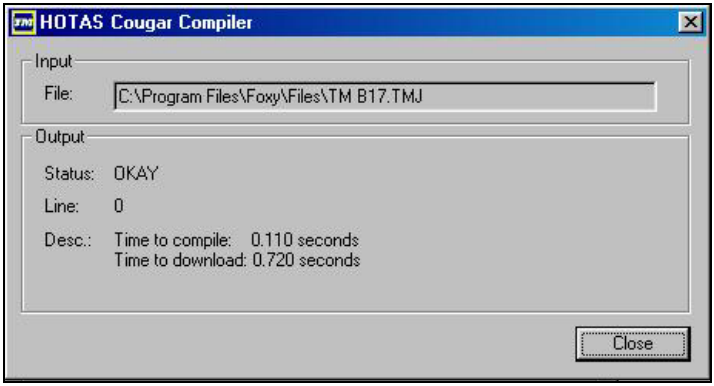


Figura 15: HOTAS Cougar Compiler per controllare i file di emulazione

Poll device

Questa checkbox serve ad abilitare la funzione di polling del CCP. La funzione di polling è quella che controlla periodicamente che il joystick sia connesso ed il suo stato. L'intervallo di tempo è modificabile nel campo sottostante il checkbox.

Hide / Taskbar Icon (Nascondi / Icona nella barra di avvio)

Il CCP può essere nascosto riducendosi ad icona nella barra di avvio di Widnows. Cicca sul pulsante Hide per nascondere il CCP. Questa funzione può essere usata solo quando il polling è abilitato. Per ripristinare la finestra dopo averla nascosta, premere sull'icona con il tasto sinistro del mouse e scegliere 'Open HOTAS Cougar Control Panel…' dal menu che apparirà. Le altre opzioni disponibili permettono di chiudere il CCP e di aprire l'HOTAS Cougar Loader. È possibile variare il modo di emulazione premendo sull'icona con il tasto destro del mouse. I colori dell'icona cambieranno seguendo questo schema:

Icona	Colore	Descrizione
	*Caccia verde / Sfondo giallo*	Emulazione attiva (**ON**), Risposta degli assi definite dall'utente (**User**)
	*Caccia rosso / sfondo giallo*	Emulazione disattivata (**OFF**), Risposta degli assi definite dall'utente (**User**)
	*Caccia verde / Sfondo grigio*	Emulazione attiva (**ON**), Risposta degli assi in modo **Windows**
	*Caccia rosso / Sfondo grigio*	Emulazione disattivata (**OFF**), Risposta degli assi in modo **Windows**

Tabella 1: Descrizione dell'icona nella barra di avvio

THRUSTMASTER

HOTAS COUGAR

MANUALE UTENTE

INDICE

1. Ecco cosa abbiamo in serbo per te!

1.1 Introduzione

Benvenuti nella nuova generazione di periferiche di alto livello di Thrustmaster – l'HOTAS Cougar. Nella confezione di questo controller, oltre allo stesso, si trova un CD-ROM contenente un set completo di utilità, il Manuale di riferimento e la tradizionale Guida d'installazione rapida.

Insieme ai file contenuti nel CD-ROM, si trova tutto il software necessario per far funzionare l'HOTAS Cougar: l'HOTAS Cougar Control Panel (descritto accuratamente nella sezione precedente), così come la suite Foxy con il programma principale, il Foxy HOTAS Cougar Edition, tutti i componenti (come il Composer e Korgy) e gli accessori come il Foxy GUI, il Launcher, ecc.

In questa sezione non verrà introdotto nulla riguardo l'HOTAS Cougar Control Panel, per tutte le informazioni su questo software basterà continuare a leggere!
Foxy e la Foxy GUI, non verranno trattati in questo manuale. Per tutte le spiegazioni necessarie, riferirsi al rispettivo help ondine.
Foxy è la chiave per accedere semplicemente alla programmazione delle funzioni del joystick, definire potenti e complicati file di macro con impareggiata semplicità.
La Foxy GUI permetterà di assegnare con precisione, con pochi click, una sequenza di tasti ad una singola azione dell'HOTAS Cougar; la programmazione delle funzioni di tastiera ai vari (e molteplici!) pulsanti della barra avverrà senza una particolare conoscenza del codice di programmazione Thrustmaster.

1.2 Impostazione del controller

Riferirsi alla Guida d'installazione rapida fornita con l'HOTAS Cougar.

1.3 FAMILIARIZZARE con il MANUALE DI RIFERIMENTO

Chiaramente, una periferica allo stato dell'arte come l'HOTAS Cougar necessita di ampia documentazione e di un supporto all'altezza del suo potenziale. Una volta installato il Cougar e tutto il software fornito sul CD, controllato il Cougar Control Panel (CCP), così come la sezione Periferiche di gioco del Pannello di Controllo di Windows, sicuramente il passo successivo sarà come iniziare con tutto questo materiale.

Iniziando con le basi, ecco il primo passo.

Livello 1: Uso di base

La prima nota positiva è che non è necessario fare nulla per poter iniziare ad usare il Cougar. Si raccomanda di muovere gli assi agli estremi della loro corsa e tenerli in quella posizione per qualche secondo per permettere alla calibrazione automatica di configurare tutto correttamente. Adesso è possibile uscire dal software del Cougar (se sta girando), aprire un gioco e usare l'HOTAS. Se il gioco permette di assegnare le funzioni agli hat ed ai pulsanti, allora si possono programmare direttamente dal gioco. Il gioco vedrà la barra e la manetta per quello che realmente sono e, normalmente, assegnerà loro le normali funzioni di volo. È possibile che il gioco riconosca anche gli assi aggiuntivi (Range ed Antenna) e gli assegni autonomamente delle funzioni. Questo è tutto, il Cougar è in quello che normalmente viene chiamato modo Windows o DirectX il che sta a significare che pulsanti e hat non sono programmati e possono essere assegnati liberamente alle funzioni del gioco.

Livello 2: Programmare il Cougar con file precostruiti per i giochi.

Il passo successivo sarà quello di configurare il Cougar con funzioni preprogrammate che sono state create e distribuite insieme all'HOTAS. Sono state sviluppate configurazioni per oltre 30 giochi diversi ed esse sono state create in modo che siano molto più efficienti che la programmazione diretta da dentro il gioco. I file sono all'interno del CD e possono essere caricati nell'HOTAS tramite il CCP, ma esistono altre due vie più semplici per ottenere lo stesso risultato. Esse prevedono l'uso di Foxy e FoxyGUI.

Foxy: andare nel menu Editor's Favourites e cliccare sul gioco a cui siamo interessati. Ciò aprirà due file nel Foxy, quello alla sinistra (nonché il principale) è chiamato il file joystick, quello alla destra è il file macro. Non c'è bisogno di fare altro a questo punto. Per caricare i file appena aperti nel joystick, basta andare nel menu Download e cliccare sulla voce Download. Altre funzioni utili sono il Graphical Layout e View ReadMe della barra Apps (la seconda dall'alto) per capire cosa gli sviluppatori di quei file suggeriscono per l'uso di quella configurazione nel simulatore.

FoxyGUI: Ancora più semplice: basta seguire le istruzioni sullo schermo. Come prima verrà selezionare il gioco desiderato, premuto il pulsante Download e quindi chiudere FoxyGUI per dedicarsi al gioco. Anche qui sarà possibile vedere i file di grafici e testuali di layout premendo i rispettivi pulsanti.

Livello 3: Imparare a programmare il Cougar.

Esiste una consistente di documentazione che è stata sviluppata per imparare come programmare il Cougar, ecco una lista delle possibilità – basta scegliere la più appropriata.

1. La prossima sezione del manuale, *(2.1 – Capire le basi della Programmazione Thrustmaster)* così come i file di help del Foxy, introdurranno le basi della programmazione del Cougar in maniera semplice ed intuitiva.

2. Nel menu Wizard del Foxy, provare il Macro wizard seguito dal Joystick wizard. Molte persone hanno imparato come programmare i controller TM da questi due semplici wizard.

3. Sempre nel menu Wizard ci sono dei Tutorial, questi aprono dei file Foxy che spiegano le basi ed oltre. È possibile caricare questi file nell'HOTAS, modificarli e osservare l'effetto delle modifiche. Un modo molto efficace di imparare programmare.

4. FoxyGUI fornisce un sistema molto semplice per programmare il Cougar nonché le spiegazioni su come la programmazione risulterebbe se fosse fatta in Foxy. Sarà poi utile migrare verso Foxy perché è più potente e più rapido una volta capiti i concetti di base.

5. In qualsiasi momento, all'interno di Foxy, è possibile premere F1 per far comparire un aiuto, oppure evidenziare una parola nel file di configurazione e premere F1 per ottenere una spiegazione sul significato. Il file di help è molto grande, ricco di informazioni e contiene tutte le spiegazioni dettagliate di ciò che è presente in questo manuale.

6. Introduzione al Composer e Korgy dal menu Insert di Foxy … sarà necessario approfondire molto lo studio sul manuale con questi due componenti di Foxy.

Una cosa è molto importante: programmare il Cougar è molto, molto semplice. Basta impiegare un po' di tempo nel leggere le istruzioni che sono state fornite per ottenere dei successi in poco tempo. Una volta usato l'approccio testuale alla programmazione, grazie a tutti i vantaggi che offre, raramente si ritorna all'approccio puramente grafico che è così comune nei controller più semplici.

2. CAPIRE LE BASI

2.1 Capire le basi della Programmazione Thrustmaster

2.1.1 Introduzione

Parlando di programmazione, i joystick e le manette Thrustmaster hanno sempre stabilito gli standard con cui gli altri controller vengono misurati. Sfortunatamente essi hanno sempre avuto la reputazione di essere difficili da programmare, idea scatenata dal fatto che le versioni precedenti del software giravano sotto DOS e che gli utenti non fossero disposti ad impiegare del tempo per capire queste periferiche. Questo controller è estremamente più semplice che imparare a pilotare uno dei complessi simulatori di volo moderni.

Quello che verrà fatto sarà di iniziare dalle basi come se l'utente non avesse alcuna esperienza con i controller Thrustmaster e la loro programmazione. È una pessima coincidenza che in questo caso venga usato il termine programmazione: esso è normalmente associato allo sviluppo software e a complessi linguaggi. Quello che realmente verrà fatto non sarà altro che creare dei file che assegnano dei caratteri della tastiera ai pulsanti presenti sul joystick e sulla manetta.

2.1.2 Il concetto di HOTAS

In qualsiasi simulatore di volo, è possibile volare e controllare le armi, il cockpit, ecc. totalmente dalla tastiera. Si rende il tutto più realistico aggiungendo un joystick ed una manetta, magari una pedaliera i quali vengono normalmente chiamati "controller". Sfortunatamente sarà sempre necessario abbassare lo sguardo per premere dei tasti sulla tastiera, ma se si includessero dei pulsanti e degli hat sul joystick che, se premuti, assumessero le stesse funzioni della tastiera? Non ci sarebbe più bisogno di levare le mani dal joystick e dalla manetta, non sarebbe più necessario toccare la tastiera e ci si potrebbe concentrare esclusivamente sul volo, sul controllo degli armamenti, ecc.
Questo è il concetto di HOTAS, marchio registrato di Thrustmaster che permette di tenere le mani sulla manetta e sulla barra sempre (**H**ands **O**n **T**hrottle **A**nd **S**tick).

2.1.3 Come applicare il concetto di HOTAS nei simulatori di volo ed in altri giochi?

Abbastanza semplice – si programmano i controller per simulare i tasti premuti sulla tastiera che sono usati nel simulatore di volo. Questo è possibile tramite due file, il **file joystick** che determina quali pulsanti ed hat assegnare a quali caratteri ed il file macro, che contiene le *"macro"*. Le macro sono semplicemente descritte come le azioni del simulatore associate ai caratteri di tastiera.

2.1.4 Introduzione al file joystick – le basi della programmazione

Il file joystick è usato per assegnare i caratteri ai vari pulsanti ed hat sul controller. Non è completamente corretto chiamarlo file joystick, in effetti non serve solo a programmare il joystick, ma programmerà **tutti** i controller (joystick, manetta, pedaliera).

I pulsanti e hat sul joystick e sulla manetta hanno nomi speciali per riconoscerli quando si desidera programmarli. Non è il caso di elencarli tutti adesso perché Foxy ed il suo Composer renderanno tutto più naturale con l'uso. Verranno introdotti solo alcuni dei loro nomi così come le basi per poterli programmare.

Se si vuole programmare il pulsante S2 per poter gestire l'autopilota, il pulsante è quello rosso vicino all'hat chiamato Hat 1 nella parte superiore del joystick. Tornando ad S2, si ipotizza che per abilitare e disabilitare l'autopilota sia necessario premere il tasto "a" sulla tastiera (abbastanza comune nei simulatori di volo).

Ciò che è da "insegnare" al joystick è il fatto che ad ogni pressione di S2 è necessario produrre la pressione del tasto "a". In un file joystick (che è un semplice file di testo) i pulsanti vengono identificati con il temine "BTN" e in particolare ad S2. Si desidera che BTN S2 invii il carattere "a". Ecco quello che va scritto nel file joystick per programmare questo comportamento:

```
BTN S2 a
```

Facile! Passare a programmare un hat quando viene spinto verso l'alto in modo che generi la pressione di F1 sulla tastiera lo è altrettanto. Questa funzione

potrebbe essere il “look forward” (guarda avanti) del simulatore. Gli hat sono a tutti gli effetti pulsanti, quindi seguendo l’esempio precedente si può programmarli come pulsanti (BTN). HAT 1 Up (alto) viene abbreviato con H1U, quindi la stringa risultante è:

```
BTN H1U F1
```

Questi sono gli elementi di base della programmazione Thrustmaster.

Nei simulatori moderni è facile trovare più di 100 comandi per poter controllare l’aereo. Ricordare l’associazione dei tasti può diventare un incubo. È possibile aggiungere un commento (REM) che ricordi qual è la funzione associata al pulsante. Proprio così:

```
BTN S2 a   REM Abilita e disabilita l’autopilota
BTN H1U F1    REM Mostra la visuale frontale
```

Esiste un sistema migliore: usare le macro ed il file macro. Prima di passare a questi concetti è bene ricordare che i commenti possono essere messi ovunque nei file e che tutti i commenti vengono ignorati dal joystick. È comune usare i commenti all’inizio dei file per titoli, descrizioni, commenti, ecc...

“Verso l’infinito ed oltre!” Bene … almeno verso i file macro!

2.1.5 Introduzione alle macro ed ai file macro – basi della programmazione

Prima di arrivare al file macro, va definito il termine macro. Nella sezione precedente si è parlato del file joystick e sono stati definiti i seguenti elementi:

```
BTN S2 a   REM Abilita e disabilita l’autopilota
BTN H1U F1    REM Mostra la visuale frontale
```

Una macro non è altro che una parola facile da ricordare che rappresenta un tasto (della tastiera) o una sequenza di tasti.

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

Ora è possibile sostituire i nostri elementi nel file joystick in questo modo:

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

Può non essere ovvio il perché questo sia il sistema migliore, ma con un file joystick che contiene più di 100 elementi le macro rendono il tutto molto più comprensibile.

Dove inserire le macro?

Le macro hanno il loro file, il file macro appunto. Esso contiene tutte le macro che descrivono tutto ciò che i tasti fanno nel simulatore, mentre il file joystick assegna queste macro ai pulsanti sul joystick e sulla manetta. Normalmente si vedranno i file Thrustmaster sempre in coppie per ogni configurazione ed ecco perché in Foxy, la videata principale è l'Editor che visualizza i file joystick e macro in due cartelle separate. L'esempio precedente, associato ad esempio a Falcon 4, genererebbe il file joystick Falcon 4.tmj ed il file macro Falcon 4.tmm (tmj = **TM J**oystick File, tmm = **TM M**acrofile).

2.1.6 Com'è associato un file macro ad un file joystick?

Affrontate le basi dei file joystick e macro e scoperto che non sono difficili da capire e produrre, è tempo di capire come gestire diverse configurazione, tipo la presenza di 30 simulatori e quindi 30 file joystick e 30 file macro nella cartella File di Foxy.

Il titolo cita, "Com'è associato un file macro ad un file joystick?" Attualmente non esiste nessuna associazione. Nel file joystick dovremo specificare quale file macro usare. Per completare il nostro esempio dovremo definire all'interno del file joystick quale file macro contiene le Macro DEFinitions (MDEF) da usare. Dopotutto in un file di macro i freni possono essere attivati tramite il tasto "w" ed in un altro essere attivati tramite il tasto "b". L'elemento da usare **nel file joystick** che definisce il file macro da usare è:

```
USE MDEF Falcon 4.tmm
```

Quando Foxy legge la linea USE MDEF, cerca il corrispondente file macro (nell'esempio Falcon4.tmm) e usa le macro in esso contenute per programmare il file joystick.

2.1.7 Riepilogando quanto imparato finora …

Ecco il file joystick ed il file macro che prendono forma.

Joystick file *(Falcon 4.tmj)*	**Macro file** *(Falcon 4.tmm)*
REM ---------- REM Falcon 4.tmj REM Falcon 4 file joystick REM REM gli elementi Rem non fanno nulla. REM Vengon usati per aggiungere REM commenti REM ---------- REM Specifichiamo al file joytick quale REM file macro usare USE MDEF Falcon 4.tmm REM Adesso programmiamo alcuni REM pulsanti assegnandogli delle macro REM prese dal file macro BTN S2 Autopilot BTN H1U Forward_view	REM ---------- REM Falcon 4.tmm REM REM Falcon 4 file macro REM ---------- REM Le Macro permettono di ricordare REM facilemnte la funzione dei tasti nel REM nostro simulatore REM La definizione delle macro inizia REM qui Autopilot = a Forward_view = F1

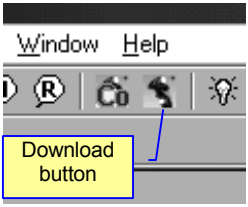
Si, queste macro non sono corrette per Falcon4, ma lo scopo è solo quello di spiegare il funzionamento delle cose!

Finora sono stati affrontati i seguenti punti:

1. I file macro contengono delle macro che descrivono semplicemente lo scopo di un tasto nel simulatore.
2. I file joystick assegnano le macro trovate nel file macro associato ai pulsanti del joystick e della manetta tramite l'elemento BTN.
3. Il file joystick conosce quale file macro usare attraverso l'elemento USE MDEF.
4. Gli elementi REM aggiungono dei commenti ai file.
5. I file macro e joystick sono dei semplici file di testo con estensioni .tmm e .tmj che si trovano nella stessa cartella del disco fisso. Di default questa è la cartella Files di Foxy.

2.1.8 Caricare il file joystick nel controller

Bene! Finite le basi dello sviluppo dei file rimane una domanda: come caricare i file nel controller?



Il modo più semplice è di usare il pulsante “Download" presente nella toolbar di Foxy o premere il tasto F12 sulla tastiera. Dopo poco il file verrà trasferito o, per usare il termine corretto, caricato nel controller. Questo è quanto, tutto è pronto per volare con i pulsanti programmati come descritto nel file joystick. Prima di continuare è meglio vedere con maggior dettaglio cosa avviene quando un file joystick viene caricato nel controller.

Ciò che accade è questo: il file joystick viene inviato ad una delle applicazioni Thrustmaster Cougar chiamata il Compiler (compilatore). Questa ha il compito di convertire il file joystick, combinandolo con il file macro, da testo in un formato che il controller è in grado di capire. Questa conversione è chiamata compilazione e quando giunge al termine senza errori il risultato viene mandato al controller. Alla fine viene inviato un messaggio a Foxy per indicare che il processo è terminato e che il controller è stato inizializzato con la programmazione fornita.

2.1.9 Struttura dei file joystick e macro

Prima di concludere questa sezione introduttiva, ecco alcune regole generali per strutturare correttamente i file. Questi esempi potrebbero non risultare chiarissimi, ma l’intenzione è quella di portare un esempio di struttura e non quello di dettagliare tutto quello che compare. Un altro elemento a cui fare attenzione è che il file joystick ha una sezione per i parametri di configurazione che verrà trattata più avanti.

Sezioni	File iovstick *(Falcon 4.tmj)*	File Macro *(Falcon 4.tmm)*
Titolo Non obbligatorio, ma è una buona idea.	Rem ---------- Rem Falcon 4.tmj Rem Rem Falcon 4 file joystick Rem Rem ultima modifica 1 Gen 01 Rem Rem ----------	Rem ---------- Rem Falcon 4.tmm Rem Rem Falcon 4 file macro Rem Rem ultima modifica 1 Gen 01 Rem Rem ----------
Configurazione *(**solo** nel file joystick)*	Rem Rem Configurazione Rem USE MDEF Falcon 4 USE RATE (60) USE TG1 AS DX1 USE S2 AS DX2	Rem Rem La configurazione non Rem Rem compare nei file macro Rem Rem quindi le macro iniziano qui
Sintassi dei comandi File joystick Assegnamento pulsanti, Assi, Logica. File Macro Definizioni delle macro	Rem ---------- Rem Assegnamento pulsanti 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 Manetta 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	Rem ---------- Rem Controlli delle viste Rem ---------- View_up = KP8 View_Down = KP2 View_Left = KP4 View_Right = KP6 Rem ---------- Rem Armi Rem ---------- Cycle_MSL_hardpt = SHF / Pickle_weapon = SPC Rem ---------- Rem Varie 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

Questa è una veloce introduzione alle basi della programmazione. Ora è il momento di affrontare i dettagli degli elementi che compongono un file joystick. Si ritornerà all'elemento BTN, identificando come sono chiamati i pulsanti e gli assi e discutendo le macro in profondità.

3. Elementi dei pulsanti e macro

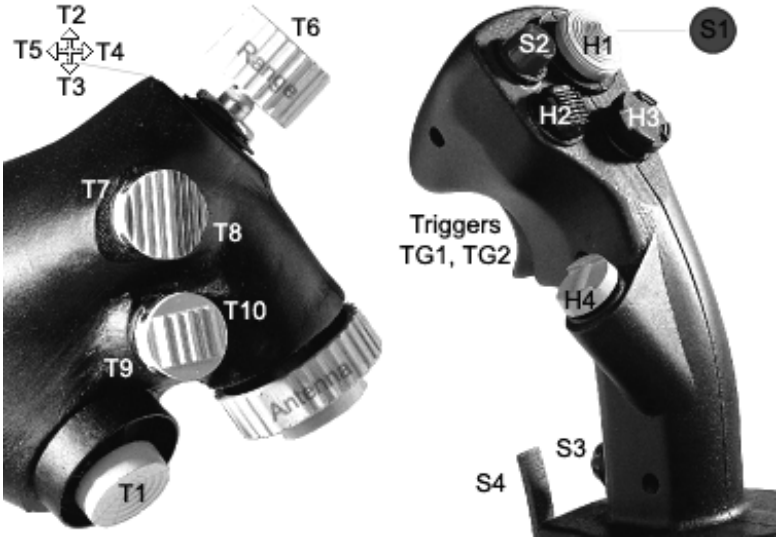
3.1 Elementi dei pulsanti e sintassi dei tasti TM

L'HOTAS Cougar consiste essenzialmente di diversi assi, qualche hat, pulsanti, grilletto, ecc. Tutto ciò che non è un asse è programmabile tramite l'elemento pulsante con questa sintassi:

Sintassi

BTN Nome_del_pulsante SequenzadiTasti e/o macro

dove:
Nome_del_pulsante identifica il pulsante da programmare:

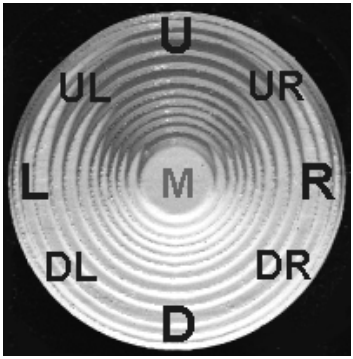


La manetta ha:	10 pulsanti:	da T1 a T10
Il joystick ha:	4 hat :	da H1 a H4
	4 pulsanti:	da S1 a S4
	grilletto a 2 livelli:	TG1, TG2

Esempi:

```
BTN T3 y            Rem "Roger, capito!!"
BTN S2 Eject
BTN T4 Chaff Flare  Rem L'ora di muoversi
BTN S4 h e l l o    Rem Notare gli spazi – non è una macro
```

Ogni hat ha 9 posizioni programmabili:



In genere solo le 4 posizioni principali sono programmate. L'HAT 1 per esempio potrebbe essere:

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

Ma gli angoli potrebbero essere programmati così:
es. `BTN H1UL  View_UL`

e la posizione centrale:

```
BTN H1M  View_forward
```

NOTE

1. *BTN non deve essere all'estremo sinistro della linea, ma solo un BTN è consentito per ogni linea e ogni pulsante può essere programmato solo una volta nel file joystick.*

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

Fin qui tutto è corretto, ma aggiungendo:

```
BTN S3 g h I
```

Il compilatore genererà un errore di duplicate button statement (elemento pulsante duplicato)

2. *Deve esserci un solo spazio tra BTN ed il Nome_del_pulsante, quindi:*

   BTNS2

   Genererà un errore.
3. *Deve esserci uno spazio dopo BTN Nome_del_pulsante, quindi:*

   BTN S2a b c

   Genererà un errore.

4. *Una macro carattere viene prodotta solo nell'elemento pulsante, che il pulsante venga tenuto premuto o meno. Volendo ripetere la macro è da valutare l'uso dei modificatori* /A (*auto-repeat) o* /H (mantenimento)*. Questo è un comportamento differente dall'originale Thrustmaster. I modificatori verranno trattati più avanti in questo manuale.*

NOTE AVANZATE

Divagando leggermente verrà affrontata la posizione centrale dell'hat. Non è suggeribile affrontare questo adesso, ma se quanto segue fosse un problema in futuro ecco la spiegazione. La posizione centrale di ogni hat può essere programmata aggiungendo M all'hat come nell'esempio qui sotto. Notare che se /P e/o /R *(spiegati più avanti) vengono programmati in una delle posizioni dell'hat, i tasti* /R verranno generati nello stesso momento dei tasti M*. Quindi:*

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

Genererà allo spostamento verso l'alto e rilascio la sequenza:
***"1", quindi "a" e "2"** nello stesso momento.*

Volendo assicurarsi che H1M *venga eseguito dopo l'* H1U /R*, è possibile aggiungere un ritardo (vedere note seguenti) nell'* H1M*:*

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

3.2 Sintassi di tastiera Thrustmaster

Ritornando alla sintassi

Sintassi
BTN Nome_del_pulsante SequenzadiTasti e/o macro

Verrà analizzata la parte SequenzadiTasti.

Assegnando caratteri di tastiera, sia nelle macro che negli elementi nel file joystick, è necessario capire che esiste uno stile di identificazione dei caratteri di tastiera definito da Thrustmaster chiamato appunto sintassi di tastiera Thrustmaster. Le basi di questa sintassi sono quelle che permettono di differenziare la pressione del tasto “5” da quella del tasto “5” presente sul tastierino numerico. In effetti essi vengono identificati come “5” e “KP5” nell’esempio. Ecco la sintassi di tastiera 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.	

Il modo migliore per assicurare la corretta corrispondenza è quello di usare Korgy, la tastiera virtuale di Foxy. È possibile visualizzare queste informazioni dall’Editor di Foxy tramite il menu Help e scegliendo la voce "Keyboard Syntax".

NOTE

1. *La sintassi per i comandi modificati con Shift, ALT o CTRL è questa: SHF a, ALT b, CTL c. Questi non sono equivalenti a LSHF a, LALT b o LCTL c. Ad esempio LSHF a è equivalente a premere lo Shift di sinistra, rilasciarlo e poi premere e rilasciare il tasto "a". Il compilatore usa come modificatori i tasti Shift, ALT e CTRL di* ***sinistra*** *come default.*

2. I *tasti ( ) { } < > sono riservati e vanno programmati nel seguente modo:*

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

3. *Una buona abitudine è quella di usare tasti modificati con shift invece che configurarli in maiuscolo:*

BTN S1 SHF p
BTN S1 P

Entrambe sono corrette ma la prima è più rigorosa.

4. *La sintassi della tastiera è basata sul layout della tastiera US. Ci possono però essere situazioni in cui risulta necessario produrre dei caratteri non presenti in tale layout. In questo caso è possibile usare i codici USB direttamente … questo argomento verrà trattato successivamente e verrà usato raramente, ma in caso di bisogno la funzione è presente!* ☺

5. *La sintassi è cambiata dalla originale presente nel Thrustmaster F22. Ad esempio il suffisso AUX di alcuni tasti è stato rimosso.*

3.3 Macro e regole

Nell'introduzione è stato spiegato il concetto di macro e di file macro. Sono stati forniti due esempi:

Autopilot = a
Forward_view = F1

Avendo spiegato la sintassi di tutti i tasti sulla tastiera è possibile generare file macro per i simulatori. Foxy ha diverse utilità interessanti che possono aiutare a creare macro che seguano le regole elencate. Queste sono: Macro Wizard, Speedy and Korgy. *Per ulteriori informazioni riferirsi alla documentazione di Foxy*. Prima di proseguire con le regole ecco un suggerimento: è spesso noioso e

difficile dover consultare il foglio riassuntivo dei comandi del simulatore durante la creazione delle macro. Se il simulatore ha un file di help è possibile provare a copiare tutte le definizioni per poi convertirle in macro. Nel 90% dei casi di errore di compilazione, la causa è dovuta ad errori nei file di macro. Durante la creazione di un file di macro è consigliabile verificare con cura quanto scritto perché poi questo file non verrà più modificato come invece accadrà al file joystick.

Ed ecco le regole …

1. I nomi delle macro **non** possono contenere spazi. Usare _ o – al posto degli spazi.

La mia macro = b

Non è valido, mentre:

La_mia_macro = b
La-mia-macro = b *è corretto.*

2. Assicurarsi di usare uno spazio prima e dopo l'uguale che segue il nome della macro:

Autopilot= a
Autopilot =a *sono errati.*

3. I seguenti caratteri sono vietati nei nomi delle macro:

= < > { } ( ) ^ , *spazio*

4. È sconsigliabile l'uso delle maiuscole nei nomi delle macro. Ad esempio RADAR_RANGE_INCREASE non genererà un errore, ma il file sarà più leggibile se verrà scritto in minuscolo. È buona norma usare le maiuscole solo per i comandi Thrustmaster (come MDEF), o abbreviazioni (come HUD).

5. I nomi delle macro sono case insensitive. Quindi:

MiaMacro = a
miamacro = a sono equivalenti.

NOTE AVANZATE

Questo è un esempio per chi ha veramente voglia di scendere nei dettagli!

Le macro possono essere nidificate, quindi è possibile usare macro all'interno di altre macro. Ad esempio:

```
Macro_1 = a b c
Macro_2 = Macro_1 d e f
```

È possibile nidificare fino a 20 livelli di macro.

```
Macro-1 = a Macro-2
Macro-2 = b Macro-3
Macro-3 = c Macro-4
... ecc...
Macro-20 = d
```

Non più di 20. Scrivendo questo:

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

Il compilatore genererà un errore "Macro looping too long; two macros might be calling themselves." (Il ciclo delle macro è troppo lungo; due macro potrebbero richiamarsi a vicenda)

3.4 Modificatori degli elementi

Finora sono stati affrontati elementi semplici come:

```
BTN T4 a
```

che produce la pressione del tasto “a” quando il pulsante T4 sulla manetta viene premuto e rilasciato. Ricorda che ci saranno situazioni in cui nono sarà sufficiente la pressione di un solo tasto. Su una tastiera puoi premere più tasti per generare una stringa di caratteri con diversi intervalli tra uno, si può tenere premuto uno o più tasti, ecc. **La base su cui realizzare un controller di successo è che deve essere in grado di emulare tutto ciò che è possibile fare con una tastiera.**

Qui introdurremo gli Statement Modifiers (Modificatori d’Espressione). Vogliamo essere in grado di andare oltre la semplice pressione di un singolo carattere emulato dai pulsanti.
I modificatori d’espressione sono espressioni che possono essere usate per cambiare il comportamento di un carattere programmato su un pulsante. I modificatori sono di cinque tipi:

1. Modificatori a barra /U, /M, /D, /I, /O, /P, /R, /T, /A, /H

```
BTN S4 /H b   Rem Freni
BTN T3 /A c f  Rem Chaff e flares
```

2. Espressione di ritardo e ripetizione DLY( ), RPT( )

```
BTN T6 1 DLY(60) 1 DLY (60) 2 Rem Vettore per il recupero TAW
BTN S2 RPT(6) c  Rem 6 chaff …
```

3. Raggruppamento dei caratteri – tramite parentesi ( ), { }, < >

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

4. Lavorare e definire i pulsanti DirectX (Direct Input) DX

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

5. Usare KeyDown, KeyUp ed i codici USB KD( ), KU( ), USB( )

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

Osserveremo i modificatori a barra per iniziare …

6. *Non è possibile usare le parole riservate Thrustmaster nel definire i nomi delle macro. Questo include la sintassi dei tasti di tastiera e altre sintassi usate nelle espressioni Thrustmaster. Ad esempio un file joystick così:*

```
BTN S2 HOME
```

ed il file macro:

```
HOME = k
```

genereranno un errore durante la compilazione o il caricamento del file joystick perché la parola "HOME" è la sintassi usata per rappresentare il tasto "HOME" della tastiera.

3.5 Modificatori a barra

Esistono 10 modificatori di questo tipo che possono essere raggruppati in tre gruppi principali basati sull'effetto che hanno sui pulsanti e sulle espressioni:

Incrementare il numero di posizioni programmabili:

/U, /M, /D usando il selettore Dogfight della manetta (T7, T8)
/I, /O usando il pulsante S3 del joystick

Separare le macro di un pulsante:

/T Impulso
/P, /R Pressione e Rilascio

Ripetizione e mantenimento:

/A Ripetizione automatica
/H Mantenimento

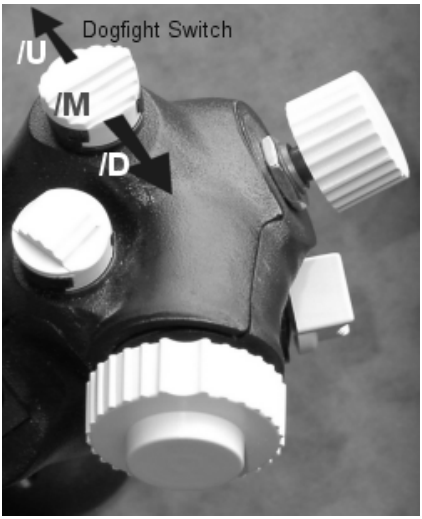
Gerarchia e regole dei codici:

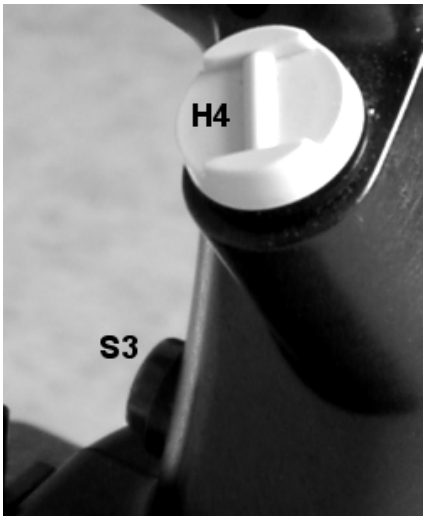
Infine, è necessario capire dove è possibile usare i modificatori a barra, dove no e in che ordine.

3.5.1 Aumentare il numero di posizioni programmabili:

Il selettore Dogfight nella manetta, può essere mosso in 3 posizioni: - Alta (/U), Centrale (/M), Bassa (/D). Analogamente, il pulsante S3 può essere premuto (/I) o sollevato (/O), *notare i grafici*. È possibile usare una combinazione di questi per aumentare il numero di posizioni dei pulsanti/hat/assi fino a sei volte.

Modificatori delle Espressioni
/U, /M, /D usando il selettore Dogfight della manetta (T7, T8) /I, /O usando il Pulsante S3 del Joystick





3.5.1.1 /U, /M, /D – Up (Alto), Middle (Centrale), Down (Basso)

È possibile aumentare il numero di posizioni programmabili di un pulsante, hat o asse usando la posizione del selettore Dogfight con i modificatori a barra /U, /M e /D. Per esempio:

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

Quando l' Hat 4 viene mosso in alto, verrà generato:

- un carattere "a" se il selettore di dogfight è in posizione alta (/U)
- un carattere "b" se il selettore di dogfight è in posizione centrale (/M)
- un carattere "c" se il selettore di dogfight è in posizione bassa (/D)

È **necessario** inserire le espressioni /U, /M, /D su linee separate. Se esse apparissero sulla stessa linea verrebbe generato un errore di compilazioni. Esse devono apparire nell'ordine usato. Quindi:

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

genererebbe un errore di compilazione.

3.5.1.2 /I, /O – In (premuto), Out (rilasciato)

È possibile aumentare il numero di posizioni programmabili di un pulsante, hat o asse usando il pulsante S3 tramite i modificatori a barra /I e /O. Per esempio:

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

Quando l'hat 4 verrà abbassato, genererà:

- un carattere "1" se il pulsante S3 è premuto (/I = **I**n)
- un carattere "2" se il pulsante S3 è rilasciato(/O = **O**ut)

Combinare /U, /M, /D con /I, /O nelle espressioni

È possibile combinare questi modificatori a barra. Per esempio:

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

In questo caso l'hat 4 a destra avrà sei diversi comportamenti, in base alla posizione del selettore Dogfight (/U, /M, /D) e del pulsante S3 (/I, /O).

NOTE

1. I *modificatori* /U, /M, /D*, se presenti dovranno sempre precedere gli* /I, /O*.*

2. *Non è possibile usare i modificatori* /U, /M, /D *sul selettore Dogfight (pulsanti T7 e T8),visto che sono evidentemente usati per determinarne la posizione dello stesso selettore.*

3. *È* ***necessario*** *usare le espressioni* /I, /O *su linee separate. Questo differisce dalla sintassi originale degli HOTAS Thrustmaster.*

4. *È necessario usare le espressioni* /I ***prima*** *delle* /O*. Quindi:*

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

è un'espressione valida, ma sia:

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

che:

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

genereranno un errore di compilazione. Questo differisce dalla sintassi originale degli HOTAS Thrustmaster.

5. *Usando i modificatori* /I, /O di *S3, qualsiasi espressione nella posizione* /O *verrà normalmente ignorato dal compilatore. Dico normalmente perché usando l'espressione* S3_LOCK *(vedere le note successive) è possibile usare espressioni* /O *su S3. Definendo un diverso pulsante per essere usato come S3, usando l'espressione* USE *Btn* AS SHIFTBTN *(vedere le note successive) si applicano le stesse note.*

6. *Se un file viene scritto per il joystick e la manetta, ma solo il joystick è disponibile, il compilatore userà la posizione* /M *su tutti i pulsanti/hat, e le espressioni* /U, /D *verranno ignorate.*

NOTE AVANZATE – COM'È STATO EVITATO IL BLOCCAGGIO DEGLI ASSI

Alcune note tecniche che spiegano cosa succede al cambiamento di stato di S3 (da rilasciato a premuto e viceversa) quando un pulsante viene programmato con le espressioni /I *e* /O. *Se la posizione del selettore dogfight o quella di S3 cambiano, questo è ciò che farà il controller:*

1. *Cougar riconosce il cambiamento di stato*
2. *Cougar controlla i pulsanti premuti*
3. *Cougar controlla se i pulsanti premuti hanno una diversa programmazione nel nuovo stato.*
4. *Se si aggiunge la macro di rilascio per il precedente stato all'uscita. Viceversa non viene generato alcun carattere, ma le restanti espressioni non subiscono variazioni (es. mouse, DirectX, assi)*
5. *Cougar passa al nuovo stato.*

3.5.2 Separare le macro di un pulsante:

Modificatori d'espressione

/T Ciclo tra diverse macro in un pulsante
/P, /R Pressione e Rilascio di un pulsante

3.5.2.1 /T – Modificatore a barra di Toggle (ciclo)

Il modo più semplice per capire che cos'è il modificatore /T è di osservare un esempio ed il risultato:

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

Supponiamo che S2 venga premuto 3 volte. Alla prima pressione verrà generato un carattere "a", "b" verrà generato alla seconda e "c" alla terza. Alla successiva la sequenza ricomincerà da "a".
Sono possibili 16 posizioni ciclabili per ogni espressione e, includendo anche i modificatori /U, /M, /D, /I e /O avremo:

```
BTN S2 /U /I   16 posizioni qui
         /O   16 posizioni qui
       /M /I   16 posizioni qui
         /O   16 posizioni qui
       /D /I   16 posizioni qui
         /O   16 posizioni qui
```

NOTE

1. *Il toggle* ***non*** *è consentito dopo le espressioni /P o /R. Quindi:*

Esempio 1.
```
BTN TG1 /T /P a
          /R b
        /T c
```

Esempio 2.
```
BTN TG1 /P /T a /T b
          /R c
```
L'esempio 1 è valido, ma il 2 ed il 3 genereranno degli errori.

Esempio 3.
```
BTN TG1 /P a
          /R /T b /T c
```

2. *Non è consentito usare l'espressione /T su T1 se l'espressione* USE T1_SENSITIVITY *compare nel file joystick (vedere note seguenti).*

3. *Non è consentito usare l'espressione* /T *su un hat del joystick se l'espressione* USE *HatID*_SENSITIVITY compare nel file joystick *(vedere note seguenti).*

4. *Non è consentito usare l'espressione* /T *con espressioni relative ad assi digitali (vedere note seguenti).*

5. *Non è consentito usare l'espressione* /T *con un'espressione di programmazione logica (vedere note seguenti).*

6. *Non è consentito usare un solo modificatore* /T *in un'espressione. Quindi:*

BTN T4 /T a

genererà un errore.

7. *Espressioni* /T *possono essere inserite sulla stessa linea o su più linee. Quindi:*

BTN S2 /T a /T b /T c *è consentito*

8. *È possibile reimpostare l'ordine di scorrimento del ciclo con l'espressione* RESET_TOGGLES *e invertire l'ordine con l'espressione* REVERSE_TOGGLES*, che verranno affrontate più avanti.*

9. *È possibile usare altri modificatori insieme a* /T*. Questa possibilità è nuova rispetto alle precedenti versioni. Ad esempio:*

BTN S2 /T a /T /H b *è consentito.*

10. *Non è possibile usare i toggle nella posizione centrale di un hat. La linea seguente genererà un errore:*

BTN H1M /T a /T b

3.5.2.2 Reimpostare la posizione dei toggle

Quando è necessario azzerare l'indice del toggle in modo che lo scorrimento dello stesso riprenda dall'inizio, è necessario usare la seguente espressione:

Sintassi

RESET_TOGGLES

Avendo un'espressione come questa:

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

e avendo premuto diverse volte il pulsante S2 in modo che sia già stato un carattere a metà della sequenza (ad esempio il '7'), sarà possibile riprendere la sequenza dall'inizio. In questo caso è sufficiente premere il pulsante S3 insieme a S3 (che attiva l'espressione /I).
Premere S2 insieme a S3 non genera alcun carattere, ma alla successiva pressione di S2 (da solo) verrà generato il carattere '1'.

NOTE

1. *Questa espressione* ***deve*** *apparire subito dopo un'espressione* /I *o* /O. *Questo esempio genererà un errore:*

```
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 Macro
```

ma questo invece funzionerà:

```
BTN S4    /U 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. *L'espressione* RESET_TOGGLES *non può essere seguita da nulla sulla stessa linea.*

3.5.2.3 Invertire la direzione di scorrimento

Quando è necessario invertire la direzione dello scorrimento di un'espressione toggle, usare la seguente espressione:

Sintassi

```
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
```

Normalmente la sequenza dei caratteri generata premendo S2 sarà da '1' a '0'. Premendo S2 ed S3 contemporaneamente l'ordine verrà invertito partendo dalla dall'ultimo carattere generato.

NOTE

1. *Questa espressione* ***deve*** *comparire direttamente dopo un* /I *o un* /O *statement, con la sequenza nell'altra espressione* /I *o* /O*. Questo genererà un errore:*

```
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 Macro
```

mentre questo funzionerà regolarmente:

```
BTN S4 /U 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
       /D Macro
```

2. *L'espressione* REVERSE_TOGGLES *non può essere seguita da altro sulla stessa linea. Quindi modificando l'esempio precedente:*

```
BTN S4 /U 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_che_genera_l'errore
       /D Macro
```

La macro inserita nella terza linea genererà un errore.

3.5.2.4 /P, /R – Pressione e Rilascio

Il modificatore /P indica che la macro specificata verrà eseguita alla pressione del pulsante relativo. Nello stesso modo il modificatore /R specifica che la macro andrà eseguita al rilascio del pulsante.

Ad esempio:
```
BTN S2 /P Chaff
       /R Flare
```

Nell'esempio precedente il pulsante S2 viene usato con entrambi i modificatori, lanciando la macro Chaff alla pressione e la macro Flare al rilascio.

NOTE

Il modificatore /R *va usato insieme a* /P*. Se* /P *non è presente, allora usare* /R *genererà un errore e viceversa.*

NOTE AVANZATE

1. *Usare questi modificatori su un hat insieme alla definizione della posizione centrale dell'hat stesso porterà all'esecuzione contemporanea delle macro di* /R *e di* BTN HxM*. Quindi:*

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

al movimento in alto e rilascio di HAT 1, genererà:
"1" seguito da ***"a 2"*** *insieme*

2. *Usando i modificatori nella definizione della posizione centrale di un hat, allo stesso modo, eseguirà l'espressione relativa alla direzione del movimento dell'hat insieme all'espressione* HxM /R*. Quindi*

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

allo spostamento in alto dell'hat, genererà:
"b 1" *insieme, seguiti da "a"*

3. *È possibile bloccare (ripetizione continua) i tasti usando le espressioni* /P *ed* /R*. Ad esempio:*

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

Premendo il pulsante S4 verrà generato un ritardo di 2 secondi, seguito dalla pressione del tasto "p" (senza il rilascio). Il rilascio del tasto "p" verrà emulato con il rilascio del pulsante S4.

Cosa accadrebbe se il pulsante S4 venisse rilasciato prima dello scadere della pausa di due secondi?
Il ritardo verrebbe comunque portato a termine e a seguire la pressione di "p", ma l'emulazione del rilascio di "p" verrebbe eseguita immediatamente, quindi prima dell'effettiva pressione che avrebbe l'effetto bloccare il tasto "p" in posizione premuta. Esistono due vie per eliminare questo problema. La prima è quella di usare e programmare l'HOTAS Cougar con buon senso. Se sai di dover tenere il dito su un tasto per un certo periodo di tempo, allora fallo! Questo eviterà ***sempre*** *problemi di questo tipo. L'altra via è quella di racchiudere le espressione* /P *tra* < >*:*

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

ciò obbliga ad ultimare la sequenza tra < > prima che l'espressione /R *venga eseguita. È importante notare che tutte le macro di tutti i pulsanti (se non già in esecuzione) verranno bloccate finché l'espressione < > non sarà terminata.*

3.5.3 Ripetizione e trattenuta dei caratteri:

Modificatori

/A Ripetizione-automatica
/H Trattenimento

3.5.3.1 Caratteri non ripetibili

Il comportamento delle macro su caratteri singoli su un pulsante è variata rispetto alla sintassi originale. Ogni espressione su pulsanti o espressione digitale, siano esse caratteri o macro **non** verrà ripetuta. Per chi fosse abituato alla vecchia sintassi, è possibile considerare come se ci fosse un modificatore /N davanti ad ogni espressione.

Questo comporta che il modificatore /N non è più necessario – di fatto esso viene ignorato dal compilatore. Si raccomanda di cancellarli dai file ove presenti.

3.5.3.2 /A – Ripetizione-automatica

Questo modificatore ha l'effetto contrario al precedente. Specifica che i caratteri o le macro andranno ripetute finché il pulsante non verrà rilasciato.

```
BTN S2 /A Fire_Missiles
```

Premendo S2 la macro Fire_Missiles verrà ripetuta esattamente come se stessimo premendo e rilasciando molto velocemente il tasto sulla tastiera. Se altri pulsanti venissero premuti, il loro risultato verrebbe mischiato a quello generato dalla ripetizione. Premere altri pulsanti sul controller non interrompe la sequenza generata dalla ripetizione. Le macro che seguono un /A possono essere formate da più caratteri. Il modificatore /A rimpiazza la vecchia sintassi THRUSTMASTER che consisteva nel mettere parentesi per forzare la ripetizione.

3.5.3.3 /H - Trattenimento

```
BTN S4 /H b  Rem Freni
```

Un'espressione /H (Hold) può essere usata per forzare il trattenimento di un tasto sulla tastiera (*attenzione alle Note qui sotto*). Per esempio, molti simulatori di volo prevedono la pressione continuata del tasto "b" per attivare i freni sul carrello. Rilasciando il tasto, questi freni vengono rilasciati.

Alcuni esempi:

```
BTN S4 /H b  Rem Freni
BTN T6 /H Freni
BTN T1 /H ALT F6
```

È possibile usare /H in espressioni più complesse. Nell'esempio qui sotto, l'espressione è equivalente alla pressione e rilascio del carattere "c" e, a seguire, la pressione (trattenuta) del carattere "f":

```
BTN S4 /H c f Rem 1 Chaff, molti Flares
```

Così come:

```
BTN S4 /H {c f} Rem Chaff and Flares
```

produce KD(c) KD(f) fino al rilascio del pulsante quando vengono prodotti KU(c) KU(f).

Data questa espressione:

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

Per definizione il carattere "C" non è altro che "SHF c" e mantenendo premuto LSHF anche il carattere "f" diventa "F". E quindi impossibile produrre il carattere "f" mantenendo premuto uno SHIFT.

L'efficacia dell'uso di /H con più di un carattere può essere dimostrata con la seguente espressione che permette di selezionare un'arma secondaria e continuare a far fuoco:

```
BTN S4 /H Select_Rockets DLY(600) Fire Rem Seleziona arma secondaria e spara
```

NOTE

1. *A questo punto diventa molto importante capire la differenza tra* /H *e* /A. *Normalmente premendo un carattere sulla tastiera, ad esempio "a", verrà prodotto un carattere "a", una piccola pausa, e una serie di "a" che durerà fintanto che il tasto verrà tenuto premuto. Ad esempio:* a *pausa* aaaaaaaaaaaaaaaa
   Questo comportamento è lo stesso che si può ottenere usando /H. *Quindi:*

```
BTN T3 /H a
```

Volendo eliminare la pausa tra il primo ed il secondo carattere, è possibile utilizzare /A. Quindi:

BTN T2 /A b *produce bbbbbbbbbbbbbbbbbbb senza attese.*

2. *La frequenza di ripetizione dei caratteri generati da /A è determinata dall'espressione* USE RATE (time_ms). *Se quest'espressione non compare nel file, il compilatore genererà automaticamente un* USE RATE (0) *e verrà usata la frequenza di default della tastiera.*

3. *Se /H è seguito da caratteri multipli, solo l'ultimo viene mantenuto premuto mentre i precedenti vengono generati una sola volta. Quindi:*

BTN S2 /H a b c

alla pressione di S2 genererà: una "a", una "b" e tratterrà "c".

4. *Non è possibile usare /H o /A dopo /R.*

3.5.4 Gerarchia e regole dei codici barra

Usando i modificatori a barra (es. /T, /P, /U), è importante capire che esistono semplici regole che stabiliscono l'ordine in cui possono essere inseriti.

3.5.4.1 Regole

1. Vanno messi dopo la definizione del pulsante/selettore (BTN S2 /H).
2. La prima macro (BTN S2 /H SomeMacro) ad eccezione di /T che può essere ripetuto più volte all'interno.
3. Deve esserci un solo spazio prima e dopo il modificatore (vedi esempio precedente).
4. Quando più di un modificatore a barra viene usato devono apparire in un ordine ben definito (*vedere "Gerarchia" qui sotto*).
5. I modificatori /U /M /D devono essere su linee separate.
6. I modificatori /I /O devono essere su linee separate.

3.5.4.2 Gerarchia

Usando più modificatori sullo stesso pulsante o selettore è importante usare la corretta gerarchia. L'ordine in cui inserire i modificatori è il seguente:

1. /U /M /D, se presenti, dovranno precedere tutti gli altri.
2. /I e /O saranno i seguenti.
3. /T precederà /P e /R.
4. /P e /R sono sempre dopo i precedenti.

5. /H e /A sono sempre gli ultimi.

Ecco alcuni esempi:

```
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
```

NOTE

Non c'è alcun problema nell'usare un'espressione vuota. Non è necessario usare Null come nelle versioni precedenti. Quindi:

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

non genererà alcun errore.

3.6 Pausa e ripetizione

Modificatori di espressione
DLY (Durata) RPT (Numero)

dove:

Durata è un tempo in millisecondi (1 secondo = 1000 millisecondi) e accetta valori tra 0 e approssimativamente 82800000. (*Per informazione, 82800000 millisecondi equivalgono a 23 ore!*)

Numero è un valore il cui Massimo cambia al variare della lunghezza della macro e può essere compreso tra 2 e 127.

Seguono alcuni esempi.

3.6.1 DLY( )

```
BTN T6 1 DLY(60) 1 DLY (60) 2 Rem Request Vector For Recovery TAW
```

Genererà "1 1 2", con una pausa di 60 millisecondi tra i caratteri. Perché usare le pause in questo modo? I simulatori (ed i giochi in generale) stanno diventando così complessi che è normale dover usare gruppi di caratteri per eseguire un'azione. In un simulatore questo è normale se si parla di comunicazioni che spesso vengono rappresentate da menu. Ad esempio Falcon4:

```
VectorToHomePlate = q DLY(60) q DLY(60) 6
```

Scrivendo la macro senza pause:

```
VectorToHomePlate = q q 6
```

C'è un'alta probabilità che il simulatore non riesca a ricevere tutti e tre i caratteri: il controller li invierebbe troppo velocemente, mentre il simulatore è impegnato in altri calcoli. Inserire delle pause tra i caratteri rallenta la sequenza rappresentando meglio la velocità con cui la mano preme i tre tasti sulla tastiera. Inserendo le espressioni di pausa tra i caratteri rallenta la generazione. La situazione si complica in base all'uso di USE RATE (time) (*vedere le note*). Questa espressione determina il la frequenza con cui i caratteri vengono inviati durante una ripetizione.

Quindi:

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

È lo stesso che:

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

Il compilatore userà USE RATE(0) ogni volta che l'espressione USE RATE non è presente nel file.

Usando espressioni DLY, non è necessario tenere il pulsante premuto per completare l'espressione. Ad esempio:

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

genererà "hello", con un'attesa di due secondi tra un carattere e l'altro. Una cosa interessante è che premendo due volte S2 nell'arco di due secondi, il risultato sarà:

"hheellllloo"

Questa è una delle caratteristiche salienti di HOTAS Cougar – la capacità di elaborare espressioni in parallelo. Per prevenire questa cosa, è necessario usare con cautela i comandi o racchiudere l'espressioni tra < >.

3.6.2 RPT( )

```
BTN S2 RPT(6) c  Rem 6 chaff
```

Usando RPT il carattere o la macro che seguirà RPT (nnn) verranno ripetuti un numero infinito di volte. L'elemento ripetuto sarà quello che seguirà RPT (sarà comunque possibile racchiudere più caratteri o macro all'interno di parentesi per ripetere tutto il gruppo più volte).

L'esempio precedente genererà 6 "c" alla pressione di S2. La frequenza sarà determinata dall'espressione USE RATE(x).

Ecco altri esempi:

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

invierà dieci "a" seguite da una "b".

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

invierà dieci volte "a b".

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

invierà 10 "a" con 60 millisecondi di attesa tra una e l'altra. A seguire una pausa di 2 secondi. Notare che quest'espressione verrà ripetuta fintanto che il pulsante rimarrà premuto S2 (effetto dell'espressione /A).

RPT possono essere nidificati – ad esempio:

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

è valido, così come:

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

Avendo una macro:

```
Macro1 = a b c
```

E un'espressione:

BTN S2 RPT (3) Macro1

Alla pressione di S2 verrà generato:

a a a b c

Per risolvere questo problema, racchiudere la macro o i caratteri tra parentesi tonde:

BTN S2 RPT (3) (Macro1) *o*

BTN S2 RPT (3) Macro1 *dove*

Macro1 = (a b c)

3.7 Raggruppamento dei caratteri – parentesi

Modificatori di espressione

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

Prima di vedere l'uso di queste parentesi per il raggruppamento dei caratteri, è importante sapere che i simboli **( ) { }** e **< >** sono espressioni **riservate**.
Non è quindi possibile assegnarli direttamente ai pulsanti e alle macro. Al loro posto andranno usate delle espressioni *shiftate*:

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

Una macro così:

Left_ToeBrake = <

Genererà un errore, mentre l'espressione corretta è:

Left_ToeBrake = SHF .

3.7.1 ( ) Parentesi

Le parentesi sono usate in diverse espressioni come DLY ( ), RPT ( ), USB ( ), KU ( ) ecc., per raggruppare più elementi. Abbiamo già visto come usarle in un esempio abbastanza complesso con RPT e DLY:

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

È fondamentale notare che con la sintassi originale, mettere parentesi intorno a gruppi di elementi voleva dire forzarne la ripetizione. Con HOTAS Cougar questa sintassi è stata abbandonata a favore del modificatore /A e quindi le parentesi hanno l'unico scopo di raggruppare caratteri e macro. Quando le parentesi vengono usate in espressioni Thrustmaster, il compilatore interpreterà i seguenti nello stesso modo: DLY (20), DLY(20), DLY (20), ecc.

Infine, se si vuole generare un carattere "(" o ")" è impossibile scrivere le macro in questo modo:

```
Macro1 = (
Macro2 = )
```

Le parentesi ( ), < > e { } sono caratteri riservati perché vengono usati nelle espressioni per modificarne il comportamento.

Avendo:

```
BTN S1 Macro1
```

Dovremo usare la combinazioni shiftata per poter generare il carattere "(". Le definizioni delle macro saranno quindi

```
Macro1 = SHF 9
Macro2 = SHF 0
```

Vedere la sezione Macro e regole. In questo caso Korgy è un ausilio molto importante per essere sicuri di generare correttamente la sintassi Thrustmaster per i caratteri desiderati.

3.7.2 { } Parentesi graffe

Questo tipo di raggruppamento è usato per indicare che il set di caratteri contenuto va generato nello stesso istante. I gruppi contenuti nelle parentesi graffe sono trattati come una singola entità. Ad esempio:

BTN S4 {a b c}

È inviato come pressione di "a", pressione di "b", pressione di "c", rilascio di "a", rilascio di "b", rilascio di "c", come se venissero premuti tutti i tasti prima di rilasciarli nello stesso ordine.

È possibile usare il modificatore /H con le parentesi graffe, in questo caso, tutto il gruppo verrà mantenuto premuto fino al rilascio del pulsante. Ad esempio:

BTN S4 /H {CTL ALT DEL} è corretto

NOTE AVANZATE

1. *Non è possibile usare DLY all'interno delle parentesi graffe.*

2. *L'implementazione delle espressioni { } sui dispositivi USB differisce molto dai precedenti HOTAS Thrustmaster gameport dove l'ordine d'invio dei caratteri rimaneva immutato rispetto all'espressione. Quindi:*

BTN S4 {a b c}

generava (e genera) pressione di "a", pressione di "b", pressione di "c", rilascio di "a", rilascio di "b", rilascio di "c".
Su HOTAS USB, l'ordine dei caratteri inviati rispecchia quello dei codici USB degli stessi (e per le lettere questo equivale all'ordine alfabetico). Quindi:

BTN S4 {c b a}

genera comunque pressione di "a", pressione di "b", pressione di "c", rilascio di "a", rilascio di "b", rilascio di "c". In ogni caso il risultato finale è che questi vengono generati ***nello stesso istante*** *(tecnicamente nello stesso frame) quindi la situazione è molto vicina a quando i tasti vengono premuti contemporaneamente. Volendo separare le operazioni di pressione e rilascio è necessario usare* KD *e* KU *in questo modo:*

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

3.7.3 < > minore e maggiore

Questi caratteri sono stati introdotti con HOTAS Cougar. Tutto ciò che appare tra questi caratteri forza l'HOTAS a non elaborare altre espressioni prima del completamento della corrente. Ad esempio:

```
BTN H1D q DLY(60) q DLY(60) 6 Rem Vector for Homeplate in Falcon 4
```

può portare a un risultato disastroso se venisse premuto un altro tasto contemporaneamente visto che questi potrebbe interferire con la sequenza. Aggiungendo < >:

```
BTN H1D <q DLY(60) q DLY(60) 6> Rem Vector for Homeplate
```

all'abbassamento di Hat 1, tutta l'espressione dovrà essere stata generata prima che qualsiasi altra possa essere interpretata. Questo aiuta molto nella prevenzione di tasti bloccati:

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

Premendo il pulsante T4 ci si assicura che l'espressione /P venga ultimata. Anche se il tasto T4 venisse rilasciato prima, KU (b) verrà generato al termine dell'espressione tra < >. Se non fossero stati usati questi caratteri di raggruppamento, la pressione edi rilascio di T4 in rapida sequenza avrebbero portato a generare prima KU(b) e poi KD(b) con il risultato che il tasto b sarebbe rimasto bloccato in posizione premuta..

NOTE

1. *È impossibile usare < > all'interno di { }, quindi:*

```
BTN S2 { < a b > }
```
genererà un errore.

2. *È impossibile nidificare < >, quindi:*

```
BTN T4 < < a b > >
```
genererà un errore.

3. *< > non necessariamente devono contenere tutta l'espressione, quindi:*

```
BTN S1 a b <c d > e f
```
è corretto.

4. *Un'espressione di questo tipo:*

```
BTN S2 /H <a b c>
```

Appena generati "a" e "b" manterrà la pressione su "c" e la parte forzata dell'espressione terminerà. L'espressione /H *su più caratteri interviene solo sull'ultimo della sequenza.*

5. *Il funzionamento di < > prevede che se altre espressioni sono in esecuzione, esse continueranno ad essere eseguite. Quindi, mantenendo premuto un pulsante legato ad un modificatore* /H, *questo continuerà a lavorare anche se si dovesse premere un pulsante con un'espressione tra < >. Altri pulsanti verranno memorizzati ed interpretati al termine dell'espressione tra < >. È ovvio che va fatta molta attenzione ad usare lunghe pause con* DLY ( ) in espressioni tra < > *poiché è possibile realmente bloccare il controller.* Vedere la parte di *Risoluzione problemi per maggiori dettagli*.

NOTE AVANZATE

Partendo da questo esempio:

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

L'espressione verrà generata usando sei frame. Avendo impostato il rate a default è possibile che serva velocizzare l'espressione rilasciano contemporaneamente i tre tasti e non seguendo la velocità impostata del rate. Questo è possibile in questo modo:

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

I KU vengono inviati nello stesso frame e risultano contemporanei.

3.8 Usare e definire i pulsanti DirectX (Direct Input)

Usando un joystick semplice, magari con solo un tasto, risulta immediata l'associazione tra questo tasto e la funzione di fuoco. Questo avviene perché sui dispositivi non programmabili è Windows a indicare la pressione del tasto secondo questa logica: "*Questo joystick ha un grilletto – decidi tu come usarlo"* Il grilletto è semplicemente un pulsante. Questo tipo di pulsanti sono chiamati pulsanti DirectX ed hanno funzioni che vengono assegnate esclusivamente all'interno del gioco..

Espressione di configurazione	
Sintassi	USE *identificatore_pulsante* o *flag* AS DX*n* BTN *identificatore_pulsante* DX*n*

dove:

identificatore del pulsante è H1U, T6, S2 etc.
flag è X1 to X32 (*spiegato più avanti*)
n vale da 1 a 28

HOTAS Cougar consiste di 10 assi analogici (compresa la pedaliera con freni), 28 pulsanti e un POV HAT (Hat 1). Quando Cougar è in modalità Windows i pulsanti possono essere assegnati solo all'interno del gioco o simulatore. Questo avviene perché Windows informa il software delle capacità della periferica: quanti assi, pulsanti, POV, ecc sono presenti. Quando programmiamo un file invece, configuriamo direttamente la periferica in modo che essa stessa comunichi dei comandi al simulatore (emulando la tastiera ad esempio).

Nel modo programmato (programmed mode), per default, nessun pulsante viene programmato come DirectX. Il pulsante che più comunemente viene configurato come DirectX è il grilletto (trigger):

```
USE TG1 AS DX1
```

Non è necessario fare altro – il grilletto verrà visto e gestito dal simulatore. Normalmente la sua funzione è quella di fuoco. Altri esempio sono:

```
USE H1U AS DX2
USE X4 AS DX3   Rem assegnato ad un flag – vedere note a seguito
USE T4 AS DX5
```

È possibile programmare espressioni DXn all'interno di altre espressioni, ad esempio:

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

In questo esempio, premendo (e mantenendo la pressione) su S2, verrà generato un carattere "a", una pausa di due secondi e il pulsante DX2 verrà mantenuto premuto. Volendo definire rigorosamente S2 come DX2:

```
USE S2 AS DX2
```

I due differiscono perché in questo caso DX2 viene generato appena S2 viene premuto, mentre nel primo caso esso viene generato dopo "a DLY(2000)".

3.8.1 USE ALL_DIRECTX_BUTTONS

Abbiamo visto come assegnare singoli pulsanti ai comandi DirectX. Ora è giusto introdurre l'espressione USE ALL_DIRECTX_BUTTONS.

Espressione di configurazione
USE ALL_DIRECTX_BUTTONS

Questa assegna tutti i pulsanti ai corrispondenti DirectX. In un file con alcuni pulsanti programmati, questa espressione assegnerà solamente quelli non programmati alle funzioni DirectX, lasciando inalterata la capacità di modificare curve, il funzionamento del mouse, ecc. Qualsiasi opzione di default definita in Foxy verrà ignorata.

Ecco un semplice esempio:

```
Rem Tutti i pulsanti associati ai rispettivi DirectX
USE ALL_DIRECTX_BUTTONS
Rem Assegna il mouse al microstick – vedere le note specifiche
USE MTYPE A3
Rem Assegna l'Hat1 del joystick al POV
USE HAT1 AS POV
```

Caricando il file nel joystick, la mappatura sarà: un joystick con manetta con un POV (Hat 1) e con tutti i pulsanti configurati come DirectX. Da questo punto è possibile iniziare a costruire un file specifico per il simulatore.

NOTE

1. Questa nuova sintassi rimpiazza la precedente *espressione* PORT Bx IS

2. *Queste sono di default le associazioni tra pulsanti del Cougar e pulsanti DirectX in modo 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

con Hat1 che diventa il POV (Point Of View – punto di vista) Hat.

3. *Usando* USE ALL_DIRECTX_BUTTONS *e programmando alcuni pulsanti con funzioni diverse, questi ultimi assumeranno le funzioni programmate e non l'associazione DirectX. Il compilatore converte l'espressione* USE ALL_DIRECTX_BUTTONS *in questo modo:*

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

Se il file S2 venisse ridefinito, la priorità andrebbe alla definizione e non all'associazione DirectX. Questo è un sistema semplice per avere la maggior parte dei pulsanti utili e poter comunque ridefinire le funzioni di alcuni .

4. *Usando* USE ALL_DIRECTX_BUTTONS, USE MTYPE *e/o* USE HATn AS POV (*vedere le note specifiche*)*, queste espressioni* ***devono*** *comparire* ***prima*** *di qualsiasi espressione di definizione dei pulsanti, viceversa il compilatore genererà un errore. È importante definire le espressioni di configurazione prima della definizione di pulsanti e assi.*

5. USE MTYPE, *che vedremo più avanti, permette di assegnare i pulsanti del mouse ai pulsanti da T1 e T6 del controller in base a quale tipo di espressione* MTYPE *viene inserita. Usare* USE MTYPE *insieme a* USE ALL_DIRECTX_BUTTONS*, porterà a non assegnare come pulsanti DirectX quelli riservati da* MTYPE*.*

MTYPE (A1.. A5) *assegna i pulsanti nel seguente modo:*

A1: *T1 = Pulsante sinistro del mouse, T6 = Pulsante destro del mouse*
A2: *T1 = Pulsante destro del mouse, T6 = Pulsante sinistro del mouse*
A3: *T1 = Pulsante sinistro del mouse*
A4: *T6 = Pulsante sinistro del mouse*
A5: *Nessun pulsante assegnato al mouse.*

6. *Usare un HAT diverso da Hat1 come POV:*

```
USE HAT3 AS POV
```

Porterà il compilatore a non assegnare i pulsanti DirectX a quell'Hat. (maggiori dettagli più avanti)

NOTE AVANZATE

1. Prendendo uno degli esempi precedenti*:*

```
USE TG1 AS DX1
```

Il compilatore produrrà il seguente risultato:

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

Dimostra che è possibile gestire gli eventi KeyUp e KeyDown anche con due pulsanti DirectX. Ecco un'applicazione pratica di quanto appena detto:

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

Se nel simulatore i cannoni sono assegnati al pulsante DirectX 1, quando il trigger verrà premuto con S3 sollevato, si sparerà normalmente, mentre con S3 premuto si sparerà a raffiche.

3.9 Usare KD, KU e i codici USB

Ci saranno casi in cui sarà necessario avere più controllo sugli eventi KeyDown e KeyUp, o semplicemente sarà necessario inviare dei codici che definiscono dei caratteri speciali (ad esempio quelli non presenti nella tastiera US). Tutto ciò può essere fatto tramite questi comandi:

Sintassi dei comandi

```
KD(Caratteri / pulsanti DX / pulsanti Mouse)
KU(Caratteri / pulsanti DX / pulsanti Mouse)

USB(eventocodiceHID)
```

3.9.1 KD, KU

Queste espressioni servono a programmare i due eventi principali collegati alla pressione di un tasto sulla tastiera, la pressione tramite il **K**ey**D**own (**KD**), e il rilascio tramite il **K**ey**U**p (**KU**). A volte sarà necessario poter programmare altri caratteri in mezzo a questi due eventi, questo è possibile nel seguente modo:

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

In questo alzando l'HAT1 verrà emulata la pressione della freccia in alto, una pausa di 20ms e il rilascio della freccia in alto. KD and KU possono essere usati su qualsiasi tasto, basta usare la corretta sintassi Thrustmaster.

È possibile combinare più tasti all'interno di KD e KU.

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

È possibile usare KD e KU su pulsanti DirectX, pulsanti del mouse e flag logici (*trattati più avanti*). Ecco un esempio con il pulsante sinistro del mouse (MOUSE_LB):

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

Premendo T6 verrà emulata la pressione del tasto sinistro del mouse, una pausa di 2 secondi e il rilascio del tasto.

3.9.2 Codici USB

È possibile inviare codici USB per generare eventi non previsti dalla sintassi Thrustmaster. Questo può essere molto utile per le tastiere non US. I codici USB vengono forniti nell'Appendice 3 in fondo a questo manuale. Ogni codice ha un prefisso che può essere "D" per rappresentare un KeyDown, o "U" per rappresentare un KeyUp. Ad esempio:

```
BTN T3 /P USB (D51) /R USB (U51) Rem 'Freccia giù'
BTN T4 USB (DE1 D04 UE1 U04) Rem 'Shift a'
```

I codici USB in questo esempio generano pressione di tasti in frame diversi. Volendo usare lo stesso frame (per rendere più eventi contemporanei) è possibile usare le parentesi graffe. Ad esempio:

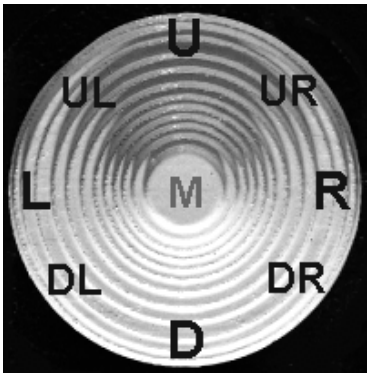
```
BTN T4 USB (DE1 D04) DLY (2000) USB ({UE1 U04}) Rem 'Shift a'
```

Genererà la pressione Left Shift e 'a' , un ritardo di 2 secondi ed il rilascio contemporaneo. È comunque utile sapere che la distanza tra due frame di circa 30 ms quindi l'effetto di queste espressioni è praticamente lo stesso..

4. Programmazione e HAT

4.1 Programmare gli HAT del Joystick

4.1.1 Posizioni programmabili di un hat



Gli hat hanno 9 posizioni programmabili anche se in generale verranno programmate solo le Quattro direzioni principali. Per l'HAT1 ad esempio:

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

anche gli angoli sono programmabili:

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

Così come la posizione centrale:

BTN H1M View_forward

È importante notare che tutte queste posizioni programmabili sono separate e che gli angoli non sono per default il prodotto di quanto programmato nelle due posizioni a lato. Programmando le direzioni alto, basso, destra e sinistra sul tastierino numerico 8, 2, 4 e 6, premendo l'hat in alto a destra (UR) non verranno generati i caratteri 8 e 6, né il 9. Non verrà fatto assolutamente nulla. Chiaramente questo dipende dalla capacità di muoversi in una posizione angolare direttamente.

4.1.2 Hat a 4 vie contro 8 vie: USE HatID FORCED_CORNERS

Se si volesse forzare una posizione angolare a generare la combinazione di caratteri delle due posizioni a lato, ecco l'espressione da usare:

Espressione di configurazione
USE *HatID* FORCED_CORNERS

dove: *HatID* può essere HAT1, HAT2, HAT3, HAT4
es. USE HAT1 FORCED_CORNERS

(*Esiste un'espressione per semplificare l'uso delle posizioni angolari: vedi USE HatID_SENSITIVITY(nnnn) più avanti*)

Un HAT può essere assegnato alla programmazione, in modo normale, come mouse, come POV (Point Of View), come frecce o come tastierino numerico. Esistono 4 espressioni di configurazione da usare con gli hat. Esse sono:

Espressione di configurazione
USE *HatID* AS MOUSE (rateo) [- *modificatori opzionali*] USE *HatID* AS POV [- *modificatori opzionali*] USE *HatID* AS ARROWKEYS [-*modificatori opzionali*] USE *HatID* AS KEYPAD [-*modificatori opzionali*]

dove:

HatID può essere HAT1, HAT2, HAT3, HAT4, RADIOSWITCH

(*RADIOSWITCH anche se non sembra un hat è formato da T2 (Su), T3 (Giù), T5(Sinistra), T4 (Destra). Differisce leggermente dai 4 hat normali in quanto ha l'espressione FORCED_CORNERS cablata nella sua definizione*)

Rateo vale da 1 a 127 e si usa solo con USE *HatID* AS MOUSE (rateo).

[*-modificatori opzionali*] Possono essere usati nelle espressioni per modificarne il comportamento. Consistono in:

REVERSE_UD, REVERSE_LR, FORCED_CORNERS, NOHOLD, KP5.

Non tutte le precedenti possono essere usate in tutte le espressioni – vedere più avanti per i dettagli o usare il Foxy Composer.

4.1.3 Controllare il mouse con un HAT.

Espressione di configurazione

USE *HatID* AS MOUSE (rateo) [- *modificatori opzionali*]

Rateo: - la velocità del movimento del mouse, da 1 a 127
Modificatori opzionali permessi: REVERSE_UD, REVERSE_LR

es. USE HAT1 AS MOUSE (2)

In questo modo si assegna il controllo del mouse all'HAT1. Il valore tra parentesi determina la velocità di spostamento del cursore sullo schermo – un valore basso genererà un movimento lento, uno alto in un movimento più rapido. Se l'hat viene mosso in un angolo il mouse si muoverà in diagonale.

Volendo invertire le direzioni Su e Giù del mouse si può usare:

USE HAT1 AS MOUSE (2) - REVERSE_UD

similmente per Destra e Sinistra:

USE HAT1 AS MOUSE (2) - REVERSE_LR

È possibile usare i due REVERSE insieme:

USE HAT1 AS MOUSE (2) - REVERSE_UD, REVERSE_LR

NOTE

È possibile controllare il mouse con un hat e contemporaneamente con il microstick o qualsiasi altro pulsante. È possibile anche controllare il mouse per una determinata combinazione del selettore di dogfight sulla manetta (/U, /M, /D) e/o di *S3* (/I, /O). *Questo verrà spiegato più avanti nella sezione: Capire il Mouse Device ed il Microstick.*

4.1.4 Un HAT come Point Of View (POV)

Espressione di configurazione

USE *HatID* AS POV [- *modificatori opzionali*]

Modificatori opzionali permessi: REVERSE_UD, REVERSE_LR

es. USE HAT3 AS POV

Usando il controller in modo Windows o in modo programmato (senza definire BTN H1x nel file joystick), l'HAT 1 si comporterà come un POV HAT. Il POV HAT è un tipo specifico di HAT usato da molti simulatori. Ad esempio in Falcon4, non programmando HAT 1, questo si comporterà come POV HAT e controllerà le visuali interne. In maniera similare, è possibile invertire le direzioni del POV:

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

È altresì possibile programmare le singole posizioni del POV usando direttamente POVU, POVD ecc., anche senza aver assegnato un hat alla funzione POV. È possibile programmare qualsiasi posizione sia necessaria (e omettere quelle non necessarie). È come usare i pulsanti DX - esistono, ma devono essere programmati per funzionare. Appena l'hardware rileva un POV sulla barra in uso, è possibile programmare le posizioni POV. Vedere le Note per maggiori dettagli.

NOTE

È molto semplice assegnare un hat alla funzione POV, ma è molto importante sottolineare che le singole posizioni POV possono essere assegante separatamente. La sintassi di queste ultime è simile a quella di un normale hat:

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

un esempio pratico:

BTN T5 POVL
BTN T4 POVR

4.1.5 Usare un HAT per emulare i tasti freccia

Espressione di configurazione

USE *HatID* AS ARROWKEYS [- *modificatori opzionali*]

Modificatori opzionali permessi: REVERSE_UD, REVERSE_LR, NOHOLD

es. USE HAT2 AS ARROWKEYS

È molto frequente nei simulatori di volo la necessità di associare le frecce ad un hat. Questo è lo scopo di questa espressione. Le frecce verranno tenute premute per la durata della pressione dell'hat. Così come le precedenti, questa espressione permette di invertire le direzioni:

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

Muovendo l'hat in un angolo verranno generati i due tasti corrispondenti alla posizione. In effetti questa espressione ha un FORCED_CORNERS cablato.
Se non si volesse il mantenimento della pressione sui tasti è possibile usare il modificatore NOHOLD in questo modo:

USE HAT3 AS ARROWKEYS - NOHOLD

Questo produrrà una singola pressione (e rilascio) delle frecce anche se l'hat venisse mantenuto premuto. Anche questo modificatore può essere usato insieme agli altri:

USE HAT1 AS ARROWKEYS - REVERSE_UD, REVERSE_LR, NOHOLD

4.1.6 Usare un HAT per emulare il tastierino numerico

Espressione di configurazione

USE *HatID* AS KEYPAD [- *modificatori opzionali*]

Modificatori opzionali permessi:
REVERSE_UD, REVERSE_LR, FORCED_CORNERS, NOHOLD, KP5

es. USE HAT4 AS KEYPAD

È altrettanto comune dover usare un HAT per emulare il tastierino numerico. Con questa espressione, HAT4 genererà i tasti associati al tastierino numerico, comprese le posizioni d'angolo (UL UR, DL, DR) che genereranno i tasti "7 9 1 3" rispettivamente.

Il problema con il tastierino numerico è che il suo comportamento varia profondamente con il simulatore o il gioco. Non solo, simulatori diversi si comportano differentemente in base anche allo stato del Num Lock (Bloc Scorr). In molti casi comunque assumere che il tastierino generi semplicemente dei numeri è più che sufficiente..

Similmente a prima è possibile invertire le direzioni:

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

È stato detto che l'espressione genera la pressione dei tasti sul tastierino numerico, ma ce n'è uno che manca all'appello: il numero "5" (KP5). Alcuni simulatori assegnano tale tasto alla funzione "centra" ed è possibile fare in modo che la generazione di questo tasto sia associata alla posizione centrale dell'hat:

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

E anche possibile forzare le posizione d'angolo,

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

Questo farà in modo che le posizioni d'angolo vengano generate in maniera composta: muovendo l'hat in posizione UR, invece di generare il KP9, verranno generati KP8 e KP6 insieme..
Questo potrebbe non aver molto senso se si pensa all'emulazione del tastierino numerico, ma nel contesto di giochi e simulatori è corretto.

Se non si volesse mantenere la pressione su tasti, possibile usare anche il modificatore NOHOLD:

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

Questo produrrà delle singole pressioni dei tasti anche se l'hat venisse mantenuto spostato.

È anche possibile combinare questi modificatori:

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

4.1.7 Come il compilatore converte gli USE HatID AS

Questa sezione è per gli utenti avanzati con una elevata curiosità per i dettagli!

È necessario aver letto alcune sezioni più avanti di questo manuale prima di poter capire appieno queste note, ma per i tecnici vedremo come il compilatore converte i vari USE HAT*n* AS *qualcosa*. Iniziamo con USE HATx AS MOUSE.

USE HAT1 AS MOUSE (2) è convertito in:

```
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+)
```

Similarmente, USE HAT1 AS MOUSE (2) - REVERSE_UD

in:
```
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+)
```

infine: USE HAT1 AS MOUSE (2) - REVERSE_UD, REVERSE_LR

in:
```
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-)
```

Ecco la conversione di ARROWKEYS: USE HAT2 AS ARROWKEYS

convertito in:

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

Allo stesso modo: USE HAT2 AS ARROWKEYS - REVERSE_UD, NOHOLD

in:

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

Osservate come le posizioni Up and Down sono invertire, e come l'aggiunta di NOHOLD rimuova gli /H.

Ecco il KEYPAD: USE HAT4 AS KEYPAD

convertito in:

```
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
```

similarmente: USE HAT4 AS KEYPAD - REVERSE_UD, NOHOLD

in:

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

Le posizioni d'angolo vengono invertite quando viene invertita una direzione e l'uso di NOHOLD rimuove gli /H.

Tutto ciò è vero anche per REVERSE_LR:

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

convertito in:

```
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
```

Le posizioni vengono nuovamente invertite con l'inversione delle direzioni. Forzando le posizioni angolari con:

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

Il compilatore produrrà:

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

Per concludere:

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

Farà aggiungere un'espressione così:

```
BTN H4M KP5
```

Notare che non è presente nessun /H. Il KP5 è un carattere non ripetibile. Questo è abbastanza per ora, passiamo ad altro….se non lo avete già fatto!

5. Espressioni di configurazione

5.1 Introduzione

All'inizio di questo manuale, abbiamo discusso il layout di un file joystick. Nonostante le espressioni possano apparire in tutta la lunghezza del file, abbiamo definito un'area che precede le espressioni di programmazione e l'abbiamo riservata alle espressioni di configurazione. Alcune di queste espressioni sono già state introdotte, ad esempio USE MDEF. Semplicemente non abbiamo voluto fare confusione spiegando questa differenza precedentemente. Ora è il momento di chiarire le cose visto che dovreste essere più a vostro agio con la programmazione del controller.

Le espressioni di configurazione si applicano all'intero file, e non applicano a posizioni specifiche del controller nonostante alcune possano essere programmate dentro un espressione di pulsante. Esse indicano al compilatore come configurare il controller per il simulatore. Includono espressioni che indicano quale file macro usare, a che velocità generare i caratteri durante le ripetizioni, quali assi disabilitare, ecc.

Molte espressioni di configurazione sono descritte altrove in questo manuale, proprio nel posto dove è più significativo che siano. Qui vorrei descrivere quelle che non hanno trovato posto altrove.

La sintassi è cambiata per HOTAS Cougar parlando di espressioni di configurazione – ecco la regola d'oro:

Tutte le espressioni di configurazione iniziano con USE o DISABLE

Tutte le espressioni di configurazione della programmazione logica iniziano con DEF

Questo differisce dalle precedenti versioni della sintassi Thrustmaster. Non fatevi intimorire dal termine "Programmazione logica" che più volte è apparso finora. Viene menzionato per completezza ed è pertinente a cose che discuteremo più avanti visto che sono pane per i programmatori più esperti ed esigenti.

5.2 MDEF - Macro DEFinition File

Espressione di configurazione
USE MDEF *macro_filename*

Questa espressione è necessaria solo se il file joystick contiene macro (*cosa consigliabile*). Il *macro_filename* è il nome del file che contiene le macro senza la sua estensione (.tmm).

Ad esempio, avendo un file joystick: Janes WW2 Fighters.tmj e il suo file macro Janes WW2 Fighters.tmm, allora l'espressione MDEF sarà:

USE MDEF Janes WW2 fighters

È fondamentale che entrambi i file stiano nella stessa directory e, per default, la cartella File di Foxy. È comunque possibile realizzare il software come se così non fosse, ma abbiamo ritenuto questo sistema più pratico e simile all'originale HOTAS Thrustmaster.

NOTE

1. *I nomi dei file possono contenere testo lungo e con spazi.*

2. *È irrilevante il fatto di mettere o meno l'estensione del file nell'*MDEF*. Entrambe queste espressioni non valide:*

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

3. *I nomi dei file macro e joystick non sono case sensitive*

Mig Alley.tmm

È lo stesso file anche se scritto cosi:

mig alley.tmm

4. *Nel HOTAS Thrustmaster originale, era possibile usare più file macro e quindi usare più* MDEF*. Questo non è più possibile con HOTAS Cougar.*

5.3 RATE

Escludendo le sottostanti, la maggior parte delle espressioni di configurazione riguardano la programmazione degli assi che è spiegata più avanti in questa sezione.

Espressione di configurazione

USE RATE (*nnnn*)

Sintassi

RATE (*nnnn*)

dove *nnnn* è un tempo in millisecondi (1000ms = 1 secondo) che determina il periodo usato per la ripetizione dei caratteri. Se omesso, il compilatore userà il default USE RATE (0) producendo i caratteri alla frequenza di ripetizione di default della tastiera. Più **grande** sarà il valore, più **lenta** sarà la ripetizione. I valori possono variare tra 0 e 655350 (*oltre 10 minuti!)*.

È anche possibile cambiare il valore di RATE in tempo reale, programmandolo su un pulsante:

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

o in un'espressione di asse:

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

(*vedere le note più avanti sulla programmazione degli assi digitali*)

NOTE AVANZATE

HOTAS Cougar produce caratteri in gruppi chiamati frame. I frame sono generati ad intervalli di circa 30ms se il RATE *è impostato a 0 o non è impostato direttamente. All'interno di un frame, il Cougar può generare 16 caratteri. Se vengono prodotti più caratteri del massimo per un frame, i restanti verranno inviati nel frame successivo. Il* RATE *determina l'intervallo di tempo tra un frame ed il successivo.*

5.4 S3_LOCK e S3_UNLOCK

Espressione di configurazione

USE S3_LOCK

Sintassi

S3_LOCK
S3_UNLOCK

Normalmente BTN S3 sul joystick viene usato in modo che quando premuto vengano generate le espressioni /I mentre quando rilasciato subentrino le /O.

USE S3_LOCK significa che quando S3 viene premuto, Cougar userà solo le espressioni /I. Ripremendolo Cougar genererà le espressioni /O.

Volendo cambiare tra i due stati tramite un altro pulsante è possibile usare la seguente espressione:

BTN S2 /T S3_LOCK /T S3_UNLOCK

Non è necessario usare l'espressione USE S3_LOCK se sono già presenti espressioni che usano direttamente S3_LOCK, S3_UNLOCK su un pulsante. Qual è la differenza tra USE S3_LOCK e solo S3_LOCK? USE S3_LOCK si applica all'intero file ed è attivo appena il file viene caricato ed abilitato nel Cougar. Al contrario S3_LOCK su un pulsante si applica solo a quel pulsante e viene attivato solo quando il pulsante stesso viene premuto.

NOTE

1. *Assegnando un diverso pulsante/hat come S3 (vedere la prossima sezione) le espressioni precedenti si applicheranno a quel pulsante usando la stessa sintassi.*

2. *Non è possibile usare il modificatore di trattenimento* (/H) *con* S3_LOCK. *Quindi:*

BTN S1 /H S3_LOCK
BTN S4 /H *macro* S3_LOCK

genereranno errori di compilazione.

5.5 Assegnare un diverso pulsante per /I e /O con SHIFTBTN

Espressione di configurazione
USE *identifcatore_del_pulsante* AS SHIFTBTN
Sintassi
SHIFTBTN (*identifcatore_del_pulsante)*

Esempi:

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

Determina quale pulsante usare al posto di S3 per selezionare le espressioni /I. Se questa espressione mancasse (situazione molto probabile) allora il pulsante usato sarà S3.

5.6 USE HAT SENSITIVITY – SENSIBILITÀ AGLI ANGOLI

A volte può essere molto difficile trovare la posizione d'angolo di un hat (ad esempio H4UL) Gli hat sono semplicemente del tasti a quattro vie. Il controller elabora i segnali molto velocemente e può capitare che vengano generate le macro delle due posizioni adiacenti a quella angolare prima che si riesca a centrare l'angolo con l'hat. È possibile ridurre la sensibilità dell'hat in modo da eliminare questo problema di transizione dal riposo all'angolo.

Espressione di configurazione
USE *HatID*_SENSITIVITY (*nnnn*)

dove:

HatID è uno dei 4 hat: HAT1, HAT2, HAT3, HAT4
nnnn è un valore tra 0 (più sensibilità) a 1000 (meno sensibilità). *nnnn* è attualmente un ritardo in millisecondi. Ad esempio:

```
USE HAT1_SENSITIVITY (100)
```

vuol dire che le espressioni di HAT1 verranno generate solo dopo che una posizione3 verrà mantenuta per 100 millisecondi. In questo modo si ha più tempo per raggiungere la posizione voluta senza rischiare di eseguire altre macro indesiderate.

NOTE

Non è possibile usare il modificatore /T *sulle posizione degli hat se su questi è stata usata un'espressione* USE *HatID*_SENSITIVITY (*nnnn*).

Quindi: USE HAT1_SENSITIVITY (*60*)
BTN H1U /T a /T b /T c *genererà un errore di compilazione.*

5.7 USE T1 SENSITIVITY

Se anche il tasto T1 risultasse troppo sensibile (o soggetto a pressioni accidentali) è possibile renderlo meno sensibile con:

Espressione di configurazione

USE T1_SENSITIVITY (*nnnn*)

dove:

nnnn è un valore numerico che può variare tra 0 (più sensibile) a 1000 (meno sensibile). *nnnn* è un ritardo in millisecondi e rappresenta la durata minima della pressione necessaria perché il pulsante T1 venga riconosciuto. Questa funzione è stata implementata per chi preme accidentalmente il pulsante T1.

Ecco un esempio:

USE T1_SENSITIVITY (1000)

In questo caso T1 verrà considerato premuto solo dopo una pressione di almeno un secondo di durata.

NOTE

Non è consentito usare il modificatore /T *su un* BTN T1 *se precedentemente è stato definito un* USE T1_SENSITIVITY (*nnnn*).

5.8 USE FOXY GRAPHIC e README

Espressione di configurazione
USE FOXY GRAPHIC *fileimmagine* USE FOXY README *filetesto*

Queste espressioni vengono ignorate dal compilatore e sono usate solo all'interno di Foxy. Quando Foxy apre i file li legge ed interpreta queste due espressioni caricando l'immagine definita in USE FOXY GRAPHIC *fileimmagine* nell'Image Viewer. Questo è molto utile per la distribuzione dei propri file quando si ha anche un'immagine che rappresenta la mappatura tra macro e pulsanti. Un esempio di questa espressione è:

USE FOXY GRAPHIC Total Air War.bmp

I tipi di immagine permessi sono: bitmap (.bmp), jpeg (.jpg) o gif (.gif).

Allo stesso modo, USE FOXY README, fa caricare a Foxy il file di testo per il Template Editor, che può essere utile come promemoria di come sono stati realizzati i file, quali settaggi sono necessari sul simulatore per il corretto funzionamento, ecc. Un esempio di questa espressione è:

USE FOXY README Total Air War.rtf

I tipi di testo permessi sono i testi semplici con estensione .txt o i Rich Text File, con estensioni .rtf. I Rich Text Files permettono di avere testo colorato, formattazione, font diversi, ecc. e sono molto più facili da leggere.

NOTE

I file immagine e di testo appena descritti devono essere messi nella stessa directory dove si trovano i file macro e joystick. Ricordiamo che per default questa cartella è la Files all'interno di Foxy.

5.9 NULLCHR – Carattere Null ^

Espressione di configurazione
USE NULLCHR *carattere*

Un carattere null è un carattere in una espressione che non genera alcun output. Il carattere null di default è il **^** (caret). A cosa serve? Alcune espressioni richiedono un numero fisso di parametri perché l'espressione sia valida. Alcuni esempio sono le espressioni Digital Type che vedremo nella prossima sezione.. Ad esempio:

```
RDDR 3 L ^ R
```

Programma il timone in modo che generi una pressione del carattere "L" quando viene premuto il pedale sinistro, un carattere "R" quando viene premuto il destro. Quando il timone è nella posizione centrale, non è necessario premere nulla quindi è stato aggiunto un carattere null (^). Non è possibile lasciare vuota la posizione centrale:

```
RDDR 3 L  R
```

Perché l'espressione richiede 3 parametri dopo "RDDR 3". Quindi è importante pensare al carattere null come *riempimento* quando è necessario inserire un carattere, ma praticamente non si vuole intraprende nessuna azione.

Ritornando all'espressione di configurazione, il carattere null di default è **^**. Volendo usare un carattere diverso è possibile scrivere un'espressione di questo tipo:

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

Se il carattere ^ fosse usato nel gioco, è possibile assegnarlo indirettamente tramite l'espressione SHF 6, ad es. BTN S1 SHF 6

NOTE

1. *Con i precedenti controller Thrustmaster, era sbagliato lasciare degli spazi vuoti nelle espressioni e per questo si usava molto il carattere null. Ad esempio:*

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

Questo non è il caso di Cougar. Non c'è alcun problema nel scrivere:

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

nel file joystick.

2. *Il carattere null produce il codice USB (00). Questo equivale a non generare nulla, è anche possibile creare una macro di questo tipo:*

```
Fai_Nulla = USB(00)
```

e usarla nell'espressione in questo modo:

```
RDDR 3 L Fai_Nulla R
```

Chiaramente è molto più veloce e semplice usare il carattere null ed è per questo che è stato creato.

3. *È impossibile usare caratteri modificati con* USE NULLCHR*. Questi esempi genereranno degli errori:*

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

5.10 KEYBOARD (AZERTY, QWERTY)

Espressione di configurazione

USE KEYBOARD *tipo*

Dove *tipo* può essere: AZERTY o QWERTY.

Usando una tastiera francese AZERTY con un gioco che rimappa i tasti, è possibile che questi non funzionino correttamente. Per ovviare a questo inconveniente esiste l'espressione USE KEYBOARD AZERTY. Vedere la sezione Key Tester per avere più informazioni.

NOTE

Non è necessario usare l'espressione USE KEYBOARD QWERTY *perché il compilatore la inserirà per default quando non diversamente specificato.*

5.11 Usare i profili del CCP - USE PROFILE

Espressione di configurazione
USE PROFILE *Profilo* (Modo di calibrazione)

dove:

Profilo: è il profilo creato dal Cougar Control Panel e salvato con l'estensione .tmc nella cartella Profiles.
Modo di calibrazione: può essere AUTO o CUSTOM. Va usato per specificare se si desidera l'autocalibrazione (AUTO) o la calibrazione personalizzata (CUSTOM) che si trova nel profilo.

Esempi:

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

Come vedremo nella programmazione degli assi, è possibile cambiarne la configurazione in modo che risultino invertiti, scambiati, disabilitati, con curve diverse, ecc. Questo può risultate molto complesso da gestire. Se si usa il CCP per configurare un profilo (e lo si salva) si può semplificare molto il processo. Usare un profilo salvato permette anche di definire le zone morte (cosa non possibile diversamente). Per esperienza, il caricamento è più veloce se si usa un profilo al posto di DISABLE o USE AXES_CONFIG (*vedere le note più avanti*)

5.11.1 Sui profili

Vorrei spendere qualche parola sui file .tmc (i profili) e perché è una buona idea includerli nei file joystick. Per iniziare, i profili sono creati usando il Cougar Control Panel (CCP). Essi contengono le informazioni sugli assi configurabili dal CCP (mappature, zone morte, ecc.). Caricando un file che contiene queste informazioni o usando un file joystick che modifica questi parametri durante il volo, una volta usciti dal simulatore rimarranno all'interno di Cougar. Questo perché il controller non ha driver e quindi memorizza tutte le informazioni al suo interno. Aprendo un secondo simulatore, gli assi non verranno resettati e verranno mantenute le impostazioni precedenti. Per ovviare a questo problema ci sono due modi. È possibile usare l'espressione USE PROFILE in ogni file joystick usando il file DEFAULT.tmc (*il profilo di default creato dal CCP al primo avvio*) o un profilo scelto tra quelli presenti per quel simulatore. In Foxy è anche possibile indicare al compilatore di resettare i valori degli assi prima di caricare i file. Spero che tutto ciò abbia senso adesso.

Per finire, visto che un profilo contiene anche la calibrazione, questa è la vera ragione per cui è necessario specificare al compilatore la necessità o meno di usare la calibrazione automatica o quella presente nel file.

NOTE

1. *Foxy ed il compilatore assumono che il file dei profili sia nella cartella Profiles del Cougar Software. In effetti questa è la cartella dove finiscono tutti i profili salvati dal Cougar Control Panel. Per definizione questa cartella è* C:\Program Files\Hotas\Profiles *(o* C:\Programmi\Hotas\Profiles *per la versione italiana di Windows). Quando un file che contiene* USE PROFILE *viene compilato/caricato, allora il compilatore:*

- ***(a)*** *Cercherà il profilo nella cartella default dei profili.*
- ***(b)*** *Se il profilo esiste, lo userà (e non cercherà altrove).*
- ***(c)*** *Se il profilo non esiste, lo cercherà nella cartella dove si trova il file joystick ad esempio la cartella Files di Foxy.*
- ***(d)*** *Se il profilo non verrà trovato neanche al punto precedente, verrà generato un errore.*

2. *I profili hanno estensione* ***tmc****. È ininfluente se questa viene specificata o meno nell'espressione* USE PROFILE *. Quindi:*

USE PROFILE Crimson Skies.tmc *è equivalente a*
USE PROFILE Crimson Skies

3. *Se viene usato Foxy per aprire un file zip contenente i file di qualcun'altro profilo, tutti i file dello zip verranno estratti nella cartella Files di Foxy. È possibile muovere il profilo nella cartella Profiles del Cougar Software, ma questa operazione andrà eseguita a mano tramite l'Explorer di Windows. Si consiglia di tenere tutti i profili nell'apposita cartella Profiles del Cougar Software* (*normalmente* C:\Program Files\HOTAS\Profiles o C:\Programmi\HOTAS\Profiles).

4. *Un profilo consiste principalmente di queste informazioni sugli assi: Mappature, Direzioni, Posizioni centrali, Calibrazioni, Zone Morte, Curve, Trim, Assi disabilitati, l'opzione Apply Axis Disable/Enable View.*

5.12 Espressioni di configurazione descritte altrove in questo manuale

Alcune espressioni sono al di fuori dallo scopo di questo capitolo e vengono trattate nel contesto più adatto per essere chiarite. Le seguenti espressioni sono discusse altrove:

Espressione di configurazione	Descritta in
USE *Btn* AS DXn	Sezione 3.8: Usare e definire i pulsanti DirectX (Direct Input)
USE ALL_DIRECTX_BUTTONS	Sezione 3.8.1: USE ALL_DIRECTX_BUTTONS
USE HAT AS MOUSE, POV, ARROWKEYS, KEYPAD	Sezione 4.1: Programmare gli HAT del Joystick
USE CURVE	Sezione 6.3: Curve di risposta (CURVE)
DISABLE AXIS	Sezione 6.5: Disabilitare gli Assi
USE SWAP	Sezione 6.6: Mappatura degli assi (SWAP)
USE REVERSE	Sezione 6.7: Invertire la direzione di un asse (REVERSE, FORWARD)
USE AXES_CONFIG	Sezione 6.8: L'uso di USE AXES_CONFIG
USE MTYPE	Sezione 7.2: USE MTYPE – la via più facile per assegnare il mouse al microstick
USE Axis_Identifier AS Mouse_Axis	Sezione 7.3.1: Assegnare altri assi al mouse
USE ZERO_MOUSE	Sezione 7.5: USARE ZERO_MOUSE
DISABLE MOUSE	Sezione 7.7: Disabilitare l'assegnamento di default del mouse al microstick
USE SCREEN_RESOLUTION	Sezione 7.8.1: Definire la risoluzione dello schermo
DEF Xn	Sezione 8.2: Definire i flag logici e le espressioni corrispondenti

6. Programmazione degli assi

6.1 Principi di base

6.1.1 Capire la differenza tra analogico e digitale

Andremo ora a vedere come programmare i vari assi del Cougar, sia in digitale che modificando il loro comportamento analogico. Prima però di fare ciò, dobbiamo spiegare la differenza tra un asse analogico e uno digitale, vista la confusione su tali termini.

Molti dei joystick attualmente in commercio lavorano meccanicamente nella stessa maniera. All'interno contengono due potenziometri, o **pots** come vengono chiamati all'inglese. Se possedete una radio e su di essa è presente una manopola per controllare il volume, quando la ruotate state variando la sua resistenza.

In un joystick, tali potenziometri sono connessi perpendicolarmente tra di loro, in modo da misurare il movimento sinistro/destro della barra, l'asse X, e il movimento avanti/indietro, l'asse Y. La posizione in cui viene mosso il joystick può essere determinata tramite tali assi, ad es. di quanto si è mosso lungo l'asse X e Y. I pots dunque forniscono un *range (campo)* di valori quando muovete la barra, e per questo sono chiamate device *analogiche*, al contrario di device *digitali*, che possono fornire solo valori on/off come i tasti della tastiera o i pulsanti del joystick.

Ora, se mi state ancora seguendo e non vi siete spostati sulla prossima sezione, complichiamoci un po' la vita. In un mondo perfetto, i pots forniranno valori precisi e stabili ad ogni punto della rotazione. i pots però possono subire degradazioni da polvere e uso. L'effetto di ciò si traduce in valori leggermente variabili o peggio in un "salto" (o "spike") dei valori. Con l'HOTAS Cougar, i segnali di un potenziometro non vengono passati direttamente al simulatore. Un processore digitale al suo interno legge i valori di un potenziometro e quindi filtra i valori errati per fornire un valore più preciso e stabile. Quindi, anche se i potenziometri sono *analogici*, i segnali sono processati in maniera *digitale.*

Presto arriveremo alle potenzialità degli assi del Cougar e cosa possiamo fare programmandoli digitalmente. Con tutto questo bene in mente, vale la pena di spiegare come fare, visto che può non essere di immediata comprensione la programmazione digitale di un asse analogico. Poniamo ad esempio che la manetta produca valori analogici nel range 0-100. È possibile dividere tale asse in

(per esempio) 5 bande, quindi la banda 1 conterrà i valori da 0 a 19, la banda 2 da 20 a 39 ecc... possiamo poi ideare un'espressione che dica "Quando sei in banda 1, genera il carattere 'a', in banda 2 il carattere 'b' ecc.... Quando parliamo di programmare un asse analogico con un'espressione digitale, ci riferiamo proprio a questo, ma ve lo spiegheremo ancora più avanti.

6.1.2 Gli assi di Cougar

HOTAS Cougar (Joystick, Manetta, Pedali con freni) ha 10 assi analogici fisici. Questi assi possono essere trattati come:

- Periferiche completamente analogiche, *(assumendo che il vostro gioco e le DirectX li supportino)* ed è quindi possibile assegnarvi controlli
- Periferiche completamente analogiche, quindi possono essere programmati per generare caratteri da tastiera
- Oppure una combinazione dei due
-

Inoltre possiamo influire sugli assi in termini di:

- rimuoverli completamente
- applicare curve di risposta differenti
- applicare valori di aggiustamento
- invertire il funzionamento degli assi
- rimapparli su altri assi, sia permanentemente che in riferimento alla posizione del selettore dogfight e di S3.

Uno dei punti di forza di Cougar è proprio quello che si può fare con gli assi. Tuttavia questo può risultare un po' complicato molto velocemente! Quindi, per provare e semplificare la comprensione di cosa possiamo ottenere da un file di configurazione in termini di programmabilità degli assi, dobbiamo prima definire le **6 espressioni digitali di configurazione.** Queste sono utilizzate per programmare gli assi digitalmente per ottenere la generazione di caratteri. Dopo averle definite, saremo già a buon punto. Potremo iniziare a capire come fare in modo che il simulatore "veda" gli assi analogici nella maniera che vogliamo noi.
Prima di andare avanti, definiamo la sintassi riguardante questi 10 assi fisici:

Sintassi TM	Asse
JOYX	Joystick X
JOYY	Joystick Y
THR	Manetta
RNG	Range knob (Manopola più piccola sulla manetta)
ANT	Antenna knob (Manopola più grande sulla manetta
MIX	Microstick X
MIY	Microstick Y
LBRK	Left Toe Brake (Freno pedale sinistro)
RBRK	Right Toe Brake (Freno pedale destro)
RDDR	Rudder (Pedaliera)

* Note: le posizioni X e Y del microstick sono un'altra cosa rispetto a Mouse X,Y.

NOTE

1. *Con HOTAS Thrustmaster originale, un asse era o analogico o digitale – ad es. era visto come un asse di default da un gioco e gli veniva assegnata una funzione (es. il TQS = Manetta nel gioco) oppure programmato digitalmente per generare caratteri da tastiera. Questo non è il caso di Cougar. Di default gli assi sono visti come analogici, ma se li programmate tramite le espressioni digitali, saranno* ***sia*** *analogici* ***sia*** *digitali. Se si richiede un asse totalmente digitale, l'asse in questione dovrebbe essere disabilitato.*

2. *Non c'è bisogno di modificare nulla nel caso vogliate usare un asse digitale. Né nel pannello di controllo o nell'applet di controllo delle periferiche. Questo doveva essere fatto con i joystick Thrustmaster precedenti, ma non è il caso di Cougar.*

3. *Con tutte le espressioni digitali per gli assi:*

- *Potete usare i modificatori* /U, /M, /D, /I, /O.
- *Ogni modifica analogica dell'asse (curva, mappatura, inversione) non influenza le espressioni digitali. Queste rimangono sugli assi fisici e sono lineari*

6.2 Espressioni Digitali

in questa sezione spieghiamo come programmare digitalmente gli assi, in modo che riproducano caratteri di tastiera. Questo comportamento si ottiene utilizzando una delle sei espressioni digitali. La via più semplice per capire queste espressioni, è guardare un esempio per ogni tipo e vedere cosa produce.

Nota: *non voglio complicarvi le cose proprio ora, ma tenete in mente che non sono solo i caratteri che possono essere usati in queste espressioni, ma anche flag logici, espressioni per il mouse e espressioni per le curve degli assi.*

Nota: per semplicità le espressioni verranno identificate con la notazione americana, quindi un'espressione di tipo 1 sarà una Type 1, la tipo 2 una Type 2 ecc.... questo per semplificarne l'utilizzo ed unificare le varie notazioni.

6.2.1 Type 1: ripetere la generazione del carattere

Un'espressione Type 1 avrà questa sintassi:

Identif. asse	Tipo Espress. Digitale	Numero di caratteri o macro (max. 50)	Carattere SU o macro	Carattere Giù o macro	Carattere di centro o macro *(optional)*	FORCE_ MACROS *(optional)*
eg. ANT	1	10	u	d	c	- FORCE_MACROS

Oppure, come apparirà nel file di configurazione:

ANT 1 10 u d c - FORCE_MACROS

Ruotare la manopola ANT in senso orario e quindi in senso antiorario produrrà i seguenti caratteri:

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

Non siamo obbligati a utilizzare caratteri singoli nelle espressioni digitali, quindi potremmo avere definite delle macro (nel file apposito) come:

Chaff_Flare = c DLY(30) f
Getting_desperate = RPT(20) (c f)

ed avere un'espressione Type 1 per la manopola RNG nel file joystick:

RNG 1 5 Chaff_Flare Getting_desperate

Notate che il Numero di caratteri ora definisce il numero totale di caratteri generati (*escludendo ogni carattere identificante la posizione centrale)* per la **completa** escursione dell'asse. Questo è differente rispetto alle espressioni Type 1 originarie di Thrustmaster, nelle quali il numero di caratteri definiva il numero di caratteri generati da tale asse passando da un'estremità al centro e poi dal centro all'altra estremità. La ragione del cambio della sintassi è perché ora il carattere centrale è un optional nell'espressione. Ad esempio:

RNG 1 6 u d

Produce i seguenti caratteri ruotando la manopola RNG:

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

Notate che escludere un carattere di centro **è diverso** rispetto all'utilizzo di un carattere nullo (di default è ^) per il centro. Quindi l'espressione:

RNG 1 6 u d ^

produce:

u u u *dead-zone* u u u d d d *dead-zone* d d d

il carattere "^" risulterà in un ...nulla di fatto! Una specie di zona morta.
Prima di lasciare questo punto, il Numero di caratteri **deve essere un numero preciso** *se* viene fornito un carattere centrale, altrimenti se non c'è il carattere centrale, può essere qualunque cosa. La ragione spero sia ovvia: se in numero o il carattere richiesto è 20 e c'è un carattere di centro, vorrete avere a disposizione 10 caratteri per ogni lato del centro!

6.2.1.1 Comprendere il modificatore - FORCE_MACROS

Questo modificatore è opzionale e può essere utilizzato solo con espressioni digitali Type 1, 2, 5 e 6. utilizzerò un'espressione Type 1 per spiegarvi il suo significato ma sappiate che si applica in egual modo anche alle altre espressioni con cui può essere usato.

Poniamo di avere:

RNG 1 50 u d

La rotazione della manopola RNG da un estremo all'altro, produrrà 50 caratteri "u" o 50 caratteri "d", a seconda della direzione in cui si muove. Ora, se muovete velocemente la RNG e controllate i risultati in Notepad o nel Foxy's Key Tester, non vedrete 50 caratteri prodotti. Ne vedrete 10 o 20, ma non 50. che sta succedendo? È un bug – voglio dire, Cougar non è progettato per processare le espressioni molto velocemente? Si, Cougar processa le espressioni molto velocemente e in parallelo, e questa è la reale ragione per cui vedere questo effetto. Il problema è che quando la manopola viene ruotata molto velocemente, non viene saltato nessun carattere. Quando vengono generati più di 16 pressioni/rilasci di tasti contemporaneamente (processo parallelo) il computer molto probabilmente li vedrà come una sola pressione/rilascio. Possiamo modificare questo comportamento forzando il computer a vedere ogni singolo carattere, così

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

e questo è essenzialmente ciò che il modificatore - FORCE_MACROS fa. Esso incapsula ogni carattere/macro in una espressione digitale con (**<** *carattere/macro* **>**). State attenti ad utilizzare questo modificatore però – se forzate delle espressioni in una posizione altre espressioni potrebbero non venire prodotte finché non è terminata la precedente. Ancora, fate attenzione che nessuna macro che viene forzata con - FORCE_MACROS abbia nella sua definizione, i modificatori "< >". Non potete concatenare modificatori force e lo farete se avrete:

Macro_1 = < a b c>
Macro_2 = d

RNG 1 50 Macro_1 Macro_2 - FORCE_MACROS

visto che il compilatore lo convertirà in:

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

NOTE

1. *Non potete avere:* RNG 1 3 a b c d e f

ma potete avere: RNG 1 3 (a b c) (d e f)
oppure RNG 1 3 ABC_macro DEF_macro

e, nel vostro file macro: ABC_macro = a b c
DEF_macro = d e f

2. *Potete usare* /U, /M, /D, /I, /O *con ogni espressione digitale, ad esempio:*

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. *Potete anche usare* /P, /R *e* /H *all'interno delle vostre macro o all'interno di parentesi, direttamente nelle espressioni Type 1,2,5 e 6. ad esempio:*

RNG 1 3 Macro_1 Macro_2

e, nel vostro file macro:

Macro_1 = /P a /R b
Macro_2 = /H d Rem Ma perché dovreste farlo?!

4. *Non potete usare il modificatore* /T *all'interno di espressioni Type 1 (o* ***ogni*** *asse digitale).*

5. *Un'espressione Type 6 è un caso particolare del Type 1. quindi tali espressioni producono risposte identiche dalla manopola dell'Antenna* (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 Considerazioni importanti sull'utilizzo di FORCE_MACROS

NOTE AVANZATE

Può sembrare una buona idea usare FORCE_MACROS *sempre e comunque, ma sotto certi aspetti questo non è vero. Il primo problema a cui dovete pensare è l'effetto non solo sulla risposta di altri pulsanti e hat programmati, ma anche la risposta degli assi a cui avete aggiunto* FORCE_MACROS. *Vediamo un esempio che mi aiuterà a spiegarvi.*
Considerate la seguente espressione Type 2 per gli assi:

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

Ora, non serve un genio per capire che, ignorando la sintassi, questa espressione sostanzialmente programma la manopola per l'antenna sulla manetta in modo da farle generare l'alfabeto.

Ora, se aggiungete questa linea al vostro file di configurazione e lo caricate sul joystick, la rotazione della manopola ANT produrrà l'alfabeto. Potete verificarlo con il Foxy's Key Tester. Se modificate l'espressione aggiungendovi il modificatore FORCE_MACROS*, in questo modo:*

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

e ricaricate il file, noterete che l'output di ANT è molto più lento – per la precisione 4 volte più lento. Questo perché per ogni carattere, diciamo la "a", il compilatore la converte in < KD(a) KU(a) >. *Ognuna di queste 4 componenti richiede il proprio "frame" temporale per essere eseguita.*

Meglio spiegare i frame prima di continuare. Quando Cougar vuole mandare in output i caratteri o le espressioni programmate, lo fa ogni 30 millisecondi. Se volete, questo è il suo frame rate. Ogni frame può contenere diversi caratteri – non è un carattere per frame (ricordate che Cougar processa in parallelo fino a 32 macro contemporaneamente). Quindi nel primo esempio ho programmato l'alfabeto su ANT e il Cougar ha inviato diversi caratteri per ogni frame, a seconda di quanto velocemente avete ruotato la manopola. Molti comandi Key Down (tasto premuto) escono insieme, seguiti dal corrispondente Key Up (tasto rilasciato). Ci torneremo più avanti....

In ogni caso, tornando al modificatore FORCE_MACROS*, ora il carattere "a" viene convertito in* < KD(a) KU(a) >. *Questi 4 componenti dell'espressione* (<, KD(a), KU(a) e >) *vanno in frame separati, ed ecco spiegato perché l'intero alfabeto viene generato* ***molto*** *più lentamente.*

Mentre stiamo parlando di questa espressione, vorrei andare un attimo "off topic" e spiegare un'ulteriore caratteristica del Cougar – questa abilità di inviare caratteri multipli nello stesso frame e uno degli effetti ad esso legati, in relazione all'ordine in cui vengono prodotti i caratteri. Vediamo l'espressione nuovamente:

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

Penserete che con questa espressione, ruotare ANT in una direzione produce l'alfabeto in ... ordine alfabetico e ruotarla nella direzione opposta produca l'esatto contrario. Bene, proviamo un attimo e vediamo cosa accade quando ruotate la manopola velocemente. Vedrete che alcuni dei caratteri saranno prodotti fuori ordine. Questo si nota molto di più ruotando la manopola in senso antiorario e la mia ha prodotto:

"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"

noterete anche che nella finestra degli eventi Up/Down/Keycode, i caratteri sono prodotti come gruppo di caratteri Down, seguiti da un gruppo di caratteri Up, ad esempio Down(z y x) Up (z y x) invece che Down (z) Up (z) Down (y) Up (y) *ecc... Perciò, cos'è questo? Un bug? No, non lo è. È il modo in cui l'USB invia i caratteri della tastiera. Con i vecchi controller Thrustmaster, i caratteri venivano inviati utilizzando la porta PS/2 standard la quale tratta i caratteri come eventi individuali Key Down, Key Up. Con le periferiche USB la modalità è la seguente: i caratteri non sono inviati così come sono ma il sistema operativo esegue uno scan di un buffer per capire quali tasti sono stati premuti e quali no. Quindi, quando Cougar modifica il buffer per indicare che i tasti z, x e y sono stati premuti, il sistema operativo li acquisisce durante lo scan e li riproduce in un ordine predefinito, nel caso dei caratteri l'ordine alfabetico. Quindi nel Key Test, sebbene abbiate programmato la manopola ANT per inviare i caratteri z, x e y in questo ordine, se la manopola è ruotata abbastanza velocemente da fare in modo che Cougar metta tali caratteri nello stesso frame, il computer vede questi tre tasti come premuti in contemporanea e li ordina in ordine alfabetico, ad es. x, y e poi z.*

Ovviamente, aggiungendo FORCE_MACROS *otterremo sempre l'alfabeto nell'ordine corretto, perché ogni carattere non sarà mai messo nello stesso frame insieme ad un altro. Ma li genereremo più lentamente.*

Spero che abbiate capito la maggioranza di quanto sopra. Pensate quindi molto a quando utilizzare il modificatore FORCE_MACROS *nel vostro simulatore e a quando non farlo. Ricordate la regola d'oro – non è come si comporta in Windows la cosa che importa – dovete provarlo nel vostro simulatore ed eseguire dei test prima di renderlo definitivo. Una nota conclusiva: se avete testato un'espressione (senza l'utilizzo di FORCE_MACROS) potrete aver riscontrato che alcuni caratteri non vengono generati in seguito a movimenti rapidi della manopola ANT – potrete altresì trovarvi generati dei caratteri non richiesti come SHF o CTL. Questa è una cosa di cui noi siamo a conoscenza e non sembra dipendere da un bug di Cougar. Pensiamo che sia riconducibile ai driver Microsoft per la tastiera che vengono sovraccaricati nel momento*

in cui si reinizializzano, producendo tali caratteri errati. È impossibile per Cougar generare questi caratteri errati tramite questa espressione.

6.2.2 Type 2: sequenze di caratteri personalizzate, zone costanti

Un'espressione Type 2 ha la seguente sintassi:

Identific. asse	Tipo espressione digitale	Numero di caratteri o macro (max.50)	Sequenza di caratteri e/o macro e/o flag logici	FORCE_ MACROS *(opzionale)*
eg. ANT	2	5	a b c d e	- FORCE_MACROS

oppure, come sarà nel file di configurazione:

ANT 2 5 a b c d e - FORCE_MACROS

La rotazione della manopola ANT in senso orario e successivamente antiorario produrrà la seguente sequenza di caratteri:

b c d e d c b a

Proprio come con le espressioni Type 1, notate che ogni carattere viene generato una volta sola (*ma i **flag** logici rimangono su on – vedi note seguenti)*. Ancora, questo si verificherà se utilizzate macro invece si singoli caratteri. Notate inoltre che se la manopola inizia la sua rotazione dalla posizione estrema antioraria e si muove in senso orario, genererà come primo carattere una "b" e non una "a" come si pensa. Questo è un comportamento differente rispetto ad una Type 1. Ovvero, se ruotate la manopola diverse volte da un estremo all'altro, eseguirete complessivamente il ciclo dei caratteri in output in questo modo:

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

quindi non produrrete:

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

notate che a differenza della sintassi originale Thrustmaster, il Numero di caratteri o macro può essere pari o dispari. Ovviamente se state assegnando valori alla manopola ANT probabilmente vorrete avere un numero dispari, in modo da poter assegnare un carattere di detent centrale alla posizione di centro.

Con le espressioni Type 2 potete anche usare delle macro:

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

e, nel file macro:

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

Ora, so che non abbiamo ancora parlato dei flag logici e di cosa sono, ma se volete addentrarvi in questa parte della programmazione Thrustmaster, sappiate che è anche possibile usare direttamente flag logici in espressioni Type 2, inclusi selettori (toggle) logici:

RNG 2 4 X1 X2* X3* X4

Ne parlerò più approfonditamente quando arriveremo alla sezione sulla programmazione logica. Così come i precedenti, i caratteri singoli, le macro e i flag logici possono essere combinati in espressioni Type 2 così come segue:

ANT 2 5 a Emcon-2 c ^ X1

6.2.2.1 Comprendere il modificatore - FORCE_MACROS

Vedete la sezione 6.2.1.1 per la spiegazione.

NOTE

1. *potete usare i modificatori /U, /M, /D, /I, /O con tutte le espressioni digitali. In ogni caso, fate attenzione se combinate tali modificatori con altri assegnati ad espressioni logiche (vedete le note seguenti), dato che potreste ottenere comportamenti anomali dal vostro controller. Ad esempio (questo è solo per utenti esperti!):*

```
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
```

sulla posizione /D, X1 può generare una "a", una "b" o una "c" in base al cambiamento della posizione del selettore dogfight durante il ritardo di 5 secondi.

2. *non potete usare* /U, /M, /D, /I, /O ***all'interno*** *dell'espressione (questo si applica a tutte le espressioni digitali). Questo genererà un errore di compilazione:*

RNG 2 3 a b macro_1

con macro_1 = /I KP1 /O KP2

3. *poniamo che abbiate programmato:*

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

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

Ora immaginiamo che abbiate S3 rilasciato, la manopola ANT nella parte centrale (genera e mantiene una "e"). Ora premete S3. Questo provocherà il cambiamento della "e" che non verrà mantenuta. Verrà quindi mantenuta una "b" e quando verrà rilasciato S3 tornerà ad essere mantenuta la "e".

4. *potete usare* /P, /R, /H *con espressioni di tipo 1, 2, 5 e 6.*

6.2.3 Type 3: generazione continuativa di caratteri

Un'espressione Type 3 ha la seguente sintassi:

Identificatore asse	Tipo Espressione Digitale	Carattere Sinistro	Carattere Centrale	Carattere Destro
Eg. RDDR	3	l	C	r

Oppure, come appare nel file di configurazione:

RDDR 3 l c r

Premere il pedale sinistro della pedaliera produrrà un carattere "l" che verrà mantenuto premuto, nella stessa maniera in cui viene mantenuto utilizzando il modificatore /H.

NOTE

1. *l'asse non è diviso in tre regioni uguali ma nel modo indicato qui sotto, altrimenti la regione centrale avrà un'escursione troppo estesa:*

Regione Sinistra	Regione Centrale	Regione Destra
/H l	c	/H r

2. *Potere usare flag logici con un'espressione Type 3 se volete.*

3. *Se non volete usare un carattere di centro, usate un carattere nullo (^):*

RDDR 3 l ^ r

6.2.4 Type 4: generazione di caratteri pulsanti (pulsazione)

Un'espressione Type 4 ha la seguente sintassi:

Indentificatore	Tipo Espressione Digitale	Rateo pulsazione (ms)	Carattere sinistro o macro	Carattere centrale o macro	Carattere destro o macro
eg. RNG	4	1000	L	C	r

Oppure, come appare nel file di configurazione:

RNG 4 1000 l c r

Un carattere pulsante viene generato ogni x millisecondi, un po' come il fascio di luce di un faro o una luce stroboscopica. Le espressioni Type 4 sono nuove nei controller Thrustmaster. Con l'espressione Range indicata qui sopra, quando viene ruotata la manopola RNG verso sinistra (o in senso orario, guardandola di fronte) un carattere "l" viene generato ogni 1000 millisecondi (o ogni secondo). Al contrario, ruotandola in direzione opposta, una singola "c" viene generata alla posizione centrale e poi una"r" ogni secondo.

NOTE

1. *Macro e flag logici possono anche essere usati al posto di singoli caratteri.*

2. *Se non volete usare un carattere centrale, usate un carattere nullo (^).*

RNG 4 60 l ^ r

3. l*l valore del rateo di pulsazione in millisecondi può variare tra 0 e 82800000 (23 ore!)*

6.2.5 Type 5: sequenze di caratteri personalizzati e regioni variabili

Un'espressione Type 5 ha la seguente sintassi:

Identif. Asse	Tipo espressione digitale	Numero di regioni (max. 50)	Larghezza regioni (come percentuale)	Sequenza di caratteri e/o macro e/o flag logici	FORCE_ MACROS *(opzionale)*
eg. THR	5	4	(0 20 45 70 100)	a b c d	- FORCE_MACROS

Oppure, come sarà nel file di configurazione:

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

Le espressioni Type 5 sembrano più complesse delle altre ma in realtà non sono altro che dei casi particolari delle espressioni Type 2. Ricordate che un'espressione Type 2 ha la sua sequenza di caratteri distribuita uniformemente per tutta la lunghezza dell'asse. Un'espressione Type 5 divide l'asse in regioni o bande e quindi assegna una sequenza di caratteri ad ogni banda.

Nell'esempio qui sopra sono definite 4 bande:

- da 0 a 20% della lunghezza dell'asse viene prodotta una "a"
- da 21 a 45%: carattere "b"
- da 46% a 70%: carattere "c"
- da 71% a 100%: carattere "d"

Le Type 5 seguono le stesse regole delle Type 2 sotto ogni altro aspetto. Ora, un'espressione digitale può coesistere con un asse analogico. Quindi possiamo fare in modo di avere una manetta analogica di default ma, usando una Type 5, in ogni punto possiamo farle generare dei caratteri. Quindi dovrebbe essere molto facile mappare degli inversori di spinta o attivare i freni del carrello mentre si atterra con il motore al minimo, mappandoli nella parte bassa dell'espressione relativa:

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

e nel file macro avremo:

Wheelbrakes = /P b /R b

Quindi, con il selettore dogfight in posizione down e la manetta al minimo, la macro Wheelbrakes attiverà i freni del carrello. Notate come io non voglia alcuna espressione digitale associata alla posizione centrale (/M) e Up (/U) del dogfight. In questo modo la macro Wheelbrakes viene eseguita sono quando selettore è in posizione bassa.

6.2.5.1 Comprendere il modificatore - FORCE_MACROS

Vedere la sezione 6.2.1.1 e la spiegazione per le espressioni Type 1

NOTE

1. *Con i precedenti HOTAS Thrustmaster, potevate programmare la posizione di minimo della manetta con un'espressione BTN MT, ma solo quando la manetta non veniva trattata in modo analogico. L'espressione BTN MT non è più supportata, considerata la maggiore versatilità che si ottiene potendo programmare contemporaneamente la manetta sia digitalmente che analogicamente. Se volete simulare il comportamento dell'espressione BTN MT, usate l'esempio fornito qui sopra* THR 5 1 (0 5) *Vostra_macro*

2.*Così come ogni espressione digitale, ogni curva applicata ad un asse analogico non influisce sulle espressioni digitali. Esse mantengono le loro "curve" lineari.*

3. *Potete usare* /P, /R, /H *con espressioni Type 1, 2, 5 e 6 ma inseriteli nelle macro oppure, se li usate direttamente nell'espressione, racchiudeteli tra parentesi. Quindi:*

THR 5 1 (0 5) Wheelbrakes

con la macro nel vostro file macro definita così:

Wheelbrakes = /P b /R b

Va bene. E potete anche avere:

THR 5 1 (0 5) (/P b /R b) *ma:*

THR 5 1 (0 5) /P b /R b *genererà un errore di compilazione.*

6.2.6 Type 6: generazione ripetitiva di caratteri, regioni variabili.

Un'espressione Type 6 ha la seguente sintassi:

Identif. Asse	Tipo Espress. Digitale	Numero di regioni (max. 50)	Larghezza delle regioni (%)	Up	Down	Centro *(opzionale)*	FORCE_ MACROS *(opzionale)*
eg. ANT	6	5	(8 20 40 45 70 80)	u	d	c	- FORCE_MACROS

Oppure, come apparirà nel file di configurazione:

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

Un'espressione digitale Type 6 è essenzialmente identica ad una Type 1, solo i caratteri non sono raggruppati in bande uguali come sarebbero in una Type 1. Sono invece raggruppati in bande scelte da voi.

A differenza delle espressioni Type 1, **se includete un carattere centrale, il numero delle regioni deve essere dispari.**

Nell'esempio qui sopra, sono impostate 5 bande:

- da 8 a 20% dell'escursione dell'asse
- da 21 a 40%
- da 41 a 45%
- da 46% a 70%
- da 71% a 80%

Ma, proprio come un'espressione Type 1, la rotazione della manopola ANT produce

u u c u u d d c d d

Se l'espressione era:

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

produrrà:

u u u u u d d d d d

6.2.6.1 Comprendere il modificatore - FORCE_MACROS

Vedere la sezione 6.2.1.1 e la spiegazione per le espressioni Type 1.

6.2.7 Direzioni degli assi: valori analogici ed espressioni digitali

6.2.7.1 Valori degli assi analogici

In questa sezione vi spiegherò quali valori analogici sono prodotti dagli assi, quando vengono generati e fornirò esempi per ogni espressione digitale, per dimostrarne il funzionamento.

Posizione dell'asse	Valore analogico
JOYX - left	0
JOYX - right	max
JOYY - back	max
JOYY - forward	0
THR - back	max
THR - forward	0
RNG - CCW [nota 1]	max
RNG - CW	0
ANT - CCW [nota 1]	max
ANT - CW	0
MIX - left [nota 2]	-
MIX - right	-
MIY- down	-
MIY - up	-
RDDR - left forward	0
RDDR - right forward	max
LBRK, RBRK - up	max
LBRK, RBRK - pressed	0

NOTE

1. *La manopola Range può confondere le idee quando si pensa alla direzione di funzionamento. La regola, con le manopole ANT e RNG è questa: guardatele **di fronte**, per determinare quale direzione è oraria (Clockwise o CW) e quale antioraria (CounterClockwise o CCW)*

2. *Il microstick (*MIX, MIY) *non viene visto da Windows come un controller analogico. Questo non significa che non possa essere usato come analogico, dato che altri assi possono essere mappati su di esso.*

6.2.7.2 Espressioni Digitali Type 1 per gli assi

JOYX 1 6 r l	Quando l'asse Joystick X si muove da sinistra a destra, otteniamo il carattere "r", mentre da destra a sinistra otteniamo il carattere "l".
JOYY 1 6 f b	Quando l'asse Joystick Y si muove in avanti otterremo "f", mentre indietro otterremo "b".
THR 1 6 f b	Muovendo la manetta in avanti otterremo "f" mentre muovendola all'indietro otterremo "b".
RNG 1 6 r l	Muovendo RNG da CCW a CW otterremo "r" mentre da CW a CCW otterremo "l".
ANT 1 6 r l	Muovendo ANT da CCW a CW, otterremo "r", mentre muovendola da CW a CCW otterremo "l"
MIX 1 6 r l	Muovendo l'asse Microstick X da sinistra a destra, otterremo"r" mentre da destra a sinistra otterremo "l".
MIY 1 6 u d	Muovendo l'asse Microstick Y dalla posizione inferiore a quella superiore otterremo "u", mentre dall'alto verso il basso otterremo "d".
RDDR 1 6 l r	Pedale sinistro avanti produrrà "l", pedale sinistro indietro e pedale destro avanti produrrà "r".
LBRK 1 6 d u	Freni pedaliera premuti produrrà un "d" e il rilascio produrrà una "u". (la stessa cosa si applica a RBRK).

6.2.7.3 Espressioni digitali Type 2 per gli assi

JOYX 2 5 a b c d e	Muovendo Joystick X da Sinistra a destra otterremo i caratteri "a b c d e", muovendo da destra a sinistra otterremo i caratteri "e d c b a".
JOYY 2 5 a b c d e	Muovendo in avanti Joystick Y otterremo i caratteri "a b c d e", mentre muovendolo all'indietro otterremo "e d c b a".
THR 2 5 a b c d e	Muovendo la manetta in avanti otterremo "a b c d e", Mentre all'indietro otterremo i caratteri "e d c b a".
RNG 2 5 a b c d e	Muovendo RNG da CCW a CW, otterremo "a b c d e", mentre da CW a CCW otterremo "e d c b a".

ANT 2 5 a b c d e Muovendo ANT da CCW a CW otterremo "a b c d e" , mentre andando da CW a CCW otterremo "e d c b a".

MIX 2 5 a b c d e Muovendo l'asse X del Microstick da sinistra a destra, otterremo "a b c d e", mentre da destra a sinistra otterremo i caratteri "e d c b a".

MIY 2 5 1 2 3 4 5 Muovendo l'asse Y del Microstick dalla posizione inferiore a quella superiore otterremo "1 2 3 4 5", mentre andando dall'alto al basso otterremo "5 4 3 2 1".

RDDR 2 5 a b c d e Pedale sinistro premuto e rilasciato via via che viene premuto il pedale destro fino al fine corsa, produrremo "a b c d e". Eseguendo lo stesso movimento ma invertendo i pedali (ovvero destro premuto e rilasciato via via che premiamo il sinistro) otterremo "e d c b a" .

LBRK 2 5 a b c d e La pressione dei freni pedaliera (entrambi i pedali premuti a fine corsa) produce "a b c d e" e "e d c b a" al rilascio (si applica la stessa cosa per RBRK).

6.2.7.4 Espressioni digitali Type 3 per gli assi

JOYX 3 l ^ r Quando Joystick X è a sinistra, avremo un carattere costante "l" e quando va a sinistra avremo un carattere costante "r".

JOYY 3 b ^ f Quando Joystick Y è nella posizione arretrata, avremo un carattere costante "b" e quando viene portato in avanti, avremo un carattere costante "f".

THR 3 b ^ f Quando la manetta è in posizione arretrata, otterremo un carattere costante "b" e quando verrà portata in avanti il carattere costante sarà "f".

RNG 3 l ^ r Il movimento di RNG in senso orario (CW), produrrà un carattere costante "r" e quando verrà ruotata in senso antiorario (CCW), produrrà un carattere costante "l".

ANT 3 l ^ r Quando ANT è ruotata in senso orario, genera un carattere costante "r" e quando ruota in senso antiorario produce un carattere "l".

MIX 3 l ^ r Quando l'asse X del microstick è a sinistra, otteniamo un carattere costante "l" e quando è a destra otteniamo un carattere costante "r".

MIY 3 d ^ u — Se l'asse Y del microstick viene portato verso il basso. Otterremo un carattere costante "d" e quando viene portato verso l'alto otteniamo un carattere costante "u".

RDDR 3 l ^ r — Con il pedale sinistro avanti produrremo un carattere costante "l" e con il pedale destro avanti produrremo un carattere costante "r".

LBRK 3 u ^ d — Se si premono i freni pedaliera (entrambi i pedali a fine corsa) otterremo un carattere costante "d" e quando li rilasciamo un carattere costante "u" (lo stesso si applica a RBRK).

6.2.7.5 Type 4 Digital axes statements

N.B.: per "carattere pulsante" si intende un carattere "intermittente"

JOYX 4 300 l ^ r — Quando l'asse Joystick X è a sinistra, si ottiene un carattere pulsante "l", mentre con Joystick X a destra otterremo un carattere pulsante "r".

JOYY 4 300 b ^ f — Quando Joystick X è in posizione arretrata, otteniamo un carattere pulsante "b", mentre in posizione avanzata otteniamo il carattere pulsante "f"

THR 4 300 b ^ f — Con la manetta in posizione arretrata otteniamo un carattere pulsante "b" mentre in posizione avanzata otteniamo un carattere "f"

RNG 4 300 l ^ r — Quando la manopola RNG viene ruotata in senso orario otteniamo un carattere pulsante "r", mentre in senso antiorario otteniamo un carattere "l"

ANT 4 300 l ^ r — Quando la manopola ANT viene ruotata in senso orario, otteniamo un carattere pulsante "r", mentre in senso antiorario otteniamo il carattere "l"

MIX 4 300 l ^ r — Quando l'asse Microstick X è in posizione sinistra otteniamo un carattere pulsante "l" mentre a destra otteniamo un carattere pulsante "r"

MIY 4 300 d ^ u — Quando l'asse Microstick Y è in basso, otteniamo il carattere pulsante "d" e quando è in alto otteniamo il carattere "u"

RDDR 4 300 l ^ r — Con il pedale sinistro avanti, otteniamo un carattere pulsante "l" e con il pedale destro avanti otteniamo un carattere pulsante "r"

LBRK 4 300 u ^ d — La pressione dei freni pedaliera produce un carattere pulsante "d" e quando torna in posizione neutra produce un carattere pulsante "u" (lo stesso comportamento si applica a RBRK).

6.2.7.6 Espressioni digitali Type 5

JOYX 5 5 (0 20 40 60 80 100) a b c d e	Muovendo Joystick X da sinistra a destra otterremo la sequenza di caratteri "a b c d e" ed andando da destra a sinistra "e d c b a" .
JOYY 5 5 (0 20 40 60 80 100) a b c d e	Muovendo Joystick Y in avanti, si otterrà la sequenza "a b c d e", mentre andando nella direzione opposta otterremo "e d c b a" .
THR 5 5 (0 20 40 60 80 100) a b c d e	Spingendo la manetta in avanti otterremo la sequenza "a b c d e" e tornando indietro otterremo "e d c b a".
RNG 5 5 (0 20 40 60 80 100) a b c d e	Muovendo RNG in senso orario, otterremo la sequenza "a b c d e", mentre in senso antiorario otterremo "e d c b a".
ANT 5 5 (0 20 40 60 80 100) a b c d e	Muovendo ANT in senso orario, otterremo la sequenza "a b c d e", mentre in senso antiorario otterremo "e d c b a".
MIX 5 5 (0 20 40 60 80 100) a b c d e	Muovendo l'asse Microstick X da sinistra a destra otterremo la sequenza "a b c d e" mentre da destra a sinistra otterremo "e d c b a".
MIY 5 5 (0 20 40 60 80 100) 1 2 3 4 5	Muovendo Microstick Y dalla posizione inferiore a quella superiore otterremo la sequenza "1 2 3 4 5" e al contrario otterremo "5 4 3 2 1".
RDDR 5 5 (0 20 40 60 80 100) a b c d e	Rilasciando il pedale sinistro e contemporaneamente premendo il pedale destro otterremo la sequenza "a b c d e". Muovere il pedale sinistro avanti mentre quello destro torna indietro produce "e d c b a".
LBRK 5 5 (0 20 40 60 80 100) a b c d e	La pressione dei freni pedaliera produce la sequenza "a b c d e" e la sequenza "e d c b a" quando vengono rilasciati (lo stesso si applica a RBRK).

6.2.7.7 Espressioni digitali Type 6

JOYX 6 5 (0 20 40 60 80 100) r l	Muovendo l'asse Joystick X da sinistra a destra otterremo "r" mentre da destra a sinistra otterremo "l".
JOYY 6 5 (0 20 40 60 80 100) f b	Muovendo in avanti Joystick Y otterremo "f" mentre in direzione opposta otterremo "b".
THR 6 5 (0 20 40 60 80 100) f b	Muovendo la manetta in avanti otterremo "f" mentre riportandola indietro otterremo "b".
RNG 6 5 (0 20 40 60 80 100) r l	Muovendo RNG dalla posizione CCW (estrema antioraria) a quella CW (estrema oraria) otterremo "r" e tornando indietro otterremo "l".
ANT 6 5 (0 20 40 60 80 100) r l	Muovendo RNG dalla posizione CCW (estrema antioraria) a quella CW (estrema oraria) otterremo "r" e tornando indietro otterremo "l".
MIX 6 5 (0 20 40 60 80 100) r l	Portando Microstick X da sinistra a destra otterremo "r" mentre da destra a sinistra otterremo "l".
MIY 6 5 (0 20 40 60 80 100) u d	Muovendo Microstick Y dal basso verso l'alto otterremo "u" mentre in direzione opposta otterremo "d".
RDDR 6 5 (0 20 40 60 80 100) l r	Il pedale sinistro avanti produce "l" mentre il pedale sinistro indietro e quello destro avanti producono "r".
LBRK 6 5 (0 20 40 60 80 100) d u	La pressione dei freni pedaliera produce "d" e il rilascio "u". (lo stesso si applica a RBRK).

Questa tabella spiega la programmazione digitale dei vari assi. Ora vedremo la parte analogica degli assi e come possiamo modificare la risposta analogica attraverso varie espressioni. Probabilmente la prima cosa che vorrete modificare sarà la curva di risposta di un asse analogico, attraverso la programmazione. Vediamo come....

6.3 Curve di risposta (CURVE)

Ognuno dei 10 assi ha, di default, una risposta lineare. Questo significa che se muoviamo la manetta in avanti, i valori che vengono inviati al simulatore sono direttamente proporzionali al movimento della manetta. Ad es. per ogni movimento della manetta pari al 10%, l'output aumenta del 10%. È possibile modificare il comportamento di questi 10 assi, cambiando le rispettive curve di risposta. Le curve di risposta per ogni asse sono identificate dalla loro sensibilità.
Ci sono due espressioni che possono essere usate per definire e modificare la risposta di un asse. Prima le definiremo e poi spiegheremo come usarle:

Espressione di configurazione

USE CURVE (Axis_Identifier, Sensitivity)

Sintassi

CURVE [modificatori] (Axis_Identifier, Sensitivity)

dove:

Axis_Identifier può essere:

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

Sensitivity

È un valore variabile tra -32 e 32 (*anche se valori oltre il 20 produrranno curve che non vorrete mai usare!*)
i numeri negativi (-10 ad esempio) rappresentano una riduzione della sensibilità – ottima per gli atterraggi, rifornimenti in volo e voli in formazione. Lo zero resetta completamente la curva al valore di risposta lineare di default (scavalcando ogni espressione USE CURVE) e i numeri positivi (10) aumentano la sensibilità, ottimi per i dogfight con aerei della WW2 ad esempio.

Modificatori *(opzionali)*

/U, /M, /D (Selettore Dogfight) e /I, /O (pulsante S3) sono utilizzabili

Confusi? Vediamo allora qualche esempio e chiariamo quali sono le differenze tra le espressioni USE CURVE e CURVE.

USE CURVE è una **espressione di configurazione -** (**tutte** le espressioni USE lo sono). Questo significa che vengono inserite all'inizio dei file di configurazione e non possono essere programmate su posizioni specifiche del controller. Sono utilizzate per definire le **curve predefinite degli assi**. Normalmente ogni asse è lineare, quindi impostando la curva di risposta dell'asse Joystick X in questo modo:

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

è alquanto inutile visto che il compilatore lo farà in ogni caso, assumendo che vogliate delle risposte lineari da tale asse. Invece se avessimo la seguente espressione nel file di configurazione:

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

modificheremmo entrambe le curve degli assi X e Y in modo da renderle più reattive. Notate come abbiamo usato il termine "JOYSTICK" per definire entrambi gli assi JOYX e JOYY, in modo che il compilatore invii entrambi i seguenti comandi al controller:

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

Quindi, l'espressione USE CURVE può essere usata per modificare le risposte di default degli assi.
L'espressione CURVE segue la stessa sintassi, ma dato che non è un'espressione di configurazione, può essere inserita in espressioni riferite a singoli pulsanti, assi digitali o come un'espressione speciale ed indipendente. Vedete gli esempi qui di seguito. Poniamo di voler modificare la sensibilità dell'asse Joystick Y in base alla posizione del selettore Dogfight. Possiamo farlo usando la seguente espressione CURVE:

```
CURVE  /U (JOYY, 2)  Rem Più reattivo per il dogfight
       /M (JOYY, 0)   Rem  Normale
       /D (JOYY, -2)  Rem  Meno reattivo per l'atterraggio
```

in modo simile, per il Microstick

```
CURVE  /I (MICROSTICK, 2)   Rem Più reattivo
       /O (MICROSTICK, 0) Rem  Normale
```

E potete combinarle:

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

L'espressione CURVE può essere anche utilizzata direttamente su una posizione programmabile:

```
BTN T7 /P CURVE(JOYX, 3)  Rem Reattivo
       /R CURVE(JOYX, 0)  Rem  Normale
```

L'espressione CURVE può anche essere usata all'interno di espressioni per assi digitali, quindi se io volessi cambiare la sensibilità del joystick in base alla posizione della manetta. Ad esempio:

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

a valori bassi della manetta, l'asse joystick X è meno reattivo del normale, ma diventa via via più reattivo all'aumentare della manetta. Come piccola digressione, questo è in realtà un bellissimo esempio di come la manetta può agire come una manetta analogica, sebbene sia programmata digitalmente.

NOTE

1. *se un asse viene mappato su un altro asse (vedete le note seguenti) la sua curva di risposta lo segue.*

2. *Non è possibile impostare zone morte con espressioni CURVE – dovete usare il pannello di controllo del Cougar (CCP) per questo. Se avete bisogno di tali zone morte per essere precisi in un particolare simulatore, usate il CCP per salvare le impostazioni in un profilo e usate l'espressioni USE PROFILE nel vostro file di configurazione. Vedete il punto 5.*

3. *Non potete avere più di un'espressione CURVE in un joystick file. La seconda espressione CURVE genererà un errore del compilatore.*

```
CURVE  /I (MICROSTICK, 2)   Rem Più reattivo
       /O (MICROSTICK, 0) Rem  Normale

CURVE  /I (ROTARIES, 2)   Rem Più reattivo
       /O (ROTARIES, 0) Rem  Normale
```

Da non confondere con l'utilizzo di CURVE con assi e pulsanti, situazione in cui potete usarlo più di una volta.

4. *Se non state programmando un'espressione CURVE su un pulsante o un asse, non potete usare CURVE da solo senza modificatori. Piuttosto usate un'espressione di configurazione. Quindi:*

CURVE (JOYSTICK, 10) *genererà un errore, dove invece:*

USE CURVE (JOYSTICK, 10) *va bene.*

5. *potete anche utilizzare un profilo salvato – vedete l'espressione USE PROFILE discussa in precedenza se volete applicare curve multiple agli assi nell'intero file. Avrete il vantaggio di poter incorporare le zone morte.*

6.4 Regolazione degli assi (TRIM)

"Trimmare" gli assi significa fare in modo che quando rilasciate il controller, il simulatore continui a vedere gli assi come se fossero mantenuti nella posizione precedente. Mi spiego meglio. Poniamo che siate in crociera a 15,000 piedi e per qualche ragione il vostro aereo tenda a salire di quota mentre il joystick rimane in posizione centrata. Dovrete quindi continuamente correggere questa tendenza spingendo in avanti la barra. In questo caso, la funzione TRIM può essere usata per permettervi di tenere il joystick centrato, ma visto dal simulatore come se fosse mantenuto spinto in avanti. Questo non è limitato all'asse del joystick, ma a tutti e 10 gli assi analogici.

Sintassi

TRIM (Axis_Identifier, Trim_amount)

e:

HOLDTRIM (Axis_Identifier)

dove:

Axis_Identifier è uno dei seguenti:

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

Trim_Amount

è un valore variabile tra -128 e 127 o TO_CURRENT.
Un valore zero resetterà l'asse della curva in modo da non applicare trim.
Un valore positivo aumenta il valore del trim e viceversa con valori negativi.
È un valore che varia da -128 a 127 oppure vale TO_CURRENT.

La keyword TO_CURRENT legge il valore attuale dell'asse e imposta il trim su questo valore quando l'asse è centrato.

Ok, ora vediamo un esempio che utilizza le manopole RNG e ANT per regolare il trim degli assi X e Y del joystick, utilizzando un'espressione Type 1.

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

La rotazione della manopola ANT in senso orario ad esempio sottrae continuamente 20 dal valore dell'asse Y, ovvero simula la pressione in avanti della barra. questo sarebbe utile per un aereo che tende a salire quando il joystick è centrato.

Posso impostare il pulsante S2 sul joystick per annullare gli effetti del trim, in questo modo:

```
BTN S2 TRIM (JOYX, 0) TRIM (JOYY, 0)  Rem Rimuove il trim da JOYX e
                                          JOYY
```

Oppure, in alternativa, il comando equivalente:

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

Posso anche specificare una quantità di trim per gli assi, in questo modo:

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

Per finire, posso tenere il joystick in una posizione e impostare il trim in modo che quando il joystick viene rilasciato, il trim mantenga tale posizione, in questo modo:

```
BTN S2 /I TRIM (JOYSTICK, TO_CURRENT) Rem Trim al valore attuale
       /O TRIM (JOYSTICK, 0) Rem Cancella tutti i trim
```

Qui però c'è un piccolo problema. Notate quando ho detto "quando il joystick viene rilasciato". Mi spiego. Se state volando e mantenete il joystick in posizione in modo da correggere una cabrata, quando applicate il trim al valore corrente e rilasciate la barra, vi aspettate che l'aereo voli livellato. Non è esattamente quello che succede però, a meno che non rilasciate la barra nel momento in cui applicate il trim. Perché?
Perché il valore del trim è calcolato come se il joystick fosse in posizione centrale, ma in questo istante non è così. Tu stai spingendo la barra in avanti e, applicando il trim, il valore risultante è come se la barra fosse spinta ancora più avanti. Quando riporti la barra al centro, il simulatore sentirà la barra come se fosse ancora leggermente spinta in avanti, della misura necessaria per

mantenere il volo livellato. Probabilmente dovrete rileggere il paragrafo precedente per afferrare il concetto.
Allora, come possiamo ovviare a questo problema?
Beh, possiamo farlo in maniera semplice o difficile. Questa è la via semplice:

BTN S2 HOLDTRIM (JOYSTICK)

La ragione per cui vi spiego la via "difficile" è perché rende più semplice la comprensione dell'utilizzo di questa espressione. Quindi, il modo per utilizzare questa espressione è il seguente. Tenete la barra in una posizione che garantisce un volo livellato. Ora premete S2 e tenetelo premuto. Riportate la barra in posizione centrale e solo ora rilasciate S2. Se S2 è rimasto premuto per tutta la corsa della barra, l'aereo continuerà a volare rettilineo e una volta che il joystick è centrale, potete rilasciare S2 e rilassarvi.

Ora vediamo la via "difficile" per implementare questa funzione. In realtà non è difficile, e aiuta a spiegare il comportamento delle espressioni precedenti. Abbiamo bisogno di usare un'espressione TRIM TO_CURRENT combinata con espressioni LOCK e UNLOCK , in questo modo:

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

Ora, mentre sto spingendo in avanti il joystick per mantenere un volo livellato, **se premo e mantengo premuto** S2, il velivolo manterrà un volo livellato mentre riporto la barra al centro e qui rilascerò S2. Ecco cosa succede: per prima cosa, quando premo S2 i valori del joystick vengono bloccati a quelli attuali. Il trim viene quindi calcolato in base a questi valori. Quando il joystick viene riportato in centro, sblocchiamo i valori rilasciando S2 e il velivolo vola livellato perché abbiamo già trimmato gli assi. Tutto ciò è cosa diventa BTN S2 HOLDTRIM (JOYSTICK) quando viene tradotto dal compilatore.

NOTE

1. *un'espressione TRIM ha come esito l'addizione o la sottrazione di un numero intero al valore dell'asse, i.e. sposta l'intera curva in una direzione o nell'altra. Non importa se state utilizzando curve lineari o modificate.*

2. *Una curva lineare a cui si applica un TRIM non permetterà di raggiungere i valori estremi dell'asse.*

3. **I***nvertire un asse non altera la direzione dell'espressione TRIM – essa rimane la stessa. Le espressioni digitali non vengono invertite come le loro controparti analogiche*

4. *Fate attenzione quando inserite segni + e - in un'espressione TRIM, così come nelle espressioni per gli assi. Se il segno è a sinistra del numero*

*specificherà la quantità di trim, mentre se è a destra del numero specificherà la quantità da aggiungere/togliere al valore attuale del trim. Vedete la sezione "*Il Mouse ed il Microstick", per capire meglio la differenza tra i segni "+ -" e il loro effetto in base alla posizione.

5. *Queste espressioni HOLDTRIM sono perfettamente valide:*

```
BTN T6 a b HOLDTRIM (RNG) c d
BTN S4 /P a HOLDTRIM (RNG)
       /R b
BTN S1 a { HOLDTRIM (JOYY) b HOLDTRIM (ANT) }
```

Notate altresì come comandi HOLDTRIM multipli devono essere ragruppati tra parentesi tonde.

6. *Non potete avere una macro chiamata TRIM, ma potete avere Trim e Trim_Hold ad esempio.*

7. *Potete usare il modificatore (A (Autorepeat) unito all'espressione TRIM per controllare qualunque asse tramite pulsanti o hat. Ad esempio queste espressioni controllano gli assi del joystick.*

```
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)
```

Ancora, è possibile manovrare assi che non sono fisicamente presenti (ad esempio la pedaliera o i freni) utilizzando espressioni come le precedenti mappate sui rispettivi assi. (vedete il manuale di riferimento Thrustmaster riguardante il checkbox Apply enable/disable Windows axes states *nel Cougar Control Panel per ulteriori informazioni, ad esempio come attivare assi che non sono fisicamente presenti.)*
Ad esempio:

```
BTN H4L /A TRIM (RDDR, 5-)
BTN H4R /A TRIM (RDDR, 5+)
```

6.5 Disabilitare gli Assi

Tutti gli assi analogici verranno riportati al simulatore come presenti per default, **eccetto** per quelli del Microstick. Ci sono alcune circostanze dove è conveniente disabilitarli come quando si desidera usarli per produrre dei caratteri con espressioni digitali. Gli assi possono essere disabilitati con espressione di configurazione nel file joystick in questo modo:

Espressione di configurazione
DISABLE Axis_Identifier

dove :

Axis_Identifier può essere:

THR	
RNG, ANT	(nominati **ROTARIES** – vedere sotto)
LBRK, RBRK	(nominati **TOEBRAKES** – vedere sotto)
RDDR	

In seguito può avere senso raggruppare gli assi in modo da poterli disabilitare con una singola espressione. Ad esempio:

DISABLE ROTARIES	è convertito dal compilatore in	DISABLE RNG DISABLE ANT
DISABLE TOEBRAKES		DISABLE LBRK DISABLE RBRK

Come tutte le espressioni di configurazione, queste appariranno in una singola linea nel file joystick. L'unica aggiunta possibile sarà quella di un REM.

Quindi

```
DISABLE THR Rem Disabilita la manetta
DISABLE ANT Rem Disabilita la manopola dell'antenna sul TQS
```

sono corretti, mentre:

```
DISABLE (THR, ANT, RNG)  non lo è.
```

Potrebbe sembrare semplicistico, ma ci sono alcuni vantaggi sia per il programmatore che per l'utente nel fatto di usare linee separate (*per dirne una, è più facile mettere dei REM su un'espressione unica*). Notate che questa nuova espressione DISABLE rimpiazza la USE NO delle precedenti versioni del software Thrustmaster (USE NOMOUSE, USE NOTHR, ecc.). Siccome gli assi sono presenti per default non sarà necessario usare espressioni come USE RCS, USE TQS etc. come per i precedenti HOTAS Thrustmaster.

6.5.1 Disabilitare e Abilitare un asse durante il volo con LOCK, UNLOCK

Adesso sappiamo come disabilitare un asse usando un'espressione di configurazione. Una volta disabilitato un asse, non può più essere visto come un asse analogico. Il risultato è che non produce nulla oppure deve essere programmato in modo digitale.

Ci sono alcune situazioni in cui, ad esempio, si potrebbe volere poter muovere un asse per usufruire delle espressioni digitali programmate, ma mantenendo invariato il suo valore analogico. Sfortunatamente non è possibile rimuovere un asse durante il gioco – questa operazione normalmente produce crash e blocchi del sistema o altri risultati indesiderati di questo tipo.

Non potendo rimuovere un asse durante il volo e poi ripristinarlo, l'unica strada possibile è quella di bloccarlo ad un valore ben preciso mentre si muove l'asse per usare le espressioni digitali ad esso collegate. Questo è possibile con l'uso di LOCK, mentre è possibile ripristinare il normale funzionamento dell'asse con UNLOCK. Ecco la sintassi:

Sintassi

```
LOCK (Axis_Identifier, Lock_Value%)

UNLOCK (Axis_Identifier)
```

dove:

Axis_Identifier può essere:

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

Lock_Value è:

Un valore tra 0 e 100%, o semplicemente LASTVALUE (ultimo valore)
Ad esempio:

```
BTN S2 /I LOCK (THR, 100%)
       /O UNLOCK (THR)
```

L'espressione LOCK viene usata per far generare all'asse un singolo valore tra lo 0% ed il 100% della sua estensione oppure all'ultimo valore generato con LASTVALUE. La natura analogica dell'asse può essere ripristinata con l'espressione UNLOCK.

Confusi? Bene il sistema più semplice per capirne il funzionamento è osservare qualche esempio. Eccoli:

1. Volendo che il RNG sia visto come un asse analogico escluso quando S3 viene mantenuto premuto. Alla pressione di S3 l'asse dovrà mantenere l'ultimo valore generato tramite un'espressione digitale.

```
RNG /I LOCK (RNG, LASTVALUE) 2 5 a b c d e
    /O UNLOCK (RNG)
```

Quando il pulsante S3 non è premuto, il RNG è visto come un asse analogico normale e il suo funzionamento viene determinato dal gioco. Alla pressione di S3, il RNG non cambierà più il suo valore analogico, ma genererà "a b c d e" seconde le modalità della espressione digitale Type 2 definita in /I. Spero che questo sia molto più chiaro, ora vediamo qualcosa di più potente:

2.
```
ANT  /U /I LOCK (ANT, LASTVALUE) 2 10 1 2 3 4 5 6 7 8 9 0
       /O UNLOCK (ANT) Rem Asse assegnato dal gioco
     /M UNLOCK (ANT) Rem Asse assegnato dal gioco
     /D LOCK (ANT, 0%) 3 Lower_flaps ^ Raise_Flaps
```

Quando il selettore dogfight è nella posizione bassa, la manopola Antenna è usata in modo digitale per usare i flap con un'espressione Type 3, e il gioco leggerà un valore 0 dall'asse analogico. Quando il selettore sarà in posizione centrale, la manopola reagirà come un normale asse analogico. In posizione alta (/U) invece, se il pulsante S3 non è premuto funzionerà come nel caso precedente. Con S3 premuto verrà mantenuto l'ultimo valore dell'asse analogico mentre l'espressione digitale genererà i numeri da 1 a 0 durante la rotazione della manopola Antenna.

Usando LOCK e UNLOCK, in combinazione con le espressioni digitali, si ottiene una programmazione modo molto potente degli assi, ma attenzione, è anche molto facile commettere degli errori in questo modo!

NOTE

1. *Non è possibile disabilitare gli assi del Joystick o del Microstick con l'espressione* DISABLE*. Gli assi del Joystick devono essere presenti come richiesto dalle specifiche DirectX.*

2. *Disabilitando un asse in un file, il controller dovrà riportare a Windows che quell'asse non è più presente. Ciò può richiedere un po' di tempo e quindi, caricare un file con quest'espressione può essere più lungo che in altri casi.*

3. LOCK e UNLOCK *devono essere preceduti da* /U, /M, /D, /I, *o* /O *quando usati su espressioni legate agli assi. Questo esempio genererà un errore.*

```
ANT LOCK (RNG, LASTVALUE)
```

Il seguente invece è corretto:

```
BTN S2 LOCK (RNG, LASTVALUE)
```

6.6 Mappatura degli assi (SWAP)

La mappatura permette di scambiare gli assi prima, durante e dopo il volo. L'espressione da usare è SWAP.

Espressione di configurazione

```
USE SWAP (Axis_Identifier, Axis_Identifier)
```

Sintassi

```
SWAP (Axis_Identifier, Axis_Identifier)
```

dove:

Axis_Identifier è uno dei seguenti:

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

Ad esempio:

```
USE SWAP (ANT, RNG)
```

scambierà gli assi Antenna e Range sulla manetta. Altri esempi:

```
BTN S1 SWAP(JOYY, THR)        REM scambia gli assi Y e Manetta
```

```
BTN S2 SWAP(JOYSTICK, MICROSTICK)     REM scambia gli assi X e Y del
                                      Joystick con i corrispondenti del
                                      Microstick
```

L'ultimo esempio viene convertito dal compilatore come:

```
BTN S2 SWAP(JOYX, MIX) SWAP(JOYY, MIY)
```

È importante notare che scambiando più di un asse alla volta è necessario specificare lo stesso numero di assi. Quindi:

```
BTN S2 SWAP(JOYSTICK, MICROSTICK)  è corretto, ma
```

```
BTN S2 SWAP(JOYSTICK, THR)
```

Genererà un errore di compilazione, perché JOYSTICK definisce 2 assi (JOYX and JOYY) ma THR è un asse singolo.

NOTE

Scambiare gli assi si ripercuoterà anche sulle curve di risposta, ma le espressioni digitali non verranno toccate.

6.7 Invertire la direzione di un asse (REVERSE, FORWARD)

È possibile invertire la direzione di un asse tramite l'espressione REVERSE:

Espressione di configurazione

```
USE REVERSE (Axis_Identifier)
```

Sintassi

```
REVERSE (Axis_Identifier)
FORWARD (Axis_Identifier)
```

dove:

Axis_Identifier è uno dei seguenti:

JOYX, JOYY	(nominati **JOYSTICK**)
THR	
RNG, ANT	(nominati **ROTARIES**)

MIX, MIY	(nominati **MICROSTICK**)
LBRK, RBRK	(nominati **TOEBRAKES**)
RDDR	

Questo può essere molto utile ad esempio in un simulatore di elicottero volendo usare la manetta in direzione inversa rispetto ad un jet o se sei come me e preferisci avere la pedaliera rispondere al contrario rispetto al comportamento di un aereo vero!

Se volessimo invertire il comportamento del timone, senza dover premere alcun pulsante, è necessario mettere uno USE davanti a REVERSE per convertirlo in un'espressione di configurazione:

```
USE REVERSE (RDDR)
```

Questo apparirà all'inizio del file joystick in una linea dedicata. Guardiamo altri esempi:

```
BTN H1U   /I REVERSE (JOYY)
          /O FORWARD (JOYY)
```

Muovendo in su HAT 1 con S3 premuto, l'asse Y del joystick verrà invertito. Muovendo l'HAT 1 in su, ma con S3 rilasciato, l'asse Y verrà riportato al normale funzionamento. Notate che, come per altre proprietà degli assi, l'effetto di REVERSE e FORWARD rimane finché l'asse non viene rimappato o riconfigurato altrove, ma che comunque le espressioni digitali non vengono modificate.

6.8 L'uso di USE AXES_CONFIG

Abbiamo visto in dettaglio come disabilitare e invertire gli assi uno alla volta. È anche possibile configurare tutti gli assi in un'espressione unica: USE AXES_CONFIG. Questa verrà poi convertita dal compilatore nelle espressioni che abbiamo già visto. L'espressione AXES_CONFIG è molto complessa, ma può essere utile in alcuni casi …

Espressione di configurazione:

USE AXES_CONFIG **(***DX-axis1*, HOTASaxis1), (*DX-axis2*, HOTASaxis2) etc.

DX-axis1 è assegnato all'HOTASaxis1

Qual'è la differenza tra una asse DX e uno HOTAS? L'ultimo è il più facile da vedere ed è il primo che affronteremo. Un asse HOTAS è uno dei 10 assi fisici

presenti sul Cougar (JOYX, JOYY, THR, RDDR, ANT, RNG, MIX, MIY, LBRK, RBRK). L'asse DX è più difficile da spiegare…

L'asse DX è l'asse che il Cougar espone alle DirectX, il quale poi viene allocato dal gioco ad una specifica funzione. Pur avendo a disposizione 10 assi usabili, le DirectX 8 ne supportano solo 8 sui dispositivi USB (le DirectX 7 solo 6). Quello che possiamo fare per sempilificarci la vita è di dire a Windows "*Usa l'asse X del Joystick come asse 1 delle direct (DX-axis 1), la Y del Joystick come DX-axis 2, ecc.*". Windows non sa gli assi siano una manetta, una manopola RNG, ecc. perché questi vengono usati con gli stessi nomi anche nel caso di un volante o altri controller. Gli assi DX sono chiamati e sono assegnati al Cougar come dalla tabella qui sotto.

Asse DirectX	Asse assegnato all'HOT	Sintassi
Asse X	Asse X del Joystick	JOYX
Asse Y	Asse Y del Joystick	JOYY
Asse Z	Manetta	THR
Rotativo X	Manopola Antenna	ANT
Rotativo Y	Freno del pedale sinistro	LBRK
Rotativo Z	Timone	RDDR
Cursore 0	Manopola Range	RNG
Cursore 1	Freno del pedale destro	RBRK

Note: L'assegnazione dei freni potrebbe essere invertita nella pedaliera del Cougar quando questa sarà disponibile al pubblico.

Tornando a come usare l'espressione, ci sono Quattro regole fondamentali da ricordare:

1. Tutti gli assi (DirectX o HOTAS) non inseriti saranno disabilitati come analogici.
2. Gli assi tra parentesi quadre sono scambiati.
3. Gli assi HOTAS con un meno (-) davanti verranno invertiti.
4. Gli assi 1 e 2 dovranno essere presenti come richiesto dalle DirectX.

Vediamo un esempio:

```
USE AXES_CONFIG (1, RNG), (2, ANT), (3, -THR)
```

In questo esempio:

1. Range è assegnato all'asse DirectX 1 perciò apparirà come asse X del Joystick al simulatore.
2. Antenna è assegnato al DirectX 2 perciò apparirà come asse Y del Joystick al simulatore.
3. La Manetta è assegnata all'asse 3 (l'assegnazione normale), ma il segno meno davanti invertirà la direzione (utile per i simulatori di elicottero).

4. Gli altri assi non compaiono e quindi non saranno disponibili al simulatore. Questi ultimi possono però essere gestiti tramite le espressioni digitali.

Ora avete visto come sia possibile scambiare, disabilitare ed invertire gli assi con una sola espressione.

NOTE

1. *Non è possibile* DISABLE, USE REVERSE *o* USE SWAP *insieme a* USE AXES_CONFIG. *Viceversa verrà generato un errore. Su un'espressione di pulsante sarà possibile usare* SWAP, REVERSE a condizione che l'asse sia presente.

2. *È molto più semplice usare* USE PROFILE *con ii file di profilo come spiegato dalle note precedenti*.

3. *DirectX mappa a modo proprio gli assi, questo potrebbe portare ad un'assegnazione non corretta. In questo caso sarà necessario sperimentare per ottenere il risultato voluto.*

7. Programmazione del mouse

7.1 Il mouse e il microstick

Nella parte finale di questo capitolo viene spiegato come funziona il mouse e vi verranno spiegati alcune cose che possono essere fatte. Questa volta però non spiegheremo direttamente il codice di programmazione o la sintassi, come abbiamo fatto finora, perché abbiamo bisogno di capire realmente come funziona un mouse.

Sono sicuro che sapete come funziona il microstick sulla manetta TQS e lo paragonate ad un mouse perché effettivamente muove il cursore del mouse. Tuttavia è importante capire questo punto:

Il microstick NON È un mouse

Il microstick è come un mini-joystick. *Normalmente* controlla il mouse perché il compilatore gli dice di farlo. Il compilatore gli assegna due assi, (MIX e MIY) il device **mouse**, senza che voi ve ne accorgiate.

Allora, cos'è una mouse device? E perché non possiamo semplicemente assegnare gli assi X e Y al microstick?

La ragione è questa: il mouse come tutte le altre periferiche **non è** un asse come quelli del joystick. Confusi? Bene, se pensate al joystick, quando lo muovete esso genera coordinate fisse X e Y per conto suo. Se il joystick controllasse un cursore come succede nel Foxy Joystick Analiser, il suo movimento corrisponderebbe al movimento del joystick. Se il joystick si ferma, anche il cursore si ferma. Ma il microstick programmato come un mouse ha un comportamento differente. Se lo muovete come un joystick e lo mantenete in una posizione diversa dal centro, il mouse continua a muoversi. Non si ferma nemmeno se il microstick è fermo.
Questo succede perché il microstick sta dicendo al computer "*hey, muovi il cursore in questa posizione X e Y e poi fermati,"* invece di dirgli "*continua a muovere il mouse alla velocità 3 sull'asse X e alla velocità 2 sull'asse Y finché non ti ordino diversamente".* Quindi, quando il microstick ritorna alla sua posizione centrale, il cursore non si muove verso il centro: si ferma dov'è, perché il microstick ha cambiato l'istruzione in "*OK, puoi smettere di muovere il mouse sugli assi X e Y ora"*

Quindi un cursore del mouse può essere mosso assegnando valori non nulli alle variabili MouseX e/o MouseY (MSX e MSY). E se ordiniamo al microstick di variare i valori di MSX e MSY, il microstick controllerà il mouse.

E la ragione per cui lo abbiamo implementato in questo modo, spero inizi a chiarirsi anche per voi. Possiamo usare qualunque cosa per regolare i valori MSX e MSY. Il microstick, il joystick, un hat, un pulsante, flag logici … quello che volete. Ed essi possono agire contemporaneamente. Quindi possiamo programmare il microstick per muovere il mouse velocemente e un hat per gli aggiustamenti di precisione!

7.2 USE MTYPE – la via più facile per assegnare il mouse al microstick

Più avanti in questo capitolo, vi spiegheremo come potete assegnare il mouse al microstick per ottenere esattamente la risposta che volete da esso. Però per fare ciò, dovete prima comprendere dettagliatamente l'utilizzo delle espressioni Digital Type utilizzate nei comandi MSX e MSY. Comunque, fortunatamente ci sono un paio di espressioni che possiamo utilizzare per assegnare il movimento del mouse al microstick molto facilmente. Queste utilizzano i comandi USE MTYPE e USE MICROSTICK AS MOUSE. Partiamo col comando USE MTYPE visto che è molto semplice da capire ed usare.

Espressione di configurazione:

USE MTYPE Type **-** REVERSE_*type*

dove:

Type: va da A1 ad A5 e descrive quale pulsante della manetta sarà utilizzato per simulare i pulsanti sinistro e destro del mouse, secondo la tabella che segue:

Tipo	Pulsante Sinistro	Pulsante Destro
A1	T1	T6
A2	T6	T1
A3	T1	Nessuno
A4	T6	Nessuno
A5	Nessuno	Nessuno

T1 è il pulsante inserito nel microstick – si preme il microstick per attivarlo, e T6 è il pulsante sulla manopola Range.

REVERSE_*type* è REVERSE_UD e/o REVERSE_LR

Il REVERSE_UD inverte il movimento dell'asse Up/Down (Y) del mouse, e REVERSE_LR inverte il movimento delle direzioni destra/sinistra (Left/Right o X).

Esempi: USE MTYPE A3

Assegna il mouse al microstick e il pulsante sinistro a T1.

USE MTYPE A5 - REVERSE_UD

Assegna il mouse al microstick, invertendo l'asse Y, e non assegna alcun pulsante.

NOTE

1. *La risposta del mouse è settata automaticamente dal compilatore per ottenere un movimento accettabile per risoluzioni fino a 1024x768. non potete alterare la risposta del mouse con un'espressione USE MTYPE, quindi se utilizzate risoluzioni maggiori, o volete un mouse più pronto nella risposta, dovete utilizzare l'espressione USE MICROSTICK AS MOUSE che spiegheremo nella prossima sezione.*

2. *Come abbiamo detto in precedenza, il microstick è un controller analogico. Non è un controller a quattro pulsanti, come nell'HOTAS originale Thrustmaster, e di conseguenza i pulsanti da T11 a T14 non esistono più nella programmazione. È possibile emulare i pulsanti da T11 a T14 con espressioni Type appropriate se volete farlo – riferitevi all'help file di Foxy chiamato "Converting TQS T11-T14 statements for use wih the Cougar's Microstick". Ricordatevi che il microstick altro non è che un controller a due assi, non una serie di pulsanti. In questo modo è infinitamente più potente.*

3. *Se assegnate una qualunque curva al microstick, non influenzerà il mouse, visto che è assegnato a espressioni digitali e le espressioni digitali non sono influenzate dalle curve analogiche.*

4. *Se utilizzate un'espressione* USE MTYPE *nel vostro file di configurazione, la quale imposta un qualsiasi pulsante del mouse su T1 o T6, non potete più programmare tali posizioni. Quindi, ad esempio, se noi abbiamo:*

   USE MTYPE A3

   a cui il compilatore, in maniera trasparente, aggiunge l'espressione

   BTN T1 /H MOUSE_LB

   ora, se da qualche altra parte nel file avete:

   BTN T1 *macro_qualsiasi* *oppure* BTN T1 /H MOUSE_RB

il compilatore genererà un errore. Se volete utilizzare USE MTYPE *per impostare il mouse sul microstick, ma volete programmare T1 e/o T6 separatamente, utilizzate un'espressione del tipo* USE MTYPE A5 *o qualcosa di appropriato che non assegni alcun pulsante del mouse al pulsante della manetta che volete utilizzare.*

7.3 USARE IL MICROSTICK COME MOUSE

Il secondo modo per assegnare il mouse al microstick avviene tramite l'utilizzo dell'espressione USE MICROSTICK AS MOUSE. Tale istruzione ha il vantaggio, rispetto a USE MTYPE, di permettervi di modificare il comportamento di default del mouse, ad esempio la velocità di movimento. Inoltre, assegna il pulsante sinistro del mouse al pulsante T1 del microstick di default, sebbene questo possa essere cambiato tramite il modificatore NO_BUTTON. Per la maggior parte dei simulatori di volo ovviamente, vorremo avere il pulsante sinistro assegnato a T1. l'espressione USE MICROSTICK AS MOUSE ordina al compilatore di assegnare un'espressione digitale Type 6 per il mouse sul microstick, oppure se viene fornito un valore di partenza, espressioni Type 5.

Diamo ora un'occhiata alla sintassi:

Espressioni di configurazione

Per espressioni Type 6 : *(Nessun valore di partenza)*

USE MICROSTICK AS MOUSE (Scale value, Increment value) - *Modifier*

Per espressioni Type 5 : *(Fornito valore di partenza)*
USE MICROSTICK AS MOUSE (Scale value, Increment value, Starting value) - *Modifier*

*(**Nota**: l'espressione USE MICROSTICK AS MOUSE ( ) assegna inoltre il pulsante sinistro del mouse al pulsante T1 sul microstick a meno che non venga utilizzato il modificatore NO_BUTTON.)*

dove:

Scale value: un numero compreso tra 2 e 12. Questo indica in quante bande sono suddivisi gli assi del microstick.

Increment value: Valore incrementale per ogni banda, un numero tra 1 e 63.
(Notate che il valore di Scale Value moltiplicato Increment Value deve essere minore di 128. Non pensate nemmeno di chiedervene il motivo!)

Starting value: Il valore di partenza da cui applicare il valore incrementale. Questa è la velocità di spostamento iniziale a cui si muoverà il mouse quando il microstick lascerà la posizione neutra.

Modifier:

REVERSE_UD:	Inverte l'asse Y del microstick.
REVERSE_LR:	Inverte l'asse X del microstick.
NO_BUTTON:	Ordina al compilatore di non assegnare il pulsante sinistro del mouse al pulsante T1. Questo vi permette di programmare T1 con un'espressione BTN T1.

Ora passiamo direttamente ad alcuni esempi.

```
USE MICROSTICK AS MOUSE (12, 2)
```

Questa espressione assegna il mouse al microstick, e assegna T1 come pulsante sinistro. Guardate le due seguenti espressioni:

USE MICROSTICK AS MOUSE (6, 4)
USE MICROSTICK AS MOUSE (7, 3, 2)

Anche queste due assegnano il mouse al microstick e designano T1 come pulsante sinistro. Quindi qui abbiamo tre espressioni, chiaramente differenti, che però, vi ho detto, eseguono le stesse operazioni. Ok, devo correggermi dicendo che fanno le stesse cose *in termini* di assegnare il mouse al microstick, ma sono diverse per quanto riguarda la risposta del mouse che otterrete. Questo significa che devo cercare di spiegarvi cosa significano i numeri tra parentesi

Bene, ora andiamo a spiegare cosa *esattamente* le funzioni delle espressioni sopraccitate, il che renderà un po' difficile la lettura. Qui di seguito c'è una spiegazione più leggera, in modo che stia nel mio semplice cervello!

Consideriamo solo l'asse X del microstick. La funzione di una qualsiasi espressione USE MICROSTICK AS MOUSE è quella di dividere gli assi in una serie di bande e regioni, come nel diagramma qui sotto *(Per dovere di cronaca, ho rappresentato le bande come aventi tutte la stessa dimensione, anche se i puristi tra di voi potrebbero obiettare che questo non è proprio come nella realtà)*

Ora il verde rappresenta il centro dell'asse, ad esempio il microstick nella sua posizione centrale. In tale posizione, ovviamente, non vogliamo muovere il mouse. Ai lati del centro, ci sono due bande gialle (1), che è la direzione vogliamo che il mouse inizi a muoversi quando il microstick viene mosso all'interno di tali bande. E ci sono altre due serie di bande (2 & 3) che indicano la posizione in cui il microstick viene mosso ancora più lontano dal centro.

Ora torniamo alla sintassi:

USE MICROSTICK AS MOUSE (Scale value, Increment value, *optional* Starting value)

Lo Scale Value determina in quante bande viene suddiviso l'asse del microstick. Il valore di Scale Value non è uguale al numero di bande. In realtà è parte di un'equazione, ma sostanzialmente, maggiore è Scale Value, maggiore sarà il numero di bande.
L'Increment Value determina quanto velocemente si muoverà il mouse nel momento in cui muoveremo il microstick entro ogni banda. Fatemi spiegare con un esempio, mostrandovi cosa succede quando il microstick viene mosso dalla sua posizione centrale alla estremità, passando all'interno delle bande. Parleremo più tardi dello Starting Value.

USE MICROSTICK AS MOUSE (4, 2)

Banda C: il microstick è in posizione centrale quindi il mouse non si muove.

Banda 1: il mouse ora inizia a muoversi ad un rateo data dall'Increment Value, "*una velocità di 2*" se volete

Banda 2: il rateo del mouse è aumentata dall'Increment Value, quindi ora si muove a *una velocità di 4*

Banda 3: il rateo del mouse è aumentata dall'Increment Value, quindi ora si muove a *una velocità di 6*

Banda 4, 5, 6 etc.

A seconda di quante bande lo Scale Value ha creato, potete vedere il mouse muoversi sempre più velocemente man mano che il microstick viene spostato verso la banda successiva. Riportandolo indietro verso la posizione centrale lo farà, ovviamente, rallentare.

Ora vediamo gli effetti che ha fornire una Starting Value

USE MICROSTICK AS MOUSE (4, 2, 1)

Lo Starting Value determina il rateo a cui il mouse si muoverà nella banda 1. Dopodiché, il resto è esattamente uguale all'aumento di velocità dato dall'Increment Value come detto in precedenza. Quindi:

Banda C: il microstick è in posizione centrale quindi il mouse non si muove.

Banda 1: Il mouse inizia a muoversi al rateo fornito dallo Starting Value, *"alla velocità 1"* se volete

Banda 2: il rateo del mouse è incrementato dall'Increment Value, quindi ora si muove alla *velocità 3* (ad esempio 1 + 2, lo Starting Value + l'Increment Value)

Banda 3: il rateo del mouse è aumentato dall'Increment Value, quindi ora si muove a *velocità 5*

Banda 4, 5, 6 etc.

A seconda di quante bande lo Scale Value ha creato, potete vedere il mouse muoversi sempre più velocemente man mano che il microstick viene spostato verso la banda successiva. Riportandolo indietro verso la posizione centrale lo farà, ovviamente, rallentare

Non preoccupatevi se non capite tutto al primo colpo, sto solo cercando di spiegarvi un concetto di massima: numeri più grandi corrispondono ad un mouse più veloce.

Possiamo anche invertire le direzioni degli assi su cui si muoverà il mouse, come abbiamo fatto nelle altre espressioni spiegate nei precedenti capitoli, come queste:

```
USE MICROSTICK AS MOUSE (7, 3, 2) - REVERSE_UD
USE MICROSTICK AS MOUSE (7, 3, 2) - REVERSE_LR
```

Prima di lasciare questa sezione, devo, per dovere di completezza, spiegarvi gli altri usi di queste istruzioni e questi riguardano l'assegnazione del mouse ad altri assi.

7.3.1 Assegnare altri assi al mouse

Ora, l'uso di USE MICROSTICK AS MOUSE è di fatto solo un caso particolare della seguente espressione (particolare perché assegna anche il pulsante sinistro a T1):

Espressioni di configurazione

Per espressioni Type 6:

USE *Axis_Identifier* AS *Mouse_Axis* (Scale value, Increment value) - REVERSE_*type*

Per espressioni Type 5:

USE *Axis_Identifier* AS *Mouse_Axis* (Scale value, Increment value, Starting value) - REVERSE_*type*

dove:

Axis_Identifier è uno dei seguenti:

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

Mouse_Axis è uno dei seguenti:

MOUSE
MOUSEX

MOUSEY
MOUSEZ (la rotellina del mouse)

Scale value: un numero compreso tra 2 e 12. indica il numero di bande in cui Axis_identifier verrà suddiviso.

Increment value: Valore incrementale per ogni banda, un numero tra 1 e 63.
(Notate che il valore di Scale Value moltiplicato Increment Value deve essere minore di 128. Non pensate nemmeno di chiedervene il motivo!)

Starting value: Il valore di partenza per un'espressione Type 5 da cui applicare il valore incrementale. Questa è la velocità di spostamento iniziale in cui il mouse si muoverà quando *Axis_identifier* lascerà la posizione neutra.

REVERSE_*type*: inverte la direzione di un asse e può valere:

REVERSE_UD: quando *Axis_identifier* è composto da due assi (JOYSTICK, ROTARIES, MICROSTICK, TOEBRAKES).

REVERSE_LR: quando *Axis_identifier* è composto da due assi può essere usato da solo o insieme a REVERSE_UD

REVERSE_DIR: inverte un singolo asse. Non può essere usato insieme a REVERSE_UD o REVERSE_LR.

Possiamo quindi usare altri assi per controllare il mouse, nella stessa maniera. Qui ci sono alcuni esempi:

```
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)
```

Quest'ultimo controlla il mouse usando l'asse Y del joystick!

Ricapitolando abbiamo visto due comandi che possono essere usati per impostare il microstick come mouse e, nell'ultima sezione, su altri assi. Ricordatevi inoltre che quando abbiamo trattato la programmazione dell'HAT, abbiamo detto che possiamo assegnare il mouse ad un HAT usando la seguente espressione:

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

ad esempio. Di fatto possiamo anche avere il mouse sul microstick e sull'hat nello stesso momento, usando il microstick per portare velocemente il mouse nella zona voluta e poi usando l'hat per gli aggiustamenti di precisione! [applausi grazie!]

Ora voglio spiegarvi esattamente cosa succede, o meglio dovrei dire, come il compilatore interpreta le vostre espressioni e come crea i comandi digitali per gli assi del microstick. Risulta un po' difficile da spiegare e perciò non ce ne addentreremo subito. La prossima sezione è solo per gli utenti esperti. Se riuscirete a capire questa sezione, potrete programmarvi il vostro mouse personalizzato su qualsiasi asse, per ottenere esattamente la risposta che volete. Potrete inoltre combinare espressioni per il mouse con altre espressioni sul microstick, così per esempio con il pulsante S3 rilasciato il microstick controllerà il cursore di puntamento e con S3 premuto controllerà il mouse. Ingegnoso, eh?

7.4 Creare un mouse personalizzato sul microstick

Abbiamo visto come sia possibile assegnare il mouse al microstick nella sezione precedente con le espressioni USE MTYPE e USE MICROSTICK AS MOUSE. Cosa queste espressioni fanno è semplicemente programmare gli assi del microstick con espressioni digitali. In questa sezione dunque, voglio spiegarvi come potete creare le vostre espressioni digitali personalizzate per programmare il mouse sul microstick o altrove.

Potete programmare il microstick con qualunque espressione digitale, dato che sono dopotutto solo assi, soggetti alle stesse regole degli altri assi. Partirò dimostrandovi l'uso delle espressioni Type 1 e Type 2, benché non ci sia alcun motivo per cui non possiamo usare una degli altri sei tipi di espressioni digitali. Diamo un'occhiata alle espressioni Type 1:

```
MIX 1 14 MSX (2+) MSX (2-) MSX (0)
MIY 1 14 MSY (2-) MSY (2+) MSY (0)
```

Bene, queste somigliano a ordinarie Type 1, eccettuato alcuni segni "-" e "+" al loro interno. Per capirli vi mostrerò cosa succede quando muovete il microstick. Ricordiamoci di come funzionano le Type 1. Se abbiamo la seguente

```
MIX 1 6 r l c
```

L'asse del microstick, se mosso alla sua estrema sinistra alla sua estrema destra e ancora indietro verso la sua estrema sinistra, produrrà i seguenti caratteri:

```
r r r c r r r l l l c l l l
```

Ora, nel nostro esempio abbiamo:

```
MIX 1 14 MSX (2+) MSX (2-) MSX (0)
```

Non stiamo producendo una serie di caratteri. I numeri nelle parentesi indicano cosa aggiungere o sottrarre al buffer del mouse attuale – in altre parole, quanto più velocemente o lentamente si muoverà il mouse.

Poniamo di avere il mouse in posizione stazionaria e il microstick in posizione centrale. La posizione centrale dell'asse del microstick è data dai "caratteri" della parte centrale dell'espressione digitale – esempio MSX (0). Questo ordina al device del mouse, di mantenere velocità zero, ad esempio rimanere stazionaria sull'asse X. Appena muoviamo il microstick a destra, il mouse inizierà a muoversi verso destra ad un *rateo di 2* , visto che un 2 viene aggiunto al buffer del mouse. Muovete il microstick ancora di più verso destra ed egli muoverà il mouse con rateo 4 … muovetelo verso l'estrema destra e muoverà il mouse con rateo 14 (7 x 2). Rilasciamo il microstick e il rateo del mouse rallenterà man mano che quantità di rateo verranno sottratte dal buffer, e quindi dalla velocità, finché lo stick non sarà al centro. Si fermerà con lo stick al centro perché siamo di nuovo sul carattere di centro MSX (0).

La stessa cosa accade per la stessa ragione quando il microstick viene mosso verso l'alto o verso il basso.

Notate che se io avessi voluto far operare il microstick sull'asse Y in modo che funzionasse come un joystick (tirare lo stick fa salire il mouse), avrei dovuto cambiare l'espressione in:

MIY 1 14 MSY (2+) MSY (2-) MSY (0)

Questo perché se incrementate il buffer Y del mouse (MSY) il mouse si muove verso il basso e vice versa. Un punto importante da notare nella sintassi quindi è l'inclusione dei caratteri **+** e **–** *all'interno* delle parentesi e il fatto che essi appaiono **dopo** il numero. Essi indicano se il valore che appare prima deve essere aggiunto o sottratto dal buffer di quel particolare asse del mouse. Capirete meglio ciò che ho detto quando vedremo le espressioni Type 2. Spero di non avervi ancora persi… perlomeno non ancora!

Potrei anche usare espressioni Type 2 per assegnare il mouse agli assi del microstick:

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)

Queste sono espressioni digitali standard Type 2. esse dividono ogni asse in 9 bande uguali e assegnano il valore reale del buffer del mouse ad ogni banda. Quindi muovere il microstick sul suo asse X corrisponde a muoverlo inizialmente ad un rateo di 1, continuando si passerà ad un rateo di 2, quindi 4 e alla fine 8. Notate qui la posizione del segno "-". Stavolta è **prima** del numero, indicando che non deve essere sottratto al buffer del mouse. Piuttosto, quando il microstick entra in quest'area, assume esattamente questo valore negativo.

In termini di sintassi, notate che le seguenti espressioni sono uguali:

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)

Ad esempio, se un degno "+" viene omesso, si assume che sia indicato a sinistra del valore

mentre:

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+)

Produrrà risultati *molto* diversi

Con tutto questo in mente, vediamo cosa fa il compilatore quando incontra un'espressione USE MICROSTICK AS MOUSE nel vostro file di configurazione. Il compilatore converte questa espressione in espressioni digitali Type 5 e Type 6, le quali sono completamente uguali rispettivamente alle Type 2 e Type 1, eccetto il fatto che possiamo determinare le dimensioni delle bande. Rivediamo la sintassi di questa espressione:

USE MICROSTICK AS MOUSE (Scale value, Increment value, *optional* Starting value)

Ho detto in precedenza che Scale Value è usato per determinare il numero di bande in cui dividere l'asse. Questo viene fatto tramite la formula

Numero di bande = (Scale value x 2) - 1

Il compilatore quindi crea queste bande con dimensioni differenti, usando una formula complessa che non sto a spiegare, e crea le espressioni digitali Type 6 se non è fornito uno Starting Value o Type 5 se viene fornito lo Starting Value.

Espressioni Digitali Type 6

USE MICROSTICK AS MOUSE (2, 2)

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 Preme il pulsante sinistro del mouse quando viene premuto T1

Ora, abbiamo fatto qualcosa di diverso qui. Non ho MSX(0) e MSY(0) come caratteri di centro, come questi

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)

Ora vi spiego perché ci sono vantaggi e svantaggi nell'avere caratteri nulli (^) invece di MSX (0), MSY (0). Queste espressioni Type 6 si differenziano dalle Type 5 perché aggiungono o sottraggono dei valori X e Y dal buffer del mouse, mentre le espressioni Type 5 impostano i valori stessi del buffer. Quando utilizziamo MSX (0) e MSY (0) come caratteri di centro, questi resettano il buffer del mouse a zero quindi è garantito che il mouse si fermerà quando l'asse che li controlla raggiungerà la posizione centrale. Questo in generale va bene. Ma la bellezza di usare caratteri null come caratteri centrali è che noi possiamo assegnare o controllare il mouse da hat e pulsanti allo stesso modo del microstick, e avere valori prevedibili. Quindi posso avere:

```
USE MICROSTICK AS MOUSE (2, 2)
BTN H1L MSX(1-)
BTN H1R MSX(1+)
```

e usare il microstick per il controllo generale del mouse e le posizioni sinistra e destra dell'Hat1 per i controlli fini della direzione. Questo è molto importante da ricordare ... dipende da come volete che si comporti il vostro mouse. Sfortunatamente non c'è modo di dire al compilatore di usare MSX(0), MSY(0) come caratteri centrali con un'espressione USE MICROSTICK AS MOUSE. Nel caso vogliate farlo ugualmente, dovrete usare le espressioni MIX e MIY o fornire uno Starting Value, il quale utilizzerà un'espressione Type 5 che usano tali codici come caratteri centrali.
Ora vediamo cosa fa il compilatore con alcune altre espressioni USE MICROSTICK AS MOUSE Type 6

USE MICROSTICK AS MOUSE (3, 4)

Viene convertito in:

```
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)

Viene convertito in:

```
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)

Viene convertito in:

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

Viene convertito in:

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

Ora vediamo cosa accade quando forniamo uno Starting Value nell'espressione USE MICROSTICK AS MOUSE. Il compilatore converte in espressioni digitali Type 5

Espressioni digitali Type 5

USE MICROSTICK AS MOUSE (2, 2, 3)

Viene convertito in:

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

Notate che il numero di bande che vengono create è 3. la banda centrale è utilizzata per assicurare che il mouse sia stazionario, e le due bande ai suoi lati assumono lo Starting value, quindi effettivamente l'Increment value è ignorato in questo caso.
Ora paragoniamolo con l'espressione seguente:

USE MICROSTICK AS MOUSE (3, 2, 3)

Viene convertito in:

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

Ora potete vedere la relazione tra Starting Value e Increment Value. Oppure se non riuscite, paragonate quelle qui sotto e dovreste farcela.

USE MICROSTICK AS MOUSE (3, 8, 1)

Viene convertito in:

```
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)

Viene convertito in:

```
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

Viene convertito in :

```
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
```

E per completezza, l'assegnamento ad altri assi:

USE JOYSTICK AS MOUSE (3, 2, 3)

```
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)

Viene convertito in:

```
JOYY 6 7 (2 12 23 36 65 78 89 98) MSY(2-) MSY(2+) ^
```

Bene, spero di avervi spiegato il tutto chiaramente! Andiamo avanti.

7.5 USARE ZERO_MOUSE

Questa espressione di configurazione è utile quando impostate le vostre espressioni personalizzate sul microstick, insieme a modificatori /I e /O, per prevenire il fatto che il mouse si blocchi.

Abbiamo visto, nella sezione precedente, come possiamo usare espressioni digitali per creare un mouse personalizzato sul microstick. Un'espressione USE ZERO_MOUSE può essere usata per assicurarsi che il mouse non si pianti quando vengono combinate espressioni che contengono modificatori /I e /O. Considerate il seguente esempio:

```
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
```

in questo esempio, se premete il pulsante S3 sul vostro joystick abiliterete il microstick al movimento del mouse. Se, mentre il mouse è in movimento, rilasciate il pulsante S3, il mouse continuerà a muoversi e si bloccherà quando raggiungerà il bordo dello schermo. Inserire un'espressione USE ZERO_MOUSE eviterà tutto questo, forzando il mouse a fermarsi quando S3 viene premuto/rilasciato.

Ora bisogna dire che questo comportamento del mouse non è propriamente un baco. Il controllo del mouse tramite il microstick si comporta come è stato programmato per fare. Potete sempre evitare, nell'esempio qui sopra, il blocco riportando il microstick in posizione centrale prima di rilasciare il pulsante S3. la regola d'oro quando si piantano i movimenti del mouse o il comportamento dei tasti è questa: dovete utilizzare i pulsanti del controller, gli hat e gli assi per quello che la loro programmazione gli permette di fare. Questa regola è importante per coloro che non ci pensano mai! ☺

7.6 Programmazione dei pulsanti del mouse

Potete assegnare i pulsanti del mouse a pulsanti del joystick, assi … beh… come fate con tutti gli altri pulsanti. Qui di seguito la sintassi:

Sintassi dei pulsanti del mouse:

```
MOUSE_LB
MOUSE_RB
MOUSE_MB
```

(MB = *pulsante centrale sui mouse a tre pulsanti*)

Ecco un esempio:

```
BTN T1 /I /H MOUSE_RB
       /O /H MOUSE_LB
```

Con questa espressione, stiamo assegnando il pulsante sinistro del mouse al pulsante T1 del microstick quando il pulsante S3 non viene premuto.
Quando S3 viene premuto, a T1 viene assegnato il pulsante destro del mouse.

Potete anche usare KD e KU con i pulsanti del mouse:

```
BTN S1 KD(MOUSE_LB) DLY(2000) KU (MOUSE_LB)
```

Quando il pulsante S1 viene premuto, il pulsante sinistro del mouse viene premuto per 2 secondi e quindi rilasciato.

7.7 Disabilitare l'assegnamento di default del mouse al microstick

Nella finestra delle opzioni in Foxy (Preferences), c'è una cartella chiamata "Defaults". La funzione delle opzioni contenute in questa cartella è di ordinare al compilatore di impostare automaticamente le funzioni selezionate nel caso non vi siano espressioni che le riguardano.

Uno dei settaggi è "Assign mouse to microstick". Se questa viene selezionata, quando caricherete il file sul vostro controller, il compilatore imposterà il mouse sul microstick di default e il pulsante sinistro su T1. Questo è stato fatto perché la maggioranza degli utenti vuole il mouse sul microstick di default, ma potrebbe non aver letto a sufficienza il manuale di riferimento per capire quali espressioni usare per programmarlo.

Se volete essere in grado di usare gli assi del microstick in un gioco, ma non volete avere il mouse assegnato di default, senza dover necessariamente disabilitare l'opzione in Foxy, dovrete usare la seguente espressione:

Espressione di configurazione

```
DISABLE MOUSE
```

nl vostro file di configurazione. Questo impedirà al compilatore di assegnare le espressioni digitali di default al microstick *nel caso Foxy sia impostato per farlo.*

Potete in ogni caso assegnare il mouse ad un Hat Switch o ad altri assi. Ricordatevi che il device del mouse è sempre presente – dovete solo assegnarlo a qualcosa se volete controllare il mouse.

7.8 Espressioni avanzate per i movimenti del mouse

Abbiamo già visto che è possibile muovere il mouse attraverso particolari istruzioni. Nella parte finale di questo capitolo, andremo a vedere come effettuare movimenti più complessi.

7.8.1 Definire la risoluzione dello schermo

Con tutte le istruzioni che spiegheremo in seguito, è **essenziale** definire la risoluzione dello schermo che state utilizzando per giocare, con l'istruzione:

Espressione di configurazione:

USE SCREEN_RESOLUTION (X,Y)

Esempio:

USE SCREEN_RESOLUTION (800,600)

Il valore minore che potete inserire è 640x480, rispettivamente per X e Y.

Nota tecnica: 800 e 600 descrivono la risoluzione in pixel. Un pixel è il più piccolo punto disegnabile sullo schermo. Quindi, in questo esempio, lo schermo corrisponde a 800 linee e 600 colonne di punti. Ovviamente una risoluzione di 1600 x 1200 è maggiore e visualizza immagini più definite.

Possiamo usare i seguenti comandi con il mouse:

Sintassi:

Muovere il mouse ad una posizione specifica
MOUSEXY (*Origin*,X,Y)

Muovere il mouse in una posizione relative a quella attuale
MOUSEMOVE (X,Y)

Rotazione / Movimento poligonale
MOUSEROTATE (*Origin*, CentrePoint, Radius, Start angle, Macro1, Rotate direction, Final angle, Number of steps, Macro2)

7.8.2 Muovere il mouse in una posizione assoluta

Sintassi :

MOUSEXY (*Origin*,X,Y)
MOUSEXY (*Origin*,X%,Y%)

dove

- *Origin* è uno dei quattro angoli dello schermo UL, DL, UR, DR.
- X,Y sono le coordinate X e Y in cui vogliamo muovere il mouse
- X%, Y% è una percentuale di movimento da zero a 100% (precisione ai tre numeri decimali) in cui vogliamo muovere il mouse. Utilizzare la percentuale ci permette di modificare la risoluzione senza dover modificare la linea di comando

Non c'è modo di sapere, dall'interno di un gioco, dov'è il cursore del mouse in ogni momento, o di essere in grado di tracciarne i movimenti. Quindi, per essere in grado di muoverlo in una determinata posizione, dobbiamo prima portarlo ad uno degli angoli dello schermo, in modo da sapere esattamente le coordinate della sua posizione. Queste sono contenute nell'espressione SCREEN_RESOLUTION e il compilatore può calcolare dove muovere il mouse partendo da qui. Questo avviene molto rapidamente e non dovrebbe essere un problema, ma sarebbe meglio assicurarsi che nessun pulsante del mouse venga premuto prima di farlo.

Quindi l'istruzione:

```
BTN H3D MOUSEXY (UL, 400, 300)
```

Prima muove il mouse molto rapidamente nell'angolo in alto a sinistra, quindi lo porta al punto 400,300, ad esempio il centro dello schermo quando si usa una risoluzione di 800x600.
Poniamo ad esempio che dobbiate premere F2 per visualizzare il cockpit, muovere il mouse verso un pulsante al suo interno, premere tale pulsante, quindi tornare alla visuale esterna premendo F1. La seguente espressione eseguirà questa sequenza:

```
BTN H3D F2 MOUSEXY (UL, 400, 300) MOUSE_LB F1
```

Ora, se modifichiamo l'espressione in questo modo:

```
BTN H3D F2 MOUSEXY (UL, 50%, 50%) MOUSE_LB F1
```

Nel caso un utente cambi la sua risoluzione a 1600x1200 (magari!), e cambi il suo USE SCREEN_RESOLUTION per istruire il compilatore, l'espressione genererà lo stesso risultato che si otteneva in 800 x 600.

NOTE

1. *Un'espressione* USE SCREEN_RESOLUTION ( ) *deve essere per forza presente nel file di configurazione per poter utilizzare gli esempi qui sopra.*

2. *Ogni espressione riferita al mouse è soggetta alla restrizione di velocità imposta dall'espressione RATE. Notate che di default non è presente l'istruzione* USE RATE (*nnnn*) *nel file di configurazione e sarà posta a zero – la risposta più veloce.*

7.8.3 Muovere il mouse relativamente alla sua posizione attuale

Sintassi :

```
MOUSEMOVE (X,Y)
MOUSEMOVE (X%,Y%)
```

Dove:

- X,Y indica il numero di pixel di cui si deve spostare il mouse partendo dalla sua posizione attuale.
- X%, Y% è una percentuale di movimento da zero a 100% (precisione ai tre numeri decimali) in cui vogliamo muovere il mouse. Utilizzare la percentuale ci permette di modificare la risoluzione senza dover modificare la linea di comando

Dovrete sapere dove si trova il mouse attualmente e per questo prima dovrete impostare un'espressione MOUSEXY per posizionarlo in un punto noto.

Ora vediamo alcuni esempi che mostrano l'uso di MOUSEMOVE:

BTN TG1 MOUSEMOVE (20, 50)

La pressione del TG1 (Trigger1) muoverà il mouse di 20 pixel a destra sul suo asse orizzontale e di 50 in basso sull'asse verticale.

BTN S1 MOUSEMOVE (20%, 50%)

La pressione del pulsante S1 muoverà il mouse sul suo asse orizzontale per una distanza pari al 20% della larghezza dello schermo.
Contemporaneamente si muoverà del 50% dell'altezza schermo, verso il basso sull'asse verticale. Ora, se utilizzate una risoluzione di 800 x 600, significa che il mouse si muoverà di 160 pixel verso destra (20% di 800) e di 300 pixel verso il basso (50% di 600).

BTN H2R MOUSEMOVE (30, 0)

Lo spostamento verso destra dell'Hat 2 farà muovere il mouse di 30 pixel verso destra ... ad esempio su una linea orizzontale. Se volete farlo muovere verso sinistra dovete inserire un segno "-" prima del 30.

BTN H2U MOUSEMOVE (0%, -50%)

Lo spostamento verso l'alto dell'Hat 2 sposterà il cursore del mouse su una linea verticale (in alto) di una distanza pari al 50% della risoluzione verticale – equivalente a 300 pixel in 800 x 600.

NOTE

1. *Un'espressione* USE SCREEN_RESOLUTION( ) *deve essere per forza presente nel file di configurazione per poter utilizzare gli esempi qui sopra o verrà generato un errore dal compilatore.*

2. *Ogni espressione riferita al mouse è soggetta alla restrizione di velocità imposta dall'espressione RATE. Notate che di default non è presente l'istruzione* USE RATE (*nnnn*) *nel file di configurazione e sarà posta a zero – la risposta più veloce.*

3. *Non è possibile combinare valori percentuali e valori assoluti in queste espressioni. Quindi:*

BTN H2U MOUSEMOVE (0%, -50%)

è corretto, mentre:

BTN H2U MOUSEMOVE (0, -50%)

genererà un errore in compilazione.

4. *Con un'espressione* MOUSEMOVE *il mouse si muoverà su entrambi i suoi assi allo stesso tempo. Quindi espressioni tipo:*

BTN S1 MOUSEMOVE (100, 100)

Muoverà il mouse diagonalmente verso il basso e verso destra. ***Non*** *eseguirà il comando in due parti, ovvero prima 100 pixel verso il basso e poi 100 verso destra.*

7.8.4 Rotazione / Movimento poligonale

Sintassi:

MOUSEROTATE **(**Origin, CentrePoint, Radius, Start angle, [Macro], Rotate direction, Final angle, Number of steps **)**

Dove :

- *Origin* è uno degli angoli dello schermo UL, DL, UR, DR (Up Left, Down Left, Up Right, Down Right).
- *CentrePoint* è il centro di rotazione passato come coordinata: X,Y o come %
- *Radius* è il raggio di un cerchio che definisce l'arco di rotazione in pixel o in percentuale (vedere la sezione *notes*)
- *Start angle* Angolo di partenza, compreso tra zero e 360°
- *Macro* è una semplice macro ... tipicamente la pressione di un tasto del mouse (ma vedete le restrizioni nella sezione *notes).* Usate un carattere

nullo se non intendete usare macro. La macro deve essere inclusa tra parentesi [ ]

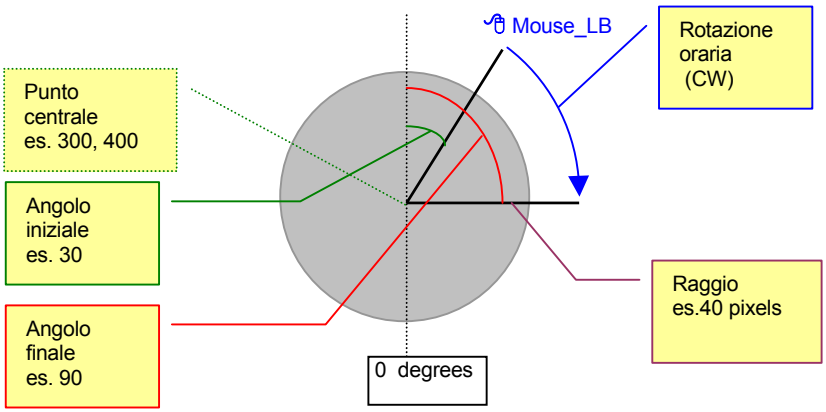
- *Rotate direction* è la direzione di rotazione, la quale può essere CW o CCW (Clockwise or CounterClockwise, oraria o antioraria)
- *Final angle* è la quantità di rotazione desiderata espressa come angolo compreso tra 0 e 1800° (*5 giri completi*)
- *Number of steps* indica il numero di passi della rotazione definisce quanto è uniforme. I valori validi sono 1 o 2. questi producono movimenti nelle seguenti forme:

  Number of steps = 1: un quadrato (*err.. 4 lati!*),
  Number of steps = 2: un ottagono (*8 lati*)

Maggiore è il numero, più lento sarà il movimento, sebbene non sia un rallentamento drammatico.
Ogni valore numerico è preciso ad una cifra decimale (ex. 244.3, 301.8)

Questa funzione è stata introdotta per permettere all'utente di essere in grado, semplicemente premendo un pulsante sul controller, di ruotare una manopola in un cockpit.

Vediamo un esempio in cui voglio ruotare, usando il mouse e il pulsante di sinistra, una manopola nel cockpit, il cui centro è posizionato sullo schermo alla posizione 300,400.



L'espressione di cui ho bisogno è:

USE SCREEN_RESOLUTION (1024,768)
BTN S2 MOUSEROTATE (DL, 300, 400, 40, 30, [MOUSE_LB], CW, 90, 2)

Ora vediamo come funziona procedendo per passi. Quando premo il pulsante S2:

1. DL – il mouse si muove nella posizione in basso a sinistra dello schermo molto velocemente per ottenere il proprio punto di riferimento. Ricordate che il mouse può essere in qualsiasi punto dello schermo quindi prima abbiamo bisogno di muoverlo in un angolo o al centro dello schermo per poter riferire i movimenti a tale posizione.
2. 300,400 – Il mouse usa il centro della manopola sullo schermo per calcolare attorno a cosa ruotare.
3. 40 – il raggio dell'arco di rotazione che disegnerà il mouse
4. 30 – l'angolo di partenza dalla verticale (0 gradi), combinato con le informazioni precedenti determina dove sarà attivata la macro1
5. [MOUSE_LB] - il pulsante sinistro del mouse viene premuto.
6. CW – il mouse ruota in senso orario attorno alla manopola
7. 90 – finché non raggiunge i 90 gradi rispetto alla verticale (0 gradi)
8. 2 – il movimento seguirà un percorso ortogonale. Utilizzate un numero che faccia ruotare la manopola nel modo che volete
9. l'espressione è finita quindi la macro è terminata – es. Il pulsante sinistro viene rilasciato

è complesso, lo so, ma è proprio il dover descrivere il percorso di rotazione del mouse che è complicato! ☺

NOTE SULLA MACRO

1. *La macro verrà sempre eseguita per intero – è come se ci fosse un modificatore /H (hold, mantieni) anteposto. Il codice della macro si interromperà una volta terminato il movimento del mouse*

2. *Qui di seguito la lista dei tasti che è possibile premere (inserire) nella macro:*

   - Caratteri semplici (a, b, 1, 2, `, *etc.*)
   - Combinazioni con SHIFT, ALT, CONTROL (A, ALT b, *etc.*)
   - *Codici DirectX e POV* (DX1, POVL, *etc.*)
   - Pulsanti del mouse MOUSE_LB / RB / MB
   - Pulsanti XFlag (X1, X2, etc.)
   - Pulsanti USB

3. *Non potete usare modificatori a barra,* DLY( ) *o* RPT( ) *nella macro.*

NOTE

1. *Non potete raggruppare espressioni* MOUSEROTATE *con altre espressioni di pressione tasti. quindi:*

BTN S2 a b DLY(30) MOUSEROTATE (*blah blah)* PRNTSCRN

Va bene ma:

BTN S2 a b DLY(30) **{**MOUSEROTATE (*blah blah*) PRNTSCRN**}**

Genererà un errore di compilazione.

2. *Deve essere presente un'espressione* USE SCREEN_RESOLUTION ( ) per *poter usare questa espressione, o verrà generato un errore di compilazione.*

3. *Ogni espressione riferita al mouse è soggetta alla restrizione di velocità imposta dall'espressione RATE. Notate che di default non è presente l'istruzione* USE RATE (*nnnn*) *nel file di configurazione e sarà posta a zero – la risposta più veloce.*

4. *Potete usare definizioni di macro invece di programmare direttamente le macro nell'espressione. Le definizioni più comuni saranno* MOUSE_LB *o* MOUSE_RB (*rispettivamente il pulsante sinistro e destro del mouse.).*

5. **I** *tasti doppi (Shift Sinistro o Shift Destro ad esempio) non funzioneranno nell'espressione* MOUSEROTATE. *Visto che ogni cosa nella macro è per il mouse è raggruppata, usate LSHF invece di SHF.*

6. **I***l raggio può essere espresso come pixel o come % della dimensione X dello schermo. Questo può sembrare a prima vista strano, quindi permettetemi di spiegarvi. Immaginiamo di avere una risoluzione di 800x600, e il raggio pari a 600 pixel (il centro di rotazione giace all'estremità bassa dell'asse X), insolito ma può capitare. Quindi il raggio in percentuale è (600/800*100) = 75%. Se ora volete utilizzare un'altra risoluzione, diciamo 1024x768, il raggio sarà il 75% di 1024, es. 768 pixel. Visto che è possibile avere un raggio più lungo della dimensione Y dello schermo, risulta quindi utile esprimerlo in percentuale dell'asse X. Qui però c'è una considerazione importante da fare. Il rapporto tra X e Y è una cosa critica qui. Per risoluzioni standard (800x600, 1024x768, 1152x864, 1600x1200 ecc..) il rapporto è fisso a 1.333. Però, se utilizzate risoluzioni come la 1280x1024 (rapporto = 1.25) dato che il rapporto è diverso, il raggio non verrà dimensionato correttamente. Fateci quindi attenzione se state inserendo una MOUSEROTATE che utilizza la % invece dei valori numerici dei pixel.*

7. *Quest espressione è complessa, quindi le possibilità di generare un errore di compilazione dimenticandosi alcune parti o una virgola sono alte! Vi raccomando quindi di utilizzare l'Advanced Mouse Programming Wizard in Foxy!*

8. Programmazione Logica

8.1 Programmazione Logica – le basi

8.1.1 Capire i Flag

La programmazione logica è tutta basata sul concetto di **flag.** Cos'è un flag? Iniziamo con alcuni concetti sui flag che potrebbero confondervi completamente le idee e poi li chiariremo repentinamente con un esempio. Ecco qui di seguito i concetti:

- Un flag può essere on o off.
- Ci sono 32 flag e sono identificati da X1 a X32
- Un flag non fa nulla. Semplicemente è on o off.

Confusi? Vi spiegherò come funzionano utilizzando il pulsante Caps Lock come esempio.

Poniamo che il vostro pulsante Caps Lock sia chiamato X1. quando lo premete, si accende la luce corrispondente sulla tastiera, quindi X1 è on. La tastiera sa che se X1 è su on, il tasto "w" invia il carattere "W" invece che "w". Se mettiamo Caps Lock su off premendo il pulsante, la luce si spegne, quindi X1 è ora su off, e il pulsante "w" invierà la lettera in minuscolo.
Possiamo quindi dire che il pulsante Caps Lock in realtà non fa nulla – semplicemente pone X1 a on o off, ma X1 influenza il tipo di carattere che la tastiera invia al computer.

Questo è il concetto importante da capire. Un flag non fa nulla. Semplicemente passa dal valore on a off. Non si nota il fatto che sia su on o off. Ma potete utilizzare il fatto che siano on o off per modificare il comportamento del vostro joystick. Quello che viene fatto nella programmazione logica è programmare gli eventi a seconda dello stato dei flag.

8.2 Definire i flag logici e le espressioni corrispondenti

Iniziamo cercando di capire come impostare un flag a on o a off. Qui c'è la sintassi:

Espressione di configurazione:

DEF X1 S2

Espressione logica:

BTN X1 /H Fire_rockets

Nell'esempio qui sopra, abbiamo DEFinito un flag chiamato X1 utilizzando il comando DEF e abbiamo specificato che è controllato dal pulsante S2. Quando il pulsante del joystick S2 viene premuto, il flag X1 è attivo (on) e quando si rilascia S2 è disattivo (off).

Una volta che avete un flag logico definito, potete programmarlo con un'espressione BTN. Nell'esempio qui sopra, quando viene premuto S2, verranno continuamente lanciati i razzi . Verranno lanciati perché gli avete detto di farlo quando X1 è su on, e X1 resta su on mentre il pulsante S2 è premuto. È importante notare che le espressioni logiche riguardanti i pulsanti sono soggette alle stesse regole a cui sono soggette le espressioni ordinarie con l'eccezione che non è possibile utilizzare il modificatore /T toggle con essi. Si comporteranno quindi come espressioni non ripetitive, anche se il flag è su on, a meno che non utilizziate un modificatore /H hold, per mantenere premuto il pulsante logico.

Ok, ora probabilmente vi starete grattando la testa e vi chiederete:
"si, ma perché non hai semplicemente usato:

BTN S2 /H Fire_rockets

nel tuo file di configurazione? Perchè servono i flag logici?" . per quanto riguarda questo esempio, avete perfettamente ragione, non c'è bisogno di usare i flag logici ed entrambe le espressioni sono equivalenti. Presto però vedrete un esempio che non è possibile realizzare con la programmazione ordinaria – per ora vi anticipo la sintassi.

È anche possibile definire flag logici direttamente in espressioni digitali Type e direttamente con espressioni per i pulsanti:

RNG 2 5 X1 X2 X3 X4 X5
BTN H1L X8

sono espressioni valide.

Ora, c'è una leggera differenza tra il definire un flag logico utilizzando un'espressione di configurazione, rispetto a definirlo direttamente sul pulsante. Quando usate un comando come il seguente:

```
DEF X20 S1
```

il flag X20 è su on e vi rimane finché il pulsante S1 resta premuto. Se invece abbiamo

```
BTN S1 X20
```

nel nostro file di configurazione (*senza un comando DEF),* si comporterà come un semplice pulsante, es. X20 andrà a on e poi a off anche se tenete premuto il pulsante. Quindi

```
BTN S1 /H X20
```

si comporterà esattamente nella maniera in cui si comporta DEF X20. notate che non è possibile avere un'espressione DEF che definisce un flag logico e poi assegnare tale flag logico su un pulsante o un espressione riferita ad un asse.

Abbiamo detto che il fatto di impostare e poi definire un flag logico direttamente su un pulsante è suscettibile delle stesse regole che si applicano alla programmazione ordinaria dei pulsanti.
Di conseguenza potete generare flag utilizzando espressioni tipo:

```
BTN S1 KD(X8) DLY(2000) KD(X6) KU(X6 X8)
```

Questo può non sembrare un vantaggio, ma se volete impostare una fase di setup, seguita da una fase di auto-ripetizione, potete utilizzare un'espressione come questa:

```
BTN X1 /A Fire_Main_Guns
BTN S1 /H Switch_to_Main_Guns X1
```

Vedete che viene utilizzato /H , applicato alla parte X1 dell'espressione. Quindi l'effetto di questa espressione è la selezione del cannone viene eseguita una volta sola, ma il fuoco è auto-ripetuto tramite il flag logico. Bello eh?

NOTE

Fate attenzione a definire lo stesso flag tramite espressioni DEF e direttamente su due pulsanti diversi. Ad esempio:

```
DEF S4 X1
BTN S2 X1
```

questo equivale a scrivere:

DEF X1 S4 OR S2

quindi sia che si prema S4 che S2, X1 sarà portato a "on".

8.3 Comparatori Logici

I comparatori logici sono AND, NOT ed OR i quali possono anche essere usati insieme a parentesi.

Espressione di configurazione

DEF X1 S2 OR T6
DEF X2 S4 AND S3 AND H1U
DEF X3 S1 AND NOT X1

Espressioni di programmazione logica:

BTN X1 Fire_missile
BTN X2 Engines_off Gather_belongings Eject
BTN X3 AutoPilot

Vediamo la prima coppia:

DEF X1 S2 OR T6
BTN X1 Fire_missile

Il flag X1 può essere portato su on, sia premendo il pulsante S2 che il pulsante T6. Quindi premere il pulsante S2 o T6 provocherà il lancio singolo di un missile. Ok, potevamo ottenere lo stesso risultato con:

BTN S2 Fire_missile
BTN T6 Fire_missile

Vediamo una situazione che non può essere programmata direttamente sul pulsante:

DEF X2 S4 AND S3 AND H1U
BTN X2 Engines_off Gather_belongings Eject
(Spegni_motore Raccogli_effetti_personali Lanciati)

In questo esempio, il flag X2 viene portato a on solo quando sono premuti sia il pulsante S4 che S3 e quando l'Hat 1 è spinto verso l'alto. Sicuramente non una cosa che capiterà accidentalmente! Però, quando lo farete, atterrerete con il fondo della vostra tuta di volo, letteralmente!

Nell'esempio finale:

```
DEF X1 S2 OR T6
DEF X3 S1 AND NOT X1
BTN X3 AutoPilot
```

abbiamo definito X3 che viene portato a on dalla pressione di S1, ma questo non avviene se i pulsanti S2 o T6 sono premuti. Quindi il pulsante S1 attiverà l'autopilota, a meno che S2 o T6 non siano premuti.

Notate che i flag logici seguono gli stessi comportamenti non-ripetitivi che hanno le espressioni per i pulsanti quando vengono programmati direttamente sui pulsanti.
Una nota finale, prima di lasciare i comparatori logici: è possibile usare delle parentesi per raggruppare tra loro delle espressioni logiche. Le espressioni DEF possono essere costituite da qualsiasi combinazione di AND, OR e NOT, parentesi aperte e chiuse e riferimenti a pulsanti o flag, ammesso che il risultato sia una equazione logica valida, ad esempio:

```
DEF X1  (S1 AND NOT S2) OR (X5 AND (H1U OR H2U))
```

è perfettamente valido.

8.4 Il selettore logico

negli esempi che abbiamo utilizzato finora abbiamo impostato dei flag logici che passavano allo stato "on" quando un pulsante, o una combinazione di pulsanti, veniva premuta e ritornavano a "off" quando questi pulsanti venivano rilasciati.
È anche possibile forzare un flag logico a comportarsi un po' come l'interruttore della luce, ovvero a passare a on e a rimanerci anche se il pulsante viene rilasciato, e poi tornare a off quando viene premuto nuovamente il pulsante. In altre parole, vogliamo farlo agire come un selettore. Possiamo fare ciò inserendo un "*" dopo il flag o il pulsante.

Espressione di configurazione:

```
DEF X1 S4*
```

Espressione logica:

```
BTN X1 /A Chaff DLY(30) Flare DLY(30)
```

Quando il pulsante S4 viene premuto, il flag X1 è portato a on. Rimane in questo stato anche quando S4 viene rilasciato, dato che è presente il simbolo * dopo di esso. Solo quando S4 viene premuto nuovamente il flag tornerà ad off. L'effetto dell'esempio qui sopra è che quando premiamo e rilasciamo S4, inizieremo a rilasciare del chaff e dei flares dal nostro velivolo finché non saremo al sicuro dai SAM e dai sidewinder (*un giorno sfortunato credo!*). Premendo nuovamente S4 fermeremo il rilascio delle contromisure (a meno che, ovviamente, nel frattempo non abbiamo esaurito le scorte!)

NOTE

1. *Non è possibile assegnare un selettore logico direttamente ad un pulsante. Ad esempio:*

```
BTN T6 X1*
```

genererà un errore di compilazione. (questo comportamento è diverso dal comportamento originario del Thrustmaster F22 PRO). I selettori logici sono permessi solamente nelle espressioni DEF. La seguente, ad esempio, è permessa:

```
DEF X1 T6*
```

2. *notate che diversamente dalle espressioni per i pulsanti, il modificatore /T* ***non*** *può essere inserito in espressioni logiche per i pulsanti. Quindi:*

```
BTN X7 /T a /T b
```

provocherà un errore del compilatore. È però permesso:

```
BTN S4 /T X1 /T X2 /T X3
```

8.5 USARE LE FUNZIONI LOGICHE DELAY E PULSE

8.5.1 La funzione DELAY

La funzione DELAY inserisce un tempo di ritardo (delay, in inglese) tra il momento in cui la funzione logica genera un risultato e il momento in cui il flag passa a on. La sintassi è:

Espressione di configurazione:

```
DEF Xflag  DELAY(Flag_on_delay) Logical equation
```

Consideriamo l'esempio:

```
DEF X1 DELAY(1000) S1 AND S4
BTN X1 Eject
```

Questa espressione fa passare X1 a ON 1 secondo (1000 millisecondi) **dopo** la pressione simultanea di S1 e S4. Se uno qualsiasi di S1 o S4 viene rilasciato, il delay verrà interrotto. Se X1 non è ancora passato a ON, rimarrà su OFF, mentre se è passato a ON, tornerà ad OFF immediatamente. Verrà inoltre resettato il delay, quindi se successivamente verranno premuti assieme S1 ed S4, il ritardo verrà ripetuto interamente.

Nell'esempio, in particolare, è inserito per far fare attenzione a non lanciarsi per errore.

NOTE

1. *La funzione logica DELAY è completamente diversa dalla funzione DLY(). I valori ammessi per la funzione DELAY sono da 0 a 327760 (in particolare 5' 46" – non pensate nemmeno di chiedervene il motivo!)*

2. *La funzione DELAY deve apparire all'inizio dell'espressione. Quindi:*

`DEF X1 DELAY(1000) S1 AND S4` *è ok, ma:*

`DEF X1 X2 AND DELAY(1000) S1 AND S4` *genererà un errore.*

8.5.2 La funzione PULSE

La funzione PULSE invia una sequenza ON-OFF ripetuta finché l'equazione logica è vera. I due numeri nella funzione definiscono i periodi in cui si ha un ON e quando si ha un OFF. Ecco la sintassi:

Espressione di configurazione:

```
DEF Xflag PULSE(Time_on Time_off) Logical equation
```

Quindi per esempio:

```
DEF X1 PULSE(100 1000) H1U
BTN X1 Trim_up_increase
```

Portare l'HAT 1 verso l'alto farà passare X1 a ON immediatamente per 1/10 di secondo, quindi a OFF per 1 secondo, quindi a ON per 1/10 di secondo ecc...

Questa operazione verrà ripetuta in continuazione finché HAT1 sarà premuto. In questo particolare esempio, aumentiamo il trim ogni 1.1 secondi. Se avessi voluto aumentare il trim ogni secondo, avrei modificato l'espressione in:

```
DEF X1  PULSE(100 900) H1U
```

Notate che un carattere SPAZIO deve separare i due valori numerici. Usare una virgola o un altro carattere, causerà un errore di compilazione.

NOTE

1. *La funzione PULSE deve essere posta all'inizio dell'espressione. Quindi:*

DEF X1 PULSE(100 1000) S1 AND TG1	*è ok, ma:*
DEF X1 X2 AND PULSE(100 1000) S1	genera un errore.

2. *La risoluzione reale per ogni funzione DELAY, RATE e PULSE è di circa 30 millisecondi e il minimo valore è anch'esso di 30 millisecondi. Di conseguenza ogni valore compreso tra 1 e 30, produce un ritardo di 30 millisecondi, da 31 a 60 produce un ritardo di 60 millisecondi, da 61 a 90 un ritardo di 90 ecc....*

3. *I valori utilizzabili nella funzione PULSE vanno da 0 a 82800000 (23 ore).*

8.6 Esempi di programmazione logica

8.6.1 Portare un'espressione Type 4 da on a off e viceversa

Utilizzando un'espressione digitale Type 4, possiamo generare comandi, ripetitivi o a pulsazione, sulla manopola RNG come qui di seguito:

```
RNG 4 1000 a ^ b
```

Ora l'unico modo per fermare la pulsazione è muovere la manopola RNG nella sua posizione centrale, dove troviamo il carattere nullo (^). Invece, quello che vogliamo fare è avere la possibilità di avviare e interrompere la pulsazione premendo e rilasciando la manopola RNG, corrispondente ad esempio al pulsante T6. Con la programmazione logica, questo è facile ed è ottenibile con le seguenti espressioni:

```
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
```

Il suo funzionamento è il seguente. La manopola del range, invece di produrre direttamente comandi a pulsazione, ora porta a on e off i flag X1 e X2. I pulsanti X4 e 5 generano i caratteri a e b, ma solo se il flag X3 non è on. Visto che il pulsante T6 porta X3 a on e off, abbiamo ora il modo di avviare e interrompere l'invio dei comandi alla manopola.

8.6.2 Una funzione di trim lenta

Devo dare credito a chi se lo merita: questa sezione è un favore di Mark!
Prendiamo in considerazione il fatto che in un simulatore di volo, usiamo KP7 e KP1 per regolare l'assetto del velivolo verso l'alto e verso il basso. Cosa andremo a fare qui, programmando un HAT, è una funzione di regolazione lenta in modo che quando premiamo Hat1 up (in alto) e lo rilasciamo, verrà inviato ogni 5 secondi un comando KP7 che regolerà l'assetto. Quando premeremo nuovamente Hat1 up, interromperemo il ciclo. La stessa cosa succederà con Hat1 down (in basso). Ecco qui come:

```
DEF X1 H1U* AND (NOT X2)
DEF X2 H1D* AND (NOT X1)
BTN X1 /A KP7 DLY(5000)
BTN X2 /A KP1 DLY(5000)
```

Ora vi spiego cosa succede qui. Quando Hat 1 e spostato verso l'alto, porta il flag X1 su on, il quale rimane in tale posizione a causa dell'"*". Se X1 è a on, viene eseguita l'espressione associata a X1, e avremo KP7 ripetuto ogni 5 secondi. Quando Hat 1 è premuto nuovamente, porta X1 a off grazie all'asterisco e verrà interrotta la sequenza di KP7. Accade esattamente la stessa cosa per X2. L'inclusione di (NOT X1) assicura che se Hat 1 è stato spostato in basso e sta generando KP1, non generi KP1 e KP7 in contemporanea se viene portato verso l'alto.

Ci sono alcuni file scritti da Mark Mooney nella vostra cartella File con alcuni altri esempi di programmazione logica.

9. In caso di problemi

9.1 Resettare i controller

Ci sono molti modi di affrontare i problemi dei controller descritti qui sotto.

9.1.1 Nel gioco: EMPTY_BUFFERS e STICK_OFF

È possibile (*però con difficoltà!*) che vengano prodotti troppi caratteri dal Cougar, sia premendo troppi pulsanti troppo velocemente (*difficile*) o usando un file programmato male (*questo è possibile*) causando un errato comportamento del joystick. Il sintomo di questa situazione è quando il controller sembra fermo: gli assi analogici continuano a funzionare, ma sembra che nessun pulsante funzioni. In questa situazione è possibile svuotare il buffer del controller senza cancellarne la programmazione. Questo permette di continuare a giocare normalmente.

Sintassi:

EMPTY_BUFFERS

Esempio:

```
BTN S2 EMPTY_BUFFERS
```

Naturalmente questo è un comando che verrà usato raramente e perciò è utile programmarlo in una combinazione di tasti premuti per un certo tempo prima che venga scatenato in modo da non lanciarlo erroneamente.

```
DEF X1  DELAY(2000) S1 AND S4
BTN X1 EMPTY_BUFFERS
```

Nell'esempio precedente è necessario tenere premuti S1 ed S4 per due secondi prima che il flag X1 diventi attivo e che quindi venga inviato un singolo EMPTY_BUFFERS al controller.

Analogamente è possibile spegnere il controller all'interno del gioco con il commando STICK_OFF. Questo metterà tutti i pulsanti in modo Windows.

Sintassi:

STICK_OFF

Esempio:

BTN S2 STICK_OFF

Come per EMPTY_BUFFERS, STICK_OFF verrà usato raramente ed è sensato usare una combinazione di tasti più complessa per evitare pressioni accidentali.

DEF X1 DELAY(2000) S1 AND S4
BTN X1 STICK_OFF

Una volta lanciato questo comando non è possibile riattivare il controller quindi STICK_OFF va usato solo in situazioni di emergenza. Ad esempio quando la programmazione del controller causa la generazione incontrollata (ed incontrollabile) di caratteri. Successivamente si dovrà uscire dal gioco, analizzare il problema dagli strumenti disponibili su Windows.

9.1.2 Da Windows

Osservando il menu Cougar di Foxy, noterete voci per resettare diversi aspetti del controller. Eccoli elencati in ordine di durezza in modo che questa sezione possa servire da checklist d'emergenza da seguire per ripristinare il vostro lo stato mentale!

1. Empty buffers

Come spiegato sopra, è possibile svuotare solo il buffer dei caratteri generati dal controller in modo da mantenere lo stato del controller.

2. Disabling programmed mode

È molto semplice da Foxy mettere il controller in modalità Windows in modo che smetta di generare caratteri se questo fosse necessario. Se sfortunatamente i caratteri che creano il problema sono quelli funzione (come F1), allora divertitevi ad osservare la combinazione di finestre generate da questa situazione mentre cercate la spina USB del Cougar per scollegarlo dal PC!

3. Clear memory

Si attiva il modo Windows e i programmi in memoria vengono cancellati.

4. Flash memory

Se ci fossero problemi seri al controller e quindi Windows riconoscesse la periferica, ma nessun asse né pulsante funzionasse, o se una nuova versione del firmware venisse rilasciata, è possibile “flashare” il firmware del Cougar.

5. Chiamare il supporto tecnico!

Se il controller non venisse riconosciuto, se nessuno dei driver nativi di Windows funzionasse o se ci fosse un problema hardware, sarà necessario contattare il supporto tecnico per verificare se il controller debba essere rispedito per eventuali riparazioni.

10. Appendici

Appendice 1. Sommario delle espressioni Thrustmaster

Espressioni pulsante e modificatori di espressioni

Espressione	Acronimo	Descrizione
BTN	Pulsante/pulsante	Definisce il pulsante da programmare; include da Hat 1 a 4, da S1 a S4, da T1 a T10, TG1 2.
REM	Osservazione	Qualsiasi testo dopo REM su una riga viene ignorato dal compilatore. Usato per commenti, titoli ecc.
RESET_TOGGLES	Toggle reset	Resetta la serie dei toggle al primo /T.
REVERSE_TOGGLES	Toggle inverso	Riproduce il toggling in direzione inverse.
DLY	Ritardo	Aggiunge una pausa tra i caratteri o le macro.
RPT	Ripetizione	Ripete un carattere o una macro.
( )	Parentesi di raggruppamento	Raggruppa caratteri/macro insieme per varie espressioni inclusi i raggruppamenti di espressioni digitali.
{ }	Parentesi graffa di raggruppamento	Raggruppa i caratteri insieme obbligando tutti i loro codici di creazione ad essere generati prima dei loro codici di chiusura. Simile a tener premuti un gruppo di pulsanti.
< >	Parentesi ad angolo	Obbligano le espressioni in esse contenute ad essere eseguite completamente prima che vengano eseguite le espressioni successive.
DX	Pulsanti DirectX	Pulsanti DirectX che possono essere programmati attraverso qualsiasi espressione.
KD KU	Tasto in giù Tasto in su	Permette il controllo degli eventi premuto e rilasciato di un tasto.
USB	USB keyboard scan codes	Usato per generare il codice di ogni carattere crea e chiudi.
REVERSE_UD REVERSE_LR REVERSE_DIR	Direzione inversa dei controller	Inverte la direzione dei controller es. in espressioni speciali degli HAT.

Espressione	Acronimo	Descrizione
NOHOLD, KP5	Effetua Espressioni USA HAT COME	NOHOLD blocca l'espressione dell'Hat che produce caratteri continui. KP5 aggiunge il centro del KP5 per espressioni USA HAT COME KEYPAD.
FORCED_CORNERS	Angoli degli HAT	Obbliga la creazione delle posizioni angolari degli hat da uno hat standard a 4 posizioni.
S3_LOCK, S3_UNLOCK	Blocca S3, Sblocca S3	Blocca S3; Sblocca S3.
SHIFTBTN	Assegna S3	Assegna un pulsante diverso per S3.
POV*n*, (n = D, L,R,UL, DL,UR, DR)	Posizioni HAT "Point Of View "	Permette il controllo in programmazione delle posizioni del POV hat.
MOUSE_LB, MOUSE_RB, MOUSE_MB	Pulsanti Mouse	Permette il controllo in programmazione dei pulsanti del mouse.

Modificatori barra e modificatori di Espressioni

Modificatore barra	Acronimo	Descrizione
/U, /M, /D	Su, Mezzo, Giù	Triplica le posizioni programmabili per ogni hat/pulsante (tranne T7 e T8) utilizzando le posizioni del selettore dogfight della manetta.
/I, /O	Dentro, Fuori	Raddoppia le posizioni programmabili di ogni pulsante/hat quando il pulsante S3 sul joystick viene premuto (non si può usare più *S3*).
/P, /R	Premi, Rilascia	Separa la programmabilità di ogni controller quando viene premuto, e poi quando viene rilasciato.
/T	Toggle	Ad ogni pressione del pulsante esegue un Toggle in avanti attraverso vari caratteri/macro.
/H	Mantieni	Produce caratteri come se venisse premuto costantemente un pulsante senza tenere in considerazione se altri caratteri sono premuti. Può venir usato con altri Modificatori a barra.
/A	Auto-Repeat	Ripete tutto di quella riga.

Espressioni di Configurazione

Sintassi	Descrizione
USE MDEF	Indica quale file macro contiene le definizioni delle macro del joystick file corrente.
USE *Btn* AS DXn	Assegna i pulsanti DirectX. Rimpiazza la sintassi PORTB1 IS.
USE ALL_DIRECTX_BUTTONS	Assegna l' Hat1 come POV e tutti gli altri pulsanti come DirectX.
USE HAT*n* FORCED_CORNERS	Converte un hat a 8 vie in un hat a 4 vie, in modo tale che la posizione ad angolo esegua l'espressione.
USE HAT AS MOUSE, POV, ARROWKEYS, KEYPAD	Definisce come usare un Hat se non come un normale BTN.
USE RATE	Rateo di Default della creazione/ripetizione dei caratteri.
USE S3_LOCK USE S3_UNLOCK	Modifica l'S3 per farlo comportare più come un selettore.
USE S3 AS SHIFTBTN	Definisce un diverso pulsante da usare come S3 /I, /O.
USE HATx_SENSITIVITY	Riduce la sensibilità per aiutare l'uso delle posizioni di angolo degli HAT.
USE T1_SENSITIVITY	Setta la sensibilità di T1.
USE FOXY	Usato all'interno di Foxy per varie funzioni, es. USE FOXY GRAPHIC USE FOXY README
USE NULLCHR	Definisce quale carattere avrà la funzione di null – il default è ^.
USE KEYBOARD	AZERTY – per la disposizione delle tastiere Azerty e problemi di rimappatura nei giochi Francesi.
USE PROFILE	Usa i profili salvati dal Control Panel del TM Cougar.
USE CURVE	Definisce la curva di risposta di un asse.
DISABLE AXIS	Disabilita un asse.
USE SWAP	Esegue lo scambio degli assi.
USE REVERSE	Inverte un asse.
USE AXES_CONFIG	Definisce quale asse usare, ed altri attributi.
USE MTYPE	Assegna il controllo del mouse al microstick, e definisce quali pulsanti usare per i tasti del mouse.
USE MICROSTICK AS MOUSE - NO_BUTTON	Assegna il mouse al microstick, ed il pulsante sinistro del mouse a T1 (tranne se - NO_BUTTON è presente).
USE Axis_Identifier AS Mouse_Axis	Assegna un asse per il controllo del mouse.
USE ZERO_MOUSE	Previene il blocco del mouse con /I, /O.
DISABLE MOUSE	Disabilita l'assegnazione di default del mouse.
USE SCREEN_RESOLUTION	Usato in movimenti complessi del mouse.

Programmazione degli Assi

Espressioni Assi	Acronimo	Descrizione
JOYX, JOYY	Joystick X e Y	Assi del Joystick.
THR	Manetta	Assi della manetta.
RDDR	Timone	Assi del Timone.
ANT	Pomello antenna	Asse del pomello antenna.
RNG	Pomello distanza	Asse del pomello distanza.
MIX, MIY	Microstick X e Y	Assi del Microstick.
LBRK and RBRK	Freno sinistro e destro	Assi dei freni sulla pedaliera.
MSX, MSY, MSZ	Mouse X, Y, Z	Assi del Mouse- Z è la rotellina del mouse.
Espressioni digitali 1 to 6		Produce caratteri della tastiera attraverso gli assi. Usato anche con le curve del mouse ed espressioni logiche delle flag.
FORCE_MACROS	Forzare le macro	Forza le caratteristiche dei caratteri/macro nelle espressioni di tipo 1, 2, 5, 6 a venir sempre generate.
CURVE	Curva dell'asse	Curve degli assi – cambia la risposta degli assi e la sensibilità.
TRIM TO_CURRENT HOLDTRIM	Trim dell'asse	Trimma un asse al valore specifico.
LOCK, UNLOCK, LASTVALUE	Blocco dell'asse	Blocca il valore di un asse in modo tale da poterlo usare solo per via digitale.
SWAP	Swap degli assi	Esegue lo scambio degli assi.
REVERSE FORWARD	Direzione degli assi	Inverte un asse, ripristina la normale direzione.

Espressioni Avanzate del mouse

Espressioni Mouse	Descrizione
MOUSEXY	Posiziona il cursore del mouse in particolari coordinate dello schermo.
MOUSEMOVE	Muove il mouse, rispetto alla sua posizione corrente.
MOUSEROTATE	Programma il mouse a muoversi in modo circolare, permettendo ai pomelli del cockpit di venir controllati tramite la programmazione.

Espressioni Logiche

Sintassi Logica	Acronimo	Descrizione
DEF	Definisci	Definisce flag logiche attraverso equazioni.
BTN	Pulsante	Usato per assegnare espressioni virtuali ai pulsanti logici da X1 a X32.
X1 to X32	Flag logiche	Flag logiche che sono o ON o OFF.
AND, OR, NOT	Comparatori logici	Usato per la costruzione di equazioni logiche.
*	Toggle	Esegue il Toggle di una flag logica tra le sue posizioni di on e di off.
DELAY	Ritardo	Esegue una pausa dopo la quale una condizione logica diventa vera.
PULSE	Pulsazione	Riproduce gli stati di caratteri/flag ogni *nn* millisecondi.

Espressioni Hardware

Sintassi	Descrizione
EMPTY_BUFFERS	Svuota il buffer di memoria dei controller.
STICK_OFF	Spegne il controller durante un gioco.

Appendice 2. Sintassi di default 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.	

NOTE

1. *La sintassi per tasti raggruppati (quando si premono insieme Shift, ALT o CTRL) è* SHF a, ALT b, CTL c. Non è LSHF a, LALT b LCTL c

2. Alcuni tasti sono riservati*:* ( ) { } < > e devono venir programmati con le espressioni SHF*:*

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

Appendice 3. Codici Keydown e Keyup USB

Esempio:

BTN S2 /P USB (D04) Rem “a” keydown
/R USB (U04) Rem “a” keyup

Nome tasto	Sintassi TM	Codice HID 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 (*Guarda le note*)		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 (*Guarda le note*)		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

NOTE

Questi tasti hanno vari significati in base al luogo di fabbricazione. Europe 1 è la tipica posizione 42 dei tasti AT-101 a fianco del tasto INVIO. Europe 2 è la tipica posizione 45 dei tasti AT-101, tra lo shift sinistro ed il tasto Z.

Appendice 4. Differenze tra i file originali Thrustmaster ed i file del Cougar

Ci sono alcune differenze sottili (e alcune non così sottili) tra i file originali Thrustmaster che supportano i chip digitali, F22, FLCS, FCS, TQS, WCS MkII. Questa Appendice farà la somma di tali differenze – cercate nella documentazione Thrustmaster per maggiori spiegazioni.

1. Cambiamenti nella sintassi dei tasti

Vecchia Sintassi	Nuova Sintassi
LSFT	LSHF
RSFT	RSHF
n.d.	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. Cambiamenti nei modificatori a barra

Modificatore a barra	Commento
/U	Come in precedenza.
/M	
/D	
/I	Come prima ma le espressioni /I devono sempre apparire prima delle espressioni /O, e /I, /O devono essere in righe diverse.
/O	
/P	Come in precedenza.
/R	
/T	Come in precedenza.
/A	Ora definisce le espressioni di auto-repeat.
/H	Come prima ma i caratteri sono generati a ripetizione, e in un espressione complessa, viene applicato all'ultimo carattere dell'espressione.
/F	Non più supportato.
/Q	Non più supportato.
/N	Non più supportato – Tutte le espressioni si comportano come se ripetono almeno che non siano presenti i modificatori /H, /A.

3. Espressioni non più supportate

Espressione	Commento
RAW ( )	Rimpiazzato da USB ( ) *(codici HID)* ma dalle più facili da usare espressioni KD( ) e KU( ).
BTN MT	Usate piuttosto le espressioni di Tipo 5 – sono più potenti.
BTN T11 – T14 (non esistono più)	Il microstick è come un joystick a 2 assi – non è un controller a 4 pulsanti. Quindi più venir programmato digitalmente con espressioni dal tipo 1 al 6.
USE RCS	Non più necessaria.
USE TQS	Non più necessaria.
USE WCS	Non più necessaria.
USE RCSPRO	Non più necessaria.
USE NOMOUSE	Non più necessaria.
USE NOTHR	Non più necessaria.
USE MTYPE (B or C)	Tipo mouse non più supportato, anche se le espressioni per mouse a 3 pulsanti possono venir generate.

4. Estensione dei file, nomi dei file

I file joystick devono finire in ".tmj" e i file macro in ".tmm". Sia i nomi dei file joystick che dei macro possono contenere spazi e non sono limitati a 8 caratteri. Come in precedenza però i file joystick devono essere nella stessa cartella/directory dei file macro file. Per default questa è la cartella Files di Foxy.

5. Azioni di default

Il compilatore eseguirà delle azioni di default per i vostri file, in base a come avete impostato le preferenze in Foxy. Queste opzioni sono saltate se selezionate in Foxy, o se avete programmato i vostri file per comportarsi in modo diverso.
E sono:

- Hat 1 o Hat di vostra preferenza come hat Point Of View (POV)
- TG1 come DX1, S2 come DX2 – es. come pulsanti DirectX, così questi pulsanti avranno una funzione assegnata direttamente dal gioco.
- Se non è presente una linea USE MDEF, e il file joystick contiene delle macro, allora il compilatore cercherà le macro da un file con lo stesso nome del file joystick, ma con l'estensione ".tmm"
- Il Microstick controllerà i movimenti del mouse, con T1 sul microstick che opererà da pulsante per mancini.

Le ragioni per questi default sono di agevolare i principianti nella programmazione. Quindi se un utente ha un file joystick con soltanto:

BTN S1 Autopilot

E un file macro con solamente:

Autopilot = a

Potrà scaricare il file joystick direttamente e trovare che il grilletto funziona, che il mouse funziona ecc.

L'unico problema sussiste se i file vengono passati a qualcun altro che ha modificato i settaggi di default in Foxy, ad esempio per generare un errore se non viene trovata la linea USE MDEF, Il file per quell'utente non funzionerà.

6. Assi Digitali e Analogici

Con i file originali Thrustmaster, un asse poteva venir programmato sia in modo digitale, producendo quindi caratteri, o lasciato nella modalità di default analogica, dove la sua funzione veniva gestita dal gioco (es. Asse manetta = spinta in un simulatore di volo). Con l' HOTAS Cougar, gli assi posso essere sia analogici che digitali allo stesso tempo se richiesto. Per avere un puro asse digitale, la sua funzione analogica deve venir disabilitata con l'espressione DISABLE. È da notare inoltre che con il Cougar, non siete obbligati a cambiare niente nell'applet Control Panel's Gaming Options, se volete usare un asse in modo digitale, Cosa che dovevate invece fare con i controller originali Thrustmaster.

7. Espressione digitale di Tipo 1

La sintassi per queste è cambiata – controllate la sezione in questo manuale.

8. Manetta non presente

Se avete un file scritto per joystick e manetta, e la manetta non è presente, allora con le espressioni /U, /M, /D solo l'espressione /M verrà compilata – le /U, /D verranno ignorate. Le espressioni riguardanti l'asse manetta verranno anch'esse ignorate.

9. Macro – Caratteri vietati

Non potete usare i seguenti caratteri per i nomi delle macro:

= < > { } ( ) ^ , *spazio*

10. RPT

Se avete una macro come la seguente: Macro1 = a b c

E un'espressione del tipo: BTN S2 RPT (3) Macro1

Allora se verrà premuto S2 otterrete: a a a b c

Per evitare ciò, o racchiudete le macro o i caratteri nella sua definizione con parentesi, es.:

BTN S2 RPT (3) (Macro1)

o

BTN S2 RPT (3) Macro1 Dove: Macro1 = (a b c)

11. I Caratteri commento //

Con i chip digitali, potevate usare i caratteri // al posto di REM. Questi non sono più supportati in Foxy.