Design Pattern #5- Queued State Machine

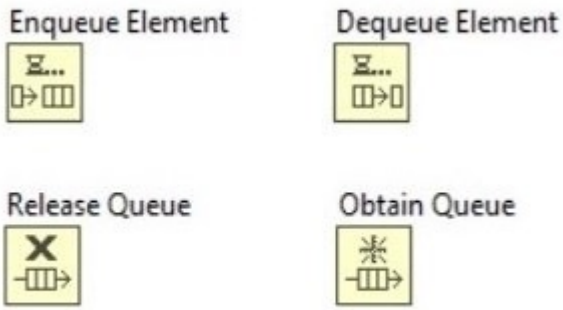
Category: LabVIEW Talk
scritto da Davide Oratelli | Marzo 23, 2020



Queued State Machine

Nel precedente articolo abbiamo visto il funzionamento della Macchina a Stati Multipla, oggi incominceremo a parlare di Queued State Machine.

Come suggerito dal nome la Queued state machine lavora con le code, per capire bene come funziona questo tipo di macchina a stati è necessario spiegarvi la funzionalità di alcune funzioni necessarie all' interno della nostra applicazione.



Obtain Queue: Crea un riferimento a una coda, è possibile specificare il nome, tipo di dato e grandezza della coda.

Trattandosi di un riferimento e non del dato potremo ulizzarlo per riferirci al dato da altre parti del codice anche da loop paralleli.

Enqueue Element: Aggiunge un elemento alla coda, necessita della Reference e dell'elemento da inserire.

Enqueue accoda i dati in modalità FIFO, utilizza "Enqueue Element At Opposite End" per accodare in LIFO.

Dequeue Element: Rimuove un elemento dal davanti della coda (First Output) e ritorna l'elemento rimosso.

Importante: Se non ci sono dati in coda e non è stato impostato il timeout (-1) rimane fermo attendendo il dato in coda.

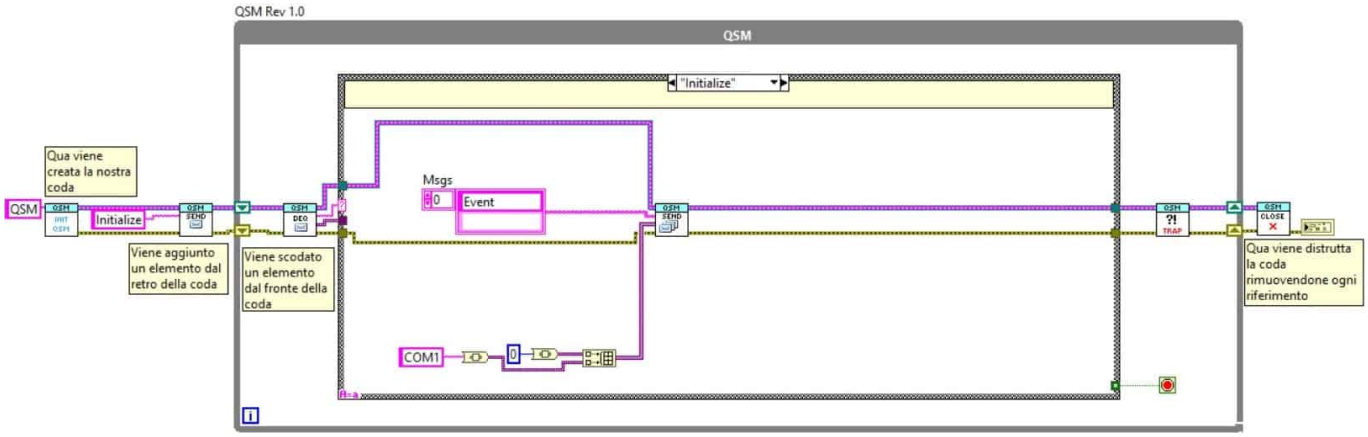
Release Queue: Rilascia il riferimento alla coda, distruggendola.

Tornando alla QSM è un particolare stile di implementazione flessibile e versatile, facile da mantenere ed efficiente dal punto di vista computazionale.

Soprattutto data la possibilità di collegarsi mediante reference agli altri loop è espandibile.

L'utilizzo delle Queues quindi porta benefici notevoli.

Si introduce uno scambio di dati message based , ovvero inviamo un messaggio e lo riceviamo da altre parti del codice(vedremo in seguito non necessariamente nello stesso loop).



Esempio di QSM

Abbiamo realizzato una piccola libreria per incapsulare un tipo di messaggio che possa portare sia lo stato sia i parametri.

Per farlo utilizziamo un cluster con una stringa (msg) e un variant(dato).

Utilizzando le code rimuovi la possibilità di perdita di dati, rendi loops paralleli sincronizzati e non necessita il polling.

Le nostra SM come quelle standard, utilizza i 5 casi principali che abbiamo visto nei precedenti articoli, ovvero: Initialize, Defaul, Event, Error ed Exit.

All' interno della nostra macchina stati abbiamo utilizzato anche un tipo di dato un pò insolito chiamato Variant, che può essere visto come un contenitore generico per gli altri tipi di dati, rendendo l'applicazione più versatile.

In conclusione si può quindi dire che la QSM è sicuramente una macchina a stati che garantisce diversi vantaggi.

Viene implementata in diversi framework in modi e meccanismi diversi, la sua principale evoluzione in framework è il QMH.

La Macchina a Stati in coda rimane il pattern di programmazione tra i più utilizzati nel mondo labVIEW, per la sua possibilità di passare dati tra loop

paraleli e per non perdere i dati.

Presenta quindi scalabilità e flessibilità.

Nel prossimo articolo vedremo la funzionalità del Produer/Consumer Pattern.